Secure Computation and Probabilistic Checking

Mor Weiss
Secure Computation and
Probabilistic Checking

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Mor Weiss

Submitted to the Senate
of the Technion — Israel Institute of Technology
Nisan 5776 Haifa April 2016
This research was carried out under the supervision of Prof. Yuval Ishai, in the Faculty of Computer Science.

Some of the results discussed in this thesis have been published by the author and research collaborators in conferences during the course of the author’s doctoral research period, the most up-to-date versions of which being:


### Acknowledgements

I am indebted to many people for making my years at the Technion so wonderful.

First and foremost, I am grateful to my advisor. Yuval, (almost) everything I know about research - I learned from you. Thank you for always making time for me, for always having an open door, and helpful advice. Working with you has taught me a great deal about abstraction, about moving away from the tantalizing details, and seeing “the big picture”. You also taught me how to pose the “right” questions, in a precise way, and which tools to use to find the “right” answer. Thank you for being so passionate about Cryptography, it inspired me to be the best researcher that I can be. Thank you for constantly pushing me, and for encouraging me so that I would succeed. Thanks to you I have numerous wonderful experiences from this period.

I was very fortunate to have wonderful collaborators during this period, both in projects related, and unrelated, to this thesis. Amit Sahai, Michael Viderman, Eric Miles, Boris Rozenberg, Muhammad Barham, Guang Yang, Daniel Genkin, Tal Rabin, and Shai Halevi - thanks for making our joint projects so productive, interesting, and also fun! I especially want to thank Amit Sahai for hosting me at UCLA for a summer internship, and introducing me to the fascinating world of obfuscation. I am especially grateful to Tal Rabin, who supported me and encouraged me throughout the last part of my PhD, and (among many other things) hosted me for a very interesting and enjoyable internship at the Cryptography group at IBM Thomas J. Watson Research Center at Yorktown Heights.

I would also like to thank all my friends at the department - you know who you are! You have been my family at the Technion during my studies. Thanks for making this period an unforgettable one!

Last, but not least - I thank my family: Aba, Ima, Or, Tomer, Lotem, and Rimon - your support and love means the world to me, I couldn’t have done this without you. Thanks for letting me “bore” you with endless details about my research, and for
reminding me there’s a life besides Crypto. Saba, thank you for your constant interest in what I do. This thesis is dedicated to you.

The generous financial help of the Technion is gratefully acknowledged.
# Contents

List of Figures

<table>
<thead>
<tr>
<th>Abstract</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction</td>
<td>3</td>
</tr>
<tr>
<td>1.1 Our Results</td>
<td>7</td>
</tr>
<tr>
<td>1.1.1 Non-Adaptively Verifiable ZKPCPs</td>
<td>7</td>
</tr>
<tr>
<td>1.1.2 ZKPCPPs for NP</td>
<td>8</td>
</tr>
<tr>
<td>1.1.3 Stable ZKLTCS</td>
<td>8</td>
</tr>
<tr>
<td>1.1.4 Cryptographic Applications of ZK Proofs and Codes</td>
<td>9</td>
</tr>
<tr>
<td>1.1.5 Leakage-Resilient Computation</td>
<td>11</td>
</tr>
<tr>
<td>1.2 Organization</td>
<td>12</td>
</tr>
<tr>
<td>2 Preliminaries</td>
<td>13</td>
</tr>
<tr>
<td>2.1 Some Facts about Probabilities</td>
<td>16</td>
</tr>
<tr>
<td>2.2 Probabilistic Proof Systems</td>
<td>16</td>
</tr>
<tr>
<td>2.2.1 Probabilistically Checkable Proofs</td>
<td>17</td>
</tr>
<tr>
<td>2.2.2 Probabilistically Checkable Proofs of Proximity</td>
<td>18</td>
</tr>
<tr>
<td>2.2.3 ZKPCPs and ZKPCPPs</td>
<td>19</td>
</tr>
<tr>
<td>2.3 Codes</td>
<td>21</td>
</tr>
<tr>
<td>2.3.1 Efficiently Encodable and Decodable Error Correcting Codes</td>
<td>22</td>
</tr>
<tr>
<td>2.3.2 Locally Testable Codes</td>
<td>22</td>
</tr>
<tr>
<td>2.3.3 ZK Codes</td>
<td>23</td>
</tr>
<tr>
<td>3 Leakage-Resilient Secure Computation</td>
<td>25</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>25</td>
</tr>
<tr>
<td>3.1.1 Our Results and Techniques</td>
<td>27</td>
</tr>
<tr>
<td>3.1.2 Chapter Organization</td>
<td>31</td>
</tr>
<tr>
<td>3.2 Preliminaries</td>
<td>32</td>
</tr>
<tr>
<td>3.2.1 Circuits</td>
<td>32</td>
</tr>
<tr>
<td>3.2.2 Additive Attacks</td>
<td>33</td>
</tr>
<tr>
<td>3.2.3 Encoding Schemes</td>
<td>33</td>
</tr>
<tr>
<td>3.2.4 Circuit Compilers</td>
<td>36</td>
</tr>
</tbody>
</table>
### 3.2.5 Leakage-Resilient Circuit Compilers
- Page 36

### 3.2.6 LRCCs with Soundness Properties
- Page 37

### 3.2.7 Multiparty LRCCs
- Page 38

### 3.3 Gadget-Based LRCCs
- Page 40
  - 3.3.1 The LRCC of Faust et al. [FRR+14]
- Page 42

### 3.4 SAT-Respecting Relaxed LRCCs Over Non-Binary Fields
- Page 45
  - 3.4.1 Roadmap Towards Proving Proposition 3.4.4
- Page 47
  - 3.4.2 The Relaxed Leakage-Resilience Property of Construction 3.15
- Page 49

### 3.5 Boolean SAT-Respecting Relaxed LRCCs
- Page 61

### 3.6 SAT-Respecting LRCCs that Resist Leakage from \( AC^0 \) Circuits Augmented with \( \oplus \) Gates
- Page 68
  - 3.6.1 A Leakage-Indistinguishable Encoding Scheme
- Page 69
  - 3.6.2 A Proof of Theorem 3.23
- Page 77

### 3.7 A Passive-Secure Multiparty LRCC
- Page 80
  - 3.7.1 The Effect of Ill-Formed Masking Inputs in Construction 3.31
- Page 83

### 3.8 A Multiparty LRCC
- Page 90
  - 3.8.1 Boolean Multiparty LRCCs
- Page 95
  - 3.8.2 Multiparty LRCCs Over Large Fields
- Page 99

### 4 Non-Adaptive Probabilistically Checkable Proofs with Zero-knowledge
- Page 101
  - 4.1 Introduction
- Page 102
  - 4.1.1 Our Results and Techniques
  - 4.1.2 Chapter Organization
- Page 104

### 5 Probabilistically Checkable Proofs of Proximity with Zero Knowledge
- Page 121
  - 5.1 Introduction
  - 5.1.1 Our Results and Techniques
  - 5.1.2 Chapter Organization
- Page 125
  - 5.2 Preliminaries
    - 5.2.1 Tools
- Page 125
  - 5.3 Exact ZKPCPPs from MPC Protocols: Feasibility
    - 5.3.1 Exact ZKPCPP Over Large Alphabets
    - 5.3.2 Exact ZKPCPPs over the Boolean Alphabet
- Page 133
  - 5.4 From Efficient PCPPs to Efficient ZKPCPPs
    - 5.4.1 From PCPPs to HVZKPCPPs
5.4.2 HVZKPCPPs with Stronger Honest-Verifier Zero-Knowledge Properties .................................................. 144
5.4.3 From HVZKPCPPs and Locking Schemes to ZKPCPPs ................................................................. 149

6 Locally Testable Codes with Zero-knowledge .......................... 157
   6.1 Introduction ....................................................................................................................... 157
   6.1.1 Our Results and Techniques ....................................................................................... 159
   6.1.2 Chapter Organization ................................................................................................. 160
   6.2 Explicit ZK Codes ............................................................................................................ 160
   6.3 General ZK Codes ........................................................................................................... 162
   6.4 Probabilistic Construction of ZK Codes from Linear Codes ........................................... 164
      6.4.1 A Proof of Theorem 6.3 ............................................................................................. 164
      6.4.2 A Proof of Corollary 6.4 .......................................................................................... 167
   6.5 Stable ZKLTCs ................................................................................................................ 168
      6.5.1 Tensor Products of Codes ......................................................................................... 170
      6.5.2 Stable LTCs via Tensor Product Codes ...................................................................... 172

7 Cryptographic Applications ............................................... 175
   7.1 Introduction ....................................................................................................................... 175
   7.1.1 Our Results and Techniques ....................................................................................... 176
   7.1.2 Chapter Organization ................................................................................................. 183
   7.2 Preliminaries .................................................................................................................... 183
   7.3 Stable Probabilistic Proof Systems .................................................................................. 184
   7.4 Quasilinear Communication VSS and Generalizations of VSS ......................................... 186
      7.4.1 Quasilinear Communication VSS from ZKLTCs ...................................................... 189
      7.4.2 Verification-Efficient Certifiable VSS from ZKPCPPs .............................................. 194
   7.5 Distributed Coin-Flipping From ZKLTCs ........................................................................ 203
   7.6 Distributed Zero-Knowledge and Witness-Indistinguishable Proofs .................................. 205
      7.6.1 Distributed Witness-Indistinguishable Proof Systems ............................................. 207
      7.6.2 Distributed Computational Zero-Knowledge Proof Systems ................................ 210
   7.7 Two-party Commit-and-Prove .......................................................................................... 213
      7.7.1 Statistical Commit-and-Prove with Super-Constant Many Rounds ......................... 219
      7.7.2 Computational Commit-and-Prove with a Constant Number of Rounds ............... 232
      7.7.3 Generalized (Multiparty) Commit-and-Prove .......................................................... 238
      7.7.4 Updatable Databases ................................................................................................. 246

8 Conclusions ................................................................. 249

A The Complexity of the Arora-Safra PCP [AS92] .............. 251
   A.1 The Construction over Large Fields ............................................................................... 251
   A.2 The Construction over GF(2) .......................................................................................... 254
A.3 Setting the Parameters ................................. 255

B Amplifying Transformations for Locking Schemes 257

B.1 Equivocation Amplification .............................. 257
B.2 Amplifying Binding ...................................... 262
B.3 Strong Equivocation of the Locking Scheme of [IMS12] .............................. 265
B.4 Exponentially Hiding and Binding Locking Schemes with Short Locks . 269

Hebrew Abstract
List of Figures

2.1 List of Symbols ............................................................. 14
2.2 List of Abbreviations ....................................................... 15
Abstract

In the past few decades, probabilistic checking techniques were shown to yield dramatic efficiency improvements in verifying proofs and approximating distance from error-correcting codes. Concurrently, so-called “side channel” attacks launched on physical devices resulted in major security breaches, and have led to a surge of interest in “leakage-resilient” hardware. In this thesis we study “zero-knowledge” variants of probabilistic checking techniques, and generalizations of leakage-resilient hardware, and apply them towards improving the efficiency, and generalizing the scope, of cryptographic protocols. Our contributions lie in two main areas.

Primarily, we study “zero-knowledge” variants of three probabilistic checking techniques. First, Probabilistically Checkable Proofs (PCPs), which are proofs that allow a randomized verifier with oracle access to a purported proof to probabilistically verify an input statement of the form “$x \in L$”, by querying only few proof bits. Second, PCPs of Proximity (PCPPs) which have the additional feature of allowing the verifier to query only few bits of the input $x$ and a purported proof, where if the input is accepted then the verifier is guaranteed that (with high probability) $x$ is close to some $x' \in L$. Thirdly, Locally Testable Codes (LTCs) which are error correcting codes that allow a tester to probabilistically test membership in the code by probing few symbols of a purported codeword. Zero-knowledge PCPs are PCPs with the additional guarantee that the view of any (possibly malicious) verifier reading a large portion of the proof can be simulated given $x$ alone. We introduce the notion of zero-knowledge PCPPs and zero-knowledge LTCs, which possess the property that even a large portion of the input and proof, or the purported codeword, reveals no information about the encoded input or message. We construct zero-knowledge variants of PCPs, PCPPs, and LTCs, and apply them to design communication-efficient protocols for various cryptographic tasks, such as verifying a shared secret, flipping a common random coin, and zero-knowledge proofs in the two-party and multiparty setting.

Second, we investigate generalizations of leakage-resilient circuit-compilers (LRCCs), which compile any circuit into a new circuit that operates on encoded inputs, and withstands side-channel attacks in the sense that these reveal nothing about the (properly encoded) input, other than what follows from the output. We construct LRCCs that guarantee correctness of the computation even when the inputs to the compiled circuit may not be properly encoded, namely the compiled circuit emulates the original circuit
on some set of inputs (determined by the possibly ill-formed encodings given to the compiled circuit). Additionally, we use LRCCs to design a single leaky (but otherwise trusted) stateless hardware device that allows $m \geq 1$ mutually distrusting parties to jointly compute a function of their inputs, while preserving input secrecy, and the correctness of the outputs, in the presence of active corrupted parties that may also obtain leakage on the internals of the device.
Chapter 1

Introduction

The broad goal of this thesis is to improve the complexity, and underlying assumptions, of cryptographic protocols. More specifically, we show a connection between “cryptographic” probabilistic verification techniques and cryptographic primitives such as secure multiparty computation and leakage-resilient computation, and make the following two related contributions. First, we show “zero-knowledge” variants of probabilistic verification and testing techniques that offer information-theoretic security against computationally unbounded corrupted parties, and apply these towards improving the complexity of cryptographic protocols. Second, we use leakage-resilient circuits to construct leakage-resilient hardware, such as a single leaky (but otherwise trusted) hardware device that enables \( m \geq 1 \) parties to compute a general joint function of their inputs.

This study is motivated by recent dramatic developments in the nature of computations, as well as by our changing perception of the adversarial entity. Indeed, as more and more computations are delegated to “The Cloud” - an extremely powerful external service provider - and performed by powerful, but possibly unreliable, servers, users need efficient techniques of verifying that “The Cloud” correctly performed the requested computations. Moreover, the increasing asymmetry between the computational powers of “The Cloud”, and the weak users (e.g., smartphones, tablets, etc.), necessitates security to hold against all-powerful cheating servers, while requiring very little computational resources from users.

Concurrently, numerous recent examples have exhibited that physical devices (such as those used to implement cryptographic protocols) inadvertently leak information regarding the data that they store, and the computations that they perform, through various side-channels. So-called “side-channel” attacks exploit such leakage, and often lead to a complete security breach. Consequently, leakage has been incorporated into the definitions of several formal computation models. However, current solutions that offer security with no information leakage require the participating parties to employ multiple leaky hardware devices, leaving open the question of basing leakage-resilient secure computation on a single leaky (but otherwise trusted) device. Such implementations
would be much preferable to basing security on _multiple_ devices, since the ability to secure _multiple_ devices against leakage seems less probable.

Probabilistic proofs, and leakage-resilient hardware, are fundamental building blocks for tackling many issues arising in these fields. Indeed, verification and testing techniques often lie at the heart of cryptographic protocols, since the mutually distrusting participating parties are required to prove that they followed the protocol correctly. Thus, improving and generalizing such techniques can potentially significantly improve various cryptographic tasks. Basing leakage-resilient secure computation on a single leaky hardware device may similarly improve cryptographic protocols, since securing a single device, even one that performs _complex_ computations, seems easier than securing multiple (even _computationally simple_) devices against leakage, and thus may lead to simpler, and more efficient, protocols.

Zero-knowledge verification techniques, and leakage-resilient hardware, are not just “a means to an end”. Rather, these powerful objects are interesting on their own accord, and our research is therefore also motivated by the theoretical interest in several natural generalizations, and open questions left unanswered by previous research, as we explain next.

**Probabilistically Checkable Proofs and Proofs of Proximity**

Probabilistically Checkable Proof (PCP) systems [ALM+92, AS92] are proof systems that allow an efficient randomized verifier, with oracle access to a purported proof generated by an efficient prover (that is also given the witness), to probabilistically verify claims of the form “$x \in L$” (for an NP-language $L$) by probing only few proof bits. The verifier accepts the proof of a true claim with probability 1 (the _completeness_ property), and rejects false claims with high probability (the probability that the verifier accepts a false claim is called _the soundness error_). The celebrated PCP theorem [ALM+92, AS92, Din06] asserts that any NP language admits a PCP system with soundness error $1/2$ in which the verifier reads only a _constant_ number of proof bits (where soundness can be amplified through repetition). Moreover, the verifier is _non-adaptive_, namely its queries are determined solely by its randomness (whereas each query of an _adaptive_ verifier may also depend on the oracle answers to previous queries).

Probabilistically Checkable Proofs of Proximity (PCPPs), also known as assignment testers, are proof systems that allow probabilistic verification of claims by probing few bits of _the input_ and a purported proof. Needless to say, the verifier of such a system cannot generally be expected to distinguish inputs in the language from inputs that are not in the language, but rather it should accept every $x \in L$ with probability 1, and reject (with high probability) every input that is “far” from all $x' \in L$. First introduced in [BGH+04, DR04, Din06] as building blocks for the construction of more efficient PCPs, there are currently known PCPP systems for NP with parameters comparable to those of the best known PCP systems [BS08, Mie09].
A very different kind of proofs are zero-knowledge (ZK) proofs [GMR85], namely proofs that carry no extra knowledge other than being convincing. Combining the advantages of ZK proofs and PCPs, a zero-knowledge PCP (ZKPCP) is defined similarly to a traditional PCP, except that the proof is also randomized and there is the additional guarantee that the view of any (possibly malicious) verifier who makes a bounded number of queries can be efficiently simulated up to a small statistical distance. ZKPCPs were first constructed by Kilian et al. [KPT97], building on a previous weaker “honest-verifier” notion of ZKPCPs implicit in [DFK92]. More concretely, previous ZKPCP constructions [KPT97, IMS12] are constructed in two steps: first, the weaker variant of ZKPCPs of [DFK92] is amplified to yield PCPs that are zero-knowledge against the honest verifier (which is much easier to achieve than full-fledged zero-knowledge). Then, these honest-verifier zero-knowledge PCPs are combined with an unconditionally secure oracle-based commitment primitive called a “locking scheme” to obtain ZKPCPs for NP that guarantee statistical zero-knowledge against query-bounded malicious verifiers, namely ones who are only limited to asking at most \( p(|x|) \) queries, for some fixed polynomial \( p \) that is much smaller than the proof length, but can be much bigger than the (polylogarithmic) number of queries required to verify the proof. A simpler construction of locking schemes was recently given in [IMS12].

These constructions have the common limitation that they require adaptive verification, even if the underlying non-ZK PCP can be non-adaptively verified. Moreover, adaptive verification is inherent to any locking-scheme-based ZKPCP due to the unconditional security of locking schemes. This state of affairs gives rise to the following natural question, which has been open since the first ZKPCP construction of [KPT97] almost 20 years ago:

**Question 1.** Do there exist PCPs for NP that can be verified non-adaptively, and guarantee ZK against malicious verifiers?

We note that constructing non-adaptively verifiable ZKPCPs requires a new approach towards ZKPCP construction which eliminates the use of locking schemes. Since locking schemes inherently incur a polynomial blow-up in the PCP length, an additional advantage of such constructions is the possibility of obtaining ZKPCPs that preserve the proof length (which is important when these are used for cryptographic applications as described in Section 1.1.4).

Another interesting question is that of extending the previous notion of ZKPCP to the realm of proofs of proximity:

**Question 2.** Do there exist zero-knowledge PCPPs (ZKPCPPs) for NP with comparable parameters to known ZKPCPs?

ZKPCPPs are a natural, and more useful, generalization of ZKPCPs. Indeed, they offer the additional guarantee of hiding not only the NP witness, but also most of the
Thus, ZKPCPPs can be employed in various cryptographic settings in which input privacy is required (which is not guaranteed by ZKPCPs).

**Locally Testable Codes**

Locally Testable Codes (LTCs) are codes associated with a randomized tester algorithm, that has oracle access to a purported codeword \( w \), and can test membership in the code by probing only few symbols of \( w \). The tester accepts codewords with probability 1 (the *completeness* property), and rejects words that are far from the code with high probability. LTCs, which were implicit already in [BFLS91] (cf. [Gol05, Sec. 2.4]), and first explicitly studied by Goldreich and Sudan [GS06], are of interest in computer science due to their connections to PCPs and property testing.

LTCs are further motivated by the possible applications of their efficient verification feature in the contexts of distributed storage, and cryptographic protocols. However, LTCs lack a privacy guarantee, namely that the secrecy of the encoded message is preserved during (and after) the testing procedure. This naturally gives rise to the notion of *zero-knowledge* LTCs (ZKLTCs), which are LTCs equipped with a randomized encoding function that hides the encoded data against any (possibly malicious) tester who observes a bounded number of codeword symbols. However, (even the zero-knowledge variant of) the standard notion of testability is generally insufficient for cryptographic applications, in which the purported codeword is distributed between multiple untrusted servers. Indeed, malicious servers may adaptively determine their answers to the queries of the tester *after seeing* the queries. This calls for a stronger notion of testability that restricts the influence of such attacks on the outcome of the testing procedure, which we call *stability*. This raises the following question:

*Question 3.* Do there exist stable ZKLTCs with comparable parameters to the best known LTCs?

It is instructive to note that the testability property of ZKLTCs is weaker than the verifiability property of ZKPCPs and ZKPCPPs, since ZKLTCs can only be used to test membership in a given code, whereas ZKPCPs and ZKPCPPs can be used to verify general NP statements. However, current constructions of LTCs achieve better tradeoffs between the different parameters of the constructions, compared to the best-known constructions of PCPs and PCPPs. Therefore, employing ZKLTCs in cryptographic contexts that only require testing membership in a code (such as distributed storage, secret sharing, etc.) can yield significant computational improvements over similar constructions based on ZKPCPs or ZKPCPPs.

**Leakage-Resilient Hardware**

A Leakage-Resilient Circuit Compiler (LRCC) compiles any circuit into a new circuit that operates on encoded inputs, and withstands side-channel attacks in the sense that
these reveal nothing about the (properly encoded) input, other than what follows from
the output. (This notion is formalized in the simulation-based paradigm, where the
output of leakage functions can be simulated given only the structure of the circuit,
and its output.) Following [ISW03, FRR⁺14, MV13], we restrict the complexity class
from which leakage functions are chosen. More specifically, our focus is on LRCCs
for stateless circuits, achieving information-theoretic security against low complexity
leakage.

A significant limitation of previous constructions that offer information-theoretic
security against low complexity leakage [FRR⁺14, MV13] is that the correctness of the
computation relies on the assumption that the inputs to the computation were honestly
encoded. More specifically, these constructions require that the parties encode their
inputs in a “special form”, which (in addition to actually encoding the inputs) also
includes “encoded randomness” used to mask the internal computations in the compiled
circuit. The “encoded randomness” (which is used to achieve leakage resilience) is
independent of the inputs, and could be generated using a simple trusted hardware
device. However, it is not simply a sequence of uniformly-random bits, but rather has
specific structure, and the compiled circuit emulates the original circuit only if the
randomness is “well-formed” (namely, has the required structure). Consequently, by
providing ill-formed randomness as part of the input encoding, malicious parties can
manipulate the outputs. This raises the following natural question.

Question 4. Does there exist a single leaky (but otherwise trusted) stateless hardware
device that enables $m \geq 1$ parties to compute a joint function of their inputs (where
each party locally encodes its input to the device), such that input secrecy, and output
correctness, are preserved in the presence of an all-powerful active adversary that
corrupts a subset of the parties, and obtains computationally-bounded leakage on the
internals of the device?

1.1 Our Results

We give a brief overview of our main results. More details are given in subsequent
chapters.

1.1.1 Non-Adaptively Verifiable ZKPCPs

Together with Ishai and Yang [IWY16] we answer Question 1 in the affirmative, in two
weaker zero-knowledge models which relax either: the simulation resources, allowing
the simulator to be unbounded, a notion which we call witness indistinguishability
(WI); or restrict the computational resources of the verifier, restricting also malicious

One exception is the LRCC of [ISW03], however their construction is only known to resist probing
attacks, whereas we will be interested in more general complexity classes such as low-depth boolean
circuits.
verifiers to run in polynomial time, and allowing the prover and verifier to share a common uniformly random string, which we call computation zero-knowledge (CZK) in the common random string (CRS) model.

As noted above, achieving non-adaptive verification requires eliminating the use of locking schemes, and more generally, developing a new approach towards the design of ZKPCPs. We do so by showing a novel connection between LRCCs and ZKPCPs, combining general non-ZK PCPs for NP with LRCCs to construct WIPCPs for NP. More specifically, for a zero-knowledge parameter $q^*$ our WIPCPs have length \( \text{poly}(q^*, |x|) \), are WI against verifiers making at most $q^*$ queries, and can be verified with only \( \text{polylog}(q^*, |x|) \) queries. We then apply the so-called “FLS technique” [FLS90] to convert these WIPCPs into CZKPCPs in the CRS model with similar parameters, based on the existence of one-way functions.

### 1.1.2 ZKPCPPs for NP

Together with Ishai [IW14] we introduce the notion of ZKPCPPs, and answer Question 2 by constructing ZKPCPP systems for NP. In our ZKPCPP system, the prover is given an input $x \in L$, a corresponding witness $w$, and a zero-knowledge parameter $q^*$, and efficiently generates a proof $\pi$ of length \( \text{poly}(|x|, q^*) \), which can be used to verify that $x$ is at most $\delta$-far from $L$ by making only \( \text{polylog}(|x|, q^*) \) queries to the oracles $x, \pi$. \( \delta \) can be any positive constant (or even inverse polylogarithmic), and the system has negligible soundness error. Moreover, the system has statistical zero-knowledge against (possibly malicious) verifiers that are restricting to making $q' \leq q^*$ queries to $(x, \pi)$.

### 1.1.3 Stable ZKLTCs

Together with Ishai, Sahai, and Viderman [ISVW13], we introduce the notion of stable LTCs, and initiate the study of stable LTCs and ZKLTCs, thus answering Question 3. We have three types of constructions.

First, we construct asymptotically good stable ZKLTCs, based on the testability of tensor products of codes. Our codes have constant rate, constant relative distance, sublinear query complexity, can efficiently correct a constant number of errors, and are zero-knowledge against testers probing a constant fraction of codeword symbols. Moreover, they are stable with a constant corruption threshold, namely the correctness and soundness of the testing procedure is guaranteed even if the values of a constant fraction of (pre-determined) codeword symbols can be adaptively determined based on the queried locations. (See Chapter 6 for a formal exposition of stable codes and their applications.) This construction is probabilistic, namely when applied to a non-ZK LTC the transformation works except with negligible probability error.

Second, we present a fully explicit transformation from linear codes to ZK codes. The advantage of this construction over the first construction is that the transformation is guaranteed to yield a ZK code (with probability 1). The drawback is that our second
construction does not guarantee the additional stability property, and the zero-knowledge property becomes statistical (rather than perfect, as in the first construction).

Finally, we show a transformation from non-ZK codes to ZK-codes that can be applied to any (not necessarily linear) code. This is obtained by observing that a completely random choice of an encoding function for a general code would give a ZK code with high probability. This construction is more general than our first two constructions, since it can be applied to any code (whereas the first two only apply to linear codes), but is less explicit than them, since the encoding function is inefficient.

1.1.4 Cryptographic Applications of ZK Proofs and Codes

We use our ZK probabilistic proofs and codes to improve the complexity of various cryptographic tasks. Our constructions improve either the communication complexity of known protocols, their round complexity, the assumptions on which they rely, or the properties which they guarantee. We focus on tasks of distributing secrets and verifying computations in a distributed setting, motivated by the possibility of employing such primitives in more general contexts that involve sublinear-communication zero-knowledge arguments on distributed or committed data. We believe that such codes and proofs can be used to improve numerous additional cryptographic tasks, since verification and testing lie at the heart of most cryptographic protocols.

Verifiable Secret Sharing

Together with Ishai, Sahai, and Viderman [ISVW13], we use ZKLTCs to reduce the communication complexity of protocols for Verifiable Secret Sharing (VSS), that allow a dealer $D$ to distribute a secret $x$ of size $n$ among $m$ servers such that (except with small error probability) a coalition of up to $\tau$ servers cannot learn or modify the secret, while on the other hand guaranteeing unique reconstruction, even if $D$ and up to $\tau$ servers collude. We consider a designated receiver variant of VSS which involves, in addition to $D$ and the $m$ servers, a designated receiver $R$ who may assist in the verification. We construct VSS protocols with total linear (in $n$) communication, that withstand a constant fraction $\tau$ of corrupted servers, and have statistical error that vanishes almost exponentially with $n$. Our protocols have the additional feature that the entire communication involving the receiver is sublinear. These VSS protocols can be used to flip a fair coin in a distributed setting in which a pair of clients are aided by $m$ servers, with the following guarantees. First, $\tau$ corrupted servers cannot cause the honest clients to output different, or biased, coins. Second, even when one of the clients colludes with $\tau$ corrupted servers, they can hardly bias the output of the honest client. These protocols achieve the same efficiency features as the underlying VSS protocol.

Together with Ishai [IW14], we use a similar VSS construction based on ZKPCPPs to achieve a stronger variant of designated-receiver VSS, which we call certifiable VSS. Certifiable VSS generalized traditional VSS by providing the additional guarantee
that the shared secret satisfies some NP predicate. (See Chapter 7 for the exact definition of certifiable VSS.) We construct certifiable VSS protocols for NP with total polynomial communication, where the entire communication involving the receiver is only polylogarithmic (in \( n \)). The advantage of these protocols, compared to the ZKLTC-based VSS protocols, is that they offer the additional certifiable property. However, the ZKLTC-based VSS protocols are more efficient in terms of total communication (linear, compared to polynomial), and can withstand a higher ratio of corrupted servers (the ZKPCPP-based construction withstands \( \sqrt{m} \) corrupted servers, for some constant \( \gamma > 0 \)).

3-Round Distributed Proofs with Zero-Knowledge Properties

Together with Ishai and Yang [IWY16], we use our non-adaptively verifiable ZKPCPs to construct 3-round proofs for NP with zero-knowledge guarantees, in a distributed setting in which the prover and verifier are aided by \( m \) servers. More specifically, for every input length \( n \), and corruption threshold \( \tau \), our distributed proof systems employ \( m = \text{poly}(n, \tau) \) servers; are correct even when \( \tau \) servers are corrupted; sound against a corrupted prover that colludes with \( \tau \) servers; and guarantee witness-indistinguishability (alternatively, CZK) against a corrupted verifier that colludes with \( \tau \) servers. Our systems improve over previous 2-party sublinear ZK proofs [Kil92, IMS12] in terms of the round complexity (reducing the number of rounds from 4 to 3), underlying assumptions (achieving either unconditional security, or relying on the existence of one-way functions, whereas previous constructions assume the existence of collision-resistant hash functions), and soundness type (our witness-indistinguishable distributed proofs are secure against computationally unbounded verifiers, whereas previous constructions assume that the verifier is computationally-bounded). These improvements over previous constructions in the 2-party setting are our main motivation for studying zero-knowledge proofs in a distributed setting.

2-Party Commit-and-Prove and Its Generalizations

Together with Ishai [IW14], we use ZKPCPPs to construct Commit-and-Prove protocols, a “certifiable” generalization of a commitment scheme. Informally, a commitment scheme is a two-phase protocol between a sender \( S \) and a receiver \( R \), where the interaction during the first phase “commits” \( S \) to a single secret \( x \) (namely, \( S \) cannot decommit to a different value), while completely hiding \( x \) until it is decommitted in the second phase. A Commit-and-Prove protocol is associated with an NP-relation \( R \), and is certifiable in the following sense. First, the commit phase binds \( S \) to a single \( x \in L_R \), namely it cannot successfully decommit to some \( x' \neq x \), or to \( x \notin L_R \). Moreover, the commit phase completely hides \( x \) and the corresponding NP-witness \( w \), and the entire interaction completely hides \( w \). Assuming the existence of exponentially-hard collision-resistant hash functions (CRHF), we design Commit-and-Prove protocols for
NP with \( \text{polylog}(|x|) \) communication during the commit phase, which only use the CRHF as a black-box. This construction is based on the existence of honest-verifier ZKPCPPs (whereas our constructions in the distributed setting require the existence of PCPs and PCPPs with zero-knowledge against malicious verifiers). We note that by allowing sublinear communication during the commit phase, the protocols can use super-polynomially hard CRHF.

We also generalize Commit-and-Prove to the reactive (and the multiparty) settings, and apply such schemes to design updateable databases that allow multiple clients to securely perform read and write operations a sensitive database.

1.1.5 Leakage-Resilient Computation

Together with Genkin and Ishai [GIW16], and Ishai and Yang [IWY16], we generalize the notion of LRCCs, and apply them towards answering Question 4. Our results are two-fold.

First, we construct LRCCs with soundness properties which we call SAT-respecting LRCCs. Roughly speaking, a SAT-respecting LRCC is an LRCC with the additional guarantee that if the original circuit is not satisfiable, then there exists no satisfying input for the leakage-resilient circuit. We construct SAT-respecting LRCCs that resist computationally-bounded leakage, and use these to construct SAT-respecting LRCCs that withstand leakage computable by constant-depth, polynomial-sized boolean circuits, even if these are augmented with a sublinear number of \( \oplus \) gates. More generally, the construction is based on an encoding scheme that resists leakage from some class \( \mathcal{L} \) of functions, in the sense that functions in \( \mathcal{L} \) have only negligible statistical advantage in distinguishing between random encodings of two different values. Given an encoding scheme that resists leakage from class \( \mathcal{L} \), our SAT-respecting LRCC resists leakage from any class \( \mathcal{L}' \) such that composing \( \ell \in \mathcal{L}' \) on top of a constant-depth, polynomial-sized, circuit gives a function in \( \mathcal{L} \). We note that our LRCCs acieve a relaxed notion of leakage-resilience with inefficient simulation (whereas previous LRCC constructions [ISW03, FRR\(^+\)14, MV13] have efficient simulators).

Second, we answer Question 4 in the affirmative by designing multiparty LRCCs, whose output is a leaky (but otherwise trusted) hardware device that allows \( m \geq 1 \) parties to jointly evaluate a boolean circuit \( C \) on their inputs. More specifically, each party locally encodes its input and feeds the encoded input into the device, which performs the computation on the encoded inputs, and returns a public output. The computation preserves input secrecy and output correctness in the presence of an unbounded, active adversary that corrupts a subset of the parties, and obtains computationally-bounded leakage on the internals of the device. Similar to SAT-respecting LRCCs, the construction is based on a “leakage-resilient” encoding scheme, where if the encoding resists leakage from \( \mathcal{L} \), then the multiparty LRCC resists leakage from any class \( \mathcal{L}' \) such that composing \( \ell \in \mathcal{L}' \) on top of an \( O(\log m) \)-depth, polynomial-sized, circuit gives a function in \( \mathcal{L} \). As
a corollary, we obtain multiparty LRCCs for a constant number of parties, that resist leakage computable by constant-depth, polynomial-sized boolean circuits.

We note that though multiparty LRCCs generalize the notion of SAT-respecting LRCCs, for some applications (such as the non-adaptively verifiable ZKPCPs of Chapter 4) SAT-respecting LRCCs are a more natural (and useful) building block. Indeed, some computational tasks for $m = 1$ parties (e.g., zero-knowledge proofs) require a randomized hardware device, which in the context of ZKPCPs would require an external randomness source, whereas SAT-respecting LRCCs require no additional randomness. See Chapter 3 for a comparison of these objects, and their possible applications.

1.2 Organization

Chapter 2 contains necessary preliminaries that are used throughout this thesis. In Chapter 3 we describe our generalizations of LRCCs. Chapter 4 contains the construction of non-adaptively verifiable ZKPCPs, and a discussion of the issues arising when one tries to extend this LRCC-based transformation to yield ZKPCPs with the stronger zero-knowledge guarantee of standard ZKPCP constructions. Chapter 5 discusses ZKCPPs, and in Chapter 6 we present our constructions of ZK-codes. Finally, in Chapter 7 we present various cryptographic applications of these objects. Each chapter is mostly self contained, with the exception of Chapter 4, which uses the SAT-respecting LRCCs of Chapter 3; and Chapter 7, which uses the probabilistic proof systems of Chapters 4 and 5, and the LTCs of Chapter 6.
Chapter 2

Preliminaries

We start with a few notations and definitions that will be used throughout this thesis. For a natural \( n \), we use \( \text{negl}(n) \) to denote an unspecified negligible function \( \epsilon : \mathbb{N} \to \mathbb{R} \), namely a function such that for any polynomial \( p(n) \), there exists an \( n_0 \in \mathbb{N} \) such that for every \( n \geq n_0 \), \( \epsilon(n) < \frac{1}{p(n)} \). For \( c > 0 \), we use \( O(c \cdot n) \) to denote a function of the form \( \Theta(n) \cdot f(c) \) for an unspecified function \( f : \mathbb{R} \to \mathbb{N} \) (in particular, when \( c \) is constant then \( O(c \cdot n) = O(n) \)). Similarly, for \( c, c' > 0 \) we use \( \text{poly}(c, c') \) to denote the function \( \text{poly}(n) \cdot f(c, c') \), where \( f(c, c') \) is an arbitrary function. The term “PPT” denotes probabilistic polynomial time algorithms.

We use \( \mathbb{F} \) to denote a finite field, and \( \Sigma \) to denote a finite alphabet (i.e., a set of symbols). We use \( [n] \) to denote the set \( \{1, \ldots, n\} \). For \( x \in \mathbb{F}^n \), let \( \text{supp}(x) = \{i \in [\hat{n}] : x_i \neq 0\} \). For \( S \subseteq [\hat{n}] \), \( x|_S \) denotes the restriction of \( x \) to the subset \( S \). Similarly, for \( s \subseteq \Sigma^n \), \( X|_S = \{x|_S : x \in X\} \). If \( x, y \in \Sigma^n \) then the (Hamming) distance between \( x, y \) is \( \Delta(x, y) = |\{i : x_i \neq y_i\}| \), and the relative distance is \( \delta(x, y) = \frac{\Delta(x, y)}{n} \). For a subset \( X \subseteq \Sigma^n \), and \( y \in \Sigma^n \), \( \Delta(y, X) = \min \{\Delta(x, y) : x \in X\} \), and \( \delta(y, X) = \min \{\delta(x, y) : x \in X\} \). For \( \epsilon \in [0, 1] \), we say that \( y \in \Sigma^n \) is \( \epsilon \)-far from \( X \subseteq \Sigma^n \) if \( \delta(y, X) > \epsilon \), otherwise \( y \) is \( \delta \)-close to \( X \).

Function composition is denoted as \( f \circ g \), where \( (f \circ g)(x) := f(g(x)) \). If \( F, G \) are families of functions then \( F \circ G := \{f \circ g : f \in F, g \in G\} \). By default, we assume that standard cryptographic primitives (e.g., one-way functions) are secure against non-uniform adversaries.

Following are a list of symbols (Figure 2.1), and a list of abbreviations (Figure 2.2), that are used throughout the thesis.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>input length</td>
</tr>
<tr>
<td>$k$</td>
<td>output length, number of repetitions in amplification methods</td>
</tr>
<tr>
<td>$r$</td>
<td>randomness length</td>
</tr>
<tr>
<td>$\hat{n}$</td>
<td>codeword length</td>
</tr>
<tr>
<td>$N$</td>
<td>length of NP-witness, or number of secret shares</td>
</tr>
<tr>
<td>$\ell$</td>
<td>proof length</td>
</tr>
<tr>
<td>$r$</td>
<td>randomness length</td>
</tr>
<tr>
<td>$q^*$</td>
<td>zero-knowledge parameter</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>security parameter</td>
</tr>
<tr>
<td>$t$</td>
<td>security parameter in building blocks</td>
</tr>
<tr>
<td>$l$</td>
<td>number of copies in amplification methods</td>
</tr>
<tr>
<td>$T(\cdot)$</td>
<td>time function (e.g., DTIME($T(n)$))</td>
</tr>
<tr>
<td>$s$</td>
<td>circuit size</td>
</tr>
<tr>
<td>$d$</td>
<td>circuit depth</td>
</tr>
<tr>
<td>$m$</td>
<td>number of parties in MPC protocols</td>
</tr>
<tr>
<td>$\tau$</td>
<td>corruption threshold in MPC protocols</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>the empty string</td>
</tr>
<tr>
<td>$\perp$</td>
<td>abort (error) message</td>
</tr>
<tr>
<td>$\delta$</td>
<td>proximity parameter</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>statistical distance parameter</td>
</tr>
<tr>
<td>$\gamma, \beta$</td>
<td>fractional parameter</td>
</tr>
</tbody>
</table>

Figure 2.1: List of Symbols
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPT</td>
<td>Probabilistic Polynomial Time</td>
</tr>
<tr>
<td>ZK</td>
<td>Zero-Knowledge</td>
</tr>
<tr>
<td>CZK</td>
<td>Computational Zero-Knowledge</td>
</tr>
<tr>
<td>HVZK</td>
<td>Honest-Verifier Zero-Knowledge</td>
</tr>
<tr>
<td>WI</td>
<td>Witness-Indistinguishability</td>
</tr>
<tr>
<td>PCP</td>
<td>Probabilistically Checkable Proof</td>
</tr>
<tr>
<td>PCPP</td>
<td>Probabilistically Checkable Proof of Proximity</td>
</tr>
<tr>
<td>ZKPCP</td>
<td>Zero-Knowledge PCP</td>
</tr>
<tr>
<td>CZKPCP</td>
<td>Computational Zero-Knowledge PCP</td>
</tr>
<tr>
<td>HVZKPCP</td>
<td>Honest-Verifier Zero-Knowledge PCP</td>
</tr>
<tr>
<td>WIPCP</td>
<td>Witness-Indistinguishable PCP</td>
</tr>
<tr>
<td>NA-CZKPCP</td>
<td>Non-Adaptive Computational Zero-Knowledge PCP</td>
</tr>
<tr>
<td>NA-WIPCP</td>
<td>Non-Adaptive Witness-Indistinguishable PCP</td>
</tr>
<tr>
<td>ZKCPP</td>
<td>Zero-Knowledge PCPP</td>
</tr>
<tr>
<td>HVZKCPP</td>
<td>Honest-Verifier Zero-Knowledge PCPP</td>
</tr>
<tr>
<td>LTC</td>
<td>Locally Testable Code</td>
</tr>
<tr>
<td>ZKLTC</td>
<td>Zero-Knowledge LTC</td>
</tr>
<tr>
<td>LRCC</td>
<td>Leakage-Resilient Circuit-Compiler</td>
</tr>
<tr>
<td>VSS</td>
<td>Verifiable Secret Sharing</td>
</tr>
<tr>
<td>OWF</td>
<td>One-Way Function</td>
</tr>
<tr>
<td>CRHF</td>
<td>Collision-Resistant Hash Function</td>
</tr>
<tr>
<td>BB</td>
<td>Black-Box</td>
</tr>
<tr>
<td>MHT</td>
<td>Merkle Hash Tree</td>
</tr>
<tr>
<td>MPC</td>
<td>Multi-Party Computation</td>
</tr>
</tbody>
</table>

Figure 2.2: List of Abbreviations

Relations

An NP relation $\mathcal{R}$ is a polynomial-time recognizable binary relation which is *polynomially bounded* in the sense that there is a polynomial $p$ such that if $(x, w) \in \mathcal{R}$ then $|w| \leq p(|x|)$. $x$ is called the *input*, and $w$ is called the *witness*. We measure the complexity of a relation $\mathcal{R}$ in terms of the time required to validate a claim “$(x, w) \in \mathcal{R}$”, where we say that $\mathcal{R}$ has complexity $t(n)$ if for every input-witness pair $(x, w) \in \{0, 1\}^* \times \{0, 1\}^*$, $\mathcal{R} \in \text{DTIME}(t(|x|))$.

We denote the NP language corresponding to $\mathcal{R}$ by $L_\mathcal{R} = \{x : \exists w, (x, w) \in \mathcal{R}\}$.

*Notation 2.1.* When an algorithm takes as input a non-integral value, e.g., $\epsilon \in [0, 1)$, we assume that $\frac{1}{\epsilon} \in \mathbb{N}$, and use $1^{\lfloor \epsilon \rfloor}$ to denote $\frac{1}{\epsilon}$, and $|\epsilon|$ to denote the binary representation of $\frac{1}{\epsilon}$. 

15
2.1 Some Facts about Probabilities

Let $D$ be a distribution. We use $X \leftarrow D$, or $X \in_R D$, to denote sampling $X$ according to the distribution $D$. We say that two distribution ensembles $X_n, Y_n$ are computationally (resp. statistically) indistinguishable if every computationally bounded (respectively, computationally unbounded) distinguisher achieves only a negligible advantage in distinguishing a sample drawn according to $X_n$ from a sample drawn according to $Y_n$.

Given two distributions $X, Y$, $\text{SD}(X, Y)$ denotes the statistical distance between $X$ and $Y$, $X \equiv Y$ denotes that $X, Y$ are identically distributed, and $X \approx Y$ denotes that $X, Y$ are computationally indistinguishable.

The entropy function $H(\cdot)$ is defined over $[0, 1]$ as follows:

$$H(p) = p \cdot \log \left( \frac{1}{p} \right) + (1 - p) \cdot \log \left( \frac{1}{1 - p} \right).$$

The following lemma generalizes Vazirani’s XOR Lemma [Vaz, Gol95] and can be proved similarly.

**Lemma 2.1.1 (XOR lemma).** Let $X$ and $Y$ be distributions over $\mathbb{F}_2^k$ such that $\text{SD}(X, Y) = \epsilon$. Then there exists an $\alpha \in \mathbb{F}_2^k$ such that

$$\text{SD}(\alpha^T X, \alpha^T Y) \geq \epsilon/2^{k/2}.$$

2.2 Probabilistic Proof Systems

We consider several probabilistic proof systems for NP relations and NP languages. Notice that in the context of proof systems, the advantage of considering NP relations (rather than NP languages) is that the prover can be efficient, since it is given as input the witness $w$ (and the efficiency of the prover is usually required for cryptographic applications).

A probabilistic proof system $(P, V)$ for $\mathcal{R}$ consists of a PPT prover $P$, that on input $(x, w)$ outputs a proof $\pi$ (in standard probabilistically checkable proofs the prover is deterministic, but our constructions will crucially rely on the prover being probabilistic), and a PPT verifier $V$ that given input $1^{||x||}$ and oracle access to $x$ (the input oracle) and $\pi$ (the proof oracle) outputs either accept or reject. Intuitively, $P$ tries to convince $V$ of the claim “$x \in L_\mathcal{R}$” using $w$ such that $(x, w) \in \mathcal{R}$. All the probabilistic proof systems studied in this work will have perfect completeness (i.e. $V$ accepts true claims with probability 1). The system has soundness error $\epsilon$ if every input $x \notin L_\mathcal{R}$ is accepted by $V$ with probability at most $\epsilon$, regardless of the proof oracle. Probabilistic proof systems may be parameterized by additional parameters that are given as input to both $P$ and $V$.

**Definition 2.2.1 (q-bounded verifier).** We say that a verifier $V$ in a probabilistic proof system $(P, V)$ is $q$-bounded if $\Pr[V \text{ makes more more than } q \text{ oracle queries}] = 0$. 

16
Definition 2.2.2 (Non-adaptive verifier). We say that a verifier $V$ in a probabilistic proof system is non-adaptive if its oracle queries are determined solely by its randomness (a verifier is adaptive if each of its queries may also depend on the oracle answers to previous queries).

Relations between classes of relations and languages. We will consider several classes consisting of NP relations (or NP languages) that admit probabilistic proof systems that with several additional properties; and will discuss the relations between these classes. All our proofs are constructive, namely for a statement of the form $\text{Class}_1 \subseteq \text{Class}_2$, we will show an efficient transformation from a pair $(P, V) \in \text{Class}_1$ to a pair $(P', V') \in \text{Class}_2$.

2.2.1 Probabilistically Checkable Proofs

A probabilistically checkable proof (PCP) is a probabilistic proof system $(P, V)$ in which $V$ can freely query its input oracle $x$ (that is, his queries to $x$ are not counted towards the query complexity). Formally,

Definition 2.2.3 (PCP). Let $\epsilon \in [0, 1]$ be a soundness parameter, and $\sigma$ be a security parameter. A probabilistic proof system $(P, V)$ is a probabilistically checkable proof (PCP) system for a relation $R = R(x, w)$, if the following conditions hold for every pair $(x, w)$:

- **Parameters**: The system has alphabet $\Sigma$, soundness error $\epsilon$, query complexity $q$, randomness complexity $r$, and proof length $\ell$, where all parameters may depend on $\epsilon$ and $|x|$.

- **Syntax**: Both parties are given $1^\sigma, 1^{|x|}$ as input. $P(1^\sigma, 1^{|x|}, x, w)$ outputs a distribution over strings of length at most $\ell$ over alphabet $\Sigma$, where $\pi \in P(1^\sigma, 1^{|x|}, x, w)$ is a proof for the claim “$(x, w) \in R$”. $V^{x, \pi}(1^\sigma, 1^{|x|}, |x|)$ tosses $r$ coins, makes queries to its oracle $x$, and at most $q$ queries to its oracle $\pi$, and based on the oracle answers outputs either $\text{acc}$ or $\text{rej}$.

- **Completeness**: If $(x, w) \in R$ then for every proof $\pi \in P(1^\sigma, 1^{|x|}, x, w)$,

  \[ \Pr[V^{x, \pi}(1^\sigma, 1^{|x|}, |x|) = \text{acc}] = 1 \]

  where the probability is over the randomness of $V$.

- **Soundness**: If $x \notin L_R$ then for every $\pi^*$,

  \[ \Pr[V^{x, \pi^*}(1^\sigma, 1^{|x|}, |x|) = \text{acc}] \leq \epsilon. \]

It will sometimes be useful to consider systems parameterized by a single security parameter $\sigma$. In such cases, $1^\sigma$ is given as input to both parties, and all other system
parameters (including $\epsilon$) may depend on $\sigma$. (In particular, $\epsilon$ is not given as input to the parties.)

Notation 2.2. We use $\text{PCP}_\Sigma [r, q, \epsilon, \ell]$ to denote the class of all NP relations that admit a PCP system with alphabet $\Sigma$, randomness complexity $O(r)$, query complexity $O(q)$, soundness error $\epsilon$ and proof length $O(\ell)$. When $\Sigma = \{0, 1\}$ then we will sometime omit the subscript $\Sigma$ from this notation.

2.2.2 Probabilistically Checkable Proofs of Proximity

A Probabilistically Checkable Proof of Proximity (PCPP) system is similar to a PCP system, except that the verifier tries to validate the claim “$x \in L_R$” while reading only few bits of $x$. (This is formalized by counting the number of queries to the input oracle towards the query complexity of the system.) Of course, since $V$ does not read its entire input, it cannot generally be expected to distinguish the case that $x \in L_R$ from the case that $x \notin L_R$, but is very “close” to $L_R$. Instead, $V$ is only expected to reject when $x$ is “far” from $L_R$.

Definition 2.2.4 (PCPP). Let $\sigma$ be a security parameter, $\epsilon \in [0, 1]$ be a soundness parameter, and $\delta \in [0, 1]$ be a proximity parameter. A probabilistic proof system $(P, V)$ is a probabilistically checkable proof of proximity (PCPP) system for a relation $R = R(x, w)$, if the following conditions hold for every pair $(x, w)$:

- **Parameters:** The system has alphabet $\Sigma$, proximity parameter $\delta$, soundness error $\epsilon$, query complexity $q$, randomness $r$ and proof length $\ell$, where all parameters may depend on $\epsilon, \delta$ and $|x|$.

- **Syntax:** Both parties are given as input $1^\sigma, 1^{|x|}, 1^{|\delta|}$. $P \left(1^\sigma, 1^{|x|}, 1^{|\delta|}, x, w\right)$ outputs a distribution over strings of length at most $\ell$ over alphabet $\Sigma$, where $\pi \in P \left(1^\sigma, 1^{|x|}, 1^{|\delta|}, x, w\right)$ is a proof for the claim “$(x, w) \in R$”. $V^{x, \pi} \left(1^\sigma, 1^{|x|}, 1^{|\delta|}, |x|\right)$ tosses $r$ coins, makes a total of $q$ queries to both oracles $x, \pi$, and based on their answers outputs either acc or rej.

- **Completeness:** If $(x, w) \in R$ then for every proof $\pi \in P \left(1^{|x|}, 1^{|\delta|}, x, w\right)$,

$$\Pr \left[V^{x, \pi} \left(1^{|x|}, 1^{|\delta|}, |x|\right) = \text{acc} \right] = 1$$

where the probability is over the randomness of $V$.

- **Soundness:** If $x$ is $\delta$-far from $L_R$ then for every $\pi^*$, $\Pr \left[V^{x, \pi^*} \left(1^{|x|}, 1^{|\delta|}, |x|\right) = \text{acc} \right] \leq \epsilon$.

Notation 2.3. We use $\text{PCPP}_\Sigma [r, q, \delta, \epsilon, \ell]$ to denote the class of all NP relations that admit a PCPP system with alphabet $\Sigma$, randomness complexity $O(r)$, query complexity $O(q)$, soundness error $\epsilon$, proximity parameter $\delta$, and proof length $O(\ell)$. When $\Sigma = \{0, 1\}$ then we will sometime omit the subscript $\Sigma$ from this notation.
The soundness property of PCPP systems can be strengthened by requiring that every \( x \notin L_R \) is rejected with probability that is proportional to its distance from \( L_R \). Such PCPPs are called strong PCPPs (see [BS08, Mei08]).

**Definition 2.2.5 (Strong PCPP).** A PCPP system \((P, V)\) is strong if there exists a function \( \epsilon_S(\delta, |x|) : [0, 1] \times \mathbb{N} \rightarrow [0, 1] \) such that for every \( \delta \in [0, 1] \), every \( x \) that is \( \delta \)-far from \( L_R \) is accepted by \( V \) with probability at most \( \epsilon_S(\delta, |x|) \). A strong PCPP system has rejection ratio \( \beta \) if every \( x \) that is \( \delta \)-far from \( L_R \) is rejected with probability at least \( \beta \delta \).

Notice that the soundness error, and the rejection ratio, are inverse proportional. In particular, we aim at increasing the rejection ratio (thus decreasing the soundness error).

### 2.2.3 ZKPCPs and ZKPCPPs

At a high level, a probabilistic proof system is zero-knowledge if a verifier that does not make "too many" queries to its oracle learns no (or little) information about the NP witness (and possibly also the input). Intuitively, a probabilistic proof system is \( q^* \)-zero-knowledge if whatever a (possibly malicious) verifier learns by making \( q' \leq q^* \) queries to \( x, \pi \) can be simulated by making \( q' \) queries to \( x \) alone. Therefore, zero-knowledge of a PCP system means that the witness is entirely hidden. (As the queries to the input oracle are not counted towards the query complexity, the simulator can query all of \( x \).) Zero-knowledge of a PCPP system means that not only is the witness hidden, but so is most of \( x \). Notice that the prover in a zero-knowledge proof system must be probabilistic, while the prover in standard proof systems for NP can be deterministic.

This is formalized in the real-ideal paradigm as described next.

Let \((P, V)\) be a probabilistic proof system, and let \( V^* \) be a (possibly malicious) \( q \)-bounded verifier. We compare the real-life interaction between \( P \) and \( V^* \) with an ideal-world interaction, in which a simulator Sim with oracle access to \( V^* \) interacts with a trusted third party (TTP) that knows only \( x \).

The real-life model consists of a probabilistic polynomial time prover \( P \); a (possibly malicious, possibly unbounded) verifier \( V^* \); and is parameterized by a security parameter \( \sigma \), and a zero-knowledge parameter \( q^* \). The real-life execution consists of two phases. In the first phase, \( P \) is given \( 1^\sigma, 1^{q^*} \) and a pair \((x, w) \in R\) as input, and generates a proof \( \pi \). In the second phase, \( V^* \) is given \( 1^\sigma, 1^{q^*}, |x| \) as input, and has oracle access to the input oracle \( x \), and the proof oracle \( \pi \) (generated during the first phase). The interaction of \( V^* \) with the oracles proceeds in rounds, where in every round: \( V^* \) tosses coins; based on the outcome, and the oracle answers provided in the previous rounds, \( V^* \) sends a set of queries to its oracles \( x, \pi \); and \( V^* \) receives the oracle answers. Let View_{V^* \rightarrow \sigma} denote the distribution ensemble describing the view of \( V^* \) when it has oracles \( x, \pi \) (the distribution is also over the randomness of \( P(1^\sigma, 1^{q^*}, x, w) \)), and let \( q_{V^*} \)-
denote the total number of queries that $V^*$ sent to the input and proof oracles. Let
$\text{Real}_{V^*,P}(\sigma, q^*, x, w) = (\text{View}_{V^*,P}, q_{V^*})$.

**The ideal process** consists of a simulator $\text{Sim}$ with oracle access to $V^*$, and a trusted third party (TTP) which knows the input $x$ (but does not know $w$!). $\text{Sim}$ is given $1^*, 1^r, |x|$ as input, and interacts with the TTP in rounds. In every round: $\text{Sim}$ tosses coins; based on the outcome, and the answers provided by the TTP in the previous rounds, $\text{Sim}$ sends a set of indices to the TTP; and the TTP responds with the corresponding symbols of $x$. Let $\text{Sim}(x)$ denote the distribution ensemble describing the output of $\text{Sim}$ (after making his queries to $x$), and let $q_S$ denote the number of queries that $\text{Sim}$ made. We define $\text{Ideal}_{\text{Sim}}(\sigma, q^*, x) = (\text{Sim}(x), q_S)$.

Next, we define the notion of zero-knowledge in the presence of query-bounded verifiers. (We will only consider zero-knowledge against query-bounded verifiers.)

**Definition 2.2.6** ($q^*$-zero-knowledge). Let $(P, V)$ be a probabilistic proof system for relation $\mathcal{R} = \mathcal{R}(x, w)$, let $q^* \in \mathbb{N}$ be a zero-knowledge parameter, and let $\epsilon \in [0, 1)$ be a distance parameter (both $q^*$ and $\epsilon$, may be functions of a security parameter $\sigma$, and the input size $|x|$). We say that $(P, V)$ is ($\epsilon, q^*$)-zero-knowledge (ZK) if for every real-life $q^*$-bounded verifier $V^*$ there exists an ideal-world simulator $\text{Sim}$ such that for every $(x, w) \in \mathcal{R}$, we have $SD(\text{Real}_{V^*,P}(\sigma, q^*, x, w), \text{Ideal}_{\text{Sim}}(\sigma, q^*, x)) \leq \epsilon$. We say that $(P, V)$ has perfect $q^*$-zero-knowledge, and write $\text{Real}_{V^*,P}(\sigma, q^*, x, w) \equiv \text{Ideal}_{\text{Sim}}(\sigma, q^*, x)$, if $(P, V)$ is ($0, q^*$)-zero-knowledge. For a query function $q^* : \mathbb{N} \rightarrow \mathbb{N}$, we say that $(P, V)$ has statistical $q^*(n)$-zero-knowledge if there exists a negligible function $\epsilon : \mathbb{N} \rightarrow \mathbb{R}^+$ such that for every input length parameter $n$, $(P, V)$ has $(\epsilon(n), q^*(n))$-zero-knowledge.

**Remark 2.4.** By default, we will make the stronger requirement that there exist a single, PPT black-box simulator $S$ such that for every $q^*$-bounded $V^*$, the simulator $\text{Sim} = S^{V^*}$ satisfies the above requirement. Moreover, $S$ can only interact with $V^*$ in a straight-line fashion (i.e., it cannot rewind $V^*$). In this case, we say that $(P, V)$ has *straight-line zero-knowledge*. (The latter straight-line simulation requirement will be useful for one of our cryptographic applications.)

**Remark 2.5.** The above notion of zero-knowledge requires that the number of input bits read by the simulator be the same as the total number of bits read by the verifier. One may consider stronger notions which require that the number of input bits read by the simulator coincide with the number of input bits read by the verifier, or even that the same input bits are read by the verifier and the simulator. The latter is captured by letting $\text{Real}$ and $\text{Ideal}$, instead of including the number of queries made by $V^*$ and $\text{Sim}$ (respectively), include the specific indices that $V^*$, $\text{Sim}$ queried in $x$. Our constructions do not satisfy these stronger notions.

**Notation 2.6.** We use $\text{ZKPCP}_{\Sigma}[r, q, \epsilon_{ZK}, \epsilon_S, \ell]$ to denote the class of NP relations that admit a PCP system with $(q^*, \epsilon_{ZK})$-ZK, alphabet $\Sigma$, randomness complexity $O(r)$,
query complexity $O(q)$, soundness error $\epsilon$, and proof length $O(\ell)$. Similarly, we use $\text{ZKPCPP}_{\Sigma}[r, q, \epsilon_K, \delta, \epsilon, \ell]$ to denote the class of NP relations that admit a PCPP system with $(q^*, \epsilon_K)$-ZK, alphabet $\Sigma$, randomness complexity $O(r)$, query complexity $O(q)$, soundness error $\epsilon$, proximity parameter $\delta$, and proof length $O(\ell)$. When $\Sigma = \{0, 1\}$, we omit the subscript $\Sigma$ from this notation.

One may also consider systems that only guarantee zero-knowledge against the honest verifier $V$. This (weaker) notion of honest-verifier ZK (HVZK) is often used as a simpler building block towards constructing probabilistic proof systems with ZK against malicious, query bounded verifiers, and is also of independent interest.

**Definition 2.2.7** (Honest-verifier zero-knowledge (HVZK)). We say that a probabilistic proof system $(P, V)$ for relation $R = R(x, w)$ has honest-verifier zero-knowledge (HVZK) with statistical distance $\epsilon \in [0, 1]$, if there exists a PPT simulator $\text{Sim}$ such that for every pair $(x, w) \in R$, $\text{SD} (\text{Real}_{V,P}(\sigma, q^*, x, w), \text{Ideal}_{\text{Sim}}(\sigma, q^*, x)) \leq \epsilon$. We say that $(P, V)$ has statistical honest-verifier zero-knowledge if there exists a negligible function $\epsilon : \mathbb{N} \rightarrow \mathbb{R}^+$ such that for every input length parameter $n$, $(P, V)$ has HVZK with statistical distance $\epsilon(n)$.

**Remark 2.7.** Since the input queries of both $V^*$ and $\text{Sim}$ are not counted towards the query complexity, we can think of a ZKPCP system as operating in the following setting. In the real world, $x$ is given to the verifier explicitly, and there is no input oracle. The ideal process consists of the simulator alone, who is given $x$ as input. In particular, there is no TTP in the ideal process. $\text{Real}_{V^*,P}(\sigma, q^*, x, w)$ consists of the view of $V^*$, and $\text{Ideal}_{\text{Sim}}(\sigma, q^*, x)$ consists of the output of the simulator.

### 2.3 Codes

A code over the alphabet $\Sigma$ is a subset $C \subseteq \Sigma^n$. Such codes can be associated with an injective encoding function $E_C : \Sigma^n \rightarrow \Sigma^\hat{n}$ that maps messages in $\Sigma^n$ to codewords in $\Sigma^\hat{n}$, i.e., $C = \{E_C(x) \mid x \in \Sigma^n\}$. We will also consider randomized encoding functions $E_C$ which map messages from $\Sigma^{n'}$, for some $n' < n$, into codewords of $C$. We assume that such a randomized encoding is injective, namely the codeword distributions associated with different messages have disjoint support sets.

Most of the well-studied and practically used codes are linear codes. A linear code $C \subseteq \mathbb{F}^\hat{n}$ is a linear subspace over the field $\mathbb{F}$. $n$ is called the blocklength of $C$, and $\dim(C)$ denotes the dimension of the code. The rate of the code is $\text{rate}(C) = \frac{\dim(C)}{n}$. If $C$ is linear then an encoding function for $C$ can be associated with a generator matrix $G \in \mathbb{F}^{\hat{n} \times n}$, where $n = \dim(C)$. That is, encoding is done by multiplying $G$ on the message vector such that $C = \{G \cdot x \mid x \in \mathbb{F}^n\}$.

The distance of a code $C$ is $\Delta(C) = \min_{x \neq y \in C} \Delta(x, y)$, and its relative distance is $\delta(C) = \frac{\Delta(C)}{n}$. Typically, one is interested in codes whose distance is linear in the
blocklength.

2.3.1 Efficiently Encodable and Decodable Error Correcting Codes

An error correcting code (ECC) is a code $C \subseteq \Sigma^n$, associated with an encoding algorithm Enc, and a decoding algorithm Dec that can reconstruct the original message from a “noisy” codeword in which few errors occurred. We will be interested in infinite families $C \subseteq \mathbb{F}^n$ of codes with varying block lengths, with efficient encoding and decoding algorithms. More specifically, we say that $C$ is efficiently encodable if it is associated with encoding functions $\text{Enc}_C : \mathbb{F}^n \rightarrow \mathbb{F}^n$, and given a message $x \in \mathbb{F}^n$, the codeword $\text{Enc}_C(x)$ can be computed in polynomial time. Notice that if $C$ is linear then it is always encodable in time $O(n \cdot \hat{n}) = O(\hat{n}^2)$ given its generator matrix.

We say that $C \subseteq \mathbb{F}^n$ is efficiently decodable from $l < \Delta(C)/2$ errors, if there exists a poly($\hat{n}$)-time decoding algorithm Dec that on any input $w \in \mathbb{F}^n$ such that $\Delta(w, C) \leq l$, outputs a codeword $c \in C$ such that $\Delta(w, c) \leq l$. (Namely, the decoding algorithm outputs the closest codeword.) Sometimes, instead of finding the closest codeword, we will need to obtain the original message, i.e., a message $m \in \mathbb{F}^n$ such that $\Delta(w, E_C(m)) \leq l$. We note that when $C$ is linear, this task is equivalent to “standard” decoding, since after obtaining the closest codeword $c \in C$, matrix multiplication can be used to easily find an $m \in \mathbb{F}^n$ such that $E_C(m) = c$. Thus, if a linear code $C$ is efficiently decodable from $l$ errors, then the encoded message can also be found efficiently, even if $l$ errors occurred.

2.3.2 Locally Testable Codes

Locally Testable Codes (LTCs) are codes associated with an efficient probabilistic tester algorithm that can test membership in the code by querying only few symbols of a purported codeword. A standard $q$-query tester for a linear code $C \subseteq \mathbb{F}^n$ is a randomized algorithm that, given a purported codeword $w \in \mathbb{F}^n$, non-adaptively picks a subset $\mathcal{I} \subseteq [\hat{n}]$ of size at most $q$; reads all symbols of $w|\mathcal{I}$; accepts if $w|\mathcal{I} \in C|\mathcal{I}$, and rejects otherwise (see [BHR05, Theorem 2]). Hence, a $q$-query tester can be associated with a distribution $\mathcal{D}$ over subsets $\mathcal{I} \subseteq [\hat{n}]$ such that $|\mathcal{I}| \leq q$.

**Definition 2.3.1 (Testers and LTCs).** A $q$-query tester for a linear code $C \subseteq \mathbb{F}^n$ is a distribution $\mathcal{D}$ over subsets $\mathcal{I} \subseteq [\hat{n}]$ such that $|\mathcal{I}| \leq q$.

- A $q$-query tester $\mathcal{D}$ is a $(q, \epsilon, \rho)$-tester if for all $w \in \mathbb{F}^n$ such that $\delta(w, C) \geq \rho$, $\Pr_{\mathcal{I} \sim \mathcal{D}}[w|\mathcal{I} \notin C|\mathcal{I}] \geq \epsilon$.

- A $q$-query tester $\mathcal{D}$ is a $(q, \epsilon)$-strong tester if for all $w \in \mathbb{F}^n$, $\Pr_{\mathcal{I} \sim \mathcal{D}}[w|\mathcal{I} \notin C|\mathcal{I}] \geq \epsilon \cdot \delta(w, C)$.

A code $C \subseteq \mathbb{F}^n$ is a $(q, \epsilon, \rho)$-weak LTC if it has a $(q, \epsilon, \rho)$-tester, and it is a $(q, \epsilon)$-strong LTC if it has a $(q, \epsilon)$-strong tester.
We note that although the tester in Definition 2.3.1 does not output acc or rej as a standard tester does, it can be converted to output acc, rej based on its local view \( w|\mathcal{I} \).

### 2.3.3 ZK Codes

We define zero-knowledge codes which, intuitively, are codes associated with a randomized encoding function \( \text{Enc}_C \), such that few codeword symbols reveal practically no information about the encoded message, where “practically no information” is formalized as either perfect, or statistical, zero-knowledge.

Notice that a standard (deterministic) encoding cannot even zero-knowledge against an adversary reading a single codeword symbol, since there must be at least one codeword symbol that depends on the message. For that reason, ZK codes are associated with a randomized encoding scheme.

We not that given a linear code \( C \subseteq \mathbb{F}^{\hat{n}} \) of dimension \( n \), any generator matrix \( G \) for \( C \), together with a message length parameter \( n' < n \), define a randomized encoding \( \text{Enc}_C \) which maps \( x \in \mathbb{F}^{n'} \) to \( \text{Enc}_C(x; r) = G(x; r) \) for \( r \in \mathbb{R}^{n-n'} \). In this case, we define the rate of \( \text{Enc}_C \) as \( \text{rate}(\text{enc}_C) = \frac{n'}{\hat{n}} \). More generally, \( \text{Enc}_C \) can be an arbitrary injective randomized mapping from \( \mathbb{F}^{n'} \) to \( C \). We now define this zero-knowledge property more formally.

**Definition 2.3.2 (ZK-codes).** Let \( n' \in \mathbb{N} \) be a length parameter, \( \hat{n} \in \mathbb{N} \) be a codeword length parameter, \( \tau \in \mathbb{N} \) be a zero-knowledge parameter, and \( \epsilon > 0 \) be a statistical distance parameter. Let \( C \subseteq \mathbb{F}^{\hat{n}} \) be a code, and \( \text{Enc}_C : \mathbb{F}^{n'} \rightarrow \mathbb{F}^{\hat{n}} \) be its associated randomized encoding function. We say that \( \text{Enc}_C \) is \((\tau, \epsilon)\)-ZK, if for every set \( \mathcal{I} \subseteq [\hat{n}] \) of size at most \( \tau \), and every message pair \( x, x' \in \mathbb{F}^{n'} \), \( \text{SD}(\text{Enc}_C(x)|\mathcal{I}, \text{Enc}_C(x')|\mathcal{I}) \leq \epsilon \). If \( \text{Enc}_C \) is \((\tau, 0)\)-ZK then we say that \( \text{Enc}_C \) is \( \tau \)-ZK.
Chapter 3

Leakage-Resilient Secure Computation

In this chapter we discuss leakage-resilient circuit compilers (LRCCs), and generalize them in two respects. First, we construct LRCCs with the following soundness guarantee, which we call SAT-respecting. The compiled (leakage-resilient) circuit, which operates on encoded inputs, is satisfiable (even by some ill-formed encoded input) only if the original circuit is satisfiable. Second, we provide a mechanism that enables $m \geq 1$ parties to compute a joint function $f$ of their inputs, using a single leaky (but otherwise trusted) hardware device.

3.1 Introduction

A leakage-resilient circuit compiler (LRCC) compiles any circuit into a new circuit that operates on encoded inputs. The new circuit withstands side-channel attacks, namely these attacks reveal nothing about the (properly encoded) input, other than what follows from the output. Works on LRCC compilers considered different restrictions on the class of leakage functions that could be tolerated. One line of work [ISW03, FRR+14, MV13] restricts the complexity class from which leakage functions are chosen. A different approach considers leakage that is “local” in the sense that the leakage functions operate on disjoint sets of wires of the circuit (see, e.g., [MR04, GR10, JV10, DF12, GR12]). These works have focused on transforming stateful circuits into stateful circuits that resist continuous leakage, whereas our focus is on LRCCs for stateless circuits, where the compiled circuit is also stateless, and resists one-time leakage. We note that if one is only interested in protecting against one-time leakage, then these known constructions can also be used to transform a stateless circuit into a stateless circuit. Our goal is to achieve information-theoretic security against low complexity leakage.

Our study of generalizations of LRCCs is motivated by their usefulness for solving various cryptographic tasks in the presence of leakage. One such task is zero-knowledge (ZK) proofs, a fundamental primitive for identification, and a useful building block for
cryptographic protocol design. In this context, a prover $P$ is interested in convincing the verifier $V$ of the validity of NP-statements of the form $(x, w) \in R$ for an NP-relation $R$. The parties can use a circuit $C = C_x$ computing the relation $R(x, \cdot)$, where the prover provides the NP-witness $w$ as input to the circuit. However, the prover is unwilling to provide its secret witness $w$ to the circuit “in the clear”, since its internals might leak. Instead, the prover prepares in advance a “leak-free” encoding $\hat{w}$ of $w$, which it stores on a small isolated device (such as a smartcard or USB drive). It then provides $\hat{w}$ as input to a leakage-resilient version $\hat{C}$ of $C$ (e.g., by plugging in his smartcard) which outputs the public verification outcome. This approach would guarantee security against computationally unbounded provers (whereas alternative approaches, e.g., non-interactive ZK proofs, can only protect against computationally-bounded provers.)

This natural and appealing application requires the use of LRCCs that offer information-theoretical security against (low complexity) leakage. However, previous constructions [FRR14, MV13, Rot12] require that the parties encode their inputs in a special form, thereby allowing a malicious prover to manipulate the output by providing a badly formed encoding. Moreover, in these constructions there always exists a choice of an ill-formed “encoding” that satisfies the circuit (i.e., causes it to output 1). Thus, a malicious prover can always convince the verifier of false claims (with probability 1!). Therefore, such applications require LRCCs with the additional SAT-respecting guarantee that if the original circuit is not satisfiable, then there exists no satisfying input (either well- or ill-formed) for the leakage-resilient circuit.

Our second construction generalizes the notion of SAT-respecting LRCCs to the broader regime of leakage resilient hardware. More specifically, we design a leaky (but otherwise trusted) hardware device that allows $m \geq 1$ parties to compute a joint function $f$ of their inputs. Each party locally encodes its input and feeds the encoded input into the device, which evaluates a boolean circuit on the encoded inputs, and returns a public output. This computation should preserve the secrecy of the inputs, as well as the correctness of the output, in the presence of an unbounded, active adversary that can corrupt a subset of the parties, and may also obtain leakage on the internals of the device. (Notice that the secrecy requirement necessitates some encoding of the inputs, otherwise we cannot protect even against a probing attack on a single bit.)

This generalizes the notion of SAT-respecting LRCCs in two respects. First, the input for the computation can be divided between several parties, whereas in the context of SAT-respecting LRCCs, a single party holds the entire input. Second, it guarantees output correctness of general computations, whereas a SAT-respecting LRCC guarantees computation correctness only for satisfying computations in verification circuits (namely, circuits with a single-bit output). We note, however, that SAT-respecting LRCCs are a more natural building-block for constructing ZKPCPs (as described in Chapter 4). This is because (as we explain next) using multiparty LRCCs to implement 2-party ZK proofs requires using a randomized device, which in the context of ZKPCPs will necessitate employing an external source of randomness. This is not needed when one
uses SAT-respecting LRCCs.

A single leaky trusted device can be used to perform general computations in both the single-party setting, and the multiparty setting with \( m \geq 2 \). For example, in the single-party setting the device can be used for ZK proofs as described above, where the device computes the function \( f(x, w) = (x, R(x, w)) \), and the prover provides \((x, \bar{w})\) as input to the leaky device. In both the single-party and the multiparty settings the hardware device is stateless. We note that in the multiparty setting, the device can also be deterministic (where any randomness needed to perform the computation is provided by the parties as part of the encoding of their inputs), whereas the case \( m = 1 \) may require a randomized device. (Indeed, the correctness of the computation may depend on the randomness being independent of the inputs, which cannot be enforced if we let the single party provide both the randomness and the input. For example, a malicious prover in the ZK application described above may correlate the randomness with its input thus causing the device to accept false claims.) However, even in the case \( m = 1 \) the randomness can be made public, and its amount can be reduced by using a suitable pseudorandom generator.

3.1.1 Our Results and Techniques

We now give more details regarding our generalizations of LRCCs, and the underlying techniques. In both cases, our starting point is the LRCC of Faust et al. [FRR+14]. Informally, an LRCC is associated with a function class \( \mathcal{L} \) (the leakage class) and a (randomized) input encoding scheme \( \mathbf{E} \), and compiles a circuit \( C \) into a circuit \( \hat{C} \), that emulates \( C \), but operates on encoded inputs. \( \hat{C} \) is leakage-resilient in the following sense: for any input \( x \) for \( C \), and any \( \ell \in \mathcal{L} \), the output of \( \ell \) on the wire values of \( \hat{C} \), when evaluated on \( \mathbf{E}(x) \), reveals nothing other than \( C(x) \). This is formalized in the simulation-based paradigm (i.e., the wire-values of \( \hat{C} \) can be efficiently simulated given only \( C(x) \)).

**SAT-Respecting Leakage-Resilient Circuits**

To describe our SAT-respecting LRCC, we first need to give some details regarding the LRCC of [FRR+14]. Their LRCC transforms a circuit \( C \) into a circuit \( \hat{C} \) that operates on encodings generated by a linear encoding scheme, and emulates the operations of \( C \) on these encodings. Leakage-resilience against functions in a restricted function class \( \mathcal{L} \) is obtained by “refreshing” the encoded intermediate values of the computation after every operation, using encodings of 0. (We note that the LRCCs of [ISW03, MV13] operate essentially in the same way.) The input of \( \hat{C} \) includes sufficiently many encodings of 0 to be used for the entire computation.\(^1\) However, by providing \( \hat{C} \) also with 1-encodings

\(^1\)Actually, [FRR+14] consider a model of \textit{continuous} leakage, in which the circuit is invoked multiple times on different inputs, and maintains a secret state. Their construction uses tamper-proof hardware (called \textit{opaque gates}) to generate the encodings of 0 used for refreshing. We consider the simpler model of \textit{one-time} leakage on circuits that operated on \textit{encoded} inputs [ISW03, MV13], and as a result we can
(i.e., encodings of 1), one can change the functionality emulated by $\hat{C}$. (In particular, if the encoding “refreshing” the output gate is a 1-encoding, the output is flipped.) This is not just an artifact of the construction, but rather is essential for their leakage-resilience argument. Concretely, to simulate the wire values of $\hat{C}$ without knowing its input, the simulator sometimes uses 1-encodings, which rules out the natural solution of verifying that the encodings used for “refreshing” are 0-encodings.

We observe that if $C$ were emulated twice, it would suffice to know that at least one copy used only 0-encodings, since then $\hat{C}$ is satisfiable only if the honestly-evaluated copy is satisfiable (in which case $C$ is also satisfiable). At first, this may seem as no help at all, but it turns out that by emulating $C$ twice, we can construct what we call a relaxed LRCC, which is similar to an LRCC, except that the simulator is not required to be efficient. Specifically, assume that before compiling $C$ into $\hat{C}$, we would replace it with a circuit $C'$ that computes $C$ twice, and outputs the AND of both evaluations. Then $\hat{C}'$ (complied from $C'$) would be relaxed leakage-resilient. Indeed, an unbounded simulator could simulate the wire values of $\hat{C}'$ by finding a satisfying input $x_S$ for $C$, and honestly evaluating $\hat{C}'$ on a pair of encodings of $x_S$. Using a hybrid argument, we prove that functions in $L$ cannot distinguish the simulated wire values $W_S$ from the actual wire values $W_R$ of $\hat{C}'$ when evaluated on a satisfying input $x_R$. More specifically, we can first replace the input in the first copy from $x_R$ to $x_S$ (using the leakage-resilience of the LRCC of [FRR+14] to claim that functions in $L$ cannot distinguish this hybrid distribution from $W_R$), then do the same in the second copy. By replacing the inputs one at a time, we only need to use 1-encodings in a single copy. However, holding two copies of the original circuit still does not guarantee that the evaluation in at least one of them uses only 0-encodings.

The natural solution would again be to add a sub-circuit verifying that the encodings used are 0-encodings, but this sub-circuit should hide the identity of the “correctly evaluated” copy. This is because the hybrid argument described above first uses 1-encodings in the first copy (and 0-encodings in the second), and then uses 1-encodings in the second copy (and only 0-encodings in the first). Therefore, if functions in $L$ could determine which copy uses only 0-encodings, they could also distinguish between the hybrids. Instead, we describe an “oblivious” checker $T_0$, which at a high-level operates as follows. To check that either the first or the second copy use only 0-encodings, it checks that for every pair of encodings, one from the first copy, and one from the second, the product of the encoded values is 0. To guarantee that leakage on $T_0$ reveals no information regarding which copy uses only 0-encodings, we use the LRCC of [FRR+14] to compile $T_0$ into a leakage-resilient circuit $\hat{T}_0$. This introduces the additional complication that now we must also verify the encodings used to “refresh” the computation in $\hat{T}_0$ (otherwise 1-encodings may be used, potentially altering the

\footnote{incorporate the necessary encodings (used for refreshing) into the encoded input.}

\footnote{This technique is reminiscent of the “2-key trick” of [NY90], used to convert a CPA-secure encryption scheme into a CCA-secure one.}
functionality of $T_0$ and rendering it useless). However, since $T_0$ does not operate directly on the inputs to $\hat{C}$ (it operates only on the encodings used for “refreshing”), we show that the “refreshing” encodings used in $T_0$ can be checked directly (by decoding the encoded values and verifying that they are 0). Further technicalities arise since introducing these additional components prevents us from using the LRCC of [FRR+14] as a black box (see Section 3.4 for more details on the analysis). Finally, we note that our circuit-compiler is relaxed-leakage-resilient because in all hybrids the honestly-evaluated copy should be satisfied, so the simulator needs to find a satisfying input for $C$.

Instantiating our SAT-respecting relaxed-LRCC with the parity encoding scheme, we obtain a SAT-respecting relaxed-LRCC that withstands leakage computable by constant-depth, polynomial-size boolean circuits with multiple output gates. This is captured by the next theorem, where $\text{AC}^0_k$ denotes the family of all constant-depth, poly$(n)$-size, boolean circuits $C : \{0, 1\}^n \rightarrow \{0, 1\}^k$ with $\wedge, \vee$ gates of unbounded fan-in and fan-out, and $\neg$ gates of unbounded fan-out; and an $(L, s, \epsilon)$-relaxed LRCC is an LRCC that given any circuit of size at most $s$, outputs a circuit whose wire values are $(L, \epsilon)$-relaxed leakage-resilient. (See Theorem 3.22 for the formal statement.)

**Theorem 3.1** (SAT-respecting relaxed LRCC for $\text{AC}^0_{s, \delta}$ leakage, informal). Let $s \in \mathbb{N}$ be a leakage size parameter. Then for every constant $\delta \in (0, 1)$ there exists a function $\epsilon_{\delta}(s) = \text{negl}(s)$ such that there exists a SAT-respecting $(\text{AC}^0_{s, \delta}, s, \epsilon_{\delta}(s))$-relaxed-LRCC that on input a circuit $C$ outputs a circuit of size poly$(|C|)$.

Additionally, by employing a different encoding scheme we obtain SAT-respecting relaxed-LRCCs that withstands leakage computable by constant-depth, polynomial-size boolean circuit, augmented with a sublinear number of $\oplus$ gates. (The analysis in this case is much more involved, see Section 3.6 for details.)

**Secure Computation with a Single Leaky Hardware Device**

Recall that the setting consists of $m \geq 1$ parties that utilize a leaky (but otherwise trusted) device to compute a joint function of their inputs, while protecting both the privacy of the inputs and the correctness of the output against an active adversary that corrupts a subset of the parties, and may also obtain leakage on the internals of the device. We think of the device as a *stateless* boolean circuit $C$, where in the case $m > 1$ the circuit can be deterministic. (As noted above, for the case $m = 1$, which is useful for the case of zero-knowledge, we need the circuit to generate randomness in order to guarantee soundness; however, small amount of randomness suffices and this randomness can be safely leaked.)

To guarantee input privacy in the presence of leakage, $C$ must operate on encoded inputs. Moreover, in addition to the output of the computation, $C$ should output a flag indicating whether there was an abort, and these outputs (namely, the output of the computation, and the flag) should also be encoded. (The reason for this is explained
below, and is inherent to any construction based on algebraic detection-manipulation circuits, which are a central tool that we use.) Thus, $C$ is associated with an input encoder $\text{Enc}$, and an output decoder $\text{Dec}$ (used to encode the inputs to, and the output of, $C$).

The natural solution to guarantee leakage resilience is to replace $C$ with its leakage-resilient version $\hat{C}$, and have each party internally encode its input to $\hat{C}$. However, securely implementing this high-level idea introduces complications even in the passive setting (in which the adversary learns the inputs of corrupted parties, and can obtain leakage on the device, but even corrupted parties follow the protocol). More specifically, the leakage-resilience property of $\hat{C}$ crucially relies on the fact that its internal computations are randomized, where the randomness used for these operations is unknown to the leakage function. However, in our setting each party knows the random inputs that it provided for the computation, and so the adversary (who chooses the leakage function) knows the random inputs of all corrupted parties. We overcome this by constructing a new LRCC, based on the LRCC of $\text{FRR}^{+14}$, which combines the randomness provided by various parties into a single set of random values which is unknown to every subset of parties.

In the active case we are faced with further obstacles. Indeed, the solution of replacing $C$ with $\hat{C}$ crucially relied on the fact that $\hat{C}$ emulates $C$ whereas, as noted above, in current LRCC constructions this holds only if the encoded inputs of $\hat{C}$ were honestly generated. As noted above, the use of “ill-formed” encodings is essential for the leakage-resilience argument, so as in the case of SAT-respecting LRCCs, we cannot simply verify that the input encodings were honestly generated (namely, that they are valid encodings).

To overcome this, we observe that when $\hat{C}$ is generated from $C$ using the LRCC of Faust et al. $\text{FRR}^{+14}$, then the effect of ill-formed input encodings on the computation in $\hat{C}$ is equivalent to applying an additive attack (that may blindly flip the value of every internal wire) to the wires of $C$. To overcome this, we use an additively-secure implementation which, roughly speaking, offers the “best possible protection” against additive attacks in the sense that the effect of every such attack on the computation can be simulated by an ideal attack that applies only to the inputs and outputs of the circuit. Thus, by replacing $C$ with an additively-secure implementation $C'$ before applying the LRCC, the effect of ill-formed encoded inputs is further restricted to an additive attack on the inputs and output alone. Finally, we use Algebraic-Manipulation Detection (AMD) encodings $\text{CDF}^{+08}$ to guarantee that such attacks are detected with high probability, where $C'$ is modified to operate on AMD-encodings of the inputs, and output an AMD-encoding of the output. Intuitively, an AMD encoding is an encoding scheme

\[\text{We note that “ill-formed” encodings do not pose a problem for stateful circuits (intuitively, the compiled circuit can use the secret state to overcome the influence of ill-formed encoded inputs, by using well-formed randomness that is part of the internal state). However, we are interested in stateless circuits.}\]
that guarantees that any additive attack on an encoding is detected during decoding, except with small probability. Consequently, ill-formed encoded inputs to the compiled circuit are either detected, or do not influence the computation. (Notice that the use of additively-secure implementations requires that the final circuit output encoded outputs. Otherwise, the adversary can use ill-formed encoded inputs to additively attack the flag indicating whether the computation failed, thus compromising the correctness of the output.)

Thus, combining AMD codes, additively-secure circuit compilers, and the LRCC of [FRR+14] withstanding leakage from a class $\mathcal{L}$ of leakage functions, we obtain a circuit compiler that given a circuit $C$ taking inputs from $m$ parties, generates a circuit $\hat{C}$ that guarantees input privacy and output correctness in the presence of malicious parties, and resists leakage from a somewhat weaker family of leakage functions. We call such circuit compilers $m$-party LRCCs. Roughly speaking, to guarantee that our $m$-party LRCC resist leakage from depth-$d$, size-$s$ boolean circuits, the LRCC of [FRR+14] should resist leakage from boolean circuits of depth $O(d + \log m)$, and size $\text{poly}(s, |C|)$. We note that the leakage resilience guarantee of the LRCC of [FRR+14] holds only when $\hat{C}$ is evaluated on honestly-generated input encodings. However, we analyze their LRCC, and show that $\hat{C}$ remains leakage-resilient even when evaluated on ill-formed encoded inputs, provided that at least one of the inputs was honestly encoded.

Instantiating our $m$-party LRCC with the parity encoding scheme, and using a result of Dubrov and Ishai [DI06b] which, roughly, states that for every constant $\delta \in (0, 1)$, $\text{AC}^0_{s, \delta}$ circuits cannot distinguish between random parity encodings of 0 and 1 ($s$ here denote a leakage size parameter), we obtain the following result, where an $m$-party LRCC for $\mathcal{L}$ is an $m$-party LRCC whose output resists leakage from $\mathcal{L}$. (See Theorem 3.37 for the formal statement.)

**Theorem 3.2** (multiparty LRCC for $\text{AC}^0_{s, \delta}$ leakage, informal). Let $s \in \mathbb{N}$ be a leakage size parameter. Then for every constant $\delta \in (0, 1)$, and every constant $m \in \mathbb{N}$, there exists an $m$-party LRCC for $\text{AC}^0_{s, \delta}$.

### 3.1.2 Chapter Organization

In Section 3.2 we give the necessary preliminaries. In Section 3.3 we describe the LRCC of Faust et al. [FRR+14] which is a main building block in all our LRCC constructions. Sections 3.4 and 3.5 contain the construction of SAT-respecting relaxed LRCCs over non-boolean and boolean fields, respectively. In Section 3.6 we construct a SAT-respecting LRCC that withstands leakage computable by constant depth, polynomial sized circuits with a sublinear number of $\oplus$ gates (which is used in Chapter 4 to construct ZKPCPs). In Section 3.7 we describe a multiparty LRCC that is secure in the presence of passive adversaries, and in Section 3.8 we extend this construction to withstand active corruptions, and prove Theorem 3.2.
3.2 Preliminaries

We begin with an overview of a few notions that will be used in this chapter, then formally define SAT-respecting, and multiparty, LRCCs.

3.2.1 Circuits

We consider boolean circuits $C$ over the set $X = \{x_1, \cdots, x_n\}$ of variables. $C$ is a directed acyclic graph whose vertices are called gates, and whose edges are called wires. The wires of $C$ are labeled with functions over $X$. Every gate in $C$ of in-degree 0 has out-degree 1 and is either labeled by a variable from $X$, and referred to as an input gate; or labeled by a constant $\alpha \in \{0, 1\}$, and referred to as a const$_\alpha$ gate. All other gates are labeled by one of the operations $\land, \lor, \neg, \oplus, \text{copy}, \text{id}$, where $\land, \lor, \oplus$ vertices have fan-in 2 and fan-out 1; $\neg$ has fan-in and fan-out 1; copy vertices have fan-in 1 and fan-out 2, where the labels of the outgoing edges carry the same function as the incoming edge; and id vertices that have fan-in and fan-out 1, and the label of the outgoing edge is the same as the incoming edge. We write $C : \{0, 1\}^n \to \{0, 1\}^k$ to indicate that $C$ is a boolean circuit with $n$ inputs, and $k$ outputs. The size of a circuit $C$, denoted $|C|$, is the number of wires in $C$, together with input and output gates. We will also consider the class of boolean circuits with no $\oplus$ gates.

We also consider arithmetic circuits $C$ over a finite field $\mathbb{F}$ and the set $X$. Similarly to the boolean case, $C$ has input and constant gates, and following $[\text{FRR}^+14]$, all other gates are labeled by one of the following functions $+, -, \times, \text{copy}$ or id, where $+, -, \times$ are the addition, subtraction, and multiplication operations of the field (i.e., the outgoing wire is labeled with the addition, subtraction, or product (respectively) of the labels of the incoming wires), and these vertices have fan-in 2 and fan-out 1; and copy, id vertices are defined similarly to the boolean case. We write $C : \mathbb{F}^n \to \mathbb{F}^k$ to indicate that $C$ is an arithmetic circuit over $\mathbb{F}$ with $n$ inputs, and $k$ outputs. Notice that boolean circuits can be viewed as arithmetic circuits over the binary field in a natural way. Therefore, we sometimes describe boolean circuits using the operations $+, -, \times$ instead of $\oplus, \neg, \land, \lor$.

We use Shallow$\mathbb{F}$ $(n,k,d,s)$ to denote the class of all depth-$d$, size-$s$, arithmetic circuits $C : \mathbb{F}^n \to \mathbb{F}^k$, and denote Shallow$\mathbb{F}$ $(n,d,s) = \cup_{k \in \mathbb{N}} \text{Shallow}_\mathbb{F} (n,k,d,s)$, and Shallow$\mathbb{F}$ $(d,s) = \cup_{n,k \in \mathbb{N}} \text{Shallow}_\mathbb{F}$ $(n,k,d,s)$. It will sometimes be useful to discuss boolean circuits with unbounded fan-in and fan-out. Specifically, we use UnBBool $(n,k,d,s)$ to denote the class of all depth-$d$, size-$s$, boolean circuits $C : \{0,1\}^n \to \{0,1\}^k$ that have gates of unbounded fan-in and fan-out, and no $\oplus$ gates. The classes UnBBool $(n,d,s)$, UnBBool $(d,s)$ are defined similarly to the arithmetic setting, where the arithmetic circuit is replaced by a boolean circuit with gates of unbounded fan-in and fan-out, and has no $\oplus$ gates. (UnBBool stands for UnBounded BOOL.) Somewhat abusing notation, we use the same notations to denote the families of functions computable by circuits in the respective class of circuits.
Definition 3.2.1 ($\text{AC}^0$). Let $n \in \mathbb{N}$ be an input length parameter. The class $\text{AC}^0$ consists of all boolean functions $f : \{0, 1\}^* \to \{0, 1\}$ for which there exists a family $\{C_n\}_n$ of boolean circuits such that the following holds.

- $C_n : \{0, 1\}^n \to \{0, 1\}$ contains $\land, \lor$ gates of unbounded fan-in and fan-out, $\lnot$ gates of unbounded fan-out, and no $\oplus$ gates.
- For every $n$, and every $x \in \{0, 1\}^n$, $C_n(x) = f(x)$.
- There exists a constant $d \in \mathbb{N}$, and a polynomial $p : \mathbb{N} \to \mathbb{N}$, such that for any $n$, $C_n$ has size at most $p(n)$, and depth at most $d$.

Somewhat abusing notation, we use $\text{AC}^0$ also to denote the family of circuit computing the functions in $\text{AC}^0$.

3.2.2 Additive Attacks

Following the terminology of [GIP+14], an additive attack $A$ on an arithmetic circuit $C$ over $\mathbb{F}$ is a fixed vector of field elements (which is independent from the inputs, and internal values, of $C$), containing a coordinate for every wire of $C$, and every input and output gate. The evaluation of $C$ under additive attack $A$, denoted $C^A$, is executed as follows. For every wire $\omega$ connecting gates $a$ and $b$ in $C$, let $A_\omega$ denote the attack on wire $\omega$ (as specified by $A$). Then $A_\omega$ is added to the output of $a$, and the resultant value is then used for to compute the gate $b$. Similarly, for every input (output) gate $g$, if $A_g$ denotes the attack on gate $g$ (as specified by $A$), then $A_g$ is added to the input (output).

3.2.3 Encoding Schemes

In this chapter, we will employ codes, where the encodings and decoding algorithms used to generate codewords, and reconstruct messages from codewords, are also of interest. To differentiate this from the codes described in Section 2.3 (where we thought of a code as simply the set of codewords), we use the term encoding scheme to describe an infinite family of codes with varying block lengths, associated with encoding and decoding algorithms. More specifically, an encoding scheme $E$ over alphabet $\Sigma$ is a pair $(\text{Enc}, \text{Dec})$ of algorithms, where the encoding algorithm $\text{Enc}$ is a probabilistic polynomial-time (PPT) algorithm that given a message $x \in \Sigma^n$ outputs an encoding $\hat{x} \in \Sigma^{\hat{n}}$ for some $\hat{n} = \hat{n}(n)$; and the decoding algorithm $\text{Dec}$ is a deterministic algorithm, that given an $\hat{x}$ of length $\hat{n}$ in the image of $\text{Enc}$, outputs an $x \in \Sigma^n$. Moreover, $\Pr[\text{Dec}(\text{Enc}(x)) = x] = 1$ for every $x \in \Sigma^n$. We use boldface letters (e.g., $a$) to denote encodings.

An encoding scheme is onto, if every range element $y \in \Sigma^{\hat{n}(n)}$ encodes some element of $\Sigma^n$, namely:

Definition 3.2.2. We say that an encoding scheme $E = (\text{Enc}, \text{Dec})$ over alphabet $\Sigma$ is onto, if $\text{Dec}$ is defined for every $x \in \Sigma^{\hat{n}(n)}$. 
Given an encoding scheme $E = (\text{Enc}, \text{Dec})$ over $\mathbb{F}$, and $n \in \mathbb{N}$, we say that $w \in \mathbb{F}^{\hat{n}(n)}$ is well-formed if $w \in \text{Enc}(0^n)$.

Next, we define linear-structured encoding schemes, which are encoding schemes over a finite field $\mathbb{F}$, in which the encoding of $x \in \mathbb{F}^n$ is a sequence of field elements consisting of $n$ “sections”, each encoding a coordinate of $x$; and decoding of each coordinate can be performed by computing the inner product with a fixed vector.

**Definition 3.2.3** (Linear-structured encoding schemes). Let $\mathbb{F}$ be a finite field. We say that an encoding scheme $E = (\text{Enc}, \text{Dec})$ is linear-structured over $\mathbb{F}$ if $\Sigma = \mathbb{F}$, and moreover:

- For every $n$, $n$ divides $\hat{n}(n)$.

- There exists a decoding vector $d^{\hat{n}(n)} \in \mathbb{F}^{\hat{n}(n)/n}$ such that for every $x \in \mathbb{F}^n$:
  
  - Every encoding $y \in \text{supp}(\text{Enc}(x))$ can be partitioned into $n$ equal-length parts $y = (y^1, \ldots, y^n)$.
  
  - $\text{Dec}(y) = (\langle d^{\hat{n}(n)}, y^1 \rangle, \ldots, \langle d^{\hat{n}(n)}, y^n \rangle)$ (where $\langle \cdot, \cdot \rangle$ denotes inner product).

**Example 3.2.4.** The parity encoding scheme $E_\oplus = (\text{Enc}_\oplus, \text{Dec}_\oplus)$ over $\mathbb{F}_2 = \{0, 1\}$ is linear-structured over $\mathbb{F}_2$, where:

- For every $x = (x_1, \ldots, x_n) \in \mathbb{F}_2^n$, $\text{Enc}_\oplus(x_1, \ldots, x_n) = (y^1, \ldots, y^n)$ where $y^i, 1 \leq i \leq n$ is random subject to the constraint $\sum y^i = x_i \mod 2$.

- For every $(y_1, \ldots, y_n) \in \mathbb{F}_2^{\hat{n}(n)}$, $\text{Dec}_\oplus(y_1, \ldots, y_n) = \left(\langle \overline{1}, y^1 \rangle \cdots \langle \overline{1}, y^n \rangle\right)$.

**Remark 3.3.** Every linear-structured encoding scheme $E = (\text{Enc}, \text{Dec})$ is also onto, since if the corresponding decoding vectors are $\{d^{\hat{n}(n)}\}$ then $\text{Dec}$ is defined for every $w \in \mathbb{F}^{\hat{n}(n)}$ (it computes the inner product of the $n$ parts of $w$ with $d^{\hat{n}(n)}$).

**Parameterized encoding schemes.** We consider encoding schemes in which the encoding and decoding algorithms are given an additional input $1^t$, which is used as a security parameter. Concretely, the encoding length depends also on $t$ (and not only on $n$), i.e., $\hat{n} = \hat{n}(n, t)$, and for every $t$ the resultant scheme is an encoding scheme (in particular, for every $x \in \Sigma^n$ and every $t \in \mathbb{N}$, $\text{Pr}[\text{Dec}(\text{Enc}(x, 1^t), 1^t) = x] = 1$). We call such schemes parameterized. A parameterized encoding scheme is onto if it is onto for every $t$. It is linear-structured if it is linear-structured for every $t$ (in particular, there exist decoding vectors $\{d^{\hat{n}(n,t)}\}$). For $n, t \in \mathbb{N}$, $w \in \mathbb{F}^{\hat{n}(n,t)}$ is well-formed if $w \in \text{Enc}(0^n, 1^t)$. Furthermore, we sometimes consider encoding schemes that take a pair of security parameters $1^t, 1^{t_{\text{in}}}$. ($t_{\text{in}}$ is used in cases when the encoding scheme employs an “internal” encoding scheme, where $t_{\text{in}}$ determines the input structure for the internal scheme.) In such cases, the encoding length depends on $n, t, t_{\text{in}}$, and the resultant scheme should be an encoding scheme for every $t, t_{\text{in}} \in \mathbb{N}$. The scheme is onto...
if it is onto for every \( t, t_{in} \in \mathbb{N} \), and it is linear-structured if it linear-structured for every \( t, t_{in} \in \mathbb{N} \). We will usually omit the term “parameterized”, and use “encoding scheme” to describe both parameterized and non-parameterized encoding schemes.

**Leakage-Indistinguishability of Distributions and Encodings**

We describe leakage-indistinguishability of distributions and encoding schemes. At a high level, leakage-indistinguishability against a class \( \mathcal{L} \) of leakage-functions means that leakage functions in the class have small statistical advantage in distinguishing between random samples, or encodings.

**Definition 3.2.5** (Leakage-indistinguishability of distributions). Let \( D, D' \) be finite sets, \( \mathcal{L} = \{ \ell : D \to D' \} \) be a family of leakage functions, and \( \epsilon > 0 \) be a statistical distance parameter. We say that two distributions \( X,Y \) over \( D \) are \((\mathcal{L}, \epsilon)\)-leakage-indistinguishable, if for any function \( \ell \in \mathcal{L} \), \( \text{SD}(\ell(X), \ell(Y)) \leq \epsilon \).

**Remark 3.4.** In case \( \mathcal{L} \) consists of functions over a union of domains, we say that \( X,Y \) over \( D \) are \((\mathcal{L}, \epsilon)\)-leakage-indistinguishable if \( \text{SD}(\ell(X), \ell(Y)) \leq \epsilon \) for every function \( \ell \in \mathcal{L} \) with domain \( D \).

**Remark 3.5.** Notice that if \( X,Y \) are \((\mathcal{L}, \epsilon)\)-leakage indistinguishable, then \( X,Y \) are also \((\mathcal{L}', \epsilon)\)-leakage indistinguishable for every \( \mathcal{L}' \subseteq \mathcal{L} \).

**Definition 3.2.6** (Leakage-indistinguishability of functions and encodings). Let \( \mathcal{L} \) be a family of leakage functions, and \( \epsilon > 0 \) be a statistical distance parameter. A randomized function \( f \) with domain \( \Sigma^n \) is \((\mathcal{L}, \epsilon)\)-leakage-indistinguishable if for every \( x, y \in \Sigma^n \), the distributions \( f(x), f(y) \) are \((\mathcal{L}, \epsilon)\)-leakage-indistinguishable.

We say that an encoding scheme \( E \) is \((\mathcal{L}, \epsilon)\)-leakage-indistinguishable if for every large enough \( t \in \mathbb{N} \), \( \text{Enc}(\cdot, 1^t) \) is \((\mathcal{L}, \epsilon)\)-leakage indistinguishable.

Håstad [Hås86] (as cited in [Kli01, Corollary 1]) proved that \( \text{AC}^0 \) circuits (Definition 3.2.1) cannot distinguish between random parity encodings of 0 and 1:

**Theorem 3.6** ([Hås86]). Let \( d \in \mathbb{N} \) be a constant depth parameter, and \( t \in \mathbb{N} \) be a security parameter. The parity encoding scheme \( E_{\oplus} \) of Example 3.2.4 is \( \left( \text{UnBBool} \left( t, 1, d, 2^{\frac{t}{d}} \right), 2^{-\Omega \left( \frac{t}{d} \right)} \right) \)-leakage-indistinguishable.

Dubrov et al. ([DI06b, Theorem 3.3]) extended Theorem 3.6 to the broader family of leakage-functions \( \text{AC}^0_k \), consisting of circuits that have multiple outputs:

**Theorem 3.7** ([DI06b]). For a natural \( d > 1 \), and \( \delta \in (0, 1) \), the parity encoding \( E_{\oplus} \) of Example 3.2.4, with security parameter \( t \), is \( \left( \text{UnBBool} \left( t^\delta, d, 2^{O \left( t^{\frac{1-\delta}{d}} \right)} \right), 2^{-\Omega \left( t^{\frac{1-\delta}{d}} \right)} \right) \)-leakage-indistinguishable.
3.2.4 Circuit Compilers

In this section we define circuit compilers which, informally, consists of an encoding scheme and a compiler algorithm that compiles a given circuit into a circuit operating on encodings, and emulating the original circuit. Formally,

**Definition 3.2.7 (Circuit compiler over \( F \)).** A circuit compiler over \( F \) is a pair \((\text{Comp}, E)\) of algorithms with the following syntax.

- \( E = (\text{Enc}, \text{Dec}) \) is an encoding scheme, where \( \text{Enc} \) is a PPT encoding algorithm that given a vector \( x \in F^n \), and \( 1^\sigma \), outputs a vector \( \hat{x} \). We assume that \( \hat{x} \in F^{\hat{n}} \) for some \( \hat{n} = \hat{n}(n, \sigma) \).

- \( \text{Comp} \) is a polynomial-time algorithm that given an arithmetic circuit \( C \) over \( F \) outputs an arithmetic circuit \( \hat{C} \).

We require that \((\text{Comp}, E)\) satisfy the following correctness requirement. For any arithmetic circuit \( C \), and any input \( x \) for \( C \), we have \( \Pr[\hat{C}(\hat{x}) = C(x)] = 1 \), where \( \hat{x} \) is the output of \( \text{Enc}(x, 1^{\|C\|}) \).

A boolean circuit compiler is a circuit compiler over \( F_2 \). We will usually omit the prefix “boolean”, and simply use “circuit compiler” to describe a boolean circuit compiler.

3.2.5 Leakage-Resilient Circuit Compilers

We consider circuit compilers whose outputs are leakage resilient for a class \( \mathcal{L} \) of functions, in the following sense. For every “not too large” circuit \( C \), and every input \( x \) for \( C \), the wire values of the compiled circuit \( \hat{C} \), when evaluated on a random encoding \( \hat{x} \) of \( x \), can be simulated given only the output of \( C \); and functions in \( \mathcal{L} \) cannot distinguish between the actual and simulated wire values.

We first define the notion of a leakage-resilient implementation. The following notation will be useful.

**Notation 3.8.** For a Circuit \( C \), a leakage function \( \ell : F^{\|C\|} \rightarrow F^m \) for some natural \( m \), and an input \( x \) for \( C \), \( [C, x] \) denotes the wire values of \( C \) when evaluated on \( x \), and \( \ell[C, x] \) denotes the output of \( \ell \) on \( [C, x] \).

**Definition 3.2.8 (Leakage-resilient implementation).** Let \( F \) be a finite field, \( n \in \mathbb{N} \) be an input length parameter, \( f : F^n \rightarrow F \), \( \mathcal{L} \) be a family of leakage functions, and \( \epsilon > 0 \). We say that \((\text{Enc}, C)\) is an \((\mathcal{L}, \epsilon)\)-leakage-resilient implementation of \( f \) if it satisfies the following requirements.

- **Syntax:**
  - \( \text{Enc} : F^n \rightarrow F^{\hat{n}} \) is a randomized function, called the input encoder.
– \(C : \mathbb{F}^n \rightarrow \mathbb{F}\) is a deterministic circuit.

- **Correctness.** For every \(x \in \mathbb{F}^n\), \(\Pr[C(\text{Enc}(x)) = f(x)] = 1\).

- **leakage resilience.** There exists a simulator \(\text{Sim}\) such that for every input \(x \in \mathbb{F}^n\), and every leakage function \(\ell \in \mathcal{L}\) of input length \(|C|\), 
  \(\text{SD}(\ell[\text{Sim}(f,f(x))],\ell[C,\hat{x}]) \leq \epsilon\), where \(\hat{x} \leftarrow \text{Enc}(x)\).

Next, we define a leakage-resilient circuit compiler.

**Definition 3.2.9** (LRCC). Let \(\mathbb{F}\) be a finite field, \(t \in \mathbb{N}\) be a security parameter, \(n \in \mathbb{N}\) be an input length parameter, \(\mathcal{L}\) be a family of leakage functions, \(\epsilon(t) : \mathbb{N} \rightarrow \mathbb{R}^+\) be a statistical distance function, and \(S(n) : \mathbb{N} \rightarrow \mathbb{N}\) be a size function. Let \(\text{Comp}\) be a PPT algorithm that on input a circuit \(C : \mathbb{F}^n \rightarrow \mathbb{F}\), outputs a circuit \(\hat{C}\).

We say that \((\text{Comp},E = (\text{Enc},\text{Dec}))\) is an \((\mathcal{L},S(n),\epsilon(t))\)-leakage-resilient circuit compiler (LRCC) if there exists a PPT algorithm \(\text{Sim}\) such that for all sufficiently large \(n\)'s, and every circuit \(C : \mathbb{F}^n \rightarrow \mathbb{F}\) of size at most \(S(n)\) that computes a function \(f_C\), 
\((\text{Enc}(\cdot,1^t),\hat{C})\) is an \((\mathcal{L},\epsilon(t))\)-leakage-resilient implementation of \(f_C\), where the security property holds with simulator \(\text{Sim}\), and \(\text{Sim}\) takes as input the circuit \(C\), and its output.

We also define a relaxed notion of leakage-resilience, which is similar to Definition 3.2.9, except that the simulator may be inefficient.

**Definition 3.2.10** (Relaxed LRCC). Let \(\mathbb{F}\) be a finite field, \(t \in \mathbb{N}\) be a security parameter, \(n \in \mathbb{N}\) be an input length parameter, \(\mathcal{L}\) be a family of leakage functions, \(\epsilon(t) : \mathbb{N} \rightarrow \mathbb{R}^+\) be a statistical distance function, and \(S(n) : \mathbb{N} \rightarrow \mathbb{N}\) be a size function. Let \(\text{Comp}\) be a PPT algorithm that on input a circuit \(C : \mathbb{F}^n \rightarrow \mathbb{F}\), outputs a circuit \(\hat{C}\).

We say that \((\text{Comp},E = (\text{Enc},\text{Dec}))\) is an \((\mathcal{L},S(n),\epsilon(t))\)-relaxed LRCC if there exists a (not necessarily efficient) algorithm \(\text{Sim}\) such that for all sufficiently large \(n\)'s, and every circuit \(C : \mathbb{F}^n \rightarrow \mathbb{F}\) of size at most \(S(n)\) that computes a function \(f_C\), 
\((\text{Enc}(\cdot,1^t),\hat{C})\) is an \((\mathcal{L},\epsilon(t))\)-leakage-resilient implementation of \(f_C\), where the security property holds with simulator \(\text{Sim}\), and \(\text{Sim}\) takes as input the circuit \(C\), and its output.

The error in Definitions 3.2.11, 3.2.9, and 3.2.10 is defined in relation to the input length \(n\). These definitions can be naturally extended such that the compiler is also given a security parameter \(\sigma\), and the error depends on \(\sigma\) (and possibly also \(n\)).

### 3.2.6 LRCCs with Soundness Properties

The first generalization of LRCCs that we consider are LRCCs with an additional *soundness* property which, informally, guarantees that for every circuit with output domain \(\mathbb{F}\) (i.e., for every circuit whose output is a single field element), if the compiled circuit is satisfiable then so is the original circuit. We formalize this notion for general (not necessarily leakage-resilient) circuit compilers.
Definition 3.2.11 (SAT-respecting circuit compiler). We say that a circuit compiler \((\text{Comp}, E)\) is SAT-respecting if it satisfies the following soundness requirement for every circuit \(C : F^n \rightarrow F\). If \(\hat{C} = \text{Comp}(C)\) is satisfiable then \(C\) is satisfiable, i.e., if \(\hat{C}(\hat{x}^*) = 0\) for some \(\hat{x}^* \in F^{\hat{n}}\), then there exists an \(x \in F^n\) such that \(C(x) = 0\). (For \(F = \mathbb{F}_2\), we require that if \(\hat{C}\) outputs 1 on some input, then so does \(C\).)

3.2.7 Multiparty LRCCs

The second generalization of LRCCs that we consider formalizes the notion of secure computation with a single piece of trusted (but leaky) hardware device. The trusted hardware device should guarantee security with abort against an adversary that corrupts a subset of parties, and obtains leakage (from some pre-defined leakage class) on wires of the device. This raises the following points.

1. The output should include a flag signaling whether there was an abort.
2. Leakage on the wires of the device should reveal no information on the computation, and the inputs of the honest parties, other than what can be computed from the output. This requires the computation in the device to be randomized.
3. The inputs should be encoded (otherwise leakage on the input wires corresponding to the inputs of honest parties may reveal information that cannot be computed from the outputs).

To keep our trusted hardware device deterministic (which is preferable to using randomized devices), we have the parties provide the randomness. However, as will be described in the following sections, resisting leakage would require using random bits chosen according to a specific distribution (rather than uniformly random bits). In particular, by providing “random” bits that were not chosen according to the correct distribution, malicious parties may in effect tamper with the hardware device. To guarantee that such “tampering” is detected, the device would output an encoding of the output, and tampering with the device would cause a decoding failure. Thus, a secure implementation consists of an input encoding algorithm \(\text{Enc}\), an output decoding algorithm \(\text{Dec}\), and a circuit \(C\) which implements the trusted device.

We first formalize this notion for the concrete case, describing a secure implementation of a function.

Definition 3.2.12 (Secure function implementation). Let \(m \in \mathbb{N}\), \(f : (\{0, 1\}^n)^m \rightarrow \{0, 1\}^k\) be an \(m\)-argument function, \(\mathcal{L}\) be a family of leakage functions, and \(\epsilon > 0\). We say that \((\text{Enc}, C, \text{Dec})\) is an \(m\)-party \((\mathcal{L}, \epsilon)\)-secure implementation of \(f\) if it satisfies the following requirements.

- Syntax:
  - \(\text{Enc} : \{0, 1\}^n \rightarrow \{0, 1\}^\hat{n}\) is a randomized function, called the input encoder.
• **Correctness.** For every \(x_1, \ldots, x_m \in \{0,1\}^n\),

\[
\Pr [\text{Dec} (C (\text{Enc} (x_1), \ldots, \text{Enc} (x_m))) = (0, f (x_1, \ldots, x_m))] = 1.
\]

• **Security.** For every adversary \(A\) there exists a simulator \(\text{Sim}\) such that for every input \((x_1, \ldots, x_m) \in \{\{0,1\}^n\}^m\), and every leakage function \(\ell \in L\),

\[
\text{SD} (\text{Real}, \text{Ideal}) \leq \epsilon,
\]
where \(\text{Real}, \text{Ideal}\) are defined as follows.

\textbf{Real}:

- \(A\) picks a set \(B \subset [m]\) of corrupted parties, and (possibly ill-formed) encoded inputs \(x'_i \in \{0,1\}^n\) for every \(i \in B\).
- For every uncorrupted party \(j \notin B\), let \(x'_j = \text{Enc} (x_j)\).
- \(\text{Real} = (B, \ell [C, (x'_1, \ldots, x'_m)], C (x'_1, \ldots, x'_m), \text{Dec} (C (x'_1, \ldots, x'_m)))\).

\textbf{Ideal}:

- \(\text{Sim}\) picks a set \(B \subset [m]\) of corrupted parties, effective inputs \(w_i \in \{0,1\}^n\) for every \(i \in B\), and \(b \in \{0,1\}\). (Intuitively, \(b\) indicates whether to abort the computation.)
- For every uncorrupted \(j \notin B\), let \(w_j = x_j\).
- Define \(y = f (w_1, \ldots, w_m)\).
- If \(b = 0\), let \(z = (0, y)\). Otherwise, let \(z = (1, 0^k)\).
- Let \(W \leftarrow \text{Sim} (y)\), where \(W\) contains a bit for each wire of \(C\). Denote the restriction of \(W\) to the output wires by \(W\)\text{out}.
- \(\text{Ideal} = (B, \ell (W), W\text{out}, z)\).

We also consider a “passive” version of secure function implementation which, at a high level, is only secure when the adversary is passive, namely the inputs of corrupted parties in \(B\) are valid encodings of their inputs to \(f\). In this case, we are guaranteed that parties provide valid random inputs to the computation, so there is no “tampering” with the device, and therefore there is no reason to encode the output. This is formalized in the next definition.

**Definition 3.2.13** (Passive-secure function implementation). Let \(m \in \mathbb{N}\), \(f : \{\{0,1\}^n\}^m \rightarrow \{0,1\}^k\) be an \(m\)-argument function, \(L\) be a family of leakage functions, and \(\epsilon > 0\). We say that \((\text{Enc}, C)\) is an \(m\text{-party} (L, \epsilon)\text{-passive-secure implementation of } f\) if it satisfies the following requirements.

• **Syntax:**

We choose to describe the operation of these compilers over a finite field $F$.

- $\text{Enc} : \{0,1\}^n \rightarrow \{0,1\}^k$ is a randomized function, called the input encoder.
- $C : (\{0,1\}^n)^m \rightarrow \{0,1\}^k$ is a deterministic circuit.

**Leakage-privacy.** There exists a simulator $\text{Sim}$ such that for every input $(x_1,\cdots,x_m) \in (\{0,1\}^n)^m$, and every leakage function $\ell \in \mathcal{L}$,

$$\text{SD} ((\ell[C,(x_1,\cdots,x_m)],C(x_1,\cdots,x_m)),(\ell(\text{Sim}(y)),y)) \leq \epsilon$$

where $y = f(x_1,\cdots,x_m)$, and $\text{Sim}(y)$ contains a bit for each wire of $C$.

Next, we define the asymptotic notions corresponding to Definitions 3.2.12 and 3.2.13.

**Definition 3.2.14** ($m$-party circuit). Let $m \in \mathbb{N}$. We say that a boolean circuit $C$ is an $m$-party circuit if its input can be partitioned into $m$ equal-length strings, i.e., $C : (\{0,1\}^n)^m \rightarrow \{0,1\}^k$ for some $n,k \in \mathbb{N}$.

**Definition 3.2.15** (Multiparty LRCCs and passive-secure multiparty LRCCs). Let $n \in \mathbb{N}$ be an input length parameter, $m \in \mathbb{N}$, $\mathcal{L}$ be a family of leakage functions, $S : \mathbb{N} \rightarrow \mathbb{N}$ be a size function, and $\epsilon : \mathbb{N} \rightarrow \mathbb{R}^+$. Let $\text{Comp}$ be a PPT algorithm that on input $m$, and an $m$-party circuit $C : (\{0,1\}^n)^m \rightarrow \{0,1\}^k$, outputs a circuit $\hat{C}$.

We say that $(\text{Enc},\text{Comp},\text{Dec})$ is an $m$-party ($\mathcal{L},S(n),\epsilon(n)$)-LRCC ($m$-party LRCC, or multiparty LRCC) if there exists a PPT simulator $\text{Sim}$ such that for all sufficiently large $n$’s, and every $m$-party circuit $C : (\{0,1\}^n)^m \rightarrow \{0,1\}^k$ of size at most $S(n)$ that computes a function $f_C$, $(\text{Enc},\hat{C},\text{Dec})$ is an $(\mathcal{L},\epsilon(n))$-secure implementation of $f_C$, where the security property holds with simulator $\text{Sim}$, and $\text{Sim}$ takes as input the circuit $C$, its output, and black-box access to the adversary.

We say that $(\text{Enc},\text{Comp})$ is an $m$-party ($\mathcal{L},S(n),\epsilon(n)$)-passive-secure LRCC (passive-secure multiparty LRCC, or passive-secure multiparty LRCC) if there exists a PPT simulator $\text{Sim}$ such that for all sufficiently large $n$’s, and every $m$-party circuit $C : (\{0,1\}^n)^m \rightarrow \{0,1\}^k$ of size at most $S(n)$ that computes a function $f_C$, $(\text{Enc},\hat{C})$ is an $(\mathcal{L},\epsilon(n))$-passive-secure implementation of $f_C$, where the leakage-privacy property holds with simulator $\text{Sim}$, that is given as input the circuit $C$, and its output.

**Remark 3.9.** All our definitions (Definitions 3.2.12-3.2.15) naturally extend to the arithmetic setting with a finite field $\mathbb{F}$, in which $C$ is an arithmetic circuit over $\mathbb{F}$. When discussing the arithmetic setting, we explicitly state the field over which we are working (e.g., we use “multiparty LRCC over $\mathbb{F}^n$ to denote that the multiparty LRCC is in the arithmetic setting with field $\mathbb{F}$).

## 3.3 Gadget-Based LRCCs

In this section we describe gadget-based LRCCs, which are the basis to all our LRCCs. We choose to describe the operation of these compilers over a finite field $\mathbb{F}$, but the
description naturally adjusts to the boolean case as well. At a high level, given a circuit $C$, a gadget-based LRCC replaces every wire in $C$ with a bundle of wires, which carry an encoding of the wire value, and every gate with a sub-circuit that emulates the operation of the gate on encoded inputs. We now describe this technique in more details.

**Gadgets**

A bundle is a string of field elements, encoding a field element according to some encoding scheme $E$; and a gadget is a circuit which operates on bundles and emulates the operation of the corresponding gate in $C$. A gadget has both standard inputs, that represent the wires in the original circuit, and masking inputs, that are used to achieve privacy. More formally, a gadget emulates a specific boolean or arithmetic operation on the standard inputs, and outputs a bundle encoding the correct output. Every gadget $G$ is associated with a set $M_G$ of “well-formed” masking input bundles (e.g., in the LRCC of [FRR+14], $M_G$ consists of sets of 0-encodings). For every standard input $x$, on input a bundle $x$ encoding $x$, and any masking input bundles $m \in M_G$, the output of the gadget $G$ should be consistent with the operation on $x$. For example, if $G$ computes the operation $\times$, then for every standard input $x = (x_1, x_2)$, for every bundle encoding $x = (x_1, x_2)$ of $x$ according to $E$, and for every masking input bundles $m \in M_G$, $G(x, m)$ is a bundle encoding $x_1 \times x_2$ according to $E$. Since all the encoding schemes that we consider are onto, we may think of the masking input bundles $m$ as encoding some set $\text{mask}$ of values, in which case we say that $G$ takes $|\text{mask}|$ masking inputs. The privacy of the internal computations in the gadget will be achieved when the masking input bundles of the gadget are uniformly distributed over $M_G$, regardless of the actual values encoded by the masking input bundles.

**Gadget-Based LRCCs**

In our (multiparty or SAT-respecting) LRCCs, the compiled circuit $\hat{C}$ is obtained from a circuit $C$ by replacing every wire with a bundle, every gate with the corresponding gadget, and adding decoding sub-circuits (computing the decoding function of $E$) following the output gates of $C$. Recall that the gadgets also have masking inputs. These are provided as part of the encoded input of $\hat{C}$, in the following way. $E = (\text{Enc}, \text{Dec})$ uses an “inner” encoding scheme $E^{\text{in}} = (\text{Enc}^{\text{in}}, \text{Dec}^{\text{in}})$, where $\text{Enc}$ uses $\text{Enc}^{\text{in}}$ to encode the inputs of $C$, concatenated with 0$\kappa$ for a “sufficiently large” $\kappa$ (these 0-encodings will be the masking inputs of the gadgets, that are used to achieve privacy); and $\text{Dec}$ uses $\text{Dec}^{\text{in}}$ to decode its input, and discards the last $\kappa$ symbols.

The starting point of all our LRCCs is the LRCC of Faust et al. [FRR+14], which we describe next.
3.3.1 The LRCC of Faust et al. [FRR+14]

In this section we describe the gadget-based LRCC of [FRR+14], which we denote by \((\text{Comp}^{FRRTV}, \text{Enc}^{FRRTV})\). Recall that in a gadget-based LRCC, every wire is replaced with a bundle of wires, and every gate is replaced with a sub-circuit that operates on wire bundles, and emulates the operation of the corresponding gate. We first describe the gadgets used by \((\text{Comp}^{FRRTV}, \text{Enc}^{FRRTV})\).

Construction 3.10. Let \(t \in \mathbb{N}\) be a security parameter, and \(\text{Enc}^{\text{in}} = (\text{Enc}^{\text{in}}, \text{Dec}^{\text{in}})\) be a linear-structured encoding scheme which outputs encodings of length \(\hat{n}(n, t)\), with decoding vectors \(\hat{d}(n,t)\). We denote \(\hat{n}_1 = \hat{n}(1, t)\).

1. \(\times\) gadget: inputs \(a \in \text{Enc}^{\text{in}}(a, 1^t), b \in \text{Enc}^{\text{in}}(b, 1^t)\) for \(a, b \in \mathbb{F}\), and masking inputs \(r^1, \ldots, r^{\hat{n}_1+1} \in \text{Enc}^{\text{in}}(0, 1^t)\); output \(c \in \text{Enc}^{\text{in}}(a \times b, 1^t)\).
   (a) Computes an \(\hat{n}_1 \times \hat{n}_1\) matrix \(B = ab^T = (a_i \times b_j)_{i,j \in [t]}\) (using \(\hat{n}_1^2 \times\) gates).
   (b) Computes an \(\hat{n}_1 \times \hat{n}_1\) matrix \(S\) as the matrix whose columns are \(r^1, \ldots, r^{\hat{n}_1}\).
   (c) Computes an \(\hat{n}_1 \times \hat{n}_1\) matrix \(U = B + S\) (using \(\hat{n}_1^2 +\) gates and \(\hat{n}_1^2\) constant gates).
   (d) Computes a vector \(q = Ud^{\hat{n}_1}\), i.e., \(q_i\) is the value that the \(i\)’th row of \(U\) encodes. This is computed using \(\hat{n}_1 \times,\) \text{const}, and \(+\) gates
   (e) Computes \(c = q + r^{\hat{n}_1+1}\).

2. \(+, -\) gadgets: inputs \(a \in \text{Enc}^{\text{in}}(a, 1^t), b \in \text{Enc}^{\text{in}}(b, 1^t)\) for \(a, b \in \mathbb{F}\), and masking input \(r \in \text{Enc}^{\text{in}}(0, 1^t)\); output \(c \in \text{Enc}^{\text{in}}(a + b, 1^t)\) (or \(c \in \text{Enc}^{\text{in}}(a - b, 1^t)\) for the \(-\) gadget).
   (a) Computes \(q = a + b\) (using \(\hat{n}_1\) \text{+ gates}). (The \(-\) gadget computes \(q = a - b\).)
   (b) Computes \(c = q + r\) (using \(\hat{n}_1 +\) gates).

3. \text{const, gadget for } f \in \mathbb{F}:\) masking input \(r \in \text{Enc}^{\text{in}}(0, 1^t)\); output \(c \in \text{Enc}^{\text{in}}(f, 1^t)\).
   (a) Computes some fixed encoding \(f \in \text{Enc}^{\text{in}}(f)\) (using \(\hat{n}_1\) \text{const gates}.
   (b) Computes \(c = f + r\) (using \(\hat{n}_1 +\) gates).

4. \text{copy gadget}: input \(a \in \text{Enc}^{\text{in}}(a, 1^t)\) for \(a \in \mathbb{F}\), masking inputs \(r^1, r^2 \in \text{Enc}^{\text{in}}(0, 1^t)\); outputs \(b, c \in \text{Enc}^{\text{in}}(a, 1^t)\).
   (a) Computes \(b = a + r^1\) (using \(\hat{n}_1 +\) gates).
   (b) Computes \(c = a + r^2\) (using \(\hat{n}_1 +\) gates).

5. \text{id gadget}: input \(a \in \text{Enc}^{\text{in}}(a, 1^t)\) for \(a \in \mathbb{F}\), masking input \(r \in \text{Enc}^{\text{in}}(0, 1^t)\); outputs \(c \in \text{Enc}^{\text{in}}(\text{id}(a), 1^t)\).
   (a) Computes \(c = a + r\) (using \(\hat{n}_1 +\) gates).
6. **rand gadget**: input $a \in \text{Enc}^\text{in}(a,1^t)$ for $a \in \mathbb{F}$, masking input $r \in \text{Enc}^\text{in}(0,1^t)$; outputs $c \in \text{Enc}^\text{in}(\text{id}(a),1^t)$.

   (a) Computes $c = a + r$ (using $n_1 +$ gates).

The leakage-resilience property of a gadget-based LRCC follows from two properties of the underlying gadgets. First, the gadgets are locally reconstructible, namely for every encoding of a “legal” input-output pair, the internal wires of the gadget (as determined by the encoding of the inputs and outputs, and the masking inputs) could be simulated in a low complexity class. Second, gadgets are re-randomizing in the sense that the encodings that each gadget outputs are uniform subject to encoding the “correct” value. (This notion is formally defined for a more general case in Section 3.7.1, see Definition 3.7.3.)

Local reconstruction, and re-randomization, hold when the gadgets are evaluated using well-formed masking inputs. However, for the results of Section 3.8, we also need to specify the properties of gadgets when possibly using ill-formed masking inputs. Therefore, we define local reconstruction for the more general case in which gadgets may be evaluated with ill-formed masking inputs.

**Definition 3.3.1** (Gadget local-reconstruction). Let $E = (\text{Enc},\text{Dec})$ be an encoding scheme, and let $G$ be a gadget that operates on encodings generated by $\text{Enc}$, and takes $r$ masking inputs. For $\text{mask} \in \mathbb{F}^r$, we say that a pair $(x,y)$ of encodings is **plausible for $G$ with masking inputs $\text{mask}$** if there exists an $m \in \text{supp}(\text{Enc}(\text{mask}))$ such that $G$ on input $(x,m)$ outputs $y$. We say that $(x,y)$ is **plausible for $G$** if there exists an $m_0 \in \text{supp}(\text{Enc}(0^t))$ such that $G$ on input $(x,m_0)$ outputs $y$.

For a security parameter $t \in \mathbb{N}$, let $\epsilon(t) : \mathbb{N} \to \mathbb{R}^+$ be a statistical distance parameter, and $\mathcal{L}, \mathcal{L}_G$ be families of functions. We say that $G$ is $(\mathcal{L},\epsilon(t))$-reconstructible by $\mathcal{L}_G$ **with masking inputs $\text{mask}$** if the following holds. There exists a distribution $\text{REC}_{\text{mask}}$ over functions $\text{rec}$ that take as input the standard inputs of $G$, and its output, and output simulated values for the masking inputs, and internal wires of $G$, such that for every pair $(x,y)$ that is plausible for $G$ with masking inputs $\text{mask}$: (1) $\text{supp}(\text{REC}_{\text{mask}}) \subseteq \mathcal{L}_G$; and (2) if $\text{rec} \leftarrow \text{REC}_{\text{mask}}$ then $\text{rec}(x,y)$ is $(\mathcal{L},\epsilon(t))$-leakage-indistinguishable from the distribution over the wires of $G$, conditioned on $(x,y)$, when its masking inputs are random encodings $m \leftarrow \text{Enc}(\text{mask})$ of $\text{mask}$.

We say that $G$ is $(\mathcal{L},\epsilon(t))$-reconstructible by $\mathcal{L}_G^0$, if $G$ is $(\mathcal{L},\epsilon(t))$-reconstructible by $\mathcal{L}_G^0$ with masking inputs $\text{mask} = 0^t$. We say that $G$ is $(\mathcal{L},\epsilon(t))$-reconstructible by $\mathcal{L}_G$ with ill-formed masking inputs, if it is $(\mathcal{L},\epsilon(t))$-reconstructible by $\mathcal{L}_G$ with masking inputs $\text{mask}$, for every masking inputs $\text{mask}$.

Faust et al. [FRR+14] prove that the gadgets of Construction 3.10 are re-randomizing, which follows directly from the fact that the outputs of the gadgets are masked with random and independent well-formed vectors. They also prove that the gadgets are locally-reconstructible.
**Lemma 3.3.2** (Gadgets are locally reconstructible, [FRR$^+$14]). Let $n \in \mathbb{N}$ be an input length parameter, $t \in \mathbb{N}$ be a security parameter, $\mathbb{F}$ be a finite field, $\mathcal{L}, \mathcal{L}_E$ be families of functions, and $\epsilon(t) : \mathbb{N} \to \mathbb{R}^+$ be a statistical distance parameter. Then the following holds for the gadgets described in Construction 3.10, when $E^{in}$ uses security parameter $t$.

- $+$ and $-$ gadgets are $(\mathcal{L}, 0)$-reconstructible by $\text{Shallow}_R (2, O (\hat{n} (1, t)))$.
- copy, id, and $\text{const}_\alpha$ gadgets are $(\mathcal{L}, 0)$-reconstructible by $\text{Shallow}_R (1, O (\hat{n} (1, t)))$.
- If $E^{in}$ is $(\mathcal{L}_E, \epsilon(n))$-leakage-indistinguishable, and $\mathcal{L}_E = \mathcal{L} \circ \text{Shallow}_R (3, O (\hat{n} (1, t)))$, then the $\times$ gadget is $(\mathcal{L}, \hat{n} (1, t) \cdot \epsilon(t))$-reconstructible by $\text{Shallow}_R (2, O (\hat{n}^2 (1, t)))$.

Next, we describe the LRCC of [FRR$^+$14].

**Construction 3.11.** Let $t, t_{in} \in \mathbb{N}$, and $n \in \mathbb{N}$ be an input length parameter. The LRCC $(\text{Comp}^{\text{FRRTV}}, E^{\text{FRRTV}})$ is defined as follows. Let $E^{in} = (\text{Enc}^{in}, \text{Dec}^{in})$ be a linear-structured encoding scheme which outputs encodings of length $\hat{n}(n, t)$, such that for every $n, t \in \mathbb{N}$, and every $x = (x_1, \cdots, x_n) \in \mathbb{F}^n$, $\text{Enc}^{in} (x, 1^t) = (\text{Enc}^{in} (x_1, 1^t), \cdots, \text{Enc}^{in} (x_n, 1^t))$. We denote $\hat{n}_1 = \hat{n} (1, t)$, and $n' (n, t_{in}) = (\hat{n}_1 + 1) \cdot t_{in}$.

- $E^{\text{FRRTV}} = (\text{Enc}^{\text{FRRTV}}, \text{Dec}^{\text{FRRTV}})$, where for every $x \in F^n$,
  
  $\text{Enc}^{\text{FRRTV}} (x, 1^t, 1^{t_{in}}) = \left( \text{Enc}^{in} (x, 1^t), \text{Enc}^{in} (\rho (n, t_{in}), 1^t) \right)$, and
  
  $\text{Dec}^{\text{FRRTV}} ((x, m), 1^t, 1^{t_{in}}) = \text{Dec}^{in} (x, 1^t)$, where $x \in F^{n(n,t)}$.

- $\text{Comp}^{\text{FRRTV}}$ on input a circuit $C : F^n \to F^k$ containing $+, -, \times, \text{id}, \text{copy}, \text{rand}$ and $\text{const}_\alpha$ gates, outputs a circuit $C^{\text{FRRTV}}$, in which every gate is replaced with the corresponding gadget, as described in Construction 3.10, and gadgets corresponding to output gates are followed by decoding sub-circuits (computing $\text{Dec}^{in}$). The input $(x, m) \in \text{supp} (\text{Enc}^{\text{FRRTV}} (x, 1^t, 1^{|C|}))$ of $C^{\text{FRRTV}}$ is interpreted as an encoding $x$ of some input $x$ for $C$ (which, if $C$ contains $\text{rand}$ gates, includes also the random inputs to $C$), and a collection $m$ of masking inputs for the gadgets of $C^{\text{FRRTV}}$. The masking inputs used in the gadgets are taken from $m$ (every masking input in $m$ is used at most once).

The following is implicit in [FRR$^+$14]. (The proof follows by combining [FRR$^+$14, Lemma 13] with Lemma 3.3.2, and adjusting the proof to the case that output gadgets are followed by decoding sub-circuits.)

**Lemma 3.3.3.** Let $\mathbb{F}$ be a finite field, $t \in \mathbb{N}$ be a security parameter, $n \in \mathbb{N}$ be an input length parameter, $S(n) : \mathbb{N} \to \mathbb{N}$ be a size function, $\epsilon(t) : \mathbb{N} \to \mathbb{R}^+$ be a statistical distance parameter, $\mathcal{L}_E, \mathcal{L}$ be function families, and $E^{in} = (\text{Enc}^{in}, \text{Dec}^{in})$ be an encoding scheme which on inputs of length $n$, and given security parameter $t$, outputs encodings of length $\hat{n}(n, t)$. If $E^{in}$ is linear-structured over $\mathbb{F}$, and $(\mathcal{L}_E, \epsilon(t))$-leakage-indistinguishable, and $\mathcal{L} \circ \text{Shallow}_R (3, O (\hat{n}^2 (1, t))) \subseteq \mathcal{L}_E$, then Construction 3.11 is an $(\mathcal{L}, S(n), \epsilon(t) \cdot (\hat{n} (1, t) + 1) \cdot S(n))$-LRCC.
3.4 SAT-Respecting Relaxed LRCCs Over Non-Binary Fields

In this section we construct SAT-respecting relaxed LRCCs over any finite field \( F \neq \mathbb{F}_2 \), which will be used in Section 3.5 to construct a boolean SAT-respecting relaxed LRCC (which, in turn, are used to construct WIPCPs and CZKPCPs).

Our starting point is the circuit-compiler of Faust et al. [FRR+14], which we denote by \( (\text{Comp}^{\text{FRRTV}}, E^{\text{FRRTV}}) \). They present a general circuit-compiler that guarantees correctness, and a stronger notion of leakage-resilience (informally, that the wire values of the compiled circuit can be efficiently simulated). However, the correctness of their construction relies on the assumption that the inputs to the compiled circuit are honestly encoded. Therefore, their construction is not SAT-respecting, since by using ill-formed encoded inputs one can cause the compiled circuit to output arbitrary values, even if other than that the compiler was honestly applied to the original circuit. We describe a method of generalizing their construction such that the circuit-compiler is also SAT-respecting. We first give a high-level overview of the compiler of [FRR+14]. (See Section 3.3 for a more detailed description of this LRCC.)

Let \( C : \mathbb{F}^n \to \mathbb{F} \) be the circuit to be compiled. In the following, let \( r = r(\sigma) \) denote the number of masking inputs used in a circuit compiled according to \( \text{Comp}^{\text{FRRTV}} \). At a high-level, our compiler, given a circuit \( C \), generates two copies \( C_1, C_2 \) of \( C \) (that operate on two copies of the inputs); compiles \( C_1, C_2 \) into circuits \( \hat{C}_1, \hat{C}_2 \) using \( \text{Comp}^{\text{FRRTV}} \); generates the circuit \( \hat{C}' \) that outputs the AND of \( \hat{C}_1, \hat{C}_2 \); generates a circuit \( T_0 \) verifying that at least one of the copies \( \hat{C}_1, \hat{C}_2 \) uses well-formed masking inputs (i.e., its masking inputs are well-formed vectors); compiles \( T_0 \) into \( \hat{T}_0 \) using \( \text{Comp}^{\text{FRRTV}} \); and finally verifies “in the clear” that \( \hat{T}_0 \) uses well-formed masking inputs. We now describe these ingredients in more detail.

Our first ingredient checks the validity of the masking inputs used in the compiled circuit \( \hat{C}' \). If \( m_1, m_2 \) are masking inputs used in the first and second copies \( \hat{C}_1, \hat{C}_2 \) in \( \hat{C}' \), respectively (i.e., these copies are given encodings of \( m_1, m_2 \)), then we compute \( v_{ij} = m_1^i \times m_2^j \) for every \( i, j \in [r] \), and check that all the \( v_{ij} \)'s are zero. To make this check easier, we will use the following “binarization” sub-circuit, which outputs 1 if its input is 0, and outputs 0 on all other input values.

**Construction 3.12.** The “binarization” sub-circuit \( T : \mathbb{F} \to \mathbb{F} \) is defined as \( T(z) = -\prod_{0 \neq a \in \mathbb{F}} (z - a) \), computed using \( O(|\mathbb{F}|) \) many \( \times \) and constant gates arranged in \( O(\log |\mathbb{F}|) \) layers.

**Observation 3.4.1.** \( T(0) = 1 \), and for every \( 0 \neq z \in \mathbb{F} \), \( T(z) = 0 \).

The “oblivious mask-checking sub-circuit” \( T_0 \) described next checks the masking inputs \( m_1, m_2 \) used in the copies of \( \hat{C} \), and outputs 1 if and only if at least one of \( m_1, m_2 \) is the all-zero string. It computes all products of the form \( m_1^i \times m_2^j \), then applies \( T \) to every product, and computes the products of all these outputs.
Construction 3.13. The oblivious mask-checking sub-circuit $T_0 : \mathbb{F}^r \times \mathbb{F}^r \rightarrow \mathbb{F}$ is defined as follows. $T_0 (y, z) = \prod_{i,j \in [r]} T (y_i \times z_j)$, computed using a multiplication tree of size $O (r)$ and depth $O (\log r)$ (on top of the multiplication trees used to compute $T$).

Observation 3.4.2. Since the outputs of $T$ are in $\{0, 1\}$, $T_0 (y, z) = 1$ if and only if for every $i, j \in [r]$, $T (y_i, z_j) = 1$ (which by Observation 3.4.1 happens if and only if $y_i \times z_j = 0$), otherwise it outputs 0.

Our final ingredient is a non-oblivious mask-checking sub-circuit $T_V$ checking the masking inputs used in the compiled sub-circuit $\hat{T}_0$. At a high level, $T_V$ decodes every masking input; uses $T$ to map the decoded values into $\{0, 1\}$ such that only 0 is mapped to 1; and multiplies all these values, to verify that all the masking inputs are well-formed. In the following, $r_0 = r_0 (\sigma)$ denotes the number of masking inputs used in $\hat{T}_0$.

Construction 3.14. Let $n \in \mathbb{N}$ be an input length parameter, $\sigma, \kappa \in \mathbb{N}$ be security parameters, $\hat{n} = \hat{n} (n + \kappa, \sigma)$, and $\{d^\hat{n}\}$ be the decoding vectors of $E^{\hat{n}}$. We define the decoding sub-circuit $D_V : \mathbb{F}^{\hat{n}} \rightarrow \mathbb{F}$ corresponding to $d^\hat{n}$ as follows: $D_V (v) = (d^\hat{n}, v)$, where $(\cdot, \cdot)$ denotes inner-product. $D_V$ is computed using any correct decoding circuit with $O (\hat{n})$ gates arranged in $O (\log \hat{n})$ layers.

We define the non-oblivious mask-checking sub-circuit $T_V : (\mathbb{F}^{\hat{n}})^{t_0} \rightarrow \mathbb{F}$ as follows: for $R = (r_1, \ldots, r_{t_0})$ where $r_i \in \mathbb{F}^{\hat{n}}$ for every $1 \leq i \leq t_0$, $T_V (R) = \prod_{i \in [t_0]} T (D_V (r_i))$. $T_V$ is computed using $O (t_0)$ many $\times$ gates, arranged in a tree of depth $O (\log t_0)$ (on top of the sub-circuits $T \circ D_V$).

Observation 3.4.3. Let $R = (r_1, \ldots, r_{t_0}) \in (\mathbb{F}^{\hat{n}})^{t_0}$, then for every $i \in [t_0]$, $D_V (r_i) = v_i$, where $v_i$ is the value that $r_i$ encodes. Since the outputs of $T$ are in $\{0, 1\}$, $T (D_V (r_i)) = 1$ if and only if $v_i = 0$, so $T_V = 1$ if and only if all $r_i$’s are well-formed, otherwise it outputs 0.

Our circuit-compiler (Construction 3.15) uses the ingredients described above. Comp first compiles 2 copies of $C$, i.e. $C_1, C_2$, and $T_0$, into $\hat{C}_1, \hat{C}_2, \hat{T}_0$ (respectively), using Comp$^{\text{FRRTV}}$. Then, it generates a flag bit indicating whether $\hat{C}_1, \hat{C}_2$ have the same output, and the masking inputs used in $\hat{C}_1, \hat{C}_2, \hat{T}_0$ are well-formed. If so, the output is that of $\hat{C}_1$, otherwise it is 1. (Recall that an arithmetic circuit is satisfied iff its output is 0.) The encodings scheme generates encoded inputs for both copies $\hat{C}_1, \hat{C}_2$, as well as sufficient masking inputs to be used in $\hat{C}_1, \hat{C}_2, \hat{T}_0$.

Construction 3.15. Let $t, t_{in} \in \mathbb{N}$ be security parameters, and $r = r (t, t_{in})$, $r_0 = r_0 (t, t_{in}) : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ be parameters whose value will be set later. The $(L, S (n), \epsilon (n))$-relaxed LRCC over $\mathbb{F}$, (Comp, $E = (\text{Enc}, \text{Dec})$), is defined as follows.

Let $n \in \mathbb{N}$ be an input length parameter, and $E^{\hat{n}} = (\text{Enc}^{\hat{n}}, \text{Dec}^{\hat{n}})$ be a linear-structured encoding scheme over $\mathbb{F}$, with encodings of length $\hat{n}_{in} = \hat{n}_{in} (n, t)$, and decoding vectors $\{d^{\hat{n}_{in}}\}$. Then $\text{Enc} (x, 1^t, 1^{\hat{n}_{in}}) = (\hat{x}_1, \hat{x}_2)$, where $\hat{x}_i \leftarrow$
Let $\hat{\text{Comp}}_{\text{FRRTV}}$ denote the circuit compiler of \cite{FRRW14}. \text{Comp} on input a circuit $C : \mathbb{F}^n \rightarrow \mathbb{F}$, outputs the circuit $\hat{C} : \mathbb{F}^{\hat{n}(n,t,|C|)} \rightarrow \mathbb{F}$ defined as follows.

- Let $C_1, C_2$ be two copies of $C$, $\hat{C}_i = \text{Comp}^{\text{FRRTV}}(C_i)$ for $i = 1, 2$, and $\hat{T}_0 = \text{Comp}^{\text{FRRTV}}(T)$. \\
- Let $f((\hat{x}_1^{in}, R_1, R_1^0), (\hat{x}_2^{in}, R_2, R_2^0)) := \mathcal{T}\left(\hat{C}_1(\hat{x}_1^{in}, R_1) - \hat{C}_2(\hat{x}_2^{in}, R_2)\right) \times \hat{T}_0(R_1, R_2, R_1^0) \times \mathcal{T}_V(R_2^0)$. ($f = 1$ if $\hat{C}_1, \hat{C}_2$ have the same output, and in addition the masking inputs used in $\hat{T}_0$, and at least one of $\hat{C}_1, \hat{C}_2$, are well-formed. Otherwise, $f = 0$.) Then:

$$\hat{C}\left(\left((\hat{x}_1^{in}, R_1, R_1^0), (\hat{x}_2^{in}, R_2, R_2^0)\right)\right) = 1 - f((\hat{x}_1^{in}, R_1, R_1^0), (\hat{x}_2^{in}, R_2, R_2^0)))$$

$+ f((\hat{x}_1^{in}, R_1, R_1^0), (\hat{x}_2^{in}, R_2, R_2^0)) \cdot \hat{C}_1(\hat{x}_1^{in}, R_1, R_1^0)$

(Notice that the output is $\hat{C}_1(\hat{x}_1^{in}, R_1, R_1^0)$ if $f = 1$, otherwise it is 1.)

Let $r^{\text{FRRTV}}(t)$ denote the maximal number of masking inputs used in a gadget of $\text{Comp}^{\text{FRRTV}}$, and $S_0(r)$ denote the size of $\hat{T}_0$ when applied to inputs of length $r$. Then $r(t, t_{in}) = t_{in} \cdot r^{\text{FRRTV}}(t)$ and $r_0(t, t_{in}) = r^{\text{FRRTV}} \cdot S_0(r^{\text{FRRTV}} \cdot t_{in})$.

We will show that if the underlying “inner” encoding scheme $E^{\text{in}}$ is leakage-indistinguishable against a leakage family $L_E$, then Construction 3.15 is a SAT-respecting relaxed LRCC against a slightly weaker leakage family $L$. Formally,

**Proposition 3.4.4** (SAT-respecting relaxed LRCC over $\mathbb{F}$). Let $n \in \mathbb{N}$ be an input length parameter, $t \in \mathbb{N}$ be a security parameter, $L, L_E$ be families of functions, $S(n) : \mathbb{N} \rightarrow \mathbb{N}$ be a size function, and $\epsilon(t) : \mathbb{N} \rightarrow \mathbb{R}^+$ be a statistical distance parameter. Let $E^{\text{in}} = (\text{Enc}^{\text{in}}, \text{Dec}^{\text{in}})$ be a linear-structured, $(L_E, \epsilon(t))$-leakage-indistinguishable encoding scheme with input length parameter $n = 1$, security parameter $t$ and $\hat{n} = \hat{n}(t)$, such that $L_E = L \circ \text{Shallow}(7, O(\hat{n}^4(t) \cdot S(n)))$. Then there exists a SAT-respecting, $(L, S(n), 8\epsilon(t) \cdot S(n))$-relaxed-LRCC over $\mathbb{F}$. Moreover, For every $C : \mathbb{F}^n \rightarrow \mathbb{F}$, the compiled circuit $\hat{C}$ has size $|\hat{C}| = O\left(|\mathbb{F}| \cdot \hat{n}^4(t) \cdot |C|^2\right)$.

### 3.4.1 Roadmap Towards Proving Proposition 3.4.4

We will show that Construction 3.15 satisfies the requirements of Proposition 3.4.4. We first analyze the SAT-respecting property, showing that if $\hat{C}$ is satisfiable, then so
is $C$. At a high level, if $C$ is not satisfiable, then one could potentially satisfy $\hat{C}$ by providing ill-formed masking inputs to one of the copies $\hat{C}_1, \hat{C}_2$, or to the oblivious masking-checking circuit $\hat{T}_0$. However, if the masking inputs of $\hat{T}_0$ are ill-formed, then $\hat{T}_V$ resets the flag, so the output is 1 (i.e., $\hat{C}$ is not satisfied). Conditioned on $\hat{T}_0$ having well-formed masking inputs, the correctness of $\text{Comp}^{FRRTV}$ (applied to $\hat{T}_0$), guarantees that if the masking inputs of both $\hat{C}_1, \hat{C}_2$ are ill-formed then the flag is reset. Finally, if at least one of $\hat{C}_1, \hat{C}_2$ has well-formed masking inputs, and $\hat{C}$ is satisfied (in particular, the flag is not reset), then there exists an $x \in \mathbb{F}^n$ that satisfies the correctly evaluated copy, and therefore also satisfies $C$. We note that the encoding scheme should be onto (which is the case for any linear-structured encoding scheme), otherwise computations in compiled circuits may not correspond to computations in the original circuits (since the “encoded” input may not correspond to a valid input for the original circuit). This intuition is formalized in the following lemma.

**Lemma 3.4.5.** If $E$ is linear-structured then Construction 3.15 is SAT-respecting. That is, if $\hat{C}(\hat{x}) = 0$ for some $\hat{x} \in \mathbb{F}^n$, then $C(x) = 0$ for some $x \in \mathbb{F}^n$.

**Proof.** Assume that $\hat{C}(\hat{x}) = 0$ for some $\hat{x} \in \mathbb{F}^n$, and denote $\hat{x} = ((\hat{x}_1^1, R_1, R_1^0), (\hat{x}_2^2, R_2, R_2^2))$. Then $f(\hat{x}) = 1$ and $\hat{C}_1(\hat{x}_1^1; R_1) = 0$ by the definition of $\hat{C}$. Therefore, $\hat{C}_1, \hat{C}_2$ have the same output, and $\hat{T}_V, \hat{T}_0$ output 1. Consequently, according to Observation 3.4.3, $R_1^0$ is well-formed, so by the correctness of $\text{Comp}^{FRRTV}$, $\hat{T}_0$ emulates $T_0$. (Here, we also use the fact that $\hat{T}_V$ is independent of all other components of, and inputs to, $\hat{C}$.) Moreover, since the encoding scheme is onto then $R_1, R_2$ define inputs to $\hat{T}_0$, on which $\hat{T}_0$ outputs 1 (because $\hat{T}_0$ outputs 1). By observation 3.4.2, at least one of $R_1, R_2$ is well-formed. Assuming (without loss of generality) that $R_1$ is well-formed, then the correctness of $\text{Comp}^{FRRTV}$ guarantees that $\hat{C}_1$ emulates $C$, so $0 = \hat{C}_1(\hat{x}_1^1; R_1) = C(x)$, where $x = (x_1, \ldots, x_n) \in \mathbb{F}^n$ is $x = \text{Dec}^{in}(\hat{x}_1^1)$ ($x$ is well-defined because $E^{in}$ is onto).

Next, we analyse the relaxed leakage-resilience property of Construction 3.15, describing a simulator $\text{Sim}$ that, given a circuit $C$, and its output on some input $x$, generates simulated wire values for $C$, such that leakage functions cannot distinguish the simulated wires from the actual wire values of $C$. At a high level, the simulator operates as follows. On input $C : \mathbb{F}^n \to \mathbb{F}$, and $C(x)$ for $x \in \mathbb{F}^n$, $\text{Sim}$ finds a $y \in \mathbb{F}^n$ such that $C(y) = C(x)$ (this is the reason that $\text{Sim}$ is unbounded); generates $\hat{C} = \text{Comp}(C)$ and $\hat{y} \leftarrow \text{Enc}(y, 1^{|C|})$; honestly evaluates $\hat{C}$ on $\hat{y}$; and outputs the wire values of $\hat{C}$. We show that if $E^{in}$ is leakage-indistinguishable for a leakage class which is “somewhat stronger” than $\mathcal{L}$, then for every $\ell \in \mathcal{L}$, $\text{SD}\left(\ell\left[\hat{C}, \hat{x}\right], \ell\left[\hat{C}, \hat{y}\right]\right) \leq \epsilon(t)$, where $\hat{x} \leftarrow \text{Enc}(x, 1^t, 1^{|C|})$. Informally, this follows from a hybrid argument, where we first replace the input of $\hat{C}_1$ from $\hat{x}$ to $\hat{y}$, and then do the same for $\hat{C}_2$. (This is also the reason that we do not explicitly verify that $\hat{C}_1, \hat{C}_2$ are evaluated on encodings of the same input.)
To show that each adjacent pair of hybrids is leakage-indistinguishable, we first use an argument similar to that of [FRR+14], where we first replace the bundles of \( \hat{C}_1 \) or \( \hat{C}_2 \) (depending on the pair of hybrids in question) that are external to the gadgets (i.e., bundles that correspond to wires of the original circuit \( C \)) with random encoding of the "correct" values; and then replace the bundles internal to the gadgets of \( \hat{C}_1 \) (or \( \hat{C}_2 \)) with simulated values. However, our compiled circuit \( \hat{C} \) consists also of \( \hat{T}_0, \hat{T}_V \), so the analysis in our case is more complex, and in particular we cannot use the leakage-resilience analysis of [FRR+14] as a black box. To explain the difficulty in generating these wires values, we need to take a closer look at their leakage-resilience analysis.

Recall that the leakage-indistinguishability proof for every pair of adjacent hybrids contains in itself two series of hybrid arguments, one replacing external bundles, and the other replacing internal bundles. In the first case, leakage-indistinguishability is reduced to that of the underlying encoding scheme \( E \), whereas in the second it is reduced to the leakage-indistinguishability of the actual and simulated wire values of a single gadget. Specifically, the leakage function \( \ell \) in the reduction is given either an encoding of a single field element, or the wire values of a single gadget; uses its input to generate all the wire values of the compiled circuit; and then evaluates \( \ell \) on these wire values. Thus, if originally we could withstand leakage from some function class \( L^\text{in} \), and the additional wires can be generated by a function class \( L_R \), then after the reduction we can withstand leakage from any function class \( L \) such that \( L \circ L_R \subseteq L^\text{in} \). In particular, if \( L^\text{in} \) consists of functions computable by low-depth circuits, and computing the internal wires of \( \hat{T}_0, \hat{T}_V \) require deep circuits (consequently, \( L_R \) necessarily contains functions whose computation requires deep circuits), then we have no leakage-resilience. To overcome this, we show how to simulate these additional wires (namely, the wires of \( \hat{T}_0, \hat{T}_V \)) using shallow circuits. This is possible because (due to the way in which the hybrids are defined) the masking inputs in at least one copy are well-formed. Specifically, the structure of \( \hat{T}_0, \hat{T}_V \) guarantees that conditioned on the masking inputs of \( \hat{C}_2 \) being well-formed, these wire values can be computed by shallow circuits. When the masking inputs of \( \hat{C}_2 \) are ill-formed, we are guaranteed that the masking inputs of \( \hat{C}_1 \) are well-formed. Conditioned on this event, we show an alternative method of computing the internal wires of \( \hat{T}_0, \hat{T}_V \), which can be done by shallow circuits.

### 3.4.2 The Relaxed Leakage-Resilience Property of Construction 3.15

We now show that Construction 3.15 is relaxed leakage-resilient. Let \( S(n) : \mathbb{N} \to \mathbb{N} \) be a size function. Let \( E^\text{in} = (\text{Enc}^\text{in}, \text{Dec}^\text{in}) \) be a linear-structured encoding scheme which, for input length parameter \( n = 1 \), and security parameters \( t \), outputs encodings of length \( \hat{n}_1(t) := \hat{n}(1, t) \). Assume that \( E^\text{in} \) is \( (L_{E^\text{in}}, \epsilon_{E^\text{in}}(t)) \)-leakage-indistinguishable for some family \( L_{E^\text{in}} \) of leakage functions, and some statistical distance function \( \epsilon_{E^\text{in}}(t) : \mathbb{N} \to \mathbb{R}^+ \), and let \( L \) be a family of functions such that \( L_{E^\text{in}} = L \circ \text{Shallow}(\hat{T}, O(\hat{n}_1(t) \cdot S(n))) \). We will show that for an appropriate choice of \( \epsilon'(t) > 0 \), Construction 3.15 is an
(Ł, S (n), t’ (t))-relaxed LRCC. Towards that end, let $E = (\text{Enc, Dec})$ denote the encoding scheme obtained from $E^n$ in Construction 3.15, and $C : \mathbb{F}^n \rightarrow \mathbb{F}$ be a circuit of size $|C| \leq S (n)$. Let $L \in \mathcal{L}$ be of input length $|\hat{C}|$, and $x \in \mathbb{F}^n$. Recall that the simulator Sim is given $C$ and $C (x)$, finds a $y \in \mathbb{F}^n$ such that $C (x) = C (y)$, generates $\hat{C} = \text{Comp} (C)$, evaluates $\hat{C}$ on an honestly-generated encoding of $y$, and outputs the wire values of $\hat{C}$. Let $\hat{x} \leftarrow \text{Enc} (x, 1^t, 1^{|C|})$, and $\hat{y} \leftarrow \text{Enc} (y, 1^t, 1^{|C|})$, such that $C (x) = C (y) = 0$, where $\hat{x} = ((\hat{x}_1, R_{1i}, R_{2i}) \mid (\hat{x}_2, R_{2i}, R_{2f}))$, and $\hat{y} = ((\hat{y}_1, R_{1i}, R_{2i}) \mid (\hat{y}_2, R_{2i}, R_{2f}))$. We need to bound $\text{SD} (\ell [\hat{C}, \hat{x}] \mid \ell [\hat{C}, \hat{y}]$, which we do using a sequence of hybrids, in which we first replace the input of the first copy $\hat{C}_1$ from $\hat{x}$ to $\hat{y}$, then do the same in the second copy $\hat{C}_2$. Thus, we use the following hybrids:

$$H^x := \left( [\hat{C}_1, (\hat{x}_1; R_{1i}), [\hat{C}_2, (\hat{x}_2; R_{2i})], [\hat{T}_0, ((R_{1i}, R_{2i}); R_{2i})], [\hat{T}_V (R_{1i})] \right)$$

$$H^{y,x} := \left( [\hat{C}_1, (\hat{y}_1; R_{1i}), [\hat{C}_2, (\hat{x}_2; R_{2i})], [\hat{T}_0, ((R_{1i}, R_{2i}); R_{2i})], [\hat{T}_V (R_{1i})] \right)$$

$$H^{y} := \left( [\hat{C}_1, (\hat{y}_1; R_{1i}), [\hat{C}_2, (\hat{y}_2; R_{2i})], [\hat{T}_0, ((R_{1i}, R_{2i}); R_{2i})], [\hat{T}_V (R_{1i})] \right)$$

Next, we show that both $\ell (H^x), \ell (H^y)$ are close to the intermediate distribution $\ell (H^{y,x})$. As noted above, to do so we use additional hybrids, in which we first replace the values of bundles external to gadgets from their distribution according to either $\ell (H^x)$ or $\ell (H^y)$, to random encodings of the “correct” values, and then replace the bundles internal to the gadgets with simulated bundle values.

Bounding $\text{SD} (\ell (H^x) \mid \ell (H^{y,x}))$

We use the intermediate distributions $H^x_{\text{ext}}, H^x_{\text{mid}}$ in which the external and internal bundles (respectively) are replaced with simulated bundles. More specifically, $H^x_{\text{ext}}$ is the hybrid distribution obtained by evaluating $\hat{C}$ honestly on $\hat{x} \leftarrow \text{Enc} (x, 1^t, 1^{|C|})$; picking local reconstructors for all gadgets of $\hat{C}_1$ (see Lemma 3.3.2), and re-computing their internal wires using the reconstructors; and re-evaluating $\hat{T}_0$ on the new masking inputs generated for the gadgets of $\hat{C}_1$. $H^x_{\text{mid}}$ represents the following mental experiment. Unlike the actual simulation, $x$ is given as input to Sim, but Sim uses it only in the second copy $\hat{C}_2$, i.e., it generates the wire values of $\hat{C}$ as follows.

- **Generating the wires of $\hat{C}_2$:** Sim generates an encoding $\hat{x} = ((\hat{x}_1, R_{1i}, R_{2i}) \mid (\hat{x}_2, R_{2i}, R_{2f})) \leftarrow \text{Enc} (x, 1^t, 1^{|C|})$ of $x$, and honestly evaluates $\hat{C}_2$ on $\hat{x}_2$ with masking inputs $R_2$.

- **Generating the wires of $\hat{C}_1$:** Sim picks a random encoding $O \leftarrow \text{Enc}^{\text{in}} (1, 1^t)$ for the output of $\hat{C}_1$, and honestly computes the wires of the output decoder. Then, Sim picks $z \in \mathbb{F}^n$ and generates $\hat{z}_1 \leftarrow \text{Enc}^{\text{in}} (z, 1^t)$, and picks random

---

4Since re-evaluating $\hat{T}_0$ does not influence its masking inputs, this does not influence the computation in $\hat{T}_V$. 

50
encodings (according to \(Enc^{\text{in}}\)) for the outputs of all gadgets (except the gadgets whose outputs “touch” the output decoder, since the outputs of these gadgets have already been determined), which effectively determines the standard inputs, and outputs, of all gadgets in \(\hat{C}_1\). Next, \(Sim\) picks a local reconstructor for every gadget of \(\hat{C}_1\), and uses the reconstructors to compute the internal wires of the gadgets in \(\hat{C}_1\). The reconstructors determine the (possibly ill-formed) masking inputs of the gadgets, which (together with \(R_2\)) form the standard inputs of \(\hat{T}_0\).

- **Generating the wires of \(\hat{T}_0\):** \(Sim\) honestly evaluates \(\hat{T}_0\) on \(R_1, R_2\), with masking inputs \(R_1^0\).
- **Generating the wires of \(T_V\):** \(Sim\) honestly evaluates \(T_V\) on \(R_1^0\).
- Finally, \(Sim\) uses the outputs of \(\hat{C}_1, \hat{C}_2, \hat{T}_0, T_V\) to generate the flag \(f\), and the output of \(\hat{C}\).

We need the following result [FRR\(^+\)14, Lemma 8] regarding preservation of leakage-resilience under computation.

**Lemma 3.4.6** ([FRR\(^+\)14]). Let \(n \in \mathbb{N}\), \(W_0, W'_0\) be distributions over \(\mathbb{F}^n\), \(L, L_0\) be families of functions, and \(\epsilon > 0\). Let \(F\) be a distribution over \(n\)-input functions in \(L\). For \(f \leftarrow F\), let \(W_1 := f(W_0), W'_1 := f(W'_0)\). If \(W_0, W'_0\) are \((L_0, \epsilon)\)-leakage-indistinguishable, then \(W_1, W'_1\) are \((L_1, \epsilon)\)-leakage-indistinguishable for any family \(L_1\) of leakage functions such that \(L_1 \circ L \subseteq L_0\).

The following notation is used to identify gadgets of \(\hat{T}_0\) (resp. \(T_V\)) that are correlated to gadgets of \(\hat{C}_1, \hat{C}_2\) (resp. \(\hat{T}_0\)).

**Notation 3.16.** We say that a gadget \(G_i\) in \(\hat{C}_i\) (for \(i = 1, 2\)) and an \(\times\) gadget \(G_0\) in the first layer of \(\hat{T}_0\) (i.e., gadgets corresponding to \(\times\) gates that are evaluated before \(T\) is called) are connected, if a masking input (i.e., an encoding of that masking input) of \(G_i\) is one of the inputs to \(G_0\). Similarly, we say that a decoding sub-circuit \(D_V\) (in \(T_V\)) and a gadget \(G_0\) of \(\hat{T}_0\) are connected, if the input to \(D_V\) is a masking input of \(G_0\).

In Lemma 3.4.7 below, we bound the statistical distance between \(\ell(H^x)\) and \(\ell(H_{\text{ext}}^x)\). In Lemma 3.4.9, we bound the statistical distance between \(\ell(H_{\text{ext}}^x)\) and \(\ell(H_{\text{mid}}^x)\). (The proofs of both of these lemmas use an additional sequence of hybrids.) Then, in Remark 3.17, we show that \(\ell(H_{\text{mid}}^x)\) and \(\ell(H_{\text{ext}}^y)\) are statistically close, so \(\ell(H^x)\) is statistically close to \(\ell(H_{\text{ext}}^y)\). The proofs of these lemmas rely on the fact that the inputs used in the second copy \(\hat{C}_2\) are well formed.

**Lemma 3.4.7.** Let \(t \in \mathbb{N}\) be a security parameter, \(n \in \mathbb{N}\) be an input length parameter, \(L_G, L\) be families of functions, \(S(n) : \mathbb{N} \to \mathbb{N}\) be a size function, and \(\epsilon(t) : \mathbb{N} \to \mathbb{R}^+\) be a statistical distance parameter. If every gadget \(G\) of \(\hat{C}_1\) is \((L_G, \epsilon(t))\)-reconstructible, and \(L_G = L \circ \text{Shallow}((|G|, 2, O(n^4(1,t) \cdot S(n))))\), then \(\text{SD}(\ell(H^x), \ell(H_{\text{ext}}^x)) \leq \epsilon(t) \cdot S(n)\) for every \(\ell \in L\).
Proof. Let $M \leq S(n)$ denote the number of gadgets in $\hat{C}_1$, then we define a fixed ordering on these gadgets, and a sequence $H_0, \ldots, H_M$ of hybrids, where in $H_i$, $\hat{C}$ is honestly evaluated with input $x$, and then the internal wires of the first $i$ gadgets of $\hat{C}_1$ are recomputed using the gadget reconstructors, and the wires of $\hat{C}_0$ influenced by these computations are also recomputed. Then $H_0 = H^x, H_M = H^x_{\text{ext}}$.

If $\text{SD} (\ell(H^x), \ell(H^x_{\text{ext}})) > \epsilon(t) \cdot S(n)$ for some $\ell \in \mathcal{L}$ then $\text{SD} (\ell(H_m), \ell(H_{m-1})) > \epsilon(t)$ for some $m \in [M]$. Denote the $m$'th gadget by $G$, then $G$ is necessarily a $\times$ gadget. (Indeed, conditioned on their inputs and output, Lemma 3.3.2 guarantees that the internal wires of the reconstructors of all other gadgets are distributed identically to the internal wires in an honest evaluation of the gadget.)

Using an averaging argument, Lemma 3.4.8 below (which we can use because $\hat{C}_2, \hat{T}_0$ are honestly evaluated), and the fact that the masking inputs of $G$ are used (as standard inputs) only in $G_0$ gadgets (in $\hat{T}_0$) connected to $G$, we can fix all the wires in $H_m, H_{m-1}$ except for the masking inputs, and internal wires, of $G$; and the wires of $B_0, U_0$, and the internal wires in the computation of $B_0, U_0, q_0$, in every gadget $G_0$ (in $\hat{T}_0$) connected to $G$ (the subscript “0” here is used to denote wires internal to $G_0$). (Notice that the inputs to $\mathcal{T}_V$ are the columns of the masking inputs of every such $G_0$. As these masking inputs are fixed, then the entire computation in $\mathcal{T}_V$ can also be fixed.)

Let $W_0^R (W_0^S)$ denote the real-world (reconstructed) wire assignment to the internals of $G$. We construct a pair of distributions $W_1^R, W_1^S$, computable from $W_0^R, W_0^S$ by a function $f \in \text{Shallow} (|G|, 2, O(\hat{n}^4(1, t) \cdot S(n)))$. Given the (either real or reconstructed) internals of $G$ (with the inputs $a, b$, and the output $c$, that were hard-wired into $H_m, H_{m-1}$), $f$ “fills in the holes” in the wire assignment of $\hat{C}$, i.e., uses its input to honestly evaluate the missing wires in $\hat{T}_0$. Then $f \in \text{Shallow} (|G|, 2, O(\hat{n}^4(1, t) \cdot S(n)))$ because by Lemma 3.4.8, every gadget $G_0$ in $\hat{T}_0$ connected to $G$ can be evaluated in $\text{Shallow} (2, O(\hat{n}^2(1, t)))$ (recall that $f$ need only compute the missing wires $B_0, U_0$, and the internal values in the computation of $B_0, U_0$ and the (fixed) $q_0$); there are at most $O(\hat{n}^2(1, t) \cdot S(n))$ such gadgets $G_0$; and we can evaluate them in parallel.

By the lemma’s assumptions, the $\times$ gadget is $(\mathcal{L}_G, \epsilon(t))$-reconstructible, so $W_0^R, W_0^S$ are $(\mathcal{L}_G, \epsilon(t))$-leakage-indistinguishable. Using Lemma 3.4.6 (with the distribution $F$ that always returns the function $f$), $W_1^R, W_1^S$ are $(\mathcal{L}, \epsilon(n))$-leakage-indistinguishable for every family $\mathcal{L}$ of leakage functions such that $\mathcal{L} \circ \text{Shallow} (|G|, 2, O(\hat{n}^4(1, t) \cdot S(n))) \subseteq \mathcal{L}_G$. In particular,

$$\text{SD} (\ell(H_m), \ell(H_{m-1})) = \text{SD} (\ell(W_1^S), \ell(W_1^R)) \leq \epsilon(t).$$

The proof of Lemma 3.4.7 used the following fact: the internal wires of every $\times$ gadget $G_0$ in $\hat{T}_0$ that is connected to gadgets of $\hat{C}_1, \hat{C}_2$, are computable (from the inputs and output of $G_0$) by a shallow circuit, provided that its second input is well-formed. Formally,

**Lemma 3.4.8.** Let $G_0$ be a gadget of $\hat{T}_0$ connected to gadgets of $\hat{C}_1$ and $\hat{C}_2$, with inputs
Let $\hat{a}_0, \hat{b}_0$, and let $B_0, S_0, U_0, q_0, r_0, c_0$ denote the internal wires of $G_0$. (See Construction 3.10, Section 3.3 for the definition of gadgets.) If $b_0$ and the masking inputs of $G_0$ are well-formed, then $q_0, c_0$ are well-formed, and independent of $a_0, b_0$. Moreover, if $b_0, S_0$ are fixed, then given $q_0, c_0, a_0, r_0$, the internal and output wires of $G_0$ are computable in Shallow $(2, O(\hat{n}^2 (1, t)))$.

Proof. Let $\hat{n}_1 := \hat{n} (1, t)$. Let $c_0$ denote the output of $G_0$, and let $B_0, U_0, S_0, q_0, r_0 := r^{\hat{n}_1+1}$ denote its internal wires. Since $B_0$ encodes 0, then $q_0$ is independent of $a_0, b_0$.

Since every column of $S_0$ is well-formed and independent of $a_0, b_0$ then $q_0$ is also well-formed. Finally, since $r_0$ is well-formed and independent of $a_0, b_0$, then so is $c_0$.

Next, we show that for fixed $b_0, S_0$, and given $a_0, q_0, c_0$, the (remaining) internals of $G_0$ are computable in Shallow $(2, O(\hat{n}^2 (t)))$. Notice that we only need to reconstruct $B_0, U_0$ and the internal wires in the computation of $B_0, U_0, q_0$. $B_0$ is computable in a single layer using $\hat{n}_1 (t)$ constant gates (for the coordinates of $b_0$), $\hat{n}_1 (t)$ input gates (for the coordinates of $a_0$), and $\hat{n}_1^3 (t) \times$ gates (computing the products $a_0, b_0, j$). Once $B_0$ is known, $U_0$ is computable in a single layer with $\hat{n}_1^2 (t)$ constant gates (for the coordinates of $S_0$) and $\hat{n}_1^2 (t) +$ gates (computing $B_{0,i} + S_{0,i}$). The internal values in the computation of every $q_0,i$ are $a_i \sum_{k=1}^{l} b_{0,k} d_{k,j} + \sum_{k=1}^{l} s_{0,i,k} d_{k,j}$, where $j = 1, \ldots, \hat{n}_1 (t)$, and $d$ is the decoding vector of $E_{in}$, so for fixed $b_0, S_0, d$, these internal wires can be computed with $O(\hat{n}_1^2 (t))$ gates organized in two layers: the first with $\hat{n}_1 (t)$ constant gates (for the values $\sum_{k=1}^{l} b_{0,k} d_{k,j}$, $j = 1, \ldots, \hat{n}_1 (t)$) and $\hat{n}_1 (t) \times$ gates (computing $a_i \sum_{k=1}^{l} b_{0,k} d_{k,j}$, $j = 1, \ldots, \hat{n}_1 (t)$), and the second with $\hat{n}_1 (t)$ constant gates (for the values $\sum_{k=1}^{l} s_{i,k} d_{k,j}$, $j = 1, \ldots, \hat{n}_1 (t)$) and $\hat{n}_1 (t) +$ gates (computing $a_i \sum_{k=1}^{l} b_{0,k} d_{k,j} + \sum_{k=1}^{l} s_{i,k} d_{k,j}$, $j = 1, \ldots, \hat{n}_1 (t)$). Since the internal wires of $c_0$ can be computed in parallel to $B_0, U_0$, the entire computation is in Shallow $(2, O(\hat{n}_1^2 (t)))$.

Lemma 3.4.9. Let $t \in \mathbb{N}$ be a security parameter, $n \in \mathbb{N}$ be an input length parameter, $L_E, L$ be families of functions, $S (n) : \mathbb{N} \to \mathbb{N}$ be a size function, and $\epsilon (t) : \mathbb{N} \to \mathbb{R}^+$ be a statistical distance parameter. If $E_{in}$ is $(L_E, \epsilon (t))$-leakage-indistinguishable, and $L_E = L \circ$ Shallow $(\hat{n} (1, t), 4, O(\hat{n}^4 (1, t) \cdot S (n)))$, then $SD (\ell (H_{mid}^x), \ell (H_{ext}^x)) \leq \epsilon (t) \cdot S (n)$ for every $\ell \in L$.

Proof. Define a fixed ordering on the set of inputs bundles of $\hat{C}_1$, and bundles at the output of gadgets in $\hat{C}_1$ that do not touch the decoder, and let $M \leq S (n)$ denote the number of such bundles. We define hybrids $H_0, \ldots, H_M$, where $H_i$ is generated by drawing an assignment to the outputs of all gadgets in $\hat{C}_1$, according to the values of the corresponding wires in $C (x)$; replacing the first $i$ bundles with random encodings of random values; computing the internal wires of the gadgets of $\hat{C}_1$ using the gadget reconstructors (see Lemma 3.3.2); and evaluating $\hat{C}_2, \hat{T}_0, T_V$ honestly (in particular, the input to $\hat{C}_2$ is a random encoding of $x$). Notice that $H_0 = H_{ext}^x$ and $H_M = H_{mid}^x$.

\footnote{Indeed, if $d$ is the decoding vector then for every $1 \leq i \leq \hat{n}_1, q_0,i = U_0,i \cdot d = a_0,i \sum_{j=1}^{n_1} b_{0,i,j} d_j + \sum_{j=0}^{n_1} s_{0,i,j} d_j = a_0,i \cdot 0 + \sum_{j=0}^{n_1} S_{0,i,j} d_j = \sum_{j=0}^{n_1} S_{0,i,j} d_j$, where $U_0,i$ denotes the $i$th row of $U_0$, and $d$ denotes the decoding vector of $E$ (the equality denoted (1) holds because $b_0$ encodes 0).}
Although (re-randomization guarantees that these equivalences hold if were fixed in advance). Moreover, Remark 3.17.

\[ C^\hat{\cdot} \]

internal wires of all gadgets in \( G \). By Lemma 3.4.6, \( W_0^R \) is the distribution of the \( m \)th bundle in \( H_{m-1} \), and \( W_0^S \) is its distribution in \( H_m \), then by the lemma’s assumption, \( W_0^R, W_0^S \) are \((\mathcal{L}, \epsilon(t))\)-leakage-indistinguishable.

Let \( W_1^R := f(W_0^R), W_0^S := f(W_1^S) \) where \( f \) is chosen according to the following distribution \( F \) over Shallow \( \hat{n}(1,t), 4, O(\hat{n}_1(t) \cdot S(n)) \). \( F \) chooses functions \( \text{rec}_o, \text{rec}_i \) according to the distribution over reconstructors for \( G_o, G_i \), respectively, and the obtained function \( f \), on input \( e \in \mathbb{F}^{\hat{n}_1} \); evaluates \( \text{rec}_o \) on the inputs \( a_o, b_o \) (that were hard-wired into \( H_m, H_{m-1} \)), and output \( e \) (thus reconstructing the masking inputs, and internal wires, of \( G_o \)); evaluates \( \text{rec}_i \) on \( e \) as one of the inputs, and the other input, and output, according to the hard-wired values (thus generating the masking inputs, and internal wires, of \( G_i \)); and for every \( G_0 \) gadget in \( \hat{C}_0 \) connected to \( G_o \) or \( G_i \), honestly re-evaluates \( G_0 \).

Notice that \( W_1^R \equiv H_{m-1} \), and \( W_1^S \equiv H_m \), because the bundles are re-randomizing (re-randomization guarantees that these equivalences hold although some of the values were fixed in advance). Moreover, \( f \in \text{Shallow} \left( \hat{n}(1,t), 4, O(\hat{n}_1^2(t) \cdot S(n)) \right) \). Indeed, by Lemma 3.3.2, \( \text{rec}_o, \text{rec}_i \) are in Shallow \( 2, O(\hat{n}_2^2(1,t)) \) (and given \( e \) they can be evaluated in parallel), and given the internals of \( G_o, G_i \), the missing wires of every \( G_0 \) connected to \( G_o \) or \( G_i \) are computable in Shallow \( 2, O(\hat{n}_2^2(1,t)) \) (see the proof of Lemma 3.4.7 for a more detailed analysis). As there are at most \( O(\hat{n}_2^2(1,t) \cdot S(n)) \) such gadgets, which can be evaluated in parallel, \( f \in \text{Shallow} \left( \hat{n}(1,t), 4, O(\hat{n}_2^2(1,t) \cdot S(n)) \right) \). By Lemma 3.4.6, \( W_1^R, W_1^S \) are \((\mathcal{L}, \epsilon(t))\)-leakage-indistinguishable, and in particular, \( \text{SD}(\ell(H_{m-1}), \ell(H_m)) \leq \epsilon(t) \).

Remark 3.17. Let \( H_{\text{ext}}^{y,x} \) denote the hybrid distribution obtained by evaluating \( \hat{C}_i \), when \( \hat{C}_1, \hat{C}_2 \) are honestly evaluated on \( y, x \), respectively (i.e., picking random encodings of \( (y, 1^t, 0^{\text{rand}(t,C)}), (x, 1^t, 0^{\text{rand}(t,C)}) \) according to \( \text{Enc}_{\text{in}} \) etc.); re-computing the internal wires of all gadgets in \( \hat{C}_1 \) using their reconstructors; and re-evaluating \( \hat{T}_0 \). Then under the conditions of Lemma 3.4.9, \( \text{SD}(\ell(H_{\text{mid}}), \ell(H_{\text{ext}}^{y,x})) \leq \epsilon(t) \cdot S(n) \). Indeed, as long as both hybrids the input to \( \hat{C}_2 \) encodes the same value, the proof was independent of the actual values whose encodings were the inputs of \( \hat{C}_1, \hat{C}_2 \) (because in \( \hat{C}_2 \) the same value is used, and in \( \hat{C}_1 \) the value is compared to a random value). The
proof of Lemma 3.4.7 also relied only on the inputs to \( \hat{C}_1, \hat{C}_2 \) encoding the same values in both distributions, and was independent of the *actual* encoded values, so the same proof can be used to show that under the conditions of Lemma 3.4.7, \( \text{SD} (\ell (H_{\text{ext}}^{y,x}), \ell (H_{\text{ext}}^{y,x})) \leq \epsilon (t) \cdot S (n) \) for every leakage function \( \ell \in \mathcal{L} \). Consequently, \( \text{SD} (\ell (H^x), \ell (H_{\text{ext}}^{y,x})) \leq 4\epsilon (t) \cdot S (n) \).

**Bounding \( \text{SD} (\ell (H_{\text{ext}}^{y,x}), \ell (H^y)) \)**

The argument in this case is somewhat more complex, because here \( \hat{C}_2 \) may not have been honestly evaluated, whereas Lemmas 3.4.7 and 3.4.9 crucially relied on the fact that the second input \( b_0 \) of every \( \times \) gadget \( G_0 \) in \( \mathcal{T}_0 \) (connected to gadgets of \( \hat{C}_1, \hat{C}_2 \)) is well-formed. In particular, when only the masking inputs of \( \hat{C}_1 \) are guaranteed to be well-formed then we cannot fix the internal wires of \( G_0 \), and consequently generating the internal wires of \( \mathcal{T}_0, \mathcal{T}_V \) may require deep circuits. Therefore, we present an alternative method of generating these wires using shallow circuits. We call these alternative methods *right-reconstructors*, to emphasize that they are used *only* when the simulator in the mental experiment simulates the internal wires of the *right* copy \( \hat{C}_2 \). More specifically, we describe (Construction 3.18) a right-reconstructor for every \( \times \) gadget in first layer of \( \mathcal{T}_0 \) (recall that these are the gadgets connected to gadgets of \( \hat{C}_1, \hat{C}_2 \)), and a right-reconstructor for every decoding sub-circuit \( \mathcal{D}_V \) of \( \mathcal{T}_V \) (Lemma 3.4.11). We note that unlike the reconstructors of Lemma 3.3.2, the right-reconstructors are only used when one of the inputs, and the output, are well formed (in the case of \( \mathcal{T}_0 \)); or if the inputs are well formed, and the output is zero (in the case of \( \mathcal{T}_V \)).

**Construction 3.18.** Let \( t \in \mathbb{N} \) be a security parameter. We define the *right reconstructor*, \( \hat{C}_0 \) as follow. Denote \( \hat{n}_1 = \hat{n} (1, t) \), then we define a set of functions \( \{ \text{rec}_{0}^{r_1, \ldots, r_{\hat{n}_1}} \} \) for \( r_1, \ldots, r_{\hat{n}_1} \in \mathbb{F}^{\hat{n}_1} \). Given standard inputs \( r_0^1, r_0^2 \), and output \( c_0 \), \( \text{rec}_{0}^{r_1, \ldots, r_{\hat{n}_1}} \) operates as follows.

1. Computes \( B_0 \leftarrow r_0^1 (r_0^2)^T = \left( r_{0,i}^1 \times r_{0,j}^2 \right)_{i,j \in \hat{n}_1} \) (using \( \hat{n}_1^2 \times \) gates).
2. Sets the columns of \( U_0 \) to \( r^1, \ldots, r_{\hat{n}_1} \).
3. Computes \( S_0 \leftarrow U_0 - B_0 \) (using \( \hat{n}_1^2 + \) gates, and \( \hat{n}_1^2 \) constant gates).
4. Computes a vector \( q_0 \), where \( q_0,i \) is the decoding of the \( i \)-th row of \( U_0 \). (Notice that \( q_0 \) depends only on \( U_0 \), which is independent of \( r_0^1, r_0^2 \).
5. Computes \( r_{\hat{n}_1+1} \leftarrow c_0 - q_0 \).
6. \( \text{rec}_{0}^{r_1, \ldots, r_{\hat{n}_1}} \) outputs \( r_0^1, r_0^2, c_0, r_1, \ldots, r_{\hat{n}_1+1}, B_0, S_0, U_0, q_0 \).

The distribution REC is defined by picking a function \( \text{rec}_{0}^{r_1, \ldots, r_{\hat{n}_1}} \) where \( r_1, \ldots, r_{\hat{n}_1} \) are chosen uniformly at random from \( \text{Enc}^{m} (0, 1^t) \).
Lemma 3.4.10. Let REC denote the distribution defined in Construction 3.18, then \( \text{supp} (\text{REC}) \subseteq \text{Shallow} (3\hat{n}_1 (1, t), 2, O (\hat{n}_2^2 (1, t))) \). Moreover, for every plausible pair \( ((r_{01}^1, r_{02}^2), c_0) \) (according to Definition 3.3.1) for a \( \times \) gadget in the first layer of \( \hat{T}_0 \) such that \( r_{01}^1 \in \text{Enc}^{\text{in}} (0, 1^t) \), if \( \text{rec} \leftarrow \text{REC} \) then \( \text{rec} (r_{01}^1, r_{02}^2, c_0) \) is indistinguishable from the wire distribution in a real-world evaluation of the \( \times \) gadget in the first layer of \( \hat{T}_0 \), conditioned on the input-output pair \( ((r_{01}^1, r_{02}^2), c_0) \).

Proof. Denote \( \hat{n}_1 = \hat{n} (1, t) \).

First, \( \text{supp} (\text{REC}) \subseteq \text{Shallow} (3\hat{n}_1 (t), 2, O (\hat{n}_2^2 (t))) \), since every function \( \text{rec}^1, \ldots, \text{rec}^{\hat{n}_1} \in \text{supp} (\text{REC}) \) uses \( O (\hat{n}_2^2) \) gates which can be arranged in two layers: the first layer contains the \( O (\hat{n}_2^2) \times \) gates of step 1, the \( \hat{n}_1^2 \) constant gates of step 2 (since \( r_1^1, \ldots, r_{\hat{n}_1}^1 \) are fixed), \( O (\hat{n}_2^2) \) constant gates (for the internal wires in the computation of \( q_0 \), which is also fixed), and \( O (\hat{n}_1) \) \( \times \) gates (for the computation of \( r_{\hat{n}_1+1} \)); and the second contains the \( \hat{n}_1^2 \times \) gates of step 3.

Second, we claim that when \( r_{01}^1 \) is well formed, and \( \text{rec} \leftarrow \text{REC} \), then the wire distribution generated by \( \text{rec} \) is indistinguishable from the real-world wire assignment, conditioned on \( ((r_{01}^1, r_{02}^2), c_0) \). The only difference between the \( \times \) gadget and the output of \( \text{rec} \) is that in the real world, the columns of \( S_0 \) are uniform well-formed vectors, whereas in the output of \( \text{rec} \) this holds for \( U_0 \). However, since the columns of \( B_0 \) are well-formed (because \( r_{01}^1 \) is well-formed), \( E^{\text{in}} \) is linear-structured, and \( S_0 = U_0 - B_0 \), then even in the reconstructed wire assignment, the columns of \( S_0 \) are uniform well-formed vectors, and \( U_0 = B_0 + S_0 \), (i.e., \( S_0, U_0 \) are equally distributed in the real-world and the simulated wire assignment). As all other computations in \( \text{rec} \) (conditioned on \( S_0 \)) emulate the computation in the \( \times \) gadget, we conclude that the distributions are identical.

Next, we describe a right-reconstructor for \( \hat{T}_V \).

Lemma 3.4.11. Let \( t \in \mathbb{N} \) be a security parameter, \( \hat{n}_1 = \hat{n} (1, t), r_{01}^1, c_0, r_1^1, \ldots, r_{\hat{n}_1}^1 \in \text{Enc}^{\text{in}} (0, 1^t) \), \( G_0 = \times \) gadget in the first layer of \( \hat{T}_0 \), and \( ((r_{01}^1, r_{02}^2), c_0) \) be a plausible pair for \( G_0 \) (according to Definition 3.3.1). Then there exists a function \( \text{rec}_V \in \text{Shallow} (\hat{n}_1 (t), 2, O (\hat{n}_1 (t))) \), such that \( \text{rec}_V (\text{rec}_0^{\hat{n}_1} \ldots \text{rec}_1 \ldots \text{rec}_1 (r_{01}^1, r_{02}^2, c_0)) \) is the wire assignment to the decoding sub-circuits \( D_V \) of \( \hat{T}_V \) that are given as input \( r_1^1, \ldots, r_{\hat{n}_1+1}^1 \) (where \( r_{\hat{n}_1+1}^1 \) is as defined by \( \text{rec}_0^{\hat{n}_1+1} (r_{01}^1, r_{02}^2, c_0) \)).

Proof. The function \( \text{rec}_V \) decodes \( S_0 = U_0 - B_0 \), and \( r_{\hat{n}_1+1}^1 = c_0 - q_0 \), where \( B_0 = r_{01}^1 (r_{02}^2)^T \), for a fixed well-formed \( r_{01}^1 \); \( c_0 \) is a fixed well-formed vector; \( U_0 = (r_1^1, \ldots, r_{\hat{n}_1}^1) \) is fixed; and consequently \( q_0 \), obtained by decoding the rows of \( U_0 \), is also fixed. Let \( \mathbf{d} \) denote the decoding vector of \( \text{Enc}^{\text{in}} (\cdot, 1^t) \), then the values of the wires of \( D_V (r_{\hat{n}_1+1}^1) \) are \( \sum_{j=1}^k d_j \times (c_{0,j} - q_{0,j}), k = 1, 2, \ldots, \hat{n}_1 (t) \). These values are all fixed, so they are computable by a circuit containing \( \hat{n}_1 (t) \) constant gates, arranged in a single layer.
The values of the wires of $\mathcal{D}_{V}(S_{0,i})$ (where $S_{0,i}$ denotes the $i$’th column of $S_{0}$) are

$$\sum_{j=1}^{k} d_{j} \times (U_{0,j,i} - B_{0,j,i}) = \sum_{j=1}^{k} d_{j} \times (U_{0,j,i} - r_{0,j}^{1} \times r_{0,i}^{2}) =$$

$$\sum_{j=1}^{k} d_{j} \times U_{0,j,i} - r_{0,i}^{2} \times \sum_{j=1}^{k} d_{j} \times r_{0,j}^{1}$$

for $k = 1, 2, \ldots, \tilde{n}_{1}(t)$. $U_{i} = r^{1}, r$, and $r_{0}^{1}$ are fixed, so for every $1 \leq k \leq \tilde{n}_{1}(t)$, $\sum_{j=1}^{k} d_{j} \times U_{0,j,i}, \sum_{j=1}^{k} d_{j} \times r_{0,j}^{1}$ are fixed, and so the internal wires of $\mathcal{D}_{V}(S_{0,i})$ are computable by a depth-2 circuit whose first layer contains $\tilde{n}_{1}(t)$ constant gates (for the values $\sum_{j=1}^{k} d_{j} \times r_{0,j}^{1}, k = 1, \ldots, \tilde{n}_{1}(t)$) and $\tilde{n}_{1}(t) \times$ gates (for the values $\sum_{j=1}^{k} d_{j} \times r_{0,j}^{1}$); and the second layer contains $\tilde{n}_{1}(t)$ constant gates (for the values $\sum_{j=1}^{k} d_{j} \times U_{0,j,i}, k = 1, \ldots, \tilde{n}_{1}(t)$) and $\tilde{n}_{1}(t) \times$ gates (for the values $\sum_{j=1}^{k} d_{j} \times U_{0,j,i}$, $k = 1, \ldots, \tilde{n}_{1}(t)$). Since $\text{rec}_{V}$ performs the same computation as $\mathcal{D}_{V}$, its output is the wire assignment of $\mathcal{D}_{V}$, conditioned on the values $r_{0}^{1}, c_{0}, r^{0}, \ldots, r^{\tilde{n}_{1}}$, and $r_{0}^{2}$.

Using the right-reconstructors, we can now bound $\text{SD}(\ell(H^{y}), \ell(H^{\text{ext}}_{\text{mid}}))$. To that effect, we first define the distributions $H_{\text{mid}}^{y}, H_{\text{ext}}^{y}$, that are similar to $H_{\text{mid}}^{x}, H_{\text{ext}}^{x}$ except that $y$ is the input used for the computation, and the wires of $\hat{C}_{1}$ remain unchanged. More formally, let $H_{\text{mid}}^{y}$ be the intermediate distribution defined by a mental experiment in which $\text{Sim}$ is given the input $y$, and operates as follows.

- Generates $\hat{y} = (y_{1}, R_{1})$, $(\hat{y}_{2}, R_{2}) \leftarrow \text{Enc}(y, 1^{t}, 1^{|C|})$.
- Honestly evaluates $\hat{C}_{1}$ on $\hat{y}_{1}$ with masking inputs $R_{1}$.
- Simulates the computation in $\hat{C}_{2}$, by picking a random input, random values for the outputs of the gadgets, and reconstructors for all gadgets; and generating the internal wires of the gadgets using the reconstructors.
- Picks random well-formed vectors for the outputs of all $\times$ gadgets in first layer of $\hat{T}_{0}$, and honestly evaluates all the wire values of $\hat{T}_{0}$, except the inputs and internal wires of these $\times$ gadgets.
- Uses the right-reconstructor of $\hat{T}_{0}$, and the function $\text{rec}_{V}$ of Lemma 3.4.11, to generate an assignment to the internal wires of all $\times$ gadgets in first layer of $\hat{T}_{0}$, and to the internal wires of $\mathcal{T}_{V}$, respectively.

$H_{\text{ext}}^{y}$ is obtained by evaluating $\hat{C}$ honestly on $\hat{y} \leftarrow \text{Enc}(y, 1^{t}, 1^{|C|})$, then picking reconstructors for all gadgets of $\hat{C}_{2}$, and re-computing their internal wires using the reconstructors; re-evaluating all wires of $\hat{T}_{0}$ using its right-reconstructor; and re-computing the internal wires of $\mathcal{T}_{V}$ using the function $\text{rec}_{V}$ of Lemma 3.4.11. In Lemma 3.4.12 below, we show that $\ell(H^{y}), \ell(H_{\text{ext}}^{y})$ are statistically close. Then, in Lemma 3.4.13, we show that $\ell(H_{\text{ext}}^{y}), \ell(H_{\text{mid}}^{y})$ are statistically close. In Remark 3.19 we show that

57
\( \ell (H^y) \), \( \ell (H^{y-x}) \) are statistically close, so \( \ell (H^y), \ell (H^{y-x}) \) are also statistically close. We conclude that \( \ell (H^y), \ell (H^x) \) are statistically close.

**Lemma 3.4.12.** Let \( t \in \mathbb{N} \) be a security parameter, \( n \in \mathbb{N} \) be an input length parameter, \( \mathcal{L}_G, \mathcal{L} \) be families of functions, \( S(n) : \mathbb{N} \rightarrow \mathbb{N} \) be a size function, and \( \epsilon(t) : \mathbb{N} \rightarrow \mathbb{R}^+ \) be a statistical distance parameter. If every gadget \( G \) of \( \hat{C}_2 \) is \((\mathcal{L}_G, \epsilon(t))\)-reconstructible, and \( \mathcal{L}_G = \mathcal{L} \circ \text{Shallow} \left( |G|, 4, O \left( \hat{n}^4(1, t) \cdot S(n) \right) \right) \), then \( \text{SD} \left( \ell (H^y), \ell (H^{y-x}) \right) \leq \epsilon(t) S(n) \) for every \( \ell \in \mathcal{L} \).

**Proof.** The proof is similar to that of Lemma 3.4.7, with the additional complication that we need to reconstruct the internal wires of \( \hat{T}_0 \), \( \mathcal{T}_V \) (using their right-reconstructors). Denote \( \hat{n}_1(t) = \hat{n}(1, t) \). Assume towards negation that the claim does not hold, and we define a fixed ordering on the \( M \leq S(n) \) gadgets of \( \hat{C}_2 \), and the hybrids \( H_i, i = 0, \ldots, M \) are obtained by evaluating \( \hat{C} \) honestly with input \( y \); recomputing the internal wires of the first \( i \) gadgets of \( \hat{C}_2 \), using their reconstructors; using the right-reconstructor, re-computing the internals of \( \hat{T}_0 \) that are influenced by the internals of the first \( i \) gadgets of \( \hat{C}_2 \); and (using the function \( \text{rec}_V \) of Lemma 3.4.11) recomputing the internal wires of \( \mathcal{T}_V \) that were influenced by re-evaluating \( \hat{T}_0 \). Then \( H_0 = H^y, H_M = H^{y-x} \).

Let \( H_m, H_{m-1}, m \in [M] \) be the neighboring hybrids such that \( \text{SD} \left( \ell (H_m), \ell (H_{m-1}) \right) > \epsilon(t) \) for some \( \ell \in \mathcal{L} \), and denote the \( m \)'th gadget by \( G \). Notice that we can fix all wire values in \( H_m, H_{m-1} \), except for the following: the internal wires of \( G \); for every gadget \( G_0 \) in \( \hat{T}_0 \) that is connected to \( G \), the wires of \( B_0, S_0 \), and the wire corresponding to the computation of \( B_0, S_0 \); and for every decoding sub-circuit \( \mathcal{D}_V \) in \( \mathcal{T}_V \) that is connected to some \( G_0 \), the internal wires corresponding to the computation of \( q_0 \). Indeed, the computation in \( \hat{C}_1 \) is the same in both hybrids, so we can fix the wire assignment of \( \hat{C}_1 \). Consequently, the right input \( \mathbf{r}_0^t \) of every \( G_0 \) gadget of \( \hat{T}_0 \) that is connected to \( G \) is well-formed and fixed, so its output \( c_0 \) is well-formed. Since \( G_0 \) is evaluated using the right-reconstructor then the columns of \( U_0 \) can also be fixed to well-formed vectors, so we can also fix the internal wires in the computation of \( q_0 \) (since these depend only on \( U_0 \)), and \( c_0 \) is therefore independent of the inputs \( \mathbf{r}_0^t, \mathbf{r}_0^2 \) and can also be fixed. Consequently, \( \hat{\mathbf{r}}^{n_1+1} \) can also be fixed. Therefore, the only non-fixed internals in \( G_0 \) are \( B_0, S_0 \), and the wires corresponding to their computation. Recall that \( \hat{\mathbf{r}}^{n_1+1} \), and the columns \( S_0, i \) of \( S_0 \), constitute the inputs to a decoding sub-circuit \( \mathcal{D}_V \) of \( \mathcal{T}_V \) (that is connected to \( G_0 \)), and \( \mathcal{D}_V \) outputs the decoding of \( \hat{\mathbf{r}}^{n_1+1}, S_0, i \) (respectively), which is zero (for \( \hat{\mathbf{r}}^{n_1+1} \) this is because \( q_0, c_0 \) are well-formed, and for \( S_0, i \) this is because \( \mathbf{r}_0^t \), and all columns of \( U_0 \), are well-formed). Therefore, the outputs of these sub-circuits \( \mathcal{D}_V \), and all values computed from them, can be fixed.

Let \( \mathcal{W}_0^R \left( \mathcal{W}_0^S \right) \) denote the real-world (reconstructed) wire assignment to the internals of \( G \) after we have fixed the values described above, and let \( \mathcal{W}_1^R = f \left( \mathcal{W}_0^R \right), \mathcal{W}_1^S = f \left( \mathcal{W}_0^S \right) \), where \( f \) is chosen according to the following distribution \( F \) over \( \text{Shallow} \left( 4, O \left( \hat{n}^4(1, t) \cdot S(n) \right) \right) \). For every gadget \( G_0 \) of \( \hat{T}_0 \) that is connected to \( G \),
$F$ chooses a function $\text{rec}_{G_0}$ chosen according to the distribution $\text{REC}$ (see Construction 3.18). The obtained function $f$, on input $E \in \mathbb{F}^{|G|}$ evaluates $\text{rec}_{G_0}$ (for every $G_0$ connected to $G$) on the corresponding masking inputs of $G$, and the hard-wired values, thus generating the masking inputs, and internal wires, of $G_0$; and uses the function $\text{rec}_V$ of Lemma 3.4.11 (with the values that are hard-wired into the $\text{rec}_{G_0}$’s) to reconstruct the internals of every decoding sub-circuit $D_V$ connected to one such $G_0$.

Notice that $f$ is well-defined, because once the masking inputs of every $G_0$ have been fixed, then we can apply the function $\text{rec}_V$. $f \in \text{Shallow} \left(4, O \left(\hat{n}^4 (t) \cdot S (n)\right)\right)$ because by Lemma 3.4.10 the right-reconstructor of every $G_0$ in $\widehat{T}_0$ that is connected to $G$ is in $\text{Shallow} \left(2, O \left(\hat{n}^2 (t)\right)\right)$ (there are at most $O \left(\hat{n}^2 (t) \cdot S (n)\right)$ such gadgets $G_0$, and their reconstructors can be applied in parallel); and the function $\text{rec}_V$ used to reconstruct the internals of every decoding sub-circuit $D_V$ of $T_V$ that is connected to $G_0$, is in $\text{Shallow} \left(2, O \left(\hat{n}^1 (t)\right)\right)$ (there are at most $O \left(\hat{n}^1 (t) \cdot S (n)\right)$ such $D_V$’s, since every $G_0$ is connected to $\hat{n}^1 (t) + 1$ decoding sub-circuits $D_V$, and for all of them, $\text{rec}_V$ can be applied in parallel).

By the lemma’s assumption, $W^R_1, W^S_1$ are $(\mathcal{L}_G, \epsilon(t))$-leakage-indistinguishable, so Lemma 3.4.6 guarantees that $W^R_1, W^S_1$ are $(\mathcal{L}, \epsilon(t))$-leakage-indistinguishable for any $\mathcal{L}$ such that $\mathcal{L} \circ \text{Shallow} \left(4, O \left(\hat{n}^4 (t) \cdot S (n)\right)\right) \subseteq \mathcal{L}_G$. In particular, $\text{SD} \left(\ell (H_m), \ell (H_{m-1})\right) = \text{SD} \left(\ell (W^S_1), \ell (W^R_1)\right) \leq \epsilon(t)$. (In the left equality we also use Lemmas 3.4.10 and 3.4.11 which guarantees that the wire values generated by the right-reconstructors are leakage-indistinguishable from the actual wire values.)

**Lemma 3.4.13.** Let $t \in \mathbb{N}$ be a security parameter, $n \in \mathbb{N}$ be an input length parameter, $\mathcal{L}_E, \mathcal{L}$ be families of functions, $S (n) : \mathbb{N} \to \mathbb{N}$ be a size function, and $\epsilon(t) : \mathbb{N} \to \mathbb{R}^+$ be a statistical distance parameter. If $E^n$ is $(\mathcal{L}_E, \epsilon(t))$-leakage-indistinguishable, and $\mathcal{L}_E = \mathcal{L} \circ \text{Shallow} \left(\hat{n} (1, t), 6, O \left(\hat{n}^4 (1, t) \cdot S (n)\right)\right)$, then $\text{SD} \left(\ell (H^y_{\text{mid}}), \ell (H^y_{\text{ext}})\right) \leq \epsilon(t) S(n)$ for every $\ell \in \mathcal{L}$.

**Proof.** The proof is similar to that of Lemma 3.4.9, with the additional complication that we need to reconstruct the internal wires of $\widehat{T}_0, T_V$ (using their right-reconstructors). Denote $\hat{n}_1 (t) = \hat{n} (1, t)$. Assume towards negation that the claim does not hold, and define a fixed ordering on the $M \leq S (n)$ bundles that are either input bundles of $\hat{C}_2$, or output bundles of gadgets in $\hat{C}_2$ (that do not touch the decoder). We define the hybrids $H_0, \ldots, H_M$, where $H_i$ is generated from $H^y_{\text{ext}}$ by replacing the first $i$ bundles with random encodings of random values; recomputing the internal wires of the first $i$ gadgets of $\hat{C}_2$ using the gadget reconstructors (see Lemma 3.3.2); reconstructing the internal wires of the $\times$ gadgets $G_0$ in the first layer of $\widehat{T}_0$ (that touch the first $i$ gadgets of $\hat{C}_2$) using the right reconstructor of Lemma 3.4.10, and re-evaluating the decoding sub-circuits $D_V$ of $T_V$ that touch these gadgets $G_0$ using the function $\text{rec}_V$ of Lemma 3.4.11. Thus, $H_0 = H^y_{\text{ext}}, H_M = H^y_{\text{mid}}$.

Let $H_m, H_{m-1}$ be the neighboring hybrids such that $\text{SD} \left(\ell (H_m), \ell (H_{m-1})\right) > \epsilon(t)$ for some $\ell \in \mathcal{L}$. We can fix all wire values in $H_m, H_{m-1}$ (while preserving the statistical
were hard-wired into the input of SD reconstructors; and then re-computing \( \hat{H} \) of \( G \) for every family of leakage functions such that \( L \circ \text{Shallow} (6, O (\hat{n}_1 (t) \cdot S (n))) \) (see Construction 3.18). The obtained function \( f \), on input \( e \in \mathbb{F}^{\hat{n}_1 (t)} \) evaluates \( \text{rec}_o \) on the inputs \( a_o, b_o \) of \( G \) (that were hard-wired into \( H_m, H_{m-1} \)), and output \( e \) (thus reconstructing the masking inputs, and internal wires, of \( G \)); evaluates \( \text{rec}_i \) on \( e \) as one of the inputs, and the other input, and output, according to the hard-wired values (thus generating the masking inputs, and internal wires, of \( G \)); for every \( G \) gadget in \( \hat{T}_0 \) connected to \( G_o \) or \( G_i \), evaluates \( \text{rec}_G \) on the corresponding masking inputs (as determined by \( \text{rec}_o, \text{rec}_i \), respectively, and the hard-wired values), thus generating the masking inputs, and internal wires, of \( G \); and uses \( \text{rec}_V \) (with the values that are hard-wired into the \( \text{rec}_G \)'s) to reconstruct the internals of every decoding sub-circuit \( D_V \) connected to one such \( G \).

Then \( W_{1}^R \equiv H_{m-1} \) and \( W_{1}^S \equiv H_{m} \) (because the bundles are re-randomizing), and \( f \in \text{Shallow} (6, O (\hat{n}_1 (t) \cdot S)) \) because \( \text{rec}_o, \text{rec}_i \in \text{Shallow} (2, O (\hat{n}_1 (t))) \) (see Lemma 3.3.2), the \( O (\hat{n}_1 (t) \cdot S (n)) \) right-reconstructors \( \text{rec}_{G_o} \) are in \( \text{Shallow} (2, O (\hat{n}_1 (t))) \) (and can be evaluated in parallel), and the \( \hat{n}_1 (t) \cdot S (n) \) functions \( \text{rec}_V \) are in \( \text{Shallow} (2, O (\hat{n}_1 (t))) \) and can be evaluated in parallel (see the proof of Lemma 3.4.12 for a more detailed analysis). Therefore, by Lemma 3.4.6, \( H_{m-1}, H_{m} \) are \( (\mathcal{L}, \epsilon (t)) \)-leakage-indistinguishable for every family \( \mathcal{L} \) of leakage functions such that \( \mathcal{L} \circ \text{Shallow} (6, O (\hat{n}_1 (t) \cdot S (n))) \subseteq \mathcal{L}_E \).

**Remark 3.19.** Let \( H_{\text{ext},x}^{\text{hy},x} \) denote the hybrid distribution obtained by evaluating \( \hat{C} \), when \( \hat{C}_1, \hat{C}_2 \) are honestly evaluated on \( y, x \), respectively (i.e., picking random encodings of \( (y, 1^t, 0^{|0(t;|C)|}) \cdot (x, 1^t, 0^{|0(t;|C)|}) \) according to \( \text{Enc}^\text{in} \) etc.); re-computing the internal wires of all gadgets in \( \hat{C}_2 \) using their reconstructors; re-evaluating all \( \times \) gadgets \( G \) (in the first layer of \( \hat{T}_0 \)) that touch gadgets of \( \hat{C}_2 \) using their right-reconstructors; and then re-computing \( T_V \). Then under the conditions of Lemma 3.4.13, \( \text{SD} (\epsilon (H_{\text{mid}}^y), \epsilon (H_{\text{ext},x}^{\text{hy},x})) \leq \epsilon (t) \cdot S (n) \), since as long as the value whose encoding was the input of \( \hat{C}_1 \) is the same in both hybrids, the proof was independent of the actual values
whose encodings were the inputs of $\hat{C}_1, \hat{C}_2$. Moreover, the proof of Lemma 3.4.12 relied only on the fact that the inputs to $\hat{C}_1, \hat{C}_2$ encode the same values in both distributions, and was independent of the actual encoded values, so the same proof can be used to show that under the conditions of Lemma 3.4.12, $SD(\ell(H_{ext,x}^y), \ell(H_{y,x}^y)) \leq \epsilon(t) \cdot S(n)$ for every leakage function $\ell \in \mathcal{L}$. Consequently, $SD(\ell(H^x), \ell(H^{y,x})) \leq 4\epsilon(t) \cdot S(n)$.

We are finally ready to prove Proposition 3.4.4.

Proof of Proposition 3.4.4. We show that Construction 3.15 satisfies the properties of Proposition 3.4.4.

Soundness follows from Lemma 3.4.5, because $\mathbb{E}^{in}$ is linear-structured, and therefore onto.

As for relaxed leakage-resilience, by Lemma 3.3.2, if $\mathbb{E}^{in}$ is $({\mathcal{L}_{E}}, \epsilon(t))-reconstructible$, then all the gadgets of $\hat{C}$ are $({\mathcal{L}_{G}}, \hat{n}(1,t) \cdot \epsilon(t))-reconstructible$, for every family $\mathcal{L}_{G}$ of leakage functions such that $\mathcal{L}_{E} = \mathcal{L}_{G} \circ \text{Shallow}(3,O(\hat{n}(1,t)))$. Since

$$\text{Shallow}(4,O(\hat{n}^2(1,t) \cdot S(n))) \circ \text{Shallow}(3,O(\hat{n}(1,t) \cdot S(n)))$$

is contained in $\text{Shallow}(7,O(\hat{n}^4(1,t) \cdot S(n)))$, then using the union bound, Lemmas 3.4.7 and 3.4.9, and Remark 3.17, we know that for every $\mathcal{L}$ such that $\mathcal{L}_{E} = \mathcal{L} \circ \text{Shallow}(7,O(\hat{n}^4(1,t) \cdot S(n)))$, and for every $\ell \in \mathcal{L}$, $SD(\ell(H^x), \ell(H^{y,x})) \leq 4\epsilon(t) \cdot S(n)$. Similarly, by Lemmas 3.4.12 and 3.4.13, and Remark 3.19, $SD(H_{y,x}^y, H^y) \leq 4\epsilon(t) \cdot S(n)$. Therefore, $SD(\ell(H^x), \ell(H^y)) \leq 8\epsilon(t) \cdot S(n)$.

Regarding the size of the compiled circuit, $\hat{C}$ contains two copies of $C$, where in each copy each gate (out of at most $|C|$) is replaced with a gadget whose size is at most $O(\hat{n}^2(1,t))$, and which uses at most $O(\hat{n}(1,t))$ masking inputs, so $|\hat{C}_1|, |\hat{C}_2| \leq O(\hat{n}^2(1,t) \cdot |C|)$. $\hat{T}_0$ contains $O\left(\hat{n}^3(1,t) \cdot |C|^2\right)$ “binarization” sub-circuits $T$, each of size at most $O(|\mathbb{F}|)$, followed by a tree of $\times$ gates, so $|\hat{T}_0| \leq O\left(|\mathbb{F}| \cdot \hat{n}^2(1,t) \cdot |C|^2\right)$. As for $\hat{T}_r$, it contains a decoding sub-circuit for each of the (at most $O(\hat{n}(1,t))$) masking inputs used in the (at most $O\left(|\mathbb{F}| \cdot \hat{n}^2(1,t) \cdot |C|^2\right)$ gadgets of $\hat{T}_0$. The decoding of each masking input requires $\hat{n}(1,t) \times$ gates followed by $\hat{n}(1,t) +$ gates. In addition, we have $O\left(|\mathbb{F}| \cdot \hat{n}^3(1,t) \cdot |C|^2\right)$ constant-sized binarization circuits, followed by $O\left(|\mathbb{F}| \cdot \hat{n}^3(1,t) \cdot |C|^2\right) \times$ gates, so $|\hat{T}_r| \leq O\left(|\mathbb{F}| \cdot \hat{n}^4(1,t) \cdot |C|^2\right)$. Therefore, $|\hat{C}| \leq O\left(|\mathbb{F}| \cdot \hat{n}^4(1,t) \cdot |C|^2\right)$.

3.5 Boolean SAT-Respecting Relaxed LRCCs

In this section we construct relaxed LRCCs over $\mathbb{F}_2$. Our starting point is the circuit-compiler of Construction 3.15 over the field $\mathbb{F}$, which we apply to an “arithmetic version” of the boolean circuit. At a high-level, we construct our circuit compiler over $\mathbb{F}_2$ as follows: we represent field elements of $\mathbb{F}$ using bit-strings; and operations.
+ \),-\),\times\),\mathsf{id}, \mathsf{copy}, \mathsf{const}_a, \alpha \in \mathbb{F} as functions over \lceil \log |\mathbb{F}| \rceil\)-bit strings. (For now, we assume that there exist gates operating on \lceil \log |\mathbb{F}| \rceil\)-bit strings and computing these operations.) We “translate” boolean circuits into arithmetic circuits with such operations, and apply the circuit-compiler of Construction 3.15 (where the field operations are implemented using boolean operations) to the “translated” circuit. (We note that leakage-resilience deteriorates when an arithmetic compiler is transformed to a boolean one, but only by a constant factor in the depth and size of circuits computing the leakage functions.) Concretely, we set \( \mathbb{F} = \mathbb{F}_3 \).

From Boolean Circuits to Arithmetic Circuits

Our boolean circuit-compiler operates on boolean circuits, but employs an arithmetic circuit-compiler which operates on arithmetic circuits over \( \mathbb{F} \). Therefore, we first transform the boolean circuit into an equivalent arithmetic circuit in the natural manner (i.e., representing every bit operation as a polynomial over the arithmetic field):

**Definition 3.5.1** (Boolean-to-arithmetic “translator” \( T' \)). Given a boolean circuit \( C : \{0,1\}^n \rightarrow \{0,1\}^k \), the algorithm \( T' \) transforms it into a functionally equivalent arithmetic circuit \( C' : \mathbb{F}^n \rightarrow \mathbb{F}^k \) (where by “functionally-equivalent” we mean that for every \( x \in \{0,1\}^n \), \( C'(x) = C(x) \)). This is done by replacing the gates of \( C \) as follows.

- The negation operation \( \neg x \) is replaced with \( 1 - x \) (replacing a single boolean gate with 2 arithmetic gates, i.e. a \( \mathsf{const}_1 \) gate and a \( \neg \) gate).

- The operation \( x \land y \) is replaced with \( x \cdot y \) (replacing a single boolean gate with a single arithmetic gate).

- Using De-Morgan’s laws, the operation \( x \lor y \) is replaced with \( 1 - ((1 - x) \cdot (1 - y)) \) (replacing a single boolean gate with 7 arithmetic gates).

- The XOR operation \( x \oplus y \) is replaced with \( (1 - x) y + x (1 - y) \) (replacing a single gate with 9 arithmetic gates).

- \( \mathsf{id}, \mathsf{copy}, \mathsf{const}_0, \mathsf{const}_1 \) remain unchanged.

**Observation 3.5.2.** For every \( x \in \{0,1\}^n \), \( C'(x) = C(x) \).

Representing Field Elements as Bit Strings

We can use any transformation \( E_b : \mathbb{F}_3 \rightarrow \{0,1\}^2 \) such that every bit string is associated with a field element. This is required for the SAT-respecting property, to guarantee that whatever values are carried on the wires of the boolean circuit, they can be “translated” into wires of the arithmetic circuit over \( \mathbb{F}_3 \), and is achieved by defining a “reverse” mapping \( E_b^{-1} \). Concretely, we use the following mapping.
**Definition 3.5.3** (mod-3 mapping $E_b$). The mod-3 mapping $E_b : \mathbb{F}_3 \rightarrow \{0, 1\}^2$ is defined as follows: $E_b(0) = 00$, $E_b(1) = 01$, and $E_b(2) = 11$. The “reverse” mapping $E_b^{-1} : \{0, 1\}^2 \rightarrow \mathbb{F}_3$ is defined as follows: $E_b^{-1}(00) = 0$, $E_b^{-1}(01) = E_b^{-1}(10) = 1$, and $E_b^{-1}(11) = 2$. $E_b, E_b^{-1}$ naturally extend to longer strings, where for $v = (v_1, \ldots, v_n) \in \mathbb{F}_3^n$, $E_b(v) = (E_b(v_1), \ldots, E_b(v_n))$, and for $(b_{1,1}, b_{1,2}, \ldots, b_{n,1}, b_{n,2}) \in \{0, 1\}^{2n}$, $E_b^{-1}(b_{1,1}, \ldots, b_{n,2}) = (E_b^{-1}(b_{1,1}), \ldots, E_b^{-1}(b_{n,1}, b_{n,2}))$.

Note that the string “10” is never used as long as the compiler is honestly applied to the arithmetic circuit.

**Implementing Field Operations**

The compiled arithmetic circuit uses the field operations $+,-,\times,$ and also copy, id and $\text{const}_\alpha, \alpha \in \mathbb{F}_3$. These operations are represented using bit operations over bit strings generated by $E_b$. Specifically, we think of every field operation as a boolean function with 4 inputs (a pair of 2-bit strings representing the pair of input field elements) and 2 outputs (a 2-bit string representing the output field element). We stress that though an honest construction over bits uses only 3 of the 4 possible 2-bit strings encoding field elements (i.e., only the strings in the image of $E_b$ as defined in Definition 3.5.3), the function representing a field operation in $\mathbb{F}_3$ should be defined to output the correct values on all 2-bit strings. The truth table of each output bit has constant size, and can be represented by a constant-size, depth-3 boolean circuit. copy, id and $\text{const}_\alpha$ gates are handled similarly. Therefore, the size (depth) of each gadget (and consequently, of the entire compiled circuit) increases by a constant multiplicative factor (by a factor of 3).

Notice that representing boolean circuits using arithmetic circuits introduces the following obstacle. For a satisfiable circuit $\hat{C}$, we are only guaranteed the existence of an $x \in \mathbb{F}_3^n$ satisfying the original arithmetic circuit, whereas for boolean circuits we require that $x \in \{0, 1\}^n$. Therefore, we need an additional “input checker” sub-circuit to guarantee that the inputs to the compiled circuit encode binary strings.

**Definition 3.5.4** (Input-checker $T^{\text{in}}$). $T^{\text{in}} : \mathbb{F} \rightarrow \mathbb{F}$ is defined as follows: $T^{\text{in}}(z) = T(z^2 - z)$.

**Observation 3.5.5.** For every $z \in \mathbb{F}_3$, $T^{\text{in}}(z) \in \{0, 1\}$, and $T^{\text{in}}(z) = 1$ if and only if $z \in \{0, 1\}$.

We now use an arithmetic SAT-respecting relaxed LRCC (Construction 3.15) to construct a boolean circuit compiler with similar properties.

**Construction 3.20.** Let $E_b, E_b^{-1}$ be the mappings of Definition 3.5.3. Let $T'$ be the algorithm of Definition 3.5.1 over $\mathbb{F}_3$, and $(\text{Comp}, E = (\text{Enc}, \text{Dec}))$ be the circuit compiler over $\mathbb{F}_3$ of Construction 3.15. The boolean SAT-respecting relaxed LRCC $(\text{Comp}_b, E_b = (\text{Enc}_b, \text{Dec}_b))$ is defined as follows.
• $\text{Enc}_b = E_b \circ \text{Enc}$ and $\text{Dec}_b = \text{Dec} \circ E_b^{-1}$.

• $\text{Comp}_b$ on input $C : \{0, 1\}^n \rightarrow \{0, 1\}$:
  
  - Uses $T'$ to transform $C$ into an equivalent arithmetic circuit $C' : \mathbb{F}_3^n \rightarrow \mathbb{F}_3$.
  - Constructs the circuit $C'' : \mathbb{F}_3^n \rightarrow \mathbb{F}_3$ such that $C''(x_1, \ldots, x_n) = 1 - (C'(x_1, \ldots, x_n) \times (\bigwedge_{i=1}^n T^\text{in}(x_i)))$. (Notice that $C''(x_1, \ldots, x_n)$ outputs 0 if and only if $C'(x_1, \ldots, x_n) = 1$ and $x_1, \ldots, x_n \in \{0, 1\}$.)
  - Computes $\hat{C}'' = \text{Comp}(C'')$.
  - Replaces every gate in $\hat{C}''$ with a constant-size, depth-3 boolean circuit computing the truth table of the gate operation. $\text{Comp}_b$ can use any correct circuit, as long as these circuits are used consistently (i.e., for every gate the same circuit is used to replace all appearances of such gate in $\hat{C}''$), and operate on the bit-representation of elements of $\mathbb{F}_3$ defined in Definition 3.5.3.
  
  Denote the output of $\hat{C}''$ by $e \in \mathbb{F}_3$, represented by the string $(e_1, e_2) \in \{0, 1\}^2$. Then $\text{Comp}_b$ outputs the circuit $\hat{C}_b$ obtained from $\hat{C}''$ by applying an $\lor$ gate, followed by a $\neg$ gate, to the output of $\hat{C}''$. (This reduces the output string of $\hat{C}''$ to a single bit, and flips the output of $\hat{C}''$, which is required due to the negation added in $C''$.)

We use $\hat{C}_{1,b}, \hat{C}_{2,b}, \hat{T}_0,b, \hat{T}_V,b$ to denote the components of $\hat{C}_b$ corresponding to $\hat{C}_1, \hat{C}_2, \hat{T}_0, \hat{T}_V$, respectively.

**Observation 3.5.6.** $\hat{C}_b(\hat{x}) \in \{0, 1\}$ for every $\hat{x}$. Moreover, $\hat{C}_b(\hat{x}) = 1$ if and only if $\hat{C}''(\hat{x}) = 0$. If $\text{Comp}_b$ is SAT-respecting, then this guarantees that $C''(\hat{x}) = 0$ for some $x \in \mathbb{F}_3$. The definition of $C''$, and the correctness of $T'$, guarantees that $x \in \{0, 1\}^n$, and that $C'(\hat{x}) = C(x) = 1$.

**Remark 3.21.** $E_b$ is used to represent field elements in $\mathbb{F}_3$ through bit strings. Formally, an encoding scheme is based on a single alphabet over which both the messages, and the encodings, are defined. For $E_b$ to be consistent with the syntactic definition, and with our applications (for boolean SAT-respecting relaxed LRCCs), the input to its encoding algorithm $\text{Enc}_b$ is a 2-bit string representing a field element, and we assume that the operation of $\text{Enc}, \text{Dec}$ is implemented using this representation of field elements.

Next, we show that Construction 3.20 “inherits” the properties of the underlying arithmetic circuit compiler. We first show that Construction 3.20 is SAT-respecting.

**Lemma 3.5.7.** If Construction 3.15 is SAT-respecting then Construction 3.20 is also SAT-respecting, i.e. if $\hat{C}_b$ is satisfiable then so is $C$.

**Proof.** Assume that $\hat{C}_b(\hat{x}_b) = 1$ for some $\hat{x}_b \in \{0, 1\}^{2h}$, where $\hat{C}_b = \text{Comp}_b(C)$. Then because $\hat{C}_b$ computes the negation of the OR of the (2-bit) output of $\hat{C}'$, then the
output of $\hat{C}''$ was 0. Therefore, by the definition of $\text{Comp}_b$, the correctness of the implementation of field operations using bit operations, and since $E^{-1}_b$ is onto $\mathbb{F}_3$, $\hat{C}''(E^{-1}_b(\hat{x}_b)) = 0$ (where $\hat{C}'' = \text{Comp}(C'')$). Since $\text{Comp}$ is SAT-respecting, there exists an $x \in \mathbb{F}_3^n$ such that $C''(x) = 0$ which, by the definition of $C''$, is possible if and only if $x \in \{0,1\}^n$ and $C'(x) = 1$. By the correctness of the transformation $T'$ (Definition 3.5.1), $C(x) = C'(x) = 1$.

Next, we show that Construction 3.20 is relaxed-leakage-resilient.

**Lemma 3.5.8.** Let $t \in \mathbb{N}$ be a security parameter, $n \in \mathbb{N}$ be an input length parameter, $\mathcal{L}, \mathcal{L}_b$ be families of functions, $S(n) : \mathbb{N} \rightarrow \mathbb{N}$ be a size function, and $\epsilon(t) : \mathbb{N} \rightarrow \mathbb{R}^+$ be a statistical distance parameter. Then there exists a constant $c > 0$ such that if $\mathcal{L} = \mathcal{L}_b \circ \text{UnBBool} \left(12, O \left( \left( \hat{n}^4(1,t) \cdot S(n)^2 \right) \right)\right)$, and $(\text{Comp}, E)$ of Construction 3.15 is an $(\mathcal{L}, c \cdot S(n), \epsilon(t))$-relaxed-LRCC over $\mathbb{F}_3$, then $(\text{Comp}_b, E_b)$ of Construction 3.20 is an $(\mathcal{L}_b, S(n), \epsilon(t))$-relaxed-LRCC. Moreover, $|\hat{C}_b| = O \left( \hat{n}^4(1,t) |C|^2 \right)$.

**Proof.** Notice that the existence of an inefficient simulator in Definition 3.2.10 is equivalent to requiring that leakage functions cannot distinguish evaluations of $\hat{C}_b$ on encodings of two inputs on which $C$ has the same output. We will use this alternative definition to prove the lemma. We choose $c = 63$.

Assume towards negation that $(\text{Comp}_b, E_b)$ is not $(\mathcal{L}_b, S(n), \epsilon(t))$-relaxed leakage-resilient. Then there exist a boolean circuit $C : \{0,1\}^n \rightarrow \{0,1\}$ of size at most $S(n)$, a pair of inputs $y_b, z_b \in \{0,1\}^n$ for $C$ such that $C(y_b) = C(z_b)$, and a leakage function $\ell_b \in \mathcal{L}_b$, such that $SD \left( \ell_b(\hat{C}_b(\hat{y}_b)), \ell_b(\hat{C}_b(\hat{z}_b)) \right) > \epsilon(t)$, where $\hat{y}_b, \hat{z}_b$ are random encodings of $y_b, z_b$, respectively, according to $E_b(N, 1^t, 1^{\mathcal{C}(C)})$. We show that $SD \left( \ell(\hat{C}(\hat{y})), \ell(\hat{C}(\hat{z})) \right) > \epsilon(t)$ for some leakage function $\ell \in \mathcal{L}$, where $\hat{y}, \hat{z}$ are random encodings of $y_b, z_b$, respectively, according to $E(N, 1^t, 1^{\mathcal{C}(C)})$, and $\hat{C} = \text{Comp}(C'')$ (recall that we assume that field elements are encoded using bit strings; and that we have “gate sub-circuits”, operating on bit strings, that emulate the field operations). This contradicts the relaxed leakage-resilience of $(\text{Comp}_b, E_b)$, because $C''(y) = C''(z)$ (because in both cases the input-checker outputs 1), and $|C''| \leq c |C| \leq c \cdot S(n) = 63 \cdot S(n)$.

(Indeed, the transformation from $C$ to $C''$ blows up the circuit by a factor of at most 9, and the transformation from $C'$ to $C''$ adds at most 7 gates for every input gate, so it blows up the circuit by a factor of at most 7.)

We distinguish between two “kinds” of wires in $\hat{C}_b$: **external** wires, that appear also in $\hat{C}$ (these are the wires between the gates of $\hat{C}$), and **internal** wires (these are the wires in the sub-circuits of $\hat{C}_b$ that emulate the gates of $\hat{C}$). The function $\ell$ is given as input a wire assignment for $\hat{C}$, which is a sequence of field elements, encoded into 2-bit strings using some correct encoding. $\ell$ first “translates” every 2-bit string into the 2-bit string encoding the same field element under the mapping $E_b$ of Definition 3.5.3. This is computable by a constant-size, depth-3 circuit with gates of unbounded fan-in and fan-out, and since these computations can be done in parallel, this “translation" is
computable in $\text{UnBBool}(3, O\left(\lceil C\rceil\right))$. This defines the wire values of all the external wires of $\hat{C}_b$, encoded into bit-strings using the same mapping that $\text{Comp}_b$ uses.

Next, $\ell$ computes the internal wires of $\hat{C}_b$. Recall that the internal wires are organized in constant-sized, depth-3 “groups”, where every such “group” corresponds to the computation of a single gate of $\hat{C}$. Therefore, the wires in every “group” are computable in $\text{UnBBool}(9, O(1))$ (since these wires can be computed sequentially given the input to the gate, where every wire is computable from the previous by a depth-3, constant-size circuit with gates of unbounded fan-in and fan-out). As all “groups” can be evaluated in parallel, the internal wires are computable from the external wires in $\text{UnBBool}(9, O\left(\lceil C\rceil\right))$. Then, $\ell$ evaluates $\ell_b$ on the wire values that it generated. As $|\hat{C}| \leq O\left(\hat{n}^4(1, t) |C''|^2\right) \leq O\left(\hat{n}^4(1, t) \cdot S(n)^2\right)$ (by Proposition 3.4.4), then $\ell \in \mathcal{L}_b \circ \text{UnBBool}(12, O\left(\left(\hat{n}^4(1, t) \cdot S(n)^2\right)\right)) = \mathcal{E}$. Moreover, $\ell_b(\hat{C}_b(\hat{y}_b)) = \ell(\hat{C}(\hat{y}))$ and $\ell_b(\hat{C}_b(\hat{z}_b)) = \ell(\hat{C}(\hat{z}))$, so $\text{SD}(\ell(\hat{C}(\hat{y})), \ell(\hat{C}(\hat{z})))) > \epsilon(t)$.

Combining Lemmas 3.5.7 and 3.5.8 with Proposition 3.4.4, we have the following.

**Proposition 3.5.9** (Boolean SAT-respecting relaxed LRCC). Let $t \in \mathbb{N}$ be a security parameter, $n \in \mathbb{N}$ be an input length parameter, $\mathcal{L}, \mathcal{L}_E$ be families of functions, $S(n) : \mathbb{N} \rightarrow \mathbb{N}$ be a size function, and $\epsilon(t) : \mathbb{N} \rightarrow \mathbb{R}^+$ be a statistical distance parameter. Let $E_{\mathbb{F}_3}$ be a linear-structured encoding scheme over $\mathbb{F}_3$ with input parameter $n = 1$, and security parameter $t$, which outputs encodings of length $\hat{n} = \hat{n}(t)$. Assume that $E_n$ is $(\mathcal{L}_E, \epsilon(t))$-leakage-indistinguishable, where $\mathcal{L}_E = \mathcal{L} \circ \text{UnBBool}(33, O\left(\hat{n}^4(t) \cdot S(n)^2\right))$. Then there exists a constant $c > 0$, and a SAT-respecting, $(\mathcal{L}, S(n), c \cdot \epsilon(t) \cdot S(n))$-relaxed-LRCC. Moreover, $|\hat{C}_b| = O\left(\hat{n}^4(t) |C''|^2\right)$.

**Proof.** We show that Construction 3.20 has the required properties, when $E$ is the encoding scheme of Construction 3.15 (using the inner encoding scheme $E_{\mathbb{F}_3}$), and the size function is $63 \cdot S(n)$.

Lemmas 3.5.7 and 3.4.5 guarantee that the compiled circuit is SAT-respecting. As noted in the proof of Lemma 3.5.8, if the wires of the compiled arithmetic circuit are already encoded into bit-strings using the same mapping that $\text{Comp}_b$ uses, then every arithmetic gate can be implemented with a depth-3, constant-size boolean circuit over gates of unbounded fan-in and fan-out. Therefore, functions computable in Shallow $(d, s)$ are also computable in $\text{UnBBool}(3d, O(s))$, since all sub-circuits corresponding to gates in a single layer can be evaluated in parallel. (Recall that all field operations are implemented using boolean operations over bit strings. Therefore, in essence this means that leakage functions over $\mathbb{F}_3$ can be simulated using somewhat deeper boolean circuits.) In particular, if $E$ is $(\mathcal{L}_E, \epsilon)$-leakage-indistinguishable and $\mathcal{L}_E = \mathcal{L}' \circ \text{UnBBool}(21, O\left(\hat{n}^4(t) \cdot S(n)\right))$, then by Proposition 3.4.4, Construction 3.15 is an arithmetic SAT-respecting $(\mathcal{L}', 63 \cdot S(n), 8c \cdot 63 \cdot S(n))$-relaxed LRCC. Therefore, Lemma 3.5.8 guarantees that $(\text{Comp}_b, E_b)$ is $(\mathcal{L}, S(n), 8 \cdot 63 \cdot \epsilon(n) \cdot S(n))$-leakage-resilient as long as $\mathcal{L}_E = \mathcal{L} \circ \text{UnBBool}(3, O(1))$. 

66
UnBBool \( (21, O \left(n^4(t) \cdot S(n)\right)) \circ \text{UnBBool} \left(12, O \left(n^4(t) \cdot S(n)^2\right)\right) \). Moreover, \( |\hat{C}| = O \left(n^4(t) |C|^2\right) \) because \( |\hat{C}| = O \left(n^4(t) |C''|^2\right) \), and since the blowup in the transformations from \( C \) to \( C'' \), and from \( \hat{C} \) to \( \hat{C}_b \), is constant.

Instantiating Proposition 3.5.9 with the parity encoding \( E_{\oplus} \) of Example 3.2.4 as the underlying encoding scheme \( E^0 \), and using Theorem 3.6, we obtain an LRCC secure against leakage from \( AC^0 \) circuits (namely, constant-depth and polynomial-sized boolean circuits with unbounded fan-in \( \land, \lor \) and \( \neg \) gates). This will follow as a special case from the next theorem.

**Theorem 3.22** (Formal statement of Theorem 3.1). For every constant \( \delta \in (0, 1/7) \), security parameter \( t \in \mathbb{N} \), and boolean circuit \( C : \{0, 1\}^n \to \{0, 1\} \) of size \( s \), there exists a pair \( \left(\text{Enc}, \hat{C}\right) \), where \( \hat{C} \) is a boolean circuit of size \( \text{poly}(t, s) \), such that for every depth parameter \( d \in \mathbb{N}, \left(\text{Enc}, \hat{C}\right) \) is an \((L, \epsilon)\)-leakage-resilient implementation of the function \( f_C \) that \( C \) computes, where:

- \( L \) consists of all boolean functions computable by depth-\( d \), size-\( 2^{O((1/7-\delta)/d)} \) boolean circuits over gates of unbounded fan-in and fan-out, with \( |\hat{C}|^\delta \) output bits.
- \( \epsilon = 2^{-d\Omega((1/7-\delta)/d)} \).

Moreover, \( \text{Enc} \) can be efficiently computable given \( 1^{\hat{C}}, 1^t \), and \( \hat{C} \) can be efficiently computable given \( C, 1^t \).

**Proof.** We use the circuit compiler of Proposition 3.5.9, instantiating \( E^0 \) with the parity encoding \( E_{\oplus} \). Given a circuit \( C : \{0, 1\}^n \to \{0, 1\} \) of size \( s \), \( E_{\oplus} \) is applied with security parameter \( \tilde{t} = \max\{s, t\} \) (in particular, \( E_{\oplus} \left(\cdot, 1^{\tilde{t}}\right) \) outputs encodings of length \( \tilde{t} \)), to obtain the compiled circuit \( \hat{C} \). Then \( s = |\hat{C}| \leq c' \cdot \tilde{t}^4 \cdot s^2 = \max\{c't^6, c's^6\} \) for some constant \( c' > 0 \), by Proposition 3.5.9. Moreover, for a large enough \( \tilde{t} \), it holds that \( \tilde{s} \leq \max\{t^7, s^7\} = \tilde{t}^7 \).

Fix some family \( C_{\text{leak}} \) of boolean circuits with depth \( d \), size \( s_{\text{leak}} = 2^{c'(1/7-\delta)/d} \) (where \( \gamma \in (0, 1) \) is a constant whose value will be set later), and output length \( s^\delta \), over gates of unbounded fan-in and fan-out. Let \( \tilde{d} = d + 33 \), and \( \delta = 7\delta \in (0, 1) \). Then Theorem 3.7 guarantees that there exist constants \( c_1, c_2 > 0 \) such that \( \text{Enc}_{\oplus} \left(\cdot, 1^{\tilde{t}}\right) \) is \( \left(\text{UnBBool} \left(\tilde{t}, \tilde{d}, 2^{c_1 \cdot \tilde{t}^{(1-\delta)/d}}, 2^{-c_2 \cdot \tilde{t}^{(1-\delta)/d}}\right)\right) \)-leakage-indistinguishable. Consequently, \( \text{Enc}_{\oplus} \left(\cdot, 1^{\tilde{t}}\right) \) is \( \left(\text{UnBBool} \left(\tilde{t}, \tildes, \tilde{d}, 2^{c_1 \cdot \max\{s, t\}^{(1-\delta)/d}}\right)\right) \)-leakage-indistinguishable.

We choose \( \epsilon = \epsilon_\delta = c \cdot 2^{-c_2 \cdot \tildes^{(1-\delta)/d}} \), where \( c \) is the constant of Proposition 3.5.9. Notice that \( \epsilon_\delta \leq 2^{-d\Omega((1/7-\delta)/d)} \). We set \( \gamma \in (0, 1) \) such that for a large enough \( m, 2^{(c_1/2)\cdot m^{(1-\delta)/d}} \geq 2^{\gamma(1-7\delta)/d} \). Then for large enough \( t, s, 2^{(c_1 \cdot \max\{s, t\}^{(1-\delta)/d}} \geq s_{\text{leak}} + \max\{c't^6, c's^6\} \), which means that the composition of \( C_{\text{leak}} \) with any boolean circuit of size at most \( \max\{c't^6, c's^6\} \) and depth 33, obtains advantage at most \( 2^{-c_2 \cdot \tildes^{(1-\delta)/d}} \).
\[ 2^{-(c_2/2) \cdot s(1-\delta)/d} \] in distinguishing parity encodings of 0 and 1. By proposition 3.5.9, the simulated and actual wire values of \( \hat{C} \) are \( c \cdot 2^{-(c_2/2) \cdot s(1-\delta)/d} \cdot 2^{-(c_2/2) \cdot s(1-\delta)/d} \cdot s \) statistically close. Since \( 2^{-(c_2/2) \cdot s(1-\delta)/d} \cdot s \leq 1 \) for a large enough \( s \), the wire values are \( \epsilon_{\delta}(s) \) statistically close.

### 3.6 SAT-Respecting LRCCs that Resist Leakage from \( \text{AC}^0 \) Circuits Augmented with \( \oplus \) Gates

In this section we extend the results of Section 3.5, and present a boolean SAT-respecting relaxed LRCC that resists leakage from \( \text{AC}^0 \) circuits (namely, constant-depth, polynomial-sized boolean circuits over unbounded fan-in and fan-out \( \land, \lor, \neg \) gates), augmented with a sublinear number of \( \oplus \) gates of unbounded fan-in and fan-out. This circuit compiler will be used in Chapter 4 to construct PCP systems with zero-knowledge guarantees.

The high level idea is to use Construction 3.20, where the underlying arithmetic LRCC over \( \mathbb{F}_3 \) is instantiated with the encoding scheme \( E^n \) that maps an element \( \gamma \in \mathbb{F}_3 \) into a vector \( v \in \{0, 1\}^t \) (where \( t \in \mathbb{N} \) is the security parameter of \( E^n \)), which is random subject to the constraint that the number of 1’s in \( v \) is congruent to \( \gamma \) modulo 3. We show, by reduction to correlation bounds of \([\text{LS11}]\), that \( \text{AC}^0 \) circuits, augmented with a sublinear number of \( \oplus \) gates, have a negligible advantage in distinguishing between random encodings of 0 and 1 according to \( E^n \). Using the leakage-indistinguishability of \( E^n \), we construct a SAT-respecting circuit compiler withstanding leakage from \( \text{AC}^0 \) circuits that have several output bits and are augmented with a sublinear number of \( \oplus \) gates:

**Theorem 3.23** (SAT-respecting relaxed LRCC for \( \text{AC}^0 \) with \( \oplus \) gates). For input length parameter \( n \), leakage length bound \( \hat{n} = \hat{n}(n) \), size bound \( s = s(n) \), output length bound \( k = k(n) \), parity gate bound \( m = m(n) \), and depth bound \( d \), let \( \mathcal{L}^k_{\hat{n}, d, s, \oplus m} = \bigcup_{n \in \mathbb{N}} \mathcal{L}^k_{\hat{n}(n), d, s(n), \oplus m(n)} \), where \( \mathcal{L}^k_{\hat{n}, d, s, \oplus m} \) denotes the class of boolean circuits of input length \( \hat{n} \) over \( \neg \) gates and unbounded \( \land, \lor, \oplus \) gates, whose depth, size, output length, and number of parity gates are bounded by \( d_0, s_0, k_0, m_0 \), respectively. Then for every positive constants \( d, c \), polynomials \( k, m \), and polynomial size bound \( s' = s'(n) \), there exists a polynomial \( S(n) \), such that there exists a SAT-respecting \( \left( \mathcal{L}^k_{\hat{n}, d, s, \oplus m}, S(n), 2^{-n^c} \right) \)-relaxed LRCC, which on input a circuit \( C : \{0, 1\}^n \rightarrow \{0, 1\} \) of size \( |C| \leq s'(n) \) outputs a circuit \( \hat{C} \) of size \( |\hat{C}| \leq S(n) \).

The remainder of the section is organized as follows. In Section 3.6.1 we exhibit an encoding scheme that is leakage-indistinguishable against \( \text{AC}^0 \) circuits augmented with a sublinear number of \( \oplus \) gates. Then, in Section 3.6.2 we prove Theorem 3.23.
3.6.1 A Leakage-Indistinguishable Encoding Scheme

In this section we use correlation bounds of [LS11] to construct an encoding scheme which is leakage-indistinguishable against leakage computable by \( \text{AC}^0 \) circuits, augmented with “few” \( \oplus \) gates. This encoding scheme will be used in Section 3.6.2 to prove Theorem 3.23. We first formally define the encoding scheme which we use, where in this entire section \( n \) is used to denote both the input length of functions, and the security parameter of encoding schemes. (This dual meaning of \( n \) will not be confusing, because in this section the input length of encoding schemes will be fixed to 1, and the security parameter of the encoding scheme will determine the input length of functions.)

**Notation 3.24.** For \( \gamma \in \{0, 1, 2\} \), and \( n \in \mathbb{N} \), \( U^n_\gamma \) denotes the uniform distribution over \( \{v \in \{0, 1\}^{3n} : \#_1(v) \equiv \gamma \mod 3\} \); \( \#_1(v) \) denotes the number of 1’s in \( v \); and \( U^0_{1,2} \) denotes the uniform distribution over \( \{v \in \{0, 1\}^{3n} : \#_1(v) \not\equiv 0 \mod 3\} \).

**Definition 3.6.1.** Let \( n \in \mathbb{N} \) be a security parameter. We define an encoding scheme \( E_3 = (\text{Enc}_3, \text{Dec}_3) \) over \( \mathbb{F}_3 \) such that for every \( e \in \mathbb{F}_3 \), \( \text{Enc}_3(e, 1^n) \) is distributed according to \( U^6_e \) and \( \text{Dec}_3(v, 1^n) \) returns \( (\#_1(v) \mod 3) \). Notice that \( E_3 \) is linear-structured, with decoding vectors \( \{1^{3n}\} \), and consequently also onto.

We will be interested in withstanding leakage computed by “\( \text{AC}^0 \)” circuits, augmented with few \( \oplus \) gates”. This leakage class is formalized in the next Definition.

**Definition 3.6.2** \((\mathcal{L}^k_{n,d,s,\oplus m} \text{ leakage family})\). Let \( n \in \mathbb{N} \) be a length parameter, \( d \in \mathbb{N} \) be a depth parameter, \( s \in \mathbb{N} \) be a size parameter, and \( m \in \mathbb{N} \) be a parity gate bound. The family \( \mathcal{L}_{n,d,s,\oplus m} \) consists of all functions computable by a boolean circuit \( C : \{0, 1\}^n \rightarrow \{0, 1\} \) of size at most \( s \) and depth \( d \), with unbounded fan-in and fan-out \( \land, \lor, \neg, \oplus \) gates, out of which at most \( m \) are \( \oplus \) gates. The family \( \mathcal{L}_{d,s,\oplus m} \) of functions is defined as \( \mathcal{L}_{d,s,\oplus m} = \bigcup_{n \in \mathbb{N}} \mathcal{L}_{n,d,s,\oplus m} \).

For an output length parameter \( k \in \mathbb{N} \), and a function \( f : \{0, 1\}^n \rightarrow \{0, 1\}^k \), let \( f_i(x_1, \ldots, x_n), i \in [k] \) denote the \( i \)’th output bit of \( f \). We use the following notation: \( \mathcal{L}^k_{n,d,s,\oplus m} = \{ f : \{0, 1\}^n \rightarrow \{0, 1\}^k : \forall i \leq k, f_i \in \mathcal{L}_{n,d,s,\oplus m} \} \), and \( \mathcal{L}^k_{d,s,\oplus m} := \bigcup_{n \in \mathbb{N}} \mathcal{L}^k_{n,d,s,\oplus m} \).

We use a correlation bound of Lovett and Srinivasan [LS11, Theorem 6] which, informally, states that \( \text{AC}^0 \) circuits, augmented with a sublinear number of \( \oplus \) gates, have negligible correlation with the boolean function \( \text{MOD}_3 \) where \( \text{MOD}_3(v) = 0 \) if and only if \( \#_1(v) \equiv 0 \mod 3 \). (Their result is more general, but we state a weaker and simpler version that suffices for our needs.) We first define the notion of correlation.

---

\( ^6\text{Enc}_3 \) can be computed efficiently by repeating the following procedure \( n^2 \) times. Pick \( v \in \{0, 1\}^{3n} \) uniformly at random, compute \( \#_1(v) \), and if it equals \( e \) return \( v \). If all iterations fail, return a fixed \( v_e \in \{0, 1\}^n \) such that \( \#_1(v) = e \). Then the output of \( \text{Enc}_3 \) is statistically close to \( U^6_e \).
Definition 3.6.3 (Correlation). Let \( n \in \mathbb{N} \) be an input length parameter, \( g, f : \{0,1\}^n \rightarrow \{0,1\} \), and let \( D \) be a distribution over \( \{0,1\}^n \). The correlation of \( g \) and \( f \) in relation to \( D \) is \( \text{Corr}_D (g, f) = 2 \left| \frac{1}{2} − \Pr_{x \leftarrow D} [g(x) = f(x)] \right| \).

For a class \( \mathcal{G} \) of functions, \( \text{Corr}_D (\mathcal{G}, f) = \max_{g \in \mathcal{G}} \text{Corr}_D (g, f) \).

We are interested in correlations with the \( \text{MOD}_s \), defined following function:

Notation 3.25. Let \( n \in \mathbb{N} \) be an input length parameter, and \( s \in \mathbb{N} \). The function \( \text{MOD}_s^n : \{0,1\}^{3n} \rightarrow \{0,1\} \) is defined as: \( \text{MOD}_s^n(x) = 0 \) if and only if \( \left( \sum_{i=1}^{3n} x_i \equiv 0 \mod s \right) \). We use \( \text{MOD}_s \) to denote the family of functions \( \cup_{n \in \mathbb{N}} \text{MOD}_s^n \).

Theorem 3.26 ([LS11], Theorem 6 (rephrased)). For every constant depth parameter \( d \in \mathbb{N} \) there exist constants \( c, \epsilon \in (0,1) \), such that for every constant \( l \in \mathbb{N} \), there exists a minimal length parameter \( n_0 \in \mathbb{N} \) such that for every \( n \geq n_0 \), \( \text{Corr}_{\text{MOD}_s} (L_{3n,d,n',\oplus n'}^n, \text{MOD}_s^n) \leq 2^{-n^\epsilon} \), where \( \text{MOD}_s^n \) is the distribution induced by the following process: first pick a random bit \( b \in \{0,1\} \); if \( b = 0 \) pick \( x \in \{0,1\}^{3n} \) according to the distribution \( U_0^n \), otherwise pick \( x \in \{0,1\}^{3n} \) according to \( U_{1,2}^n \).

Next, we use Theorem 3.26 to show that \( \text{AC}^0 \) circuits, augmented with a sublinear number of \( \oplus \) gates, have a negligible advantage in distinguishing between random encodings of 0, and random encodings of either 1 or 2.

Corollary 3.27. For every constant depth parameter \( d \in \mathbb{N} \) there exist constants \( c, \epsilon \in (0,1) \), such that for every constant \( l \in \mathbb{N} \) there exists a minimal length parameter \( n_0 \in \mathbb{N} \) such that for every \( n \geq n_0 \) the encoding scheme \( \text{Enc}_d (\cdot, 1^n) \) of Definition 3.6.1 is \( (L_{3n,d,n',\oplus n'}^n, 2^{-n^\epsilon}) \)-leakage-indistinguishable.

We proceed to prove Corollary 3.27 in two steps. First, we show that Theorem 3.26 implies that \( \text{AC}^0 \) circuits, augmented with a sublinear number of \( \oplus \) gates, cannot distinguish between random encodings of 0, and random encodings of either 1 or 2. Second, we show that this implies indistinguishability of encodings of every pair of values in \( \{0,1,2\} \). The first step follows from the next lemma.

Lemma 3.6.4. Let \( \epsilon \in (0,1) \), \( n \in \mathbb{N} \), and \( \mathcal{G} \) be a class of functions from \( \{0,1\}^{3n} \) to \( \{0,1\} \). If \( \text{Corr}_{\text{MOD}_s} (\mathcal{G}, \text{MOD}_s^n) \leq \epsilon \) then \( U_0^n, U_{1,2}^n \) are \( (\mathcal{G}, \epsilon) \)-leakage-indistinguishable, where \( \text{MOD}_s^n \) is the distribution defined in Theorem 3.26.

Proof. Let \( g \in \mathcal{G} \). We first establish the connection between the probability \( p_g := \Pr_{x \leftarrow \text{MOD}_s^n} [g(x) = \text{MOD}_s^n (x)] \) that \( g \) computes \( \text{MOD}_s^n \) correctly, and the distinguishing advantage of \( g \):

\[
p_g = \Pr_{x \leftarrow \text{MOD}_s^n} [g(x) = \text{MOD}_s^n (x) | \text{MOD}_s^n (x) = 0] \cdot \Pr_{x \leftarrow \text{MOD}_s^n} [\text{MOD}_s^n (x) = 0] + \Pr_{x \leftarrow \text{MOD}_s^n} [g(x) = \text{MOD}_s^n (x) | \text{MOD}_s^n (x) = 1] \cdot \Pr_{x \leftarrow \text{MOD}_s^n} [\text{MOD}_s^n (x) = 1]
\]
observing that for $x \leftarrow D^n_3$, $\text{MOD}^n_3(x)$ is $0$ (or $1$) with probability half, and that
\[
\Pr_{x \leftarrow D^n_3} [g(x) = \text{MOD}^n_3(x) | \text{MOD}^n_3(x) = 0] = \Pr_{x \leftarrow U^n_0} [g(x) = 0]
\]
\[
\Pr_{x \leftarrow D^n_3} [g(x) = \text{MOD}^n_3(x) | \text{MOD}^n_3(x) = 1] = \Pr_{x \leftarrow U^n_{1,2}} [g(x) = 1]
\]

we get:
\[
p_g = \frac{1}{2} + \frac{1}{2} \left( \Pr_{x \leftarrow U^n_{1,2}} [g(x) = 1] - \Pr_{x \leftarrow U^n_0} [g(x) = 1] \right).
\]

By the assumption of the lemma,
\[
2 \left| \frac{1}{2} - p_g \right| = \text{Cerr}_{D^n_3}(g, \text{MOD}^n_3) \leq \epsilon.
\]

Therefore, we get:
\[
\left| \Pr_{x \leftarrow U^n_{1,2}} [g(x) = 1] - \Pr_{x \leftarrow U^n_0} [g(x) = 1] \right| \leq \epsilon.
\]

Next, we establish a connection between the distinguishing advantage of circuits between the following pairs of distributions: $U^n_0, U^n_{1,2}$ (over $6n$-bit vectors); $U^n_1, U^n_{1,2}$; and $U^n_0, U^n_1$ (over $3n$-bit vectors).

**Lemma 3.6.5.** Let $d, s, m \in \mathbb{N}$, and $c \in (0, 1)$ be a constant. If there exists an $n_0 \in \mathbb{N}$ such that for every $n \geq n_0$, $U^n_0, U^n_{1,2}$ are $(L_{3n,d,s,\oplus m}, \epsilon)$-leakage-indistinguishable for $\epsilon = 2^{-n^c}$, and $U^n_{2,0}, U^n_{2,1,2}$ are $(L_{6n,d+1,2s+1,\oplus 2m}, \epsilon)$-leakage-indistinguishable, then there exists an $n'_0$ such that for every $n \geq n'_0$, $U^n_0, U^n_1$ are $(L_{3n,d,s,\oplus m}, \sqrt{\epsilon})$-leakage-indistinguishable.

We will need the following notation and observation regarding the connection between $U^n_1, U^n_2$, and $U^n_{1,2}$.

**Notation 3.28.** Let $n \in \mathbb{N}$. For $\gamma \in \{0, 1, 2\}$, we use $S^n_0$ to denote $\text{supp}(U^n_0)$, $S^n_{1,2}$ to denote $\text{supp}(U^n_{1,2})$, and $k^n_\gamma$ to denote $|S^n_\gamma|$.

**Observation 3.6.6.** For every $n \in \mathbb{N}$, and every function $g : \{0, 1\}^{3n} \rightarrow \{0, 1\}$, by the law of total probability, and since $\Pr_{x \leftarrow U^n_{1,2}} [x \in S^n_1] = \Pr_{x \leftarrow U^n_{1,2}} [x \in S^n_2] = \frac{1}{2},$
\[
\Pr_{x \leftarrow U^n_{1,2}} [g(x) = 1] = \frac{1}{2} \left( \Pr_{x \leftarrow U^n_1} [g(x) = 1] + \Pr_{x \leftarrow U^n_2} [g(x) = 1] \right).
\]

**Proof of Lemma 3.6.5.** If the lemma does not hold, then there exist infinitely many $n$'s, for each of which $U^n_0, U^n_1$ are not $(L_{3n,d,s,\oplus m}, \sqrt{\epsilon})$-leakage-indistinguishable. Let $\epsilon' = \epsilon'(n) > \sqrt{\epsilon'}$ denote the maximal distinguishing advantage between $U^n_0, U^n_1$, let $\hat{D} = \{\hat{D}_n\}$ be a family of distinguishers obtaining this advantage, and let $\mathcal{N}$ be the infinite set of $n$'s for which $\hat{D}$ obtains this advantage. For $\gamma \in \{0, 1, 2\}$, let $p^n_\gamma := \Pr_{x \leftarrow U^n_\gamma} [\hat{D}_n(x) = 1].$
Assume first that \( p_0^n > p_1^n \) for infinitely many \( n \)'s in \( N \). There are two possible cases: either for infinitely many \( n \)'s in \( N \), \( p_0^n > p_2^n \); or \( p_0^n < p_2^n \) for infinitely many \( n \)'s in \( N \). In the first case, \( D \) has advantage at least \( \frac{\sqrt{\epsilon}}{2} > \frac{\sqrt{\epsilon}}{2} \geq \epsilon \) in distinguishing between \( U_0^n, U_{1,2}^n \); for every \( n \) such that \( p_0^n \geq p_2^n \) and \( p_0^n \geq p_1^n + \epsilon' \). Indeed, using Observation 3.6.6,

\[
\left| \Pr_{x \leftarrow U_0^n} [D_n (x) = 1] - \Pr_{x \leftarrow U_{1,2}^n} [D_n (x) = 1] \right| = \left| p_0^n - \frac{1}{2} (p_0^n + p_2^n) \right|
\]

using the case assumption that \( p_0^n \geq p_1^n, p_2^n \), this advantage is equal to:

\[
\frac{1}{2} (p_0^n - p_1^n) + \frac{1}{2} (p_0^n - p_2^n) \geq \frac{1}{2} (p_0^n - p_1^n) \geq \epsilon' / 2.
\]

Therefore, only the second case \( p_0^n < p_2^n \) remains, and Lemma 3.6.7 below shows that there exists an \( \hat{n}_0 \in \mathbb{N} \) such that for every such \( n \) which is greater than \( \hat{n}_0 \), \( U_0^{2n}, U_{1,2}^{2n} \) are distinguishable by \( L_{6n,d+1,2s+1,\epsilon;2m} \) circuits, with advantage at least \( \left( \frac{\epsilon \epsilon'}{6} \right) + E(n) > \left( \frac{\sqrt{\epsilon}}{6} \right) + E(n) = \epsilon + \frac{e^+ E(n)}{6} \), where \( E(n) = O\left( 2^{-3n} \right) \). Recall that \( \epsilon = 2^{-n/2} \), so \( E(n) = o(\epsilon) \), and let \( n' \in \mathbb{N} \) such that for every \( n \geq n' \), \( |E(n)| \leq \epsilon \) (notice that \( E(n) \) may be negative). Then for every \( n \geq \max \{ n', \hat{n}_0 \} \) in \( N \) such that \( p_0^n \geq p_2^n \geq p_1^n + \epsilon' \) (there are infinitely many such \( n \)'s by the case assumption), \( \epsilon + \frac{e^+ E(n)}{6} \geq \epsilon \), meaning that \( U_0^{2n}, U_{1,2}^{2n} \) can be distinguished in \( L_{6n,d+1,2s+1,\epsilon;2m} \) with advantage more than \( \epsilon \), a contradiction to the assumption of the lemma. Therefore, it cannot be the case that \( p_0^n \geq p_1^n + \epsilon' \) for infinitely many \( n \)'s in \( N \).

Assume now that \( p_0^n \geq p_1^n \) only for finitely many \( n \)'s in \( N \), i.e., \( p_0^n \geq p_0^n \) for infinitely many \( n \)'s in \( N \). If for infinitely many \( n \)'s in \( N \), \( p_0^n \geq p_0^n \) and \( p_0^n > p_0^n \), then the advantage of \( \hat{D}_n \) in distinguishing between \( U_0^n, U_{1,2}^n \) is at least

\[
\left| p_0^n - \frac{p_0^n + p_2^n}{2} \right| = \frac{p_1^n - p_0^n}{2} + \frac{p_2^n - p_0^n}{2} \geq \frac{p_1^n - p_0^n}{2} \geq \epsilon' / 2.
\]

The second case, where \( p_2^n < p_0^n < p_1^n \) for infinitely many \( n \)'s, follows from Lemma 3.6.7 in the same manner as before. \( \square \)

We now prove the lemma used in the proof of Lemma 3.6.5, for the case that for infinitely many \( n \)'s, \( p_2^n > p_0^n > p_1^n \) (or \( p_1^n > p_0^n > p_2^n \)). Notice that Lemma 3.6.7 uses the distributions \( U_0^{2n}, U_{1,2}^{2n} \) over \( 6n \)-bit vectors, and distinguishers over \( 3n \)-bit vectors.

**Lemma 3.6.7.** Let \( n, d, s, m \in \mathbb{N} \), \( \epsilon > 0 \), and \( \{ D_n \in L_{3n,d,s,\epsilon;2m} \}_{n \in \mathbb{N}} \). For \( \gamma \in \{ 0, 1, 2 \} \), denote \( p_{\gamma}^n := \Pr_{x \leftarrow U_0^n} [D_n (x) = 1] \). Then there exist error terms \( E^+(n), E^-(n) = O\left( 2^{-3n} \right) \), and a minimal length parameter \( n_0 \in \mathbb{N} \), such that the following holds for every \( n_0 \leq n \in \mathbb{N} \). If \( p_2^n > p_0^n > p_1^n \) and \( p_0^n - p_1^n \geq \epsilon \), then \( U_0^{2n}, U_{1,2}^{2n} \) are \( \left( L_{6n,d+1,2s+1,\epsilon;2m, \epsilon^2/6} + E^+ (n) \right) \)-distinguishable; and if \( p_2^n < p_0^n < p_1^n \) and \( p_0^n - p_1^n \geq \epsilon \), then \( U_0^{2n}, U_{1,2}^{2n} \) are \( \left( L_{6n,d+1,2s+1,\epsilon;2m, \epsilon^2/6} + E^- (n) \right) \)-distinguishable.
Proof. Let $D'_n$ be the distinguisher that interprets its input as a pair $(x, y)$ of 3n-bit vectors, and outputs $D_n(x) \land D_n(y)$. Notice that if $D_n \in \mathcal{L}_{3n, d, s, \oplus m}$, then $D'_n \in \mathcal{L}_{6n, d+1, l, s, \oplus m}$. We now analyze the advantage of $D'_n$ in distinguishing between $U_{0}^{2n}, U_{1}^{2n}$.

Using Lemma 3.6.9, $\Pr_{(x, y) \leftarrow U_{1/2}^{2n}}[D'_n(x, y) = 1] = \frac{(p_0^2)^2 + 2p_0^2p_1^2}{6} + E_0(n) + E'_0(n) \cdot p_2^2$, where $E_0(n), E'_0(n)$ are error terms, and $|E_0(n)|, |E'_0(n)| = O(2^{-3n})$. Using Lemma 3.6.10, $\Pr_{(x, y) \leftarrow U_{1/2}^{2n}}[D'_n(x, y) = 1] = \frac{2p_0^2p_1^2 + (p_1^2)^2 + 2p_1^2 + (p_1^2)^2}{6} + E_{1, 2}(n) + E'_1(n) \cdot p_2^2 + E''_1(n) \cdot (p_2^2)^2$, where $E_{1, 2}(n), E'_1(n), E''_1(n)$ are error terms, and $|E_{1, 2}(n)|, |E'_1(n)|, |E''_1(n)| = O(2^{-3n})$. Therefore, the distinguishing advantage $E_{D'_n}$ of $D'_n$ is at most:

$$E_{D'_n} = \Pr_{x \leftarrow U_{1/2}^{2n}}[D'_n(x, y) = 1] - \Pr_{x \leftarrow U_{0}^{2n}}[D'_n(x, y) = 1] = \frac{2p_0^2p_1^2 + (p_1^2)^2 + 2p_1^2 + (p_1^2)^2}{6} - 2(p_0^2)^2 - 4p_1^2p_2^2 + E(n) + E'(n) \cdot p_2^2 + E''(n) \cdot (p_2^2)^2$$

where $E(n), E'(n), E''(n)$ are error terms, and $|E(n)|, |E'(n)|, |E''(n)| = O(2^{-3n})$.

Thinking of $E_{D'_n}$ as a function of $p_2^2$, there exists an $n_0$ such that for every $n \geq n_0$, the minimal value of $E_{D'_n}$ is obtained when $p_2^2 = \frac{2p_0^2 - p_0^2 - 3E'(n)}{1 + 6E''(n)} \approx 2p_0^2 - p_0^2$. Let $n \geq n_0$, and assume first $p_2^2 > p_0^2$ and $p_0^2 < p_0^2 \geq \epsilon$. Then $\frac{2p_0^2 - p_0^2 - 3E'(n)}{1 + 6E''(n)} \approx 2p_0^2 - p_0^2 < p_0^2$, and in the domain $z \geq \frac{2p_0^2 - p_0^2 - 3E'(n)}{1 + 6E''(n)}$, $E_{D'_n}$ is monotonically increasing, so the minimal value of $E_{D'_n}$ in this section is obtained when $p_2^2 = p_0^2$ (since by the case assumption, $p_2^2 \geq p_0^2$), in which case $E_{D'_n}[p_2^2 = p_0^2] = \frac{(p_0^2 - p_0^2)^2}{6} + E(n) + E'(n) \cdot p_0^2 + E''(n) \cdot (p_0^2)^2 \geq \frac{\epsilon^2}{6} + E(n) + E'(n) \cdot p_0^2 + E''(n) \cdot (p_0^2)^2 = p_0^2 \in (0, 1) \frac{\epsilon^2}{6} + E'(n)$, where $E'(n) = O(2^{-3n})$, so $D'_n$ obtaining advantage $\delta' := \frac{\epsilon^2}{6} + E'(n)$ in distinguishing between $U_0^{2n}, U_1^{2n}$, where $E'(n) = O(2^{-3n})$.

Second, assume that $p_2^2 < p_0^2 < p_0^2$ and $p_0^2 \geq \epsilon$. Then $\frac{2p_0^2 - p_0^2 - 3E'(n)}{1 + 6E''(n)} \approx 2p_0^2 - p_0^2 > p_0$. Since by the case assumption $p_0^2 < p_0^2$ then in the domain $z \leq \frac{2p_0^2 - p_0^2 - 3E'(n)}{1 + 6E''(n)}$, the function is monotonically decreasing, so the minimal advantage is obtained when $p_0^2 = p_2^2$, and the rest of the analysis follows as in the previous case. \qed

We now state and prove the lemmas that were used in the proof of Lemma 3.6.7. We will need the following result about the values of $k_0^n, k_1^n, k_2^n$.

**Lemma 3.6.8.** Let $n \in \mathbb{N}$. Then $k_1^n = k_2^n = \frac{3n^2 + (1)^{n-1}}{3}$, and $k_0^n = \frac{2^{3n} + (1)^{n-1}}{3}$.

**Proof.** First notice that $k_1^n = |S_1^n| = |S_2^n| = k_2^n$, because the transformation that flips all the bits in the vector is a bijection between $S_1^n$ and $S_2^n$. Second, notice that $k_0^n = 2^{3n} - 2k_1^n$, and $\frac{2^{3n} + (1)^{n-1}}{3} = 2^{3n} - 2 \cdot \frac{3n^2 + (1)^{n-1}}{3}$, so it suffices to prove the claim for $k_1^n$, which we do by induction on $n$. The base case, for $n = 1$, holds because $k_1 = 3$ (since $S_1^1 = \{(0, 0, 1), (0, 1, 0), (1, 0, 0)\}$).

73
For the step, assume that the claim holds for \( i = n + 1 \) and we prove the claim for \( i = n \). Notice that the vectors in \( S_1^{n+1} \) can be divided into 3 subsets: vectors of the form “a vector in \( S_0^n \), concatenated with a vector in \( S_1^n \)” (there are \( k_1^{n-1} \cdot 3 \) such vectors); vectors of the form “a vector in \( S_1^n \), concatenated with a vector in \( S_1^n \)” (there are \( k_1^{n-1} \cdot 2 \) such vectors, because \( S_1^n = \{(0, 0, 0), (1, 1, 1)\} \)); and vectors of the form “a vector in \( S_2^n \), concatenated with a vector in \( S_2^n \)” (there are \( k_2^{n-1} \cdot 3 \) such vectors). Using the observations that \( k_1^n = k_2^n \) and \( k_0^n = 2^{3n} - 2k_1^n \), we get:

\[
k_1^{n+1} = 3k_0^n + 2k_1^n + 3k_2^n = 3 \cdot \left(2^{3n} - 2k_1^n\right) + 5k_1^n = 3 \cdot 2^{3n} - k_1^n.
\]

Using the induction hypothesis, \( k_1^n = \frac{2^{3n} + (-1)^{n-1}}{3} \), so:

\[
k_1^{n+1} = 3 \cdot 2^{3n} - k_1^n = 3 \cdot 2^{3n} - \frac{2^{3n} + (-1)^{n-1}}{3} = \frac{2^{3(n+1)} + (-1)^{(n+1)-1}}{3}.
\]

**Lemma 3.6.9.** Let \( D'_n, p_0^n, p_1^n, p_2^n \) be as defined in the proof of Lemma 3.6.7. Then

\[
\Pr_{(x,y)\leftarrow U_0^{2n}}[D'_n(x, y) = 1] = \frac{(p_0^n)^2 + 2p_1^n p_2^n}{3} + E_0(n) + E'_0(n) \cdot p_2^n
\]

where \( E_0(n), E'_0(n) \) are error terms, and \(|E_0(n)|, |E'_0(n)| = O(2^{-3n})\).

**Proof.** Since

\[
S_0^{2n} = \{(x, y) : x, y \in \{0, 1\}^{3n} \land (x, y \in S_0^n \lor x \in S_1^n, y \in S_0^n \lor x \in S_2^n, y \in S_1^n)\}
\]

then by the law of total probability,

\[
\Pr_{(x,y)\leftarrow U_0^{2n}}[D'_n(x, y) = 1] \text{ is equal to:}
\]

\[
= \left(\Pr_{x\leftarrow U_0^n}[D(x) = 1]\right)^2 \cdot \frac{|S_0^n|^2}{|S_0^{2n}|} + 2 \Pr_{x\leftarrow U_1^n}[D(x) = 1] \cdot \Pr_{x\leftarrow U_2^n}[D(x) = 1] \cdot \frac{|S_1^n|}{|S_0^{2n}|}.
\]

If \( n \) is even, then by Lemma 3.6.8: \( k_0^n = |S_0^n| = \frac{2^{3n} + 2}{3} \); \( k_2^n = |S_2^n| = \frac{2^{6n+2}}{3} \); and \( k_1^n = |S_1^n| = \frac{2^{3n} - 1}{3} \). Therefore,

\[
\frac{|S_0^n|^2}{|S_0^{2n}|} = \left(\frac{2^{3n} + 2}{3}\right)^2 = \frac{2^{6n} + 2^{4n+2} + 2}{3} = \frac{1}{3} \cdot \left(1 + \frac{2^{3n} + 2}{2^{3n} + 2}\right) = \frac{1}{3} + O(2^{-3n})
\]

74
\[
\frac{|S_1^n| \cdot |S_2^n|}{|S_0^{2n}|} = \frac{|S_1^n|^2}{|S_0^{2n}|} = \frac{(\frac{2^{3n} - 1}{3})^2}{\frac{2^{2n} + 2}{3}} = \frac{1}{3} \cdot \frac{2^{6n} - 2^{3n} + 1}{2^{6n} + 2} = \frac{1}{3} - O(2^{-3n})
\]

Otherwise, \( n \) is odd, and by Lemma 3.6.8: \( k_0^n = |S_0^n| = \frac{2^{2n} - 2}{3}; \) \( k_0^n = |S_0^n| = \frac{2^{2n} - 2}{3}; \) and \( k_1^n = |S_1^n| = \frac{2^{3n} - 1}{3}. \) Similar calculations give:

\[
\frac{|S_0^n|^2}{|S_0^{2n}|} = \frac{1}{3} - O(2^{-3n}), \quad \frac{|S_1^n| \cdot |S_2^n|}{|S_0^{2n}|} = \frac{1}{3} + O(2^{-3n})
\]

Consequently,

\[
\Pr_{(x,y) \leftarrow U_{1,2}^n}[D'_n(x,y) = 1] = \frac{(p_0^n)^2 + 2p_0^n p_2^n + (p_2^n)^2}{3} + E_0(n) + E'_0(n) \cdot p_2^n
\]

where \( E_0, E'_0 \) are error terms, and \( |E_0(n)|, |E'_0(n)| = O(2^{-3n}). \)

**Lemma 3.6.10.** Let \( D'_n, p_0^n, p_1^n, p_2^n \) be as defined in the proof of Lemma 3.6.7. Then \( \Pr_{(x,y) \leftarrow U_{1,2}^n}[D'_n(x,y) = 1] \) is equal to

\[
\frac{2p_0^n p_1^n + (p_1^n)^2 + 2p_0^n p_2^n + (p_2^n)^2}{6} + E_{1,2}(n) + E'_{1,2}(n) \cdot p_2^n + E''_{1,2}(n) \cdot (p_2^n)^2
\]

where \( E_{1,2}, E'_{1,2}, E''_{1,2} \) are error terms, and \( |E_{1,2}(n)|, |E'_{1,2}(n)|, |E''_{1,2}(n)| = O(2^{-3n}). \)

**Proof.** The proof is similar to the proof of Lemma 3.6.9: \( S_{1,2}^{2n} \) is the disjoint union of the sets

\[
S_{1,2}^{2n} = \{(x,y) \in \{0,1\}^{6n} : x, y \in \{0,1\}^{3n} \land (x \in S_0^n, y \in S_1^n \lor x \in S_1^n, y \in S_0^n \land x, y \in S_2^n)\}
\]

and

\[
S_{2,2}^{2n} = \{(x,y) \in \{0,1\}^{6n} : x, y \in \{0,1\}^{3n} \land (x \in S_0^n, y \in S_2^n \lor x \in S_2^n, y \in S_0^n \land x, y \in S_1^n)\}
\]

where a random element in \( S_{1,2}^{2n} \) belongs to each of these sets with probability \( \frac{1}{2}. \)

Therefore, by the law of total probability, \( \Pr_{(x,y) \leftarrow U_{1,2}^n}[D'_n(x,y) = 1] \) is equal to:

\[
\frac{1}{2} \left( 2 \Pr_{x \leftarrow U_0^n}[D(x) = 1] \cdot \Pr_{x \leftarrow U_1^n}[D(x) = 1] \cdot \frac{|S_0^n| \cdot |S_1^n|}{|S_0^{2n}|} + \left( \Pr_{x \leftarrow U_2^n}[D(x) = 1] \right)^2 \cdot \frac{|S_0^n|^2}{|S_1^{2n}|} \right)
\]

\[
+ \frac{1}{2} \left( 2 \Pr_{x \leftarrow U_0^n}[D(x) = 1] \cdot \Pr_{x \leftarrow U_2^n}[D(x) = 1] \cdot \frac{|S_1^n| \cdot |S_2^n|}{|S_0^{2n}|} + \left( \Pr_{x \leftarrow U_1^n}[D(x) = 1] \right)^2 \cdot \frac{|S_1^n|^2}{|S_2^{2n}|} \right)
\]

If \( n \) is even, then by Lemma 3.6.8: \( k_0^n = |S_0^n| = \frac{2^{3n} - 1}{3}; \) \( k_1^n = |S_1^n| = \frac{2^{3n} - 1}{3}; \) and
\(k_1^{2n} = |S_1^{2n}| = k_2^{2n} = |S_2^{2n}| = \frac{2^{6n}-1}{3}. \) Therefore,

\[
\frac{|S_0^n| \cdot |S_2^n|}{|S_1^{2n}|} = \frac{|S_0^n| \cdot |S_1^n|}{|S_2^{2n}|} = \frac{2^{3n} + 2 \cdot 2^{3n-1}}{2^{2n}-1} = \frac{1}{3} \cdot \frac{2^{6n} + 2^{3n} - 2}{2^{6n} - 1} = \frac{1}{3} + O(2^{-3n})
\]

and

\[
\frac{|S_0^n|^2}{|S_2^{2n}|} = \frac{|S_1^n|^2}{|S_2^{2n}|} = \left(\frac{2^{3n} - 2}{2^{2n}-1}\right)^2 = \frac{1}{3} \cdot \frac{2^{6n} - 2^{3n+1} + 1}{2^{6n} - 1} = \frac{1}{3} \left(1 - \frac{2^{3n+1} - 2}{2^{6n} - 1}\right) = \frac{1}{3} - O(2^{-3n})
\]

Otherwise, \(n\) is odd, and by Lemma 3.6.8: \(k_0^n = |S_0^n| = \frac{2^{3n}-2}{3}; k_1^n = |S_1^n| = \frac{2^{3n}+1}{3}; \)
and \(k_2^n = |S_2^n| = k_3^n = |S_2^{2n}| = \frac{2^{6n}-1}{3}. \) Therefore,

\[
\frac{|S_0^n| \cdot |S_2^n|}{|S_1^{2n}|} = \frac{|S_0^n| \cdot |S_1^n|}{|S_2^{2n}|} = \frac{2^{3n} - 2 \cdot 2^{3n+1}}{2^{2n}-1} = \frac{1}{3} \cdot \frac{2^{6n} - 2^{3n} - 2}{2^{6n} - 1} = \frac{1}{3} - O(2^{-3n})
\]

and

\[
\frac{|S_0^n|^2}{|S_2^{2n}|} = \frac{|S_1^n|^2}{|S_2^{2n}|} = \left(\frac{2^{3n} + 1}{2^{2n}-1}\right)^2 = \frac{1}{3} \cdot \frac{2^{6n} + 2^{3n+1} + 1}{2^{6n} - 1} = \frac{1}{3} \left(1 + \frac{2^{3n+1} + 2}{2^{6n} - 1}\right) = \frac{1}{3} + O(2^{-3n})
\]

Consequently, \(\Pr_{(x,y) \leftarrow U_n^{\mathbb{Z}_2}}[D'_n (x, y) = 1]\) is equal to

\[
\frac{2p_0^n p_1^n + (p_1^n)^2 + 2p_0^n p_2^n + (p_2^n)^2}{6} + E_{1,2} (n) + E_{1,2}' (n) + E_{1,2}'' (n) \cdot (p_2^n)^2
\]

where \(E_{1,2} (n), E_{1,2}' (n), E_{1,2}'' (n)\) are error terms, and \(|E_{1,2} (n)|, |E_{1,2}' (n)|, |E_{1,2}'' (n)| = O(2^{-3n}). \)

According to Lemma 3.6.5, indistinguishability of \(U_0^n, U_1^n\) and \(U_0^{2n}, U_1^{2n}\) implies indistinguishability of \(U_0^n, U_1^n\). We now show that this implies that \(E_3\) is leakage indistinguishable (against a slightly weaker family of leakage functions).

**Lemma 3.6.11.** Let \(n, d, s, m \in \mathbb{N}\), and \(\epsilon = \epsilon (n) > 0\). If there exists an \(n_0 \in \mathbb{N}\) such that for every \(n \geq n_0\), \(U_0^n, U_1^n\) are \((\mathcal{L}_{3n,d,s,\oplus m}, \epsilon)\)-leakage-indistinguishable, then for every \(n \geq n_0\), \(E_3 (\cdot, 1^n) = (\mathcal{L}_{3n,d-1,s-3n,\oplus m}, 2\epsilon)\)-leakage-indistinguishable.

**Proof.** We show first that \(\text{Enc}_3 (0, 1^n)\) or \(\text{Enc}_3 (2, 1^n)\) are \((\mathcal{L}_{3n,d-1,s-3n,\oplus m}, \epsilon)\)-leakage-indistinguishable for every \(n \geq n_0\). Otherwise, there exists an \(n > n_0\), and a distinguisher \(D_n \in \mathcal{L}_{3n,d-1,s-3n,\oplus m}\) that achieves advantage \(\epsilon' > \epsilon\) in distinguishing between the distributions \(\text{Enc}_3 (0, 1^n)\), \(\text{Enc}_3 (2, 1^n)\). We define \(D'_n\) to apply negation gates on its inputs, and run \(D_n\). Then \(D'_n \in \mathcal{L}_{3n,d,s,\oplus m}\), and notice that since the encoding length is divisible by 3, and the transformation \(v \rightarrow \overline{v}\) is 1:1 and onto (where \(\overline{v}\) denotes the vector obtained by coordinate-wise negating \(v\)) then: if \(v \leftarrow \text{Enc}_3 (0, 1^n)\) then \(\overline{v} \leftarrow \text{Enc}_3 (0, 1^n)\); and if \(v \leftarrow \text{Enc}_3 (1, 1^n)\) then


$v \leftarrow \text{Enc}_3(2, 1^n)$. Therefore, $|\text{Pr}[D'_n(\text{Enc}(0, 1^n)) = 1] - \text{Pr}[D'_n(\text{Enc}(1, 1^n)) = 1]| = |\text{Pr}[D_n(\text{Enc}(0, 1^n)) = 1] - \text{Pr}[D_n(\text{Enc}(2, 1^n)) = 1]| = \epsilon' > \epsilon$, contradicting the assumption of the lemma.

Second, since for every $n \geq n_0$, $\text{Enc}_3(0, 1^n)$, $\text{Enc}_3(2, 1^n)$ are $(\mathcal{L}_{3n, d-1, s-3n, \pm m}, \epsilon)$-leakage-indistinguishable, and $\text{Enc}_3(0, 1^n), \text{Enc}_3(1, 1^n)$ are $(\mathcal{L}_{3n, d, s, \pm m}, \epsilon)$-leakage-indistinguishable, then using the triangle inequality $\text{Enc}_3(1, 1^n), \text{Enc}_3(2, 1^n)$ are $(\mathcal{L}_{3n, d-1, s-3n, \pm m}, 2\epsilon)$-leakage-indistinguishable.

We are finally ready to prove Corollary 3.27.

Proof of Corollary 3.27. Let $d' = d + 2$, let $\epsilon, c$ be the constants for which Theorem 3.26 holds for depth parameter $d'$, and we set $c' = \frac{\epsilon}{2}$, and $\epsilon' = \frac{\epsilon}{2}$.

Given $l$, let $l' = l + 1$, and let $n_0$ be the minimal length parameter for which Theorem 3.26 holds with parameters $d', l'$. Let $n'_0$ be such that for every $n \geq n'_0$, $2(n^l + 3n) + 1 \leq n'^l$, $2n'^l \leq n^l$, and $2\sqrt{\pi} \cdot 2^{-\frac{n^l}{2}} \leq 2^{-n'^l}$. Let $n''_0$ be the minimal length parameter whose existence is guaranteed by Lemma 3.6.5 for the length parameter $\max\{n_0, n'_0\}$, constant $c$, depth parameter $d + 2$, size parameter $s = n^l + 3n$, and parity gate bound $m = n'^l$.

Let $\tilde{n}_0 = \max\{n_0, n'_0, n''_0\}$.

We show that the corollary holds for minimal length parameter $\tilde{n}_0$ and constants $c', \epsilon'$. Indeed, for every $n \geq \tilde{n}_0$ Theorem 3.26 guarantees that $\text{Corr}_{\mathbb{F}_3^n}\left(\mathcal{L}_{3n, d+2, 2(n^l + 3n) + 1, \pm 2n'^l}, \text{MOD}_2^n\right) \leq 2^{-n^l}$ (since $n \geq n_0$ and $n \geq n'_0$). By Lemma 3.6.4, this implies that for every $n \geq \tilde{n}_0$, $U^n_0, U^n_{1, 2}$ are $(\mathcal{L}_{3n, d+1, n^l + 3n, \pm n'^l}, 2^{-n^l})$-leakage-indistinguishable, and $U^{2n}_0, U^{2n}_{1, 2}$ are $(\mathcal{L}_{6n, d+2, 2(n^l + 3n) + 1, \pm 2n'^l}, 2^{-n^l})$-leakage-indistinguishable. By Lemma 3.6.5, for every $n \geq \tilde{n}_0$, $U^n_0, U^n_{1, 2}$ are $(\mathcal{L}_{3n, d+1, n^l + 3n, \pm n'^l}, \sqrt{\pi} \cdot 2^{-\frac{n^l}{2}})$-leakage-indistinguishable (because $n \geq n''_0$). By Lemma 3.6.11, $E_3(\cdot, 1^n)$ is $(\mathcal{L}_{3n, d, n'^l, \pm n'^l}, \sqrt{\pi} \cdot 2^{-\frac{n^l}{2}})$-leakage-indistinguishable. Since $\tilde{n}_0 \geq n'_0$, $E_3(\cdot, 1^n)$ is $(\mathcal{L}_{3n, d, n'^l, \pm n'^l}, 2^{-n'^l})$-leakage-indistinguishable.

3.6.2 A Proof of Theorem 3.23

In this section we prove Theorem 3.23. The proof combines Proposition 3.5.9 and Corollary 3.27. More specifically, we instantiate the encoding scheme $E$ of Proposition 3.5.9 with an “appropriate” encoding scheme. Recall that Proposition 3.5.9 uses the compiler of Construction 3.20, which is based on the arithmetic circuit compiler of Construction 3.15, and is obtained by mapping an encoding according to a linear-structured encoding scheme over $\mathbb{F}_3$ to an encoding over $\{0, 1\}$ in which every field element is represented by a 2-bit string (e.g., using the mapping $E_3$ in Definition 3.5.3). The next lemma states that the leakage-indistinguishability of an encoding is preserved under this mapping (up to a constant additive loss in the depth, and a constant multiplicative loss in the size, of the circuits computing the leakage functions).
Lemma 3.6.12. Let $n, k, m, d, s \in \mathbb{N}$, $\epsilon > 0$, and $E = (\text{Enc}, \text{Dec})$ be an $(L_{n,d+1,s+n,\oplus m}^k, \epsilon)$-leakage-indistinguishable encoding scheme. Then the encoding scheme $E_b = (E_b \circ \text{Enc}, \text{Dec} \circ E_b^{-1})$ of Construction 3.20 is $(L_{2n,d,s,\oplus m}^k, \epsilon)$-leakage-indistinguishable.

Remark 3.29. Recall that the $E_b$ operates on elements in $\mathbb{F}_3^n$, where each field element is represented using a 2-bit string. (See Remark 3.21 for additional details.)

Proof. Denote $\text{Enc}_b = E_b \circ \text{Enc}$. If $E_b = (E_b \circ \text{Enc}, \text{Dec} \circ E_b^{-1})$ is not $(L_{2n,d,s,\oplus m}^k, \epsilon)$-leakage-indistinguishable, then there exists a pair $w_1, w_2 \in \mathbb{F}_3^n$, and a leakage function $\ell_b \in L_{2n,d,s,\oplus m}^k$ such that $\text{SD}(\ell_b(w_1), \ell_b(w_2)) > \epsilon$, where $w_i \leftarrow \text{Enc}_b(w_i, 1^n)$ for $i = 1, 2$ (notice that the output length of $\text{Enc}_b(\cdot, 1^n)$ is $2n$). We show a leakage function $\ell \in L_{n,d+1,s+4n,\oplus m}^k$ that achieves advantage more than $\epsilon$ in distinguishing between $v_1 \leftarrow \text{Enc}(w_1, 1^n)$ and $v_2 \leftarrow \text{Enc}(w_2, 1^n)$. On input $z = (z_1, \ldots, z_n) \in \{0,1\}^n \subseteq \mathbb{F}_3^n$, $\ell$ first computes the string $z' = (0, z_1, 0, z_2, \ldots, 0, z_n)$ (this can be done in $L_{n,1,4n,0}$ by adding $n$ constant gates and $2n$ output gates), and outputs $\ell_b(z')$. Then $\ell \in L_{n,d+1,s+4n,\oplus m}^k$, and notice that if $z$ is distributed according to $\text{Enc}(\gamma, 1^n)$ for some $\gamma \in \mathbb{F}_3^n$, then $z'$ is distributed according to $\text{Enc}_b(\gamma, 1^n)$. Therefore, $\text{SD}(\ell(v_1), \ell(v_2)) = \text{SD}(\ell_b(v_1'), \ell_b(v_2')) = \text{SD}(\ell_b(w_1), \ell_b(w_2)) > \epsilon$. \hfill $\square$

The proof of Theorem 3.23 now follows from Corollary 3.27, Proposition 3.5.9, and Lemma 3.6.13 by an appropriate choice of the parameters. We first restate the theorem.

Theorem (Theorem 3.23, restated). For input length parameter $n$, leakage length bound $\hat{n} = \hat{n}(n)$, size bound $s = s(n)$, output length bound $k = k(n)$, parity gate bound $m = m(n)$, and depth bound $d$, let $L_{\hat{n},d,s,\oplus m}^k = \bigcup_{n \in \mathbb{N}} L_{\hat{n}(n),d,s(n),\oplus m(n)}^k$, where $L_{\hat{n},d_0,s_0,\oplus m_0}$ denotes the class of boolean circuits of input length $\hat{n}$ over $\sim$ gates and unbounded fan-in $\land, \lor$ gates, whose depth, size, output length, and number of parity gates are bounded by $d_0, s_0, k_0, m_0$, respectively. Then for every positive constants $d, c$, polynomials $k, m$, and polynomial size bound $s' = s'(n)$, there exists a polynomial $S(n)$, such that there exists a SAT-respecting $(L_{\hat{n},d,s,\oplus m}^k, s'(n), 2^{-n^c})$-relaxed LRCC, which on input a circuit $C : \{0,1\}^n \rightarrow \{0,1\}$ of size $|C| \leq s'(n)$ outputs a circuit $\hat{C}$ of size $|\hat{C}| \leq S(n)$.

Proof of Theorem 3.23. Let $C, |C| \leq s'$ be the circuit to be compiled. Let $d' = d + 33$, and denote by $\tilde{c}, \tilde{c}$ the constants whose existence is guaranteed by Corollary 3.27 for the depth parameter $d' + 1$. Let $(\text{Enc}, \text{Dec})$ be the encoding scheme obtained when Construction 3.15 is applied to $(\text{Enc}_3, \text{Dec}_3)$ as the internal encoding scheme. Let $\text{Enc}_b(\cdot, 1^n, 1^{1m}) = E_b \circ \text{Enc}(\cdot, 1^n, 1^{1m}), \text{Dec}_b(\cdot, 1^n, 1^{1m}) = \text{Dec}(\cdot, 1^n, 1^{1m}) \circ E_b^{-1}$ (here, $E_b, E_b^{-1}$ are the mapping and reverse mapping of Definition 3.5.3, and $\text{Enc}_3, \text{Dec}_3$ are the encoding and decoding algorithms of the encoding scheme of Definition 3.6.1). Since $E_3 = (\text{Enc}_3, \text{Dec}_3)$ is linear-structured, then Proposition 3.5.9 guarantees that there exist constants $z, c' \in \mathbb{N}$ such that when Construction 3.20 is instantiated with
Let $p := c + \frac{4c}{c\epsilon} + 1$ and $p' = p + 4c + 7$, then Corollary 3.27 guarantees that there exists a minimal length parameter $\tilde{\sigma}_0$ such that for every $\sigma \geq \tilde{\sigma}_0$, $\text{Enc}_3(\cdot, 1^{\sigma})$ is $(L_{3d, d' + 1, \sigma', \tilde{\sigma}, s}, 2^{-\tilde{\sigma}})$-leakage-indistinguishable. Let $\sigma'_0$ be such that for every $\sigma \geq \sigma'_0$, $z + (z')^c \leq \sigma$, and $\sigma''_0$ be such that for every $\sigma \geq \sigma''_0$, $c' \sigma 2^{-\tilde{\sigma}} \leq 1$.

We take $S(n) := S'(n, k, s', m) := z' \cdot (\max \{\tilde{\sigma}_0, \sigma_0', \sigma''_0\})^4 (4n^{c'}k's'm)^{2c + \frac{2p}{\epsilon}} = \text{poly}(n, k, s', m) = \text{poly}(n)$ (the last equality holds because $s', m, k = \text{poly}(n)$), and $\sigma = (4n^{c'}k's'm)^{\frac{2}{\epsilon}} \cdot \max \{\tilde{\sigma}_0, \sigma_0', \sigma''_0\}$, and instantiate the SAT-respecting circuit-compiler $(\text{Comp}_b, E_b) = (\text{Enc}_b, \text{Dec}_b)$ of Construction 3.20, with size parameter $s'$, and leakage class $\mathcal{L} = L_{d, S''(n), \epsilon m} (\epsilon$ will be set later), setting $t = \sigma$ and $t_m = |C|$. Then for every $C$ such that $|C| \leq s'$, the compiled circuit $\hat{C}$ satisfies $|\hat{C}| \leq z' |C|^2 \cdot \sigma^4 \leq z'(s')^2 \cdot \max \{\tilde{\sigma}_0, \sigma_0', \sigma''_0\}^4 (4n^{c'}k's'm)^{\frac{2}{\epsilon}} \leq z' \cdot (\max \{\tilde{\sigma}_0, \sigma_0', \sigma''_0\})^3 (4n^{c'}k's'm)^{2c + \frac{2p}{\epsilon}} = S'(n, k, s', m) = S(n)$. Moreover, since the length of encodings generated by the encoding scheme was taken to be $\sigma \geq \max \{\tilde{\sigma}_0, \sigma_0', \sigma''_0\} \geq \tilde{\sigma}_0$, then $\text{Enc}_3(\cdot, 1^{\sigma})$ is $(L_{3d, d' + 1, \sigma', \tilde{\sigma}, s}, 2^{-\tilde{\sigma}})$-leakage-indistinguishable.

Notice that

$$S'(n) + z \cdot (s)^2 = (z')^c \cdot (\max \{\tilde{\sigma}_0, \sigma_0', \sigma''_0\})^4 (4n^{c'}k's'm)^{2c + \frac{2p}{\epsilon}} + z \cdot (s)^2$$

which can be upper bounded by

$$( (z')^c + z) \cdot (\max \{\tilde{\sigma}_0, \sigma_0', \sigma''_0\})^4 (4n^{c'}k's'm)^{2c + \frac{2p}{\epsilon} + 2}$$

which (by the choice of $\sigma$) is at most $\sigma \cdot \sigma^{4c} \sigma^{\frac{2p}{\epsilon}} \leq \epsilon, \tilde{\epsilon} < 1 \sigma^{1 + 4c + p}$, so

$$k \cdot (S'(n) + z \cdot (s)^2 \sigma^4) + 1 \leq k \leq \sigma (S'(n) + z \cdot (s)^2) \cdot \sigma^6 \leq \sigma^{p + 4c + 7} = \sigma^{p'}$$

Moreover, $km + 1 \leq \sigma^\tilde{\epsilon}$ (because $\tilde{\epsilon} < 1$ and $s' \geq 2$). Using Lemma 3.6.13, this implies that $\text{Enc}_3(\cdot, 1^{\sigma})$ is $(L_{3d, d', s''(n), \epsilon m} (\cdot, s'), 2^{-\tilde{\epsilon}})$-leakage-indistinguishable. Therefore, the compiled circuit is $(L_{d, s''(n), \epsilon m}, c', 2^{-\tilde{\epsilon}} \cdot \sigma)$-relaxed-leakage-resilient (here, we also use the fact that $s' \leq \sigma$, which holds because $n, k, m \geq 1$ and $\tilde{\epsilon}, \tilde{\epsilon} < 1$). We conclude the proof by noticing that since $\tilde{\epsilon}, \tilde{\epsilon} < 1$ then $c' \cdot 2^{\frac{k}{\tilde{\epsilon}} - \sigma^\tilde{\epsilon}} \cdot \sigma \leq 2^{k - 2n^{c'}k's'm} \cdot c' \sigma 2^{-\frac{2p}{\epsilon}}$. Since $s', m \geq 1$, and $\sigma \geq \sigma''_0$, we can upper bound this expression by $2^{-n^{c'}}$. \[\square\]

\[\text{Note:} \text{Proposition 3.5.9 requires that } L \circ \text{UnBBool} \left(33, z \tilde{n}^4 (\sigma) \cdot (s')^2 \right) \subseteq L_3, \text{ but in } E_3, \tilde{n} (t) = O (t) \text{ for every } t, \text{ so we can replace } \tilde{n} (\sigma) \text{ with } O (\sigma).\]
The next lemma was used in the proof of Theorem 3.23.

**Lemma 3.6.13.** Let \( n,n',k,d,s,m \in \mathbb{N} \), let \( \epsilon > 0 \), and let \( f : \{0,1\}^{n'} \to \{0,1\}^n \) be a randomized function. If \( f \) is \((L_{n+1,k+1,\oplus;km+1},\epsilon)\)-leakage-indistinguishable, then \( f \) is \((L_{n,d,s,\oplus;m},\epsilon \cdot 2^k)\)-leakage-indistinguishable.

**Proof.** If \( f \) is not \((L_{n,d,s,\oplus;m},\epsilon \cdot 2^k)\)-leakage-indistinguishable, then there exist \( y,z \in \{0,1\}^n \), and a function \( \ell \in L_{n,d,s,\oplus;m}^k \), such that \( SD(\ell(f(y)),\ell(f(z))) > \epsilon \cdot 2^k \). Since \( \ell \in L_{n,d,s,\oplus;m}^k \), then there exist \( k \) circuits \( C_1,\ldots,C_k : \{0,1\}^n \to \{0,1\} \) of depth at most \( d \) and size at most \( s \), with unbounded fan-in and fan-out \( \land,\lor,\oplus \) gates, out of which at most \( m \) are \( \oplus \) gates, where \( C_i \) computes \( \ell_i \), the \( i \)th output bit of \( \ell \). Let \( X_y,Z_z \) be random variables over \( \{0,1\}^k \) distributed according to \( \ell(f(y)),\ell(f(z)) \), then \( SD(X_y,Z_z) > \epsilon \cdot 2^k \). Therefore, by Lemma 2.1.1 there exists an \( \alpha \in \{0,1\}^k \) such that \( SD(\alpha^T X_y,\alpha^T Z_z) > \epsilon \). Let \( \hat{\ell} : \{0,1\}^n \to \{0,1\} \) be the function computed by the following circuit \( C \). The first (at most) \( d \) layers contain the circuits \( C_1,\ldots,C_k \) in parallel. Layer \( d+1 \) contains a single \( \oplus \) gate, whose inputs are the outputs of the circuits \( \{C_i : \alpha_i = 1\} \). Then \( \hat{\ell} \in L_{n,d+1,k+1,\oplus;km+1} \), and notice that \( \hat{\ell}(f(y)) \) is distributed according to \( \alpha^T X_y \), while \( \hat{\ell}(f(z)) \) is distributed according to \( \alpha^T Z_z \). Therefore, \( SD(\hat{\ell}(f(y)),\hat{\ell}(f(z))) > \epsilon \), contradicting the \((L_{n,d+1,k+1,\oplus;km+1},\epsilon)\)-leakage-indistinguishability of \( f \).

### 3.7 A Passive-Secure Multiparty LRCC

In this section we describe a passive-secure multiparty LRCC. Our starting point is the LRCC \( \text{Comp}^{\text{FRRTV}},\text{E}^{\text{FRRTV}} \) of [FRR+14], described in Construction 3.11. Recall that the compiler uses an internal linear-structured encoding scheme \( E^{\text{in}} = (\text{Enc}^{\text{in}},\text{Dec}^{\text{in}}) \), where given a circuit \( C : \mathbb{F}^n \to \mathbb{F}^k \), the compiled circuit \( C^{\text{FRRTV}} \) takes both standard inputs, which are encodings (according to \( \text{Enc}^{\text{in}} \)) of the inputs of \( C \) and masking inputs, which are well-formed vectors (namely, encodings of 0 according to \( \text{Enc}^{\text{in}} \)), and are used to randomize the internal computations in \( C^{\text{FRRTV}} \).

Our passive-secure LRCC (Construction 3.31) outputs a compiled circuit \( \hat{C} \) that takes encoded inputs from \( m \) parties. The input of each party consists of an encoding of its input to \( C \), and encodings of masking inputs for \( C^{\text{FRRTV}} \). \( \hat{C} \) obtains the masking inputs needed for \( C^{\text{FRRTV}} \) by summing the encodings of masking inputs provided by all parties. We first describe the sub-circuit that combines the masking inputs.

**Construction 3.30.** Let \( m \in \mathbb{N}, t \in \mathbb{N} \) be a security parameter, \( r \) be a randomness length parameter, and \( E^{\text{in}} = (\text{Enc}^{\text{in}},\text{Dec}^{\text{in}}) \) be a linear-structured encoding scheme that on length-\( n \) inputs, and given security parameter \( t \), outputs encodings of length \( \hat{n}(n,t) \). The Mask combiner \( C^{\text{sum}} \) is defined as follows. \( C^{\text{sum}} : (\mathbb{F}^n)^m \to \mathbb{F}^{\hat{n}(r,t)} \), where \( C^{\text{sum}}(x_1,\ldots,x_m) = \sum_{i=1}^m x_i \). (The sum is computed component-wise using \( \hat{n}(r,t) \) log \( m \)-depth sub-circuits consisting of \( O(m) \) addition gates.)
Construction 3.31. Let \( m, n, t, t_{in} \in \mathbb{N} \), and \( \mathbb{F} \) be a finite field. Let \((\text{Comp}^{\text{FRRTV}}, \text{E}^{\text{FRRTV}})\) denote the LRCC of Construction 3.11, with underlying encoding scheme \( \text{E}^{\text{in}} = (\text{Enc}^{\text{in}}, \text{Dec}^{\text{in}}) \) which outputs encodings of length \( \hat{n}(n,t) \). We denote \( \hat{n}_1 = \hat{n}(1,t) \), and \( \hat{n}'(n,t,t_{in}) := \hat{n}(n + (\hat{n}_1 + 1) \cdot t_{in}, t) \). The Passive-secure multiparty LRCC (\( \text{Comp}, \text{E} = (\text{Enc}, \text{Dec}) \)) is defined as follows.

- For every \( x \in \mathbb{F}^n \), and \( t, t_{in} \in \mathbb{N} \), \( \text{Enc}(x, 1^t, 1^{t_{in}}) = \text{Enc}^{\text{in}}((x, 0^{(\hat{n}_1 + 1) \cdot t_{in}}), 1^t) \), and for every \((x, m) \in \mathbb{F}^{\hat{n}(n,t)} \times \mathbb{F}^{\hat{n}'(\hat{n}_1 + 1) \cdot t_{in}}, t)\), \( \text{Dec}((x, m), 1^t, 1^{t_{in}}) = \text{Dec}^{\text{in}}(x, 1^t) \).

- \( \text{Comp} \), on input the number of parties \( m \in \mathbb{N} \), a security parameter \( t \), and an \( m \)-party circuit \( C : (\mathbb{F}^n)^m \rightarrow \mathbb{F}^k \), outputs a circuit \( \hat{C} : (\mathbb{F}^{\hat{n}'(t,n)}, \mathbb{F})^m \rightarrow \mathbb{F}^k \), where \( \hat{C}(\langle x_1, m_1 \rangle, \ldots, \langle x_m, m_m \rangle) = \text{C}^{\text{FRRTV}}((x_1, \ldots, x_m), \text{C}_{\text{sum}}(m_1, \ldots, m_m)) \).

Instantiating Construction 3.31 over the binary field, we obtain the following result.

Theorem 3.32 (Passive-secure multiparty LRCC). Let \( m \in \mathbb{N} \) denote the number of parties, \( n \in \mathbb{N} \) be an input length parameter, \( t \in \mathbb{N} \) be a security parameter, \( \mathbb{F} \) be a finite field, \( \epsilon(t) : \mathbb{N} \rightarrow \mathbb{R}^+ \) be an error parameter, \( S(n) : \mathbb{N} \rightarrow \mathbb{N} \) be a size function, and \( L, L_{\text{E}} \) be families of functions. Let \( \text{E}^{\text{in}} \) be an encoding scheme that on input length \( n \), and security parameter \( t \), outputs encodings of length \( \hat{n}(n,t) \), and is linear-structured over \( \mathbb{F} \). If \( \text{E}^{\text{in}} \) is \((L_{\text{E}}, \epsilon(t))\)-leakage-indistinguishable, and \( L_{\text{E}} = L \circ \text{Shallow}_{\mathbb{F}}(\log m + 4, O(m \cdot \hat{n}^2(1,t) \cdot S(n))) \), then there exists a passive-secure \( m \)-party \((L, S(n), \epsilon(t) \cdot (\hat{n}(1,t) + 1) \cdot S(n)) \)-LRCC over \( \mathbb{F} \). Moreover, on input an \( m \)-party circuit \( C \), the compiler outputs a circuit of size at most \( O(m \cdot \hat{n}^2(1,t) \cdot |C|) \).

We first sketch the leakage-privacy argument. At a high level, the simulator \( \hat{\text{Sim}} \) of Construction 3.31 uses the simulator \( \text{Sim}^{\text{FRRTV}} \) whose existence follows from the leakage-resilience of the LRCC of [FRR+14] (Lemma 3.3.3). The main difference between \( \text{Sim}^{\text{FRRTV}} \) and \( \hat{\text{Sim}} \), is that \( \text{Sim}^{\text{FRRTV}} \) only simulates the wires of \( \text{C}^{\text{FRRTV}} \), whereas \( \hat{\text{Sim}} \) should simulate the entire wire values of \( \hat{C} \), including the internals of \( C_{\text{sum}} \). However, given a wire assignment to the wires of \( \text{C}^{\text{FRRTV}} \), a simulated wire assignment to the internals of \( C_{\text{sum}} \) can be generated in \( \text{Shallow}_{\mathbb{F}}(\log m + 1, O(\hat{n}^2(1,t) \cdot S(n) \cdot m)) \). This accounts for the loss in the complexity of leakage functions that our passive-secure LRCC can withstand.

Proof of Theorem 3.32. We show that Construction 3.31 satisfies the properties of Theorem 3.32. The fact that it has the correct syntax follows directly from the construction. We now show that it satisfies the leakage-privacy property, by describing the simulator \( \hat{\text{Sim}} \). Let \( \text{Sim}^{\text{FRRTV}} \) denote the simulator whose existence follows from Lemma 3.3.3. The simulator \( \hat{\text{Sim}} \), on input \( C : (\mathbb{F}^n)^m \rightarrow \mathbb{F}^k \) of size at most \( S(n) \), and \( y = C(x_1, \ldots, x_m) \), operates as follows.

- Runs \( \text{Sim}^{\text{FRRTV}}(C, y) \) to obtain simulated wire values \( W_S \) for the wires of \( \text{C}^{\text{FRRTV}} \).
- Let \( m \) denote the masking inputs of \( \text{C}^{\text{FRRTV}} \), as they appear in \( W_S \).
• Picks \( m - 1 \) random, well-formed vectors \( \mathbf{z}_1, \ldots, \mathbf{z}_{m-1} \), and sets \( \mathbf{z}_m = \mathbf{m} - \sum_{i=1}^{m-1} \mathbf{z}_i \).

• Uses \( \mathbf{z}_1, \ldots, \mathbf{z}_m \) to compute the internal wire in the computation of \( C_{\text{sum}} \). (Notice that these can be computed coordinate-wise using \( \hat{n}((\hat{n}(1,t) + 1) \cdot |C|, t) \) sub-circuits, each consisting of \( O(m) \) gates arranged in \( \log m \) layers.) Denote the wire values generated during this step by \( W_{\text{sum}} \).

• Outputs \( (W_{\text{sum}}, W_S) \). (The order in which we have written these wire values is for notational convenience only. \( \text{Sim} \) outputs these wire values in the same order as they appear in \( \hat{\mathcal{C}} \).

We now bound the advantage obtained by leakage functions in \( \mathcal{L} \) in distinguishing between the actual and simulated wire values. For that, we first present an alternative method of describing \( \text{Sim} \). For every set of \( m - 1 \) well-formed vectors \( \mathbf{z}_1, \ldots, \mathbf{z}_{m-1} \in \text{supp}(\text{Enc}^\text{in}(0^{(\hat{n}(1,t)+1)|C|}, 1^t)) \), define the function \( f_{\mathbf{z}_1, \ldots, \mathbf{z}_{m-1}} \) that on input a wire assignment \( W_S \) for the wires of \( C_{\text{FRRTV}} \), in which \( \mathbf{m} \) denotes the encoding of the masking inputs, is evaluated as follows:

- Computes \( \mathbf{z}_m = \mathbf{m} - \sum_{i=1}^{m-1} \mathbf{z}_i \) (computed in \( \text{Shallow}_{\mathcal{F}}(1, \hat{n}((\hat{n}(1,t) + 1) \cdot |S(n), t)) \) because all values except the input \( \mathbf{m} \) are hard-wired into \( f_{\mathbf{z}_1, \ldots, \mathbf{z}_{m-1}} \).

- Uses \( \mathbf{z}_1, \ldots, \mathbf{z}_m \) to compute an assignment \( W_{\text{sum}} \) to the internal wires of \( C_{\text{sum}} \), when evaluated on \( \mathbf{z}_1, \ldots, \mathbf{z}_m \) (this can be computed in \( \text{Shallow}_{\mathcal{F}}(\log m, O(\hat{n}((\hat{n}(1,t) + 1) \cdot |S(n), t) \cdot m)) \) using \( \hat{n}((\hat{n}(1,t) + 1) \cdot |S(n), t) \) trees consisting of \( O(m) \) gates arranged in \( \log m \) layers).

- Outputs \( (W_S, W_{\text{sum}}) \) (arranged in the same order as these wires appear in \( \hat{\mathcal{C}} \)).

Let \( F = \{f_{\mathbf{z}_1, \ldots, \mathbf{z}_{m-1}} : \mathbf{z}_1, \ldots, \mathbf{z}_{m-1} \in \text{supp}(\text{Enc}^\text{in}(0^{(\hat{n}(1,t)+1)|C|}, 1^t))\} \), then \( F \subseteq \text{Shallow}_{\mathcal{F}}(\log m + 1, \hat{n}((\hat{n}(1,t) + 1) \cdot |S(n), t) \cdot m)) \), and notice that for \( f \leftarrow F \), the output of \( \text{Sim} \) is equally distributed to the distribution obtained by evaluating \( f \) on the output of \( \text{Sim}^\text{FRRTV} \).

For every \( \ell \in \mathcal{L} \) and every \( f \in F \),

\[
\ell \circ f \in \mathcal{L} \circ \text{Shallow}_{\mathcal{F}}(\log m + 1, O(\hat{n}((\hat{n}(1,t) + 1) \cdot |S(n), t) \cdot m))
\]

Moreover, when \( f \leftarrow F \) then

\[
\text{Sim}(C, y) \equiv f(\text{Sim}^\text{FRRTV}(C, y))
\]

and

\[
[\hat{\mathcal{C}}, \hat{x}_1, \ldots, \hat{x}_m] \equiv f([C^\text{FRRTV}, \hat{x}'])
\]

where for every \( 1 \leq i \leq m \), \( \hat{x}_i \leftarrow \text{Enc}^\text{in}((x_i, 0^{(\hat{n}(1,t)+1)|C|}), 1^t) \), and \( \hat{x}' \leftarrow \)
Construction 3.31 has the following property. For every \( n,t,t_{\text{in}} \in \mathbb{N}, \) every \( m_1, \ldots, m_m \in \mathbb{N}^{(n(1,t)+1)\cdot t_{\text{in}}}, \) and every \( m \)-party circuit \( C : (\mathbb{F}^n)^m \rightarrow \mathbb{F}^k, \) there

\[
\text{Enc}^\text{in} \left( \left( (x_1, \ldots, x_m), 0^{(n(1,t)+1)\cdot |C|}, 1^t \right) \right).
\]

Therefore, when \( f \leftarrow F \) then for every \( \ell \in \mathcal{L}, \)

\[
\text{SD} \left( \ell \left( \text{Sim} (C, y) \right), \ell \left[ \hat{C}, \hat{x}_1, \ldots, \hat{x}_m \right] \right)
\]

is equal to

\[
\text{SD} \left( \left( \ell \circ f \right) \left( \text{Sim}^{\text{FRRTV}} (C, y) \right), \left( \ell \circ f \right) \left[ C^{\text{FRRTV}}, \hat{x}' \right] \right).
\]

Since \( \hat{n} (n, t) = n \cdot \hat{n} (1, t) \) (by the properties of Construction 3.11) then \( \hat{n} ((\hat{n} (1, t) + 1) \cdot S (n), t) = O \left( \hat{n}^2 (1, t) \cdot S (n) \right) \). Therefore,

\[
\mathcal{L} \circ \text{Shallow}_F (\log m + 1, O (\hat{n} ((\hat{n} (1, t) + 1) \cdot S (n), t) \cdot m)) \circ \text{Shallow}_F (\log 3, O \left( m \cdot \hat{n}^2 (1, t) \right))
\]

is contained in

\[
\mathcal{L} \circ \text{Shallow}_F (\log m + 4, O \left( m \cdot \hat{n}^2 (1, t) \cdot S (n) \right)) = \mathcal{L}_E.
\]

Combining this with the fact that \( C^{\text{FRRTV}} (\hat{x}') = y \) (by the correctness of Construction 3.11), and with Lemmas 3.3.3 and 3.4.6, we can bound the statistical distance by \( \epsilon (t) \cdot (\hat{n} (1, t) + 1) \cdot S (n) \). Moreover, the statistical distance does not increase even if we concatenate \( y \) to both random variables, namely

\[
\text{SD} \left( \left( \ell \left( \text{Sim} (C, y) \right), y \right), \left( \ell \left[ \hat{C}, \hat{x}_1, \ldots, \hat{x}_m \right], y \right) \right) \leq \epsilon (t) \cdot (\hat{n} (1, t) + 1) \cdot S (n)
\]

Finally, we show that \( |\hat{C}| = O \left( m \cdot \hat{n}^2 (1, t) \cdot |C| \right) \), \( |C^{\text{FRRTV}}| = O \left( \hat{n}^2 (1, t) \cdot |C| \right) \), because every gate in \( C \) is replaced by a gadget of size at most \( O \left( \hat{n}^2 (1, t) \right) \). Moreover, \( C^{\text{FRRTV}} \) uses at most \( (\hat{n} (1, t) + 1) \cdot |C| \) masking inputs (since each gadget uses as most \( \hat{n} (1, t) + 1 \) masking inputs), each consisting of \( \hat{n} (1, t) \) field elements. Therefore, the input to \( C_{\text{sum}} \) has length \( O \left( m \cdot \hat{n}^2 (1, t) \cdot |C| \right) \), and so \( |C_{\text{sum}}| = O \left( m \cdot \hat{n}^2 (1, t) \cdot |C| \right) \).

### 3.7.1 The Effect of Ill-Formed Masking Inputs in Construction 3.31

In this section we discuss the effect of ill-formed masking inputs (namely, masking inputs that are not encodings of zeros) on the computation in the compiled circuit \( \hat{C} \) generated by the passive-secure multiparty LRCC of Construction 3.31. These properties will be used in Section 3.8 to construct a multiparty LRCC which is secure against malicious parties.

We first show that evaluating \( \hat{C} \) with ill-formed masking inputs is equivalent to evaluating the original circuit \( C \) under an appropriate additive attack \( A \) (which is determined by the identity of the masking inputs used in \( \hat{C} \)). This is formalized in the next lemma.

**Lemma 3.7.1** (Ill-formed masking inputs correspond to additive attacks).

**Construction 3.31 has the following property.** For every \( n,t,t_{\text{in}} \in \mathbb{N}, \) every \( m_1, \ldots, m_m \in \mathbb{N}^{(n(1,t)+1)\cdot t_{\text{in}}}, \) and every \( m \)-party circuit \( C : (\mathbb{F}^n)^m \rightarrow \mathbb{F}^k, \) there
exists an additive attack $A_m$ on $C$ such that for every $x_1, \ldots, x_m \in \mathbb{F}^n$, and every $\hat{x}_i \in \text{supp} \left( \text{Enc}^m \left( (x_i, m_i), 1 \right) \right)$ for $1 \leq i \leq m$, $\hat{C} (\hat{x}_1, \ldots, \hat{x}_m) = C^{A_m} (x_1, \ldots, x_m)$. Moreover, there exists a PPT algorithm $\text{Alg}$ such that $\text{Alg} (m_1, \ldots, m_m) = A_m$.

Proof. Let $(m_1, \ldots, m_m)$ denote the collections of masking inputs provided to $\hat{C}$, and let $m = \sum_{i=1}^m m_i$ (the sum is computed component-wise). For every $1 \leq i \leq m$, since $m_i$ was encoded using the onto encoding scheme $\text{E}^m$ ($\text{E}^m$ is linear-structured, and therefore also onto according to Remark 3.3), then there exists some masking input $m_i$ such that $\text{Dec}^m (m_i) = m_i$. Moreover, since $\text{E}^m$ is linear-structured, then $m$ encodes $m := \sum_{i=1}^m m_i$.

We define the additive attack $A_m$ as follows. Let $G$ be a gadget in $\hat{C}$, let $g$ denote the gate which $G$ emulates, and let $m_G$ denote the restriction of $m$ to the masking inputs used by $G$. Then:

- If $g$ is a $+$, $-$, $\text{const}_f$, $\text{id}$ or $\text{rand}$ gate, and $m_G$ encodes $e \in \mathbb{F}$, then $G$ with masking input $m_G$ emulates the gate $g'$ obtained by adding $e$ to the output of $g$. Therefore, we set the attack (in $A_m$) on the output wire of $g$ to $e$.

- If $g$ is a $\text{copy}$ gadget, and $m_G$ encodes $(e_1, e_2)$, then $G$ with masking input $m_G$ emulates the gate that on input $x$ outputs $(x + e_1, x + e_2)$. Therefore, we set the attack (in $A_m$) on the output wires of $g$ to $e_1, e_2$ respectively.

- If $g$ is a $\times$ gate, and $m_G$ encodes $(e_1, \ldots, e_{n(1,t)+1})$, then $G$ with masking input $m_G$ emulates the gate that on input $x, y$ outputs $xy + e_{n(1,t)+1} + \sum_{i=1}^{\hat{n}(1,t)} e_i d_i^{n(1,t)}$. In this case, we set the attack (in $A_m$) on the output wire of $g$ to $e_{n(1,t)+1} + \sum_{i=1}^{\hat{n}(1,t)} e_i d_i^{n(1,t)}$.

Then $\hat{C} (\hat{x}_1, \ldots, \hat{x}_m) = C^{A_m} (x_1, \ldots, x_m)$. Moreover, notice that to compute the distribution of $A_m$, one needs only $m_1, \ldots, m_m$, since the identity of the additive attack is determined by the values encoded by the masking inputs, rather than by the masking inputs themselves.

Next, we show that as long as one of the collections $m_i$ of masking inputs is well-formed, then the circuit $\hat{C}$ generated by Construction 3.31 is leakage-resilient. However, recall that when the compiled circuit $\hat{C}$ is evaluated with ill-formed masking inputs, then it emulates $C^A$, for some additive attack $A$ that is determined by the masking inputs. Then leakage resilience holds when the simulator is given the output of $C^A$ (instead of the output of $C$). Formally,

**Lemma 3.7.2** (Leakage-resilience with ill-formed masking inputs). Let $n, m, t \in \mathbb{N}$, $\mathbb{F}$ be a finite field, $\varepsilon (t) : \mathbb{N} \rightarrow \mathbb{R}^+$, $S (n) : \mathbb{N} \rightarrow \mathbb{N}$ be a size function, and $\mathcal{L}, \mathcal{L}_E$ be families of functions such that $\mathcal{L}_E = \mathcal{L} \circ \text{ShallowF} (\log m + 5, O (n^2 (1,t) \cdot S (n) \cdot m))$. Let $\text{E}^m$ be an encoding scheme that on inputs of length $n$, and given security parameter $t$, outputs
encodings of length \( n (n,t) \), which is linear-structured over \( \mathbb{F} \), and \((L_E, \epsilon (t))\)-leakage-indistinguishable. Then there exists a PPT simulator \( \text{Sim} \) such that the following holds for all sufficiently large \( n \)'s, every \( m \)-party circuit \( C : (\mathbb{F}^n)^m \rightarrow \mathbb{F}^k \) of size \( |C| \leq S(n) \), every subset \( B \subset [m] \), every \( \ell \in L \), every \( x_1, \ldots , x_m \in \mathbb{F}^n \), every \( m_i \in \mathbb{F}^{(\hat{n}(1,t)+1)|C|} \), \( i \in B \), and every \( (x_i, m_i) \in \text{supp}(\text{Enc}^n((x_i, m_i), 1^t)) \), \( i \in B \). For every \( i \notin B \) define \( m_i = 0^{(\hat{n}(1,t)+1)|C|} \), let \( \hat{m} = (m_1, \ldots , m_m) \in (\mathbb{F}^{(\hat{n}(1,t)+1)|C|})^m \), and let \( A \) denote the additive attack that corresponds to \( \hat{m} \), as defined in Lemma 3.7.1. For every \( i \notin B \) let \( \hat{x}_i \leftarrow \text{Enc}^n((x_i, m_i), 1^t) \), and for every \( i \in B \) let \( \hat{x}_i = (x_i, m_i) \). Then

\[
\text{SD}(\ell (\text{Sim} (C, C^A (x_1, \ldots , x_m), B, (x_i, m_i)_{i\in B})), \ell (\hat{C}, (\hat{x}_1, \ldots , \hat{x}_m),)) \leq \epsilon ' (n,t)
\]

where \( \hat{C} = \text{Comp} (C) \), \( \text{Comp} \) is the algorithm described in Construction 3.31, and \( \epsilon ' (n,t) = \epsilon (t) \cdot (\hat{n}(1,t)+1) \cdot S(n) \).

To prove Lemma 3.7.2, we use the methodology of \cite{FRR+14}: we first show that the gadgets are re-randomizing and locally-reconstructible even when using ill-formed masking inputs (as long as these encodings are random); we then use a lemma similar to \cite[Lemma 13]{FRR+14} to show that the compiled circuit is leakage-resilient.

We first formally define the notion of re-randomization when gadgets are evaluated with ill-formed masking inputs.

**Definition 3.7.3** (Gadget re-randomization with ill-formed masking inputs). Let \( G \) be a gadget emulating a gate \( g \), and operating on encodings generated according to some encoding scheme \( E = (\text{Enc}, \text{Dec}) \). We say that \( G \) is re-randomizing with masking input mask if for every standard input \( x \) for \( g \), and every encoding \( x \in \text{supp}(\text{Enc}(x)) \),

the distribution \( G(x, m) \), when \( m \leftarrow \text{Enc}(\text{mask}) \), is random over \( \text{supp}(\text{Enc}(g^A_{\text{mask}}(x))) \), where \( A_{\text{mask}} \) is the additive attack defined by \( \text{mask} \), whose existence is guaranteed by Lemma 3.7.1. We say that \( G \) is re-randomizing with ill-formed masking inputs if it is re-randomizing with masking input \( \text{mask} \), for every \( \text{mask} \).

The next lemma states that the gadgets are re-randomizing even when using ill-formed masking inputs.

**Lemma 3.7.4** (Rerandomization with ill-formed masking inputs). Let \( E^\text{in} \) be an encoding scheme that is linear-structured over \( \mathbb{F} \). Then the gadgets described in Construction 3.10 are re-randomizing with ill-formed masking inputs.

**Proof.** Let \( G \) be a gadget in \( \hat{C} \), let \( g \) denote the gate which \( G \) emulates, and let \( m \) denote the masking inputs of \( G \), where \( m \leftarrow \text{Enc}^\text{in}(\text{mask}) \) for some values \( \text{mask} \). Since \( E^\text{in} \) is linear-structured then Lemma 3.7.1 guarantees that for every input \( x \) encoding value \( x \), \( G(x, m) \in \text{supp}(\text{Enc}^\text{in}(g^A_{\text{mask}}(x))) \). Moreover, since the output bundles of \( G \) are masked with (i.e., obtained through summation with) random encodings (because \( m \) is a random encoding of \( \text{mask} \)), and the encoding scheme is linear-structured, then \( G(x, m) \) is a random encoding of \( \text{Enc}^\text{in}(g^A_{\text{mask}}(x)) \).

85
Next, we show that the gadgets are locally reconstructible with ill-formed masking inputs (as in Definition 3.3.1).

**Lemma 3.7.5** (Local-reconstruction with ill-formed masking inputs). Let \( t \in \mathbb{N} \) be a security parameter, \( \mathbb{F} \) be a finite field, \( \mathcal{L}, \mathcal{L}_E \) be families of functions, and \( \epsilon (t) : \mathbb{N} \to \mathbb{R}^+ \). Denote \( \hat{n}_1 = \hat{n}(1,t) \), then the following holds for the gadgets described in Construction 3.10.

- Positive and negative gadgets are \((\mathcal{L}, 0)\)-reconstructible with ill-formed masking inputs by \( \text{Shallow}_\mathbb{F}(2, O(\hat{n}_1)) \).
- copy, id, and const\(_{A}\) gadgets are \((\mathcal{L}, 0)\)-reconstructible with ill-formed masking inputs by \( \text{Shallow}_\mathbb{F}(1, O(\hat{n}_1)) \).
- Let \( m \in \mathbb{N} \). If \( E^\text{in} \) is \((\mathcal{L}_E, \epsilon(t))\)-leakage-indistinguishable, and \( \mathcal{L}_E = \mathcal{L} \circ \text{Shallow}_\mathbb{F}(4, O(\hat{n}_1)) \), then the \( \times \) gadget is \((\mathcal{L}, \hat{n}_1 \cdot \epsilon(t))\)-reconstructible with ill-formed masking inputs by \( \text{Shallow}_\mathbb{F}(2, O(\hat{n}_1^2)) \).

The proof is similar to [FRR\(^+14\), Lemmas 5-7, and Lemma 9]. We include it here for completeness.

**Proof.** Let \( G \) be one of the gadgets described in Construction 3.10, and let \( \text{mask} \) denote its masking inputs. If \( G \in \{+, -, \text{const}_f, \text{copy}, \text{id}, \text{rand}\} \) then any pair \((x, y)\) that is plausible for \( G \) with masking input \( \text{mask} \) completely determines the identity of the encodings \( m \in \text{supp}(\text{Enc}^\text{in}(\text{mask}, 1^t)) \) used in \( G \), and therefore also the internal wires. Therefore, the reconstructed wire values are identically distributed to the actual wire values (and so the actual and reconstructed wire values are \((\mathcal{L}, 0)\)-leakage-indistinguishable for any family \( \mathcal{L} \) of leakage functions). Moreover, if \( G \in \{+, -\} \) then the internal wires are computable from \( x, y \) in \( \text{Shallow}_\mathbb{F}(2, O(\hat{n}_1)) \) (if \( x = (a, b) \) then first compute \( q = a + b \), then compute \( r = y - q \)); and if \( G \in \{\text{const}_f, \text{copy}, \text{id}, \text{rand}\} \) then the internal wires are computable from \( x, y \) in \( \text{Shallow}_\mathbb{F}(1, O(\hat{n}_1)) \) (since the output is simply the sum of the input, or a constant vector, with the masking inputs).

If \( G \) is a \( \times \) gadget, then for every masking input \( \text{mask} \), the family of local reconstructors \( \text{Rec}_{\text{mask}} = \{\text{rec}_{z_1, \cdots, z_{\hat{n}_1}} : z_1, \cdots, z_{\hat{n}_1} \in \text{supp}(\text{Enc}^\text{in}(\mathbb{F}, 1^t))\} \), where for every \( z_1, \cdots, z_{\hat{n}_1} \in \text{supp}(\text{Enc}^\text{in}(\mathbb{F}, 1^t)) \), the function \( \text{rec}_{z_1, \cdots, z_{\hat{n}_1}} \) on input \( x = (a, b), y \) operates as follows:

- Let \( U \) denote the matrix whose columns are \( z_1, \cdots, z_{\hat{n}_1} \). Compute the vector \( q = Ud^{\hat{n}_1} \), and fix the internal wires of this computation into \( \text{rec}_{z_1, \cdots, z_{\hat{n}_1}} \). (Notice that these wires are fixed in advance, and therefore do not “cost” towards the complexity of the function.)

- Computes the matrix \( B = ab^T \), and the vector \( r^{\hat{n}_1+1} = y - q \), using \( O(\hat{n}_1^2) \) gates arranged in parallel.

86
Computes the vectors $r^1, \ldots, r^{\hat{n}_1}$ as the columns of the matrix $S = U - B$, using $\hat{n}_1^2$ gates arranged in parallel.

Notice that $\text{rec}_{z^1, \ldots, z^{\hat{n}_1}} \in \text{Shallow}_F \left(2, O \left(\hat{n}_1^2\right)\right)$.

It remains to show that when $\text{rec}_{z^1, \ldots, z^{\hat{n}_1}} \leftarrow \text{Rec}_{\text{mask}}$ (namely, when $z^1, \ldots, z^{\hat{n}_1}$ are random encodings of random values in $\mathbb{F}$) then the actual and reconstructed wire values are $(\mathcal{L}, \hat{n} (1, t) \cdot \epsilon (t))$-leakage-indistinguishable, conditioned on the event that the masking inputs encode $\text{mask} = (\text{mask}_1, \ldots, \text{mask}_{\hat{n}_1+1})$. The only difference between the real and reconstructed wires is that in the real world, $U$ is computed as $B + S$, where the columns of $S$ are random encodings of $(\text{mask}_1, \ldots, \text{mask}_{\hat{n}_1})$, and $r^{\hat{n}_1+1}$ is a random encoding of $\text{mask}_{\hat{n}_1+1}$, whereas in the reconstructed wires, the columns of $U$ are encodings of random values, and $c$ is computed as $y - q$.

Notice that in both cases, $r^{\hat{n}_1+1}$ is completely determined given $U, y$, so it remains to consider only the distribution over $U$ in both cases. Alternatively, we can consider the distribution over $S$ in both cases (since $U, S$ are computed from one another in the same way in both cases).

We show that the wire values $W$ of the internal wires of $G$, when using a matrix $S$ whose columns are random encodings of $(\text{mask}_1, \ldots, \text{mask}_{\hat{n}_1})$, are $(\mathcal{L}, \hat{n} (1, t) \cdot \epsilon (t))$-leakage-indistinguishable from the values $W'$ of these wires, when $G$ uses a matrix $S$ whose columns are random vectors. This would suffice since when the columns of $S$ are random vectors, then so is the distribution of $U$, conditioned on $U = B + S$ (that is, $U$ will be distributed as in the reconstructed wires).

Assume towards negation that there exists an $\ell \in \mathcal{L}$ such that $\text{SD} (\ell (W), \ell (W')) > \hat{n} (1, t) \cdot \epsilon (t)$. We show that there exists a leakage function $\ell_E \in \mathcal{L}_E$ that obtains advantage more than $\epsilon (t)$ in distinguishing between a random encoding of some fixed value, and a random encoding of a random field element $e \in_R \mathbb{F}$. We define a sequence of hybrids $H_i, i = 0, \ldots, \hat{n} (1, t)$, where $H_i$ is an assignment to the internal wires of $G$, when it has inputs $x$ and output $y$, in which the first $i$ columns of $S$ are random encodings of $(\text{mask}_1, \ldots, \text{mask}_i)$, and the rest are random encodings of random values. Then $W_0 = W', W_{\hat{n} (1, t)} = W$, and by the negation assumption there exists an $0 \leq i_0 < \hat{n} (1, t)$ such that $\text{SD} (\ell (H_{i_0+1}), \ell (H_{i_0})) > \epsilon (t)$. Using an averaging argument, we can maintain the statistical distance while fixing all the wires in $H_{i_0+1}, H_{i_0}$, except for the following: the $i_0$'th column of $S$, the $i_0$'th column of $U$, $q$, and part of the internal wires in its computation, and $r^{\hat{n} (1, t)+1}$. More specifically, since the $i_0$'th column of $U$ has not been fixed, then for every coordinate $j$ of $q$, the value $U_{j, i_0}$ needed for the computation is missing (but all other values needed for this computation can be fixed).

We now describe $\ell_E$. On input $v$ (which is either an encoding of $\text{mask}_i$, or of a random $e \in_R \mathbb{F}$), $\ell_E$ sets the $i_0$'th column of $S$ to be $v$, and using $v$, and the fixed wires, computes the following.

- Computes the $i_0$'th column $U_{i_0}$ of $U$ as $U_{i_0} = B_{i_0} + v$, where $B_{i_0}$ denotes the $i_0$ column of $B$. (This is computable using $\hat{n} (1, t)$ gates arranged in parallel.)
• Uses $U_{i_0}$ to compute the internal wires in the computation of $q$. (Each of these values is obtained by summing a value that has already been fixed, with the product of a fixed coordinate of $d^{(1, t)}$ with a coordinate of $U_{i_0}$. Computing the products can be done using $O (\hat{n} (1, t))$ constant and $\times$ gates arranged in parallel, followed by $O (\hat{n} (1, t))$ gates arranged in parallel to compute the sum, so this entire computation is in $\text{Shallow}_F (2, O (\hat{n} (1, t)))$.)

• Computes $r^{\hat{n} (1, t)+1} = y - q$. (This is computable using $\hat{n} (1, t)$ gates arranged in parallel.)

• Evaluates $\ell$ on the wire values obtained by combining the fixed wires with $U_{i_0}$, $q, r^{\hat{n}1}$.

Then $\ell_E \in \{ \ell \} \circ \text{Shallow}_F (4, O (\hat{n} (1, t)))$, namely $\ell_E \in L_E$, and notice that if $v$ is a random encoding of $\text{mask}_{i_0}$ then $\ell_E (v) = \ell (W_{i_0})$, and if $v$ is a random encoding of $e \in R F$ then $\ell_E (v) = \ell (W_{i_0+1})$, contradicting the $(L_E, \epsilon (n))$-leakage-indistinguishability of $E^n$.

To prove that the compiled circuit $C'$ is leakage-resilient even when using ill-formed masking inputs, we will use the following lemma, whose proof is similar to the proof of [FRR+14, Lemma 13].

**Lemma 3.7.6.** Let $n \in \mathbb{N}$ be an input length parameter, $t \in \mathbb{N}$ be a security parameter, $\epsilon (t), \epsilon' (t) : \mathbb{N} \rightarrow \mathbb{R}^+$ be statistical distance parameters, $L_E, L', L'_{\text{rec}}$ be function families, $E = (\text{Enc, Dec})$ be an encoding scheme, and $r (n)$ denote the number of masking inputs used in a circuit compiled using Construction 3.11. Assume that:

• $E$ is $(L_E, \epsilon (t))$-leakage-indistinguishable.

• Every gadget $G$ is re-randomizing with ill-formed masking inputs.

• Every gadget $G$ is $(L', \epsilon' (t))$-reconstructible by $L'_{\text{rec}}$ with ill-formed masking inputs.

Then for every family $L$ of functions such that $L \subseteq L'$, and $L \circ (2 \times L'_{\text{rec}}) \subseteq L_E$ there exists a PPT algorithm $\text{Sim}$ such that the following holds. For all sufficiently large $n$’s, every arithmetic circuit $C$ over $F$ of input length $n$, every $B \subseteq [n]$, every $x = (x_1, \ldots , x_n) \in F^n$, every $x_i \in \text{supp} (\text{Enc}_{\text{in}} (x_i, 1^t))$, every $\text{mask} \in F^{r (n)}$, and every $\ell \in L$,

$$\text{SD} \left( \ell (\text{Sim} (C, C_{\text{mask}} (x), \text{mask}, B, (x_i)_{i \in B})) , \ell \left[ \hat{C}, \hat{x} \right] \right) \leq |C| \cdot (\epsilon (t) + \epsilon' (t))$$

where $\hat{x} = (\hat{x}_1, \ldots , \hat{x}_n, \text{mask})$ such that for every $i \in B$, $\hat{x}_i = x_i$; for every $i \notin B$, $\hat{x}_i \leftarrow \text{Enc}_{\text{in}} (x_i, 1^t)$; $\text{mask} \leftarrow \text{Enc}_{\text{in}} (\text{mask}, 1^t)$; and $A_{\text{mask}}$ denotes the additive attack whose existence is guaranteed by Lemma 3.7.1.

Using Lemma 3.7.6, we can now prove Lemma 3.7.2.
Proof of Lemma 3.7.2. Denote the simulator whose existence follows from Lemma 3.7.6 by Sim\textsuperscript{in}. The simulator Sim, on input C, C\textsuperscript{A} (x\textsubscript{1}, ..., x\textsubscript{m}), B, and (x\textsubscript{i}, m\textsubscript{i})\textsubscript{i\in B} operates as follows.

- For every i \in B, computes m\textsubscript{i} = Dec\textsuperscript{in} (m\textsubscript{i}, 1\textsuperscript{t}) to determine the (possibly ill-formed) masking input encoded by m\textsubscript{i}, and sets m = \sum\textsubscript{i\in B} m\textsubscript{i}.

- Runs Sim\textsuperscript{in} (C, C\textsuperscript{A} (x\textsubscript{1}, ..., x\textsubscript{m}), m, B, (x\textsubscript{i})\textsubscript{i\in B}) to obtain an assignment W\textsuperscript{in} to the wire values of \hat{C}\textsuperscript{in} = Comp\textsuperscript{FRRTV} (C), when evaluated with: (1) inputs x\textsubscript{i} for every i \in B; inputs x\textsubscript{i} \leftarrow Enc\textsuperscript{in} (x\textsubscript{i}, 1\textsuperscript{t}) for every i \notin B; and (3) masking inputs m\textsubscript{i} \leftarrow Enc\textsuperscript{in} (m, 1\textsuperscript{t}). Let m\textsubscript{in} \in \mathbb{F}\textsuperscript{\hat{n}(\hat{n}(1, t)+1)\cdot |C|\cdot t} denote the masking inputs used by \hat{C}\textsuperscript{in}, as these appear in W\textsuperscript{in}.

- Let i\textsubscript{0} be the minimal index not in B. For every i\textsubscript{0} \neq i \notin B, picks m\textsubscript{i} \leftarrow Enc\textsuperscript{in} (0\textsuperscript{\hat{n}(1, t)+1\cdot |C|\cdot t}, 1\textsuperscript{t}), and sets m\textsubscript{i\textsubscript{0}} = m\textsubscript{in} - \sum\textsubscript{\neq i\in [m]} m\textsubscript{i} (the sum and difference here is computed component-wise, and notice that \sum\textsubscript{i=1} m\textsubscript{i} = m\textsubscript{in}).

- Uses m\textsubscript{1}, ..., m\textsubscript{m} to compute the internal wires of C\textsubscript{sum}. Denote the wire values generated during this step by W\textsubscript{sum}.

- Outputs (W\textsubscript{sum}, W\textsubscript{in}). (The order in which we have written these wire values is for notational convenience only. Sim outputs these wire values in the same order as they appear in \hat{C}.)

We now bound the advantage obtained by leakage functions in L in distinguishing between the actual and simulated wire values. For that, we first present an alternative method of describing Sim. Let s = m - |B| - 1, then for every set of s well-formed vectors z\textsubscript{1}, ..., z\textsubscript{s} \in \text{supp (Enc\textsuperscript{in} (0\textsuperscript{\hat{n}(1, t)+1\cdot |C|\cdot t}, 1\textsuperscript{t})}, define the function f_{z_1, ..., z_s} that on input a wire assignment W for the wires of \hat{C}\textsuperscript{in}, in which m denotes the encoding of the masking inputs, operates as follows.

- Computes
  
m_{i_0} = m - \sum\textsubscript{i=1}^{s} z_i - \sum\textsubscript{i\in B} m_i
  
  (computed in Shallow\textsubscript{F} (1, O (\hat{n} ((\hat{n} (1, t) + 1) \cdot S (n), t)))) because all values except the input m are hard-wired into f_{z_1, ..., z_s}.

- Uses m_{i_0}, z_i, 1 \leq i \leq s, and m_i, i \in B to compute an assignment W' to the internal wires of C\textsubscript{sum}, when evaluated on m_{i_0}, z_i, 1 \leq i \leq s, and m_i, i \in B (this can be computed in Shallow\textsubscript{F} (\log m, O (\hat{n} ((\hat{n} (1, t) + 1) \cdot S (n), t) \cdot m)) using \hat{n} ((\hat{n} (1, t) + 1) S (n), t)\textsubscript{trees consisting of O (m) gates arranged in log m layers).

- Outputs (W, W') (arranged in the same order as these wires appear in \hat{C}.)

Let F = \{f_{z_1, ..., z_s} : z_1, ..., z_s \in \text{supp (Enc\textsuperscript{in} (0\textsuperscript{\hat{n}(1, t)+1\cdot |C|\cdot t})}\}, then F \subseteq Shallow\textsubscript{F} (\log m + 1, O (\hat{n}^2 (1, t) \cdot S (n) \cdot m)) (here we also use the fact that \hat{n} (n, t) =
\( n \cdot \tilde{n}(1,t) \) for every \( n, t \), by the properties of Construction 3.11), and notice that for \( f \leftarrow F \), the output of Sim is equally distributed to the distribution obtained by evaluating \( f \) on the output of Sim\textsuperscript{in}.

Denote \( \mathcal{L}' = \mathcal{L} \circ \text{Shallow}_F \left( \log m + 1, O \left( \tilde{n}^2(1,t) \cdot S(n) \cdot m \right) \right) \), then \( \mathcal{L}' \circ \text{Shallow}_F \left( 4, O \left( \tilde{n}(1,t) \right) \right) \subseteq \mathcal{L}_E \). Using Lemma 3.7.5, each gadget used by Comp is \( (\mathcal{L}', \epsilon(t) \cdot \tilde{n}(1,t)) \)-reconstructible by \( \text{Shallow}_F \left( 2, O \left( \tilde{n}^2(1,t) \right) \right) \) with ill-formed masking inputs.

Assume towards negation that there exists a leakage function \( \ell \in \mathcal{L} \) such that

\[
\text{SD} \left( \ell \left( \text{Sim} \left( C, C^A(x_1, \cdots, x_m), B, (x_i, m_i)_{i \in B} \right) \right), \ell \left[ \hat{C}, (\hat{x}_1, \cdots, \hat{x}_m) \right] \right)
\]

is more than \( \epsilon(t) \cdot (\tilde{n}(1,t) + 1) \cdot S(n) \), where \( \hat{x}_1, \cdots, \hat{x}_m \) are as defined in the theorem statement. We use \( \ell \) to construct a leakage function \( \ell' \in \mathcal{L}' \) such that

\[
\text{SD} \left( \ell' \left( \text{Sim} \left( C, C^A(x_1, \cdots, x_m), m, B, (x_i)_{i \in B} \right) \right), \ell' \left[ \hat{C}^\text{in}, \hat{x} \right] \right)
\]

is more than \( \epsilon(t) \cdot (\tilde{n}(1,t) + 1) \cdot S(n) \), where \( \hat{x} = (\hat{x}_1', \cdots, \hat{x}_m', \hat{m}) \) such that for every \( i \in B \), \( \hat{x}_i' = x_i \), for every \( i \notin B \), \( \hat{x}_i \leftarrow \text{Enc}^\text{in} \left( x_i, 1' \right) \), and \( \hat{m} \leftarrow \text{Enc}^\text{in} \left( m, 1' \right) \). This contradicts Lemma 3.7.6 because \( \mathcal{L}' \circ \text{Shallow}_F \left( 2, O \left( \tilde{n}^2(1,t) \right) \right) \subseteq \mathcal{L}_E \).

Let \( z_1, \cdots, z_s \) be a set of well-formed vectors that maximize the statistical distance between the output of Sim, and the actual wire values of \( \hat{C} \). Then \( \ell' \) on input an (either simulated or real) wire assignment \( \mathcal{W} \) for the wires of \( \hat{C}^\text{in} \), outputs \( \ell \left( f_{z_1, \cdots, z_s}(\mathcal{W}) \right) \). Consequently, \( \ell' \in \mathcal{L} \circ \text{Shallow}_F \left( \log m + 1, O \left( \tilde{n}^2(1,t) \cdot S(n) \cdot m \right) \right) = \mathcal{L}' \), and notice that when \( \text{Sim}^\text{in} \left( C, C^A(x_1, \cdots, x_m), m, B, (x_i)_{i \in B} \right) \) then \( \ell'(\mathcal{W}) \) is identically distributed to \( \ell \left( \text{Sim} \left( C, C^A(x_1, \cdots, x_m), B, (x_i, m_i)_{i \in B} \right) \right) \) (when conditioned on the event that Sim chose \( z_1, \cdots, z_s \)), whereas when \( \mathcal{W} \left[ \hat{C}^\text{in}, \hat{x} \right] \) (the distribution is over the encodings \( \hat{x}_1', \cdots, \hat{x}_m', \hat{m} \)) then \( \ell'(\mathcal{W}) \) is identically distributed to \( \ell \left[ \hat{C}, \hat{x} \right] \) (when conditioned on the event that Sim chose \( z_1, \cdots, z_s \)), so we have arrived at a contradiction.

### 3.8 A Multiparty LRCC

In this section we describe a multiparty LRCC that withstands active adversaries. The difference from the passive case is that the masking inputs provided by corrupted parties may be ill-formed. (Notice that since we are using onto encoding schemes, then every input encoding provided by corrupted parties corresponds to some effective input of that party to the original circuit \( C \), so that aspect is not problematic.) As discussed in Section 3.7.1, the effect of ill-formed masking inputs corresponds to applying an additive attack to the original circuit \( C \). We protect against such attacks in two steps.

First, we use AMD circuits over \( F \) which, informally, are randomized circuits that offers the “best possible protection” against additive attacks. That is, the effect of every additive attack that may blindly add an element of \( F \) to every internal wire of the circuit.
can be simulated by an ideal attack that applies only to the inputs and outputs. (See Definition 3.8.2 below for the exact definition.) By replacing $C$ with an AMD circuit before transforming it into a leakage-resilient circuit using Construction 3.31, the influence of bad masks is further restricted to additive attacks only on the inputs and outputs of $C$. Second, we use AMD encodings which, informally, guarantee that additive attacks on encodings are detected by the decoder with some non-zero probability. Therefore, by evaluating the AMD circuit on AMD encodings of the inputs, and outputting an AMD encoding of the outputs, we can guarantee that any additive attack on the inputs and outputs is caught (with high probability).

In summary, the compiling algorithm $\text{Comp}$ of our multiparty LRCC operates as follows. First, it transforms the circuit $C$ into a circuit $C^{\text{amd}}$ that emulates the operation of $C$ on AMD-encodings of its inputs. (That is, $C^{\text{amd}}$ expects its inputs to be AMD codewords. It decodes these codewords, evaluates $C$, and outputs an encoding of the outputs.) Second, it compiles $C^{\text{amd}}$ into a an additively-secure circuit $C'$. Finally, it compiles $C'$ into a leakage-secure circuit $\tilde{C}'$. This is formalized in the next construction.

Before presenting our construction, we first formally define AMD encodings, and AMD circuits.

**Definition 3.8.1** (AMD encoding scheme, [CDF+08, GIP+14]). Let $F$ be a finite field, $n \in \mathbb{N}$ be an input length parameter, $t \in \mathbb{N}$ be a security parameter, and $\epsilon(n,t) : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{R}^+$. An $(n,t,\epsilon(n,t))$-algebraic manipulation detection (AMD) encoding scheme $(\text{Enc},\text{Dec})$ over $F$ is an encoding scheme with the following guarantees.

- **Perfect completeness.** For every $\vec{x} \in F^n$, $\Pr[\text{Dec}(\text{Enc}(\vec{x},1^t),1^t) = (0,\vec{x})] = 1$.

- **Additive soundness.** For every $0^{\bar{h}(n,t)} \neq \vec{a} \in F^{\bar{h}(n,t)}$, and every $\vec{x} \in F^n$,

$$\Pr[\text{Dec}(\text{Enc}(\vec{x},1^t) + \vec{a},1^t) \notin \text{ERR}] \leq \epsilon(n,t)$$

where $\text{ERR} = (F \setminus \{0\}) \times F^n$, and the probability is over the randomness of $\text{Enc}$.

Next, we consider additively-secure circuit compilers which, at a high level, are compiler algorithms that transform a given circuit into a circuit resilient to additive attacks.

**Definition 3.8.2** (Additively-secure circuit compiler). Let $n \in \mathbb{N}$ be an input length parameter, $k \in \mathbb{N}$ be an output length parameter, and $\epsilon > 0$. We say that a randomized circuit $\hat{C} : F^n \rightarrow F^k$ is an $\epsilon$-additively-secure implementation of a circuit $C : F^n \rightarrow F^k$ if the following holds.

- **Completeness.** For every $x \in F^n$, $\Pr[\hat{C}(x) = C(x)] = 1$.

- **Additive-attack security.** For any additive attack $\mathcal{A}$ there exist $\vec{a}^{\text{in}} \in F^n$, and a distribution $\mathcal{A}^{\text{out}}$ over $F^k$, such that for every $\vec{x} \in F^n$, 

SD \left( C^A(\vec{x}), C(\vec{x} + \vec{a}^{\text{in}}) + \mathcal{A}^{\text{out}} \right) \leq \epsilon, \text{ where } C^A \text{ is the circuit obtained by subjecting } C \text{ to the additive attack } \mathcal{A}.

For } \epsilon : \mathbb{N} \to \mathbb{R}^+, \text{ we say that a PPT algorithm } \text{Comp} \text{ is an } \epsilon(n)-\text{additively-secure circuit compiler} \text{ if for every circuit } C : \{0,1\}^n \to \{0,1\}^k, \text{ the circuit } \hat{C} = \text{Comp}(C) \text{ is an } \epsilon(n)-\text{additively secure implementation of } C.

Next, using AMD encodings, additively-secure circuit compilers, and passive-secure multiparty LRCCs, we construct a multiparty LRCC.

**Construction 3.33.** Let } m \in \mathbb{N} \text{ denote the number of parties, } t \in \mathbb{N} \text{ be a security parameter, and } n \in \mathbb{N} \text{ be an input length parameter. Let } \text{Comp}' \text{ denote the compiler algorithm described in Construction 3.31, employing the encoding scheme } \text{Enc}^{\text{in}}, \text{Comp}^{\text{add}} \text{ be an additively-secure circuit compiler, and } (\text{Enc}^{\text{amd}}, \text{Dec}^{\text{amd}}) \text{ be an AMD encoding scheme. Let } \hat{n}^{\text{amd}}(n), \hat{n}(n,t) \text{ denote the length of encodings output by } \text{Enc}^{\text{amd}}, \text{ and } \text{Enc}^\text{add}, \text{ respectively. The multiparty LRCC } (\text{Enc}, \text{Comp}, \text{Dec}) \text{ is defined as follows.}

- For every } n,t,t_{\text{in}} \in \mathbb{N} \text{ and every } x \in \mathbb{F}^n,
  \begin{align*}
  \text{Enc}(x,1^t,1^{t_{\text{in}}}) &= \text{Enc}^{\text{in}} \left( \left( \text{Enc}^{\text{amd}}(x,1^n), 0^{(\hat{n}(1,1)+1)t_{\text{in}}} \right), 1^t \right).
  \end{align*}

- For every } y \in \mathbb{F}^{\hat{n}^{\text{amd}}(n)}, \text{Dec} \left( y, 1^t, 1^{t_{\text{in}}} \right) \text{ computes } (f_0, z) = \text{Dec}^{\text{amd}}(y, 1^n). \text{ If } f_0 = 0 \text{ then Dec outputs } (f_0, z), \text{ otherwise it outputs } (1, 0^{\hat{n}}).

- \text{Comp} \text{ on input } m \in \mathbb{N}, \text{ and an } m \text{-party circuit } C : (\mathbb{F}^n)^m \to \mathbb{F}^k:
  1. \text{ Constructs the circuit } C^{\text{amd}} : \left( \mathbb{F}^{\hat{n}^{\text{amd}}(n)} \right)^m \to \mathbb{F}^{\hat{n}^{\text{amd}}(k+1)} \text{ that operates as follows:}
    - Decodes its inputs using Dec^{\text{amd}}, thus obtaining the inputs } x_1, \ldots, x_m \text{ for } C, \text{ and a list } f_{\text{amd}} \text{ of flags.}
    - If } f_{\text{amd}} = \vec{0}, \text{ then } C^{\text{amd}} \text{ evaluates } C \text{ on inputs } x_1, \ldots, x_m; \text{ sets } y = C(x_1, \ldots, x_m); \text{ and } f = 0. \text{ Otherwise, } C^{\text{amd}} \text{ samples } f \in_R \mathbb{F} \setminus \{0\}; \text{ and } y \in_c \mathbb{F}^k.
    - Generates } e \leftarrow \text{Enc}^{\text{add}}((f,y), 1^n), \text{ and outputs } e.
  2. \text{Computes } C' = \text{Comp}^{\text{add}}(C^{\text{amd}}).
  3. \text{Outputs the circuit } C'' = \text{Comp}'(C').

We use } b(s,n,k) \text{ to denote the blow-up caused by Steps 1 and 2, i.e., for every } m \text{-party circuit } C : (\mathbb{F}^n)^m \to \mathbb{F}^k \text{ of size } s, \left| C' \right| \leq b(s,n,k). \text{ Notice that if } S^{\text{amd}}(n) \text{ denotes the combined size of } \text{Enc}^{\text{amd}} \text{ and } \text{Dec}^{\text{amd}} \text{ when applied to inputs of length } n, \text{ and } B \text{ denotes the blow-up caused by step 2, then there exists a constant } c \text{ such that } b(s,n,k) \leq B \left( c \cdot (s + m \cdot S^{\text{amd}}(n+k)) \right).
Theorem 3.34 (Multiparty LRCC). Let $F$ be a finite field, $m \in \mathbb{N}$ denote the number of parties, $n \in \mathbb{N}$ be an input length parameter, $t \in \mathbb{N}$ be a security parameter, $\epsilon(t) : \mathbb{N} \rightarrow \mathbb{R}^+$, $S(n), s'(n) : \mathbb{N} \rightarrow \mathbb{N}$ be size functions, and $\mathcal{L}, \mathcal{L}_E$ be families of functions. Assume that:

- $E^n$ is an encoding scheme which on inputs of length $n$, and given security parameter $t$, outputs encodings of length $\tilde{n}(n, t)$, which is linear-structured over $F$, and $(\mathcal{L}_E, \epsilon(t))$-leakage-indistinguishable, where $\mathcal{L}_E = \mathcal{L} \circ \text{Shallow}_F \left( \log m + 5, O \left( \tilde{n}^2(1, t) \cdot S(n) \cdot m \right) \right)$.
- $\text{Comp}^{\text{add}}$ is an $\epsilon'(n)$-additively-secure circuit compiler over $F$, where there exists a $B : \mathbb{N} \rightarrow \mathbb{N}$ such that for any circuit $C$, $|\text{Comp}^{\text{add}}(C)| \leq B(|C|)$; and a PPT algorithm $\text{Alg}'$ that given an additive attack $\mathcal{A}$ outputs the ideal attack $(\hat{o}^{\text{in}}, \hat{a}^{\text{out}})$ whose existence is guaranteed by the additive-attack security property of Definition 3.8.2.
- $(\text{Enc}^{\text{amd}}, \text{Dec}^{\text{amd}})$ is an $(n, t, \epsilon^{\text{amd}}(n, t))$-AMD encoding scheme over $F$, where the combined sizes of $\text{Enc}^{\text{amd}}$ and $\text{Dec}^{\text{amd}}$, on encodings of values in $F^n$, is $S^{\text{amd}}(n)$.

Then there exists a constant $c > 0$ such that there exists an m-party $(\mathcal{L}, \mathcal{L}'(n), \epsilon''(n, t))$-LRCC over $F$, where $S(n) = B \left( c \left( S'(n) + m \cdot S^{\text{amd}}(n + k) \right) \right)$, and $\epsilon''(n, t) = c'(n) + \epsilon^{\text{amd}}(n, n) + \epsilon(n, t)$ for $\epsilon(n, t) = \epsilon(t) \cdot (\tilde{n}(1, t) + 1) \cdot S(n)$. Moreover, on input an m-party circuit $C$, the compiler outputs a circuit of size at most $O \left( m \cdot \tilde{n}^2(1, t) \cdot B \left( c \left( |C| + m \cdot S^{\text{amd}}(n + k) \right) \right) \right)$.

Before proving the theorem, we first sketch the main points in the security proof. First, Lemma 3.7.1 guarantees that the effect of ill-formed masking inputs is equivalent to an additive attack on the internals of $C'$, which, by the additive-security guarantee, is (with high probability) equivalent to an attack on the inputs and outputs of $C^{\text{amd}}$. Moreover, if under this attack, the inputs of $C^{\text{amd}}$ are not valid AMD codewords, then its structure guarantees that (with high probability) the output is random, and can therefore be generated by $\hat{\text{Sim}}$. Using the simulator of Lemma 3.7.2, $\hat{\text{Sim}}$ can then simulate the internal wires of $C''$. Otherwise, the inputs to $C^{\text{amd}}$ are valid AMD codewords, and so $\hat{\text{Sim}}$ can extract effective inputs for $C$, and given the output of $C$, can again use the simulator of Lemma 3.7.2 to simulate the wires of $C$.

**Proof of Theorem 3.34.** We show that Construction 3.33, when using the algorithm $\text{Comp}'$ of Construction 3.31, $\text{Comp}^{\text{add}}$, and $(\text{Enc}^{\text{amd}}, \text{Dec}^{\text{amd}})$ as the underlying components, has the required properties. It follows directly from the construction that the compiler has the required syntax.

**Correctness.** The perfect correctness of $\text{Comp}'$, and $\text{Comp}^{\text{amd}}$, guarantee that $C''(x'_1, \cdots, x'_m) = C'(x^{\text{amd}}_1, \cdots, x^{\text{amd}}_m) = C^{\text{AMD}}(x^{\text{amd}}_1, \cdots, x^{\text{amd}}_m)$,
where for every $1 \leq i \leq m$, $x_i^{\text{amd}} \in \text{supp}(\text{Enc}^{\text{amd}}(x_i,1^n))$, and $x_i' \in \text{supp}(\text{Enc}^{\text{in}}\left(\left(x_i^{\text{amd}}, (\hat{n}(1,t)+1)|C'\right),1^t\right))$. Moreover, the perfect correctness of the AMD encoding scheme, and the structure of $C^{\text{amd}}$, guarantee that the input decoding in $C^{\text{amd}}$ succeeds, and so $C^{\text{AMD}}(x_1^{\text{amd}}, \ldots, x_m^{\text{amd}})$ outputs an AMD encoding of $(0, C(x_1, \ldots, x_m))$.

**Security.** Let $\bar{\epsilon}(n,t) = \epsilon(t)\cdot(\hat{n}(1,t) + 1)\cdot S(n)$, then the first assumption of the theorem guarantees that Lemma 3.7.2 holds with statistical distance $\bar{\epsilon}(n,t)$, and simulator $\text{Sim}^{\text{in}}$.

We describe the simulator $\text{Sim}$, that uses the adversary $A$ in a black-box manner to determine the set $B = \{i_1, \ldots, i_r\}$ of corrupted parties ($\text{Sim}$ chooses to corrupt the same set of parties), and the (possibly ill-formed) encoded inputs $x_B' = (x_{i_1}', \ldots, x_{i_r}')$ for the parties in $B$. Since $\text{Enc}^{\text{in}}$ is linear-structured, and therefore also onto by Remark 3.3), then every $x_{i_j}'$ is an encoding (according to $\text{Enc}^{\text{in}}$) of some pair $(x_{i_j}^{\text{amd}}, m_{i_j})$, where $(x_{i_1}^{\text{amd}}, \ldots, x_{i_r}^{\text{amd}})$ are supposed to be AMD encodings of the inputs of $B$ for $C'$, and $m_B = (m_{i_1}, \ldots, m_{i_r})$ are masking inputs for the compiled circuit $C''$. For every $i_j \notin B$, let $x_{i_j}^{\text{amd}} \leftarrow \text{Enc}^{\text{amd}}(x_{i_j},1^n)$, and $x_{i_j}' \leftarrow \text{Enc}^{\text{in}}\left((x_{i_j}^{\text{amd}}, 0(\hat{n}(1,t)|C')|,1^t)\right)$. Let $A_{\text{msg}}$ denote the additive attack on $C'$ that corresponds to $m_B$, which can be efficiently computed from $m_B$ by Lemma 3.7.1. (Syntactically, $A_{\text{msg}}$ is determined not only by $m_B$, but also by the masking inputs provided by the honest parties. Since these are simply well-formed vectors, $m_B$ completely determines the additive attack.) Then Lemma 3.7.1 guarantees that $C''(x_1', \ldots, x_m') = C'A_{\text{msg}}(x_1^{\text{amd}}, \ldots, x_m^{\text{amd}})$. Consequently, by the $\epsilon'(n)$-additive security of $\text{Comp}^{\text{add}}$, there exists an additive attack $a^{\text{in}}$ on the inputs of $C^{\text{amd}}$, and a distribution $A^{\text{out}}$ over additive attacks on the outputs of $C^{\text{amd}}$, such that $\text{SD}\left(C^{\text{msg}}(x_1^{\text{amd}}, \ldots, x_m^{\text{amd}}), C^{\text{amd}}(a^{\text{in}}, A^{\text{out}})(x_1^{\text{amd}}, \ldots, x_m^{\text{amd}})\right) \leq \epsilon'(n)$. Moreover, by the second assumption of the theorem, $a^{\text{in}}, A^{\text{out}}$ can be computed efficiently given $A_{\text{msg}}$.

Sim uses $m_B$ to compute $A_{\text{msg}}$, from which it computes $a^{\text{in}}, A^{\text{out}}$, and samples $A^{\text{out}} \leftarrow A^{\text{out}}$. We partition $a^{\text{in}}$ into $a^{\text{in}}_B \in \left(\mathbb{F}^{n^{\text{amd}}(n)}\right)^r$, $a^{\text{in}}_B \in \left(\mathbb{F}^{n^{\text{amd}}(n)}\right)^{m-r}$, such that $a^{\text{in}}_B, a^{\text{in}}_B$ are the “parts” of $a^{\text{in}}$ that “meet” the inputs of $B, B$, respectively, and consider 2 possible cases. (We note that Sim can efficiently check which of these cases holds, by looking at $a^{\text{in}}_B$ and $A^{\text{out}}$, and decoding $(x_1^{\text{amd}}, \ldots, x_m^{\text{amd}})$ and $a^{\text{in}}_B$.)

**Case (1):** $a^{\text{in}}_B \neq 0^{n^{\text{amd}}(n)-(m-r)}$, or $(x_1^{\text{amd}}, \ldots, x_m^{\text{amd}}) + a^{\text{in}}_B$ does not consist of $r$ valid AMD encodings. Sim picks $w_{i_j} = 0^n$ for every $1 \leq j \leq r$, and $b = 1$. Notice that in this case, at least one of the inputs of $C^{\text{amd},(a^{\text{in}}, A^{\text{out}})}$ is not a valid AMD encoding, so the $\epsilon^{\text{amd}}(n,n)$-additive soundness guarantees that except with at most $\epsilon^{\text{amd}}(n,n)$ probability the input decoding in $C^{\text{amd},(a^{\text{in}}, A^{\text{out}})}$ outputs at least one non-zero flag. Conditioned on the event that the input decoding failed, the structure of $C^{\text{amd}}$ guarantees that $C^{\text{amd}),(a^{\text{in}}, A^{\text{out}})}(x_1^{\text{amd}}, \ldots, x_m^{\text{amd}}) = \epsilon' + A^{\text{out}}$, where $\epsilon'$ is an AMD encoding of $(f, z)$ for $f \in R \setminus \{0\}$ and $z \in R^k$. Let $y' = (1, 0^k)$, and Sim sets $y'' = (f', y')$ for $f' \in R \setminus \{0\}$ and $y' \in R^k$. (Notice that in this case Sim does not use the output $y$ of $C$ which was given to it.)

**Case (2):** $a^{\text{in}}_B = 0^{n^{\text{amd}}(n)-(m-r)}$, and $(x_1^{\text{amd}}, \ldots, x_m^{\text{amd}}) + a^{\text{in}}_B$ are valid AMD encod-
ings of some $\hat{x}_{i_1}, \cdots, \hat{x}_{i_\ell} \in \mathbb{F}^n$. Sim chooses $b = 0$, and sets $w_{ij} = \hat{x}_{ij}$ for every $1 \leq i \leq j$, and for every $i_j \notin B$, let $w_{ij} = x_{ij}$. Notice that in this case the structure of $C_{\text{AMD}}$, and the perfect correctness of the AMD encoding scheme, guarantee that $C_{\text{AMD}}(\hat{w}_{\text{AMD}}, \hat{w}_{\text{AMD}}') = e + A_{\text{out}}$, where $e$ is an AMD encoding of $(0, C(w_1, \cdots, w_m))$. In case (2), Sim is given $y = C(w_1, \cdots, w_m)$, and sets $y' = (0, y)$. Let $\hat{y} = (0, y)$.

In either case, Sim samples $y_{\text{AMD}} \leftarrow \text{Enc}_{\text{AMD}}(y'', 1^n)$, and outputs $\text{Sim}^{\text{in}}(C, y_{\text{AMD}} + A_{\text{out}}, B, (x'_{ij})_{j \in [r]})$. Let $\ell \in \mathcal{L}$. By Lemma 3.7.2,

$$\text{SD}\left( \ell \left( \text{Sim}^{\text{in}}(C, y_{\text{AMD}} + A_{\text{out}}, B, (x'_{ij})_{j \in [r]}), \ell \left[ C^\prime, (x'_1, \cdots, x'_m) \right] \right) \right) \leq \tilde{c}(n, t)$$

conditioned on the event that $C^\prime(x'_1, \cdots, x'_m) = y_{\text{AMD}} + A_{\text{out}}$ (when $A_{\text{out}} \leftarrow A_{\text{out}}$).

As noted above, in case (1) $\text{SD}(C^\prime(x'_1, \cdots, x'_m), y_{\text{AMD}} + A_{\text{out}}) \leq e_{\text{AMD}}(n, n) + \epsilon'(n)$, and in case (2), $\text{SD}(C^\prime(x'_1, \cdots, x'_m), y_{\text{AMD}} + A_{\text{out}}) \leq \epsilon'(n)$ (since in both cases $\text{SD}(C_{\text{AMD}}(x_{1, \cdots, m}^\prime), y_{\text{AMD}}(x_{1, \cdots, m}^\prime)) \leq \epsilon'(n)$). Therefore,

$$\text{SD}\left( \ell \left( \text{Sim}^{\text{in}}(C, y_{\text{AMD}} + A_{\text{out}}, B, (x'_{ij})_{j \in [r]}), \ell \left[ C^\prime, (x'_1, \cdots, x'_m) \right] \right) \right)$$

is at most $\tilde{c}(n, t) + e_{\text{AMD}}(n, n) + \epsilon'(n)$.

Moreover, $B$ is equally distributed in both worlds, and conditioned on the value of the outputs $C^\prime(x'_1, \cdots, x'_m)$ and $y_{\text{AMD}} + A_{\text{out}}$, the random variables $\text{Real}'', \text{Ideal}'$ describing the distribution of $(B, C^\prime(x'_1, \cdots, x'_m), \text{Dec}(C^\prime(x'_1, \cdots, x'_m)))$, and $(B, \mathcal{W}_{\text{out}}, \hat{y})$ (where $\mathcal{W}_{\text{out}}$ is the restriction of $\text{Sim}^{\text{in}}(C, y_{\text{AMD}} + A_{\text{out}}, B, (x'_{ij})_{j \in [r]})$ to the output wires of $C^\prime$) are identically distributed, and independent of the random variables describing $\ell \left[ C^\prime, (x'_1, \cdots, x'_m) \right]$, and $\ell \left( \text{Sim}^{\text{in}}(C, y_{\text{AMD}} + A_{\text{out}}, B, (x'_{ij})_{j \in [r]}), \ell \left[ C^\prime, (x'_1, \cdots, x'_m) \right] \right)$, respectively. Therefore,

$$\text{SD}(\text{Real}, \text{Ideal}) \leq \tilde{c}(n, t) + e_{\text{AMD}}(n, n) + \epsilon'(n).$$

As for the size of the compiled circuit $C^\prime$, according to the analysis following Construction 3.33, $|C'| \leq B\left(c \cdot (|C| + m \cdot S_{\text{AMD}}(n + k))\right)$ for some constant $c > 0$. Since $C^\prime$ is obtained from $C'$ by applying the compiler of Construction 3.31, then Theorem 3.32 guarantees that $|C'| \leq O(\cdot \tilde{n}^2(1, t) \cdot B\left(c \cdot (|C| + m \cdot S_{\text{AMD}}(n + k))\right))$.

### 3.8.1 Boolean Multiparty LRCCs

In this section we use Theorem 3.34 to construct a boolean multiparty LRCC that can withstand leakage from AC$^0$ circuits with multiple output gates, thus proving Theorem 3.2. More specifically, Theorem 3.2 will follow as a corollary from the following proposition, which is obtained by instantiating Construction 3.33 with the AMD encoding scheme of [GIP+14, Corollary 2.1 in the full version], and an additively-secure boolean
Proposition 3.8.3. Let \( m \in \mathbb{N} \) denote the number of parties, \( n \in \mathbb{N} \) be a security parameter, \( k \in \mathbb{N} \) be an output length parameter, \( d \in \mathbb{N} \) be a depth parameter, \( \epsilon (t) : \mathbb{N} \to \mathbb{R}^+ \) be an error function, and \( \mathcal{L}, \mathcal{L}_E \) be families of functions. Let \( \text{E}^n = (\text{Enc}^n, \text{Dec}^n) \) be an encoding scheme that on length-\( n \) inputs, and given security parameter \( t \), outputs encodings of length \( \hat{n} (n, t) \), and is linear-structured over \( \mathbb{F}_2 \). Then there exist positive constants \( c, c' \) such that for every depth-\( d \), \( m \)-party circuit \( C : \{0, 1\}^n \to \{0, 1\}^k \) computing function \( f_C \), if \( \text{E}^n \) is \((\mathcal{L}_E, \epsilon (t))\)-leakage-indistinguishable, where \( \mathcal{L}_E = \mathcal{L} \circ \text{Shallow}_{\mathbb{F}_2} \left( \log m + 5, O \left( \hat{n}^2 (1, t) \cdot S \cdot m \right) \right) \) for \( S = |C| \cdot \log^\epsilon (n |C|) + (nkmd)^\epsilon \), then there exists an \( m \)-party \((\mathcal{L}, \epsilon'' (n, t))\)-secure implementation \( \left( \text{Enc}, \hat{C}, \text{Dec} \right) \) of \( f_C \), where \( |\hat{C}| = m \cdot \hat{n}^2 (1, t) \cdot (m \cdot |C| \cdot \text{polylog} (m \cdot |C|) + \text{poly} (mnkd)) \), and \( \epsilon'' (n, t) = \epsilon (t) \cdot (\hat{n} (n, t) + 1) \cdot S + 2^{-n+1} \).

We note that Proposition 3.8.3 naturally gives rise to an \( m \)-party LRCC. The proof uses two ingredients, the first is the following additively-secure boolean circuit compiler:

Theorem 3.35 (Additively-secure circuit compiler, [GIW16]). Let \( n \) be an input length parameter, and \( \epsilon (n) : \mathbb{N} \to \mathbb{R}^+ \) be a statistical error function. Then there exists an \( \epsilon (n) \)-additively-secure circuit compiler \( \text{Comp} \). Moreover, on input a depth-\( d \) boolean circuit \( C : \{0, 1\}^n \to \{0, 1\}^k \), \( \text{Comp} \) outputs a circuit \( \hat{C} \) such that \( |\hat{C}| = |C| \cdot \text{polylog} \left( |C|, \log \frac{1}{\epsilon (n)} \right) + \text{poly} \left( n, k, d, \log \frac{1}{\epsilon (n)} \right) \). Furthermore, there exists a PPT algorithm \( \text{Alg} \) that on input \( C, \epsilon (n) \), and an additive attack \( \mathcal{A} \), outputs a vector \( \tilde{a}^n \in \{0, 1\}^n \), and a distribution \( \mathcal{A}^{\text{out}} \) over \( \{0, 1\}^k \), such that for any \( \tilde{x} \in \{0, 1\}^n \) it holds that \( \text{SD} (\hat{C} \mathcal{A} (\tilde{x}), C (\tilde{x} + \tilde{a}^n) + \mathcal{A}^{\text{out}}) \leq \epsilon (n) \).

The second is the AMD encoding scheme of [GIP+14, Corollary 2.1 in the full version]. As such encoding schemes will also be used in Section 3.8.2 to construct multiparty LRCCs over large fields \( \mathbb{F} \), we describe the encoding scheme over general fields.

Theorem 3.36 (AMD encoding scheme, full version of [GIP+14]). Let \( \mathbb{F} \) be a finite field, \( n, t \in \mathbb{N} \), and \( \epsilon (t) = |\mathbb{F}|^{-t} \). Then there exists an \((n, t, \epsilon (t))\)-AMD encoding scheme \((\text{Enc}, \text{Dec})\) such that \( \text{Enc} \) outputs encodings of length \( \hat{n} (n, t) = O (n + t) \). Moreover, there exists a constant \( c_{\text{amd}} > 0 \) such that encoding and decoding of length-\( n \) inputs with parameter \( t \) can be performed by circuits of size \( c_{\text{amd}} \cdot (n + t) \).

Proof of Proposition 3.8.3. We instantiate Construction 3.33 with the compiler algorithm \( \text{Comp}' \) of Construction 3.31 over \( \mathbb{F}_2 \), which is based on the encoding scheme \( \text{E}^n \); the \( \epsilon (n) \)-additively-secure circuit compiler \( \text{Comp}_{\text{add}} \) of Theorem 3.35, with error function \( \epsilon (n) = 2^{-n} \) (according to Theorem 3.35, \( \text{Comp}_{\text{add}} \) has the additional guarantee that for any additive attack, the corresponding ideal additive attacks on the inputs and outputs of the compiled circuit can be found efficiently); and the \((n, t, c_{\text{amd}} (t))\)-AMD encoding scheme \((\text{Enc}_{\text{amd}}, \text{Dec}_{\text{amd}})\) of Theorem 3.36, with security parameter \( t = n \).
Let $c^{\text{amd}}$ be the constant whose existence is guaranteed by Theorem 3.36, and let $c^{\text{add}}_1, c^{\text{add}}_2$ be the constants whose existence is guaranteed by Theorem 3.35, namely for every depth-$d$ boolean circuit $C : \{0,1\}^m \rightarrow \{0,1\}^k$, $|\text{Comp}^{\text{add}}(C)| \leq m \cdot |C| \cdot \log^{c^{\text{add}}_2}(n \cdot |C|) + (nkd^m)^{c^{\text{add}}_1}$. Then for every depth-$d$, $m$-party boolean circuit $C : \{0,1\}^n \rightarrow \{0,1\}^k$, the circuit $C^{\text{amd}}$ constructed in step (1) of Construction 3.33 has size at most $c \cdot (|C| + c^{\text{amd}} \cdot m \cdot (n+k))$, for some constant $c > 0$, so the circuit $C'$ obtained in step (2) of Construction 3.33 has size $S_C := c \cdot (|C| + c^{\text{amd}} \cdot m \cdot (n+k)) \cdot \log^{c^{\text{add}}_2}(cn \cdot (|C| + c^{\text{amd}} \cdot m \cdot (n+k))) + c''(nkdm)^{c^{\text{add}}_1}$, for some constant $c'' > 0$. (The dependence on $c''$ is because the input of $C'$ are AMD encodings of the inputs of $C$, so each input has length $O(n)$.)

Using Theorem 3.34 with size function $S' = |C|$, the circuit $C''$ output by Construction 3.33 is an $m$-party ($\mathcal{L}, \ell''(n,t)$)-secure implementation of the function $f_C$ which $C$ computes, where $\ell''(n,t) = \ell'(n) + c^{\text{amd}}(n) + \tilde{\epsilon}(n,t)$, where $\tilde{\epsilon}(n,t) = \epsilon(t) \cdot (n(1,t) + 1) \cdot S_C$ (since $|C'| \leq S_C$); $\ell'(n) = 2^{-n}$; and $c^{\text{amd}}(n) = 2^{-n}$ (by Theorem 3.36). Moreover, by Theorem 3.34, the compiled circuit $\hat{C} = \text{Comp}'(C)$ satisfies

$$|\hat{C}| \leq m \cdot n^2(1,t) \cdot (m \cdot |C| \cdot \text{polylog}(m \cdot |C|)) + \text{poly}(nkdm).$$

Taking $E^{\text{in}}$ to be the parity encoding scheme $E_\oplus$ (Example 3.2.4) in Proposition 3.8.3, and using a result of Håstad [Häs86] (Theorem 3.6), we obtain a multiparty LRCC secure against leakage from $\text{AC}^0$ circuits, namely, constant-depth and polynomial-sized boolean circuits with unbounded fan-in $\land, \lor$ and $\neg$ gates. This will follow as a special case from the proof of Theorem 3.37 below.

Since Proposition 3.8.3 discusses leakage functions computable by circuits using operations of the field $F_2$, whereas the parity encoding is leakage-indistinguishable against leakage functions computable by circuits using only $\land, \lor, \neg$ gates, we will need the following lemma which, informally, states that circuits in $\text{Shallow}_{F_2}$ can be implemented using circuits in $\text{UnBBool}$.

**Lemma 3.8.4** (Emulating $\text{Shallow}_{F_2}$ circuits using $\text{UnBBool}$ circuits). Let $d \in \mathbb{N}$ be a depth parameter, $s \in \mathbb{N}$ be a size parameter, and $n \in \mathbb{N}$ be an input length parameter. Then there exist constants $c_d, c_s > 0$ such that for every circuit $C \in \text{Shallow}_{F_2}(d,s)$ of input length $n$, there exists a circuit $C_b \in \text{UnBBool}(c_d,d,c_s,s)$ of input length $n$ such that for every $x \in \{0,1\}^n$, $C(x) = C_b(x)$.

**Proof.** The lemma follows simply by noting that there exists a constant $c$ such that every boolean function $f : \{0,1\}^m \rightarrow \{0,1\}$ is in $\text{UnBBool}(3,c \cdot 2^n)$ (computable by the circuit that computes the DNF representation of $f$). Therefore, by replacing every gate $g$ in $C$ with the circuit evaluating the DNF representing the function $f_g$ that the gate computes, we obtain the circuit $C_b \in \text{UnBBool}(3d,O(s))$ (since gates in $\text{Shallow}_{F_2}$ have bounded fan-in, so $m$ is constant).
We now prove Theorem 3.2 by combining Proposition 3.8.3 with Theorem 3.7, and Lemma 3.8.4. We first formally state the theorem.

**Theorem 3.37** (Formal statement of Theorem 3.2). There exists a constant $\gamma \in (0, 1)$ such that for every constant $\delta \in (0, \gamma)$, constant number of parties $m \in \mathbb{N}$, security parameter $t \in \mathbb{N}$, and $m$-party boolean circuit $C : \{0, 1\}^m \to \{0, 1\}^k$ of size $s$ computing function $f_C$, the following holds. There exists a triplet $(\mathcal{E}, \hat{C}, \mathcal{D})$, where $\hat{C}$ is a boolean circuit of size $\text{poly}(t, s)$, such that for every depth parameter $d \in \mathbb{N}$, $(\mathcal{E}, \hat{C}, \mathcal{D})$ is an $m$-party $(\mathcal{L}, \epsilon)$-secure implementation of $f_C$, where:

- $\mathcal{L}$ consists of all boolean functions computable by depth-$d$, size-$2^{O((\gamma - \delta)/(\gamma d))}$ boolean circuits over gates of unbounded fan-in and fan-out, with $|\hat{C}|^\delta$ output bits.
- $\epsilon = 2^{-\Omega((\gamma - \delta)/(\gamma d))}$.

Moreover, $\mathcal{E}$, $\mathcal{D}$ can be efficiently computed given $1^{\lceil C \rceil}, 1^t$, and $\hat{C}$ can be efficiently computed given $C, 1^t$.

**Proof.** We use the multiparty LRCC of Construction 3.33, instantiating $\mathcal{E}_{\|}$ with the parity encoding $\mathcal{E}_{\|}$ of Example 3.2.4. Given a depth-$d_C$, $m$-party circuit $C : \{0, 1\}^m \to \{0, 1\}^k$ of size $s$ computing function $f_C$ (notice that $s \geq n \cdot m$, $s \geq k$, and $s \geq d_C$), let $c_s, c', c''$ denote the constants for which Proposition 3.8.3 holds, namely $S = s \cdot \log^{c_s}(ns) + (nkdc_m)^{c'} = \text{poly}(s)$, and $\mathcal{L} = \mathcal{L} \circ \text{Shallow}_2^\gamma(\log m + 5, c'' \cdot \hat{n}^2(1, t) \cdot S \cdot m)$, where $\hat{n}(1, t)$ denotes the length of encodings output by $\mathcal{E}_{\|}$ on inputs of length 1, and given security parameter $t$. Notice that we can assume without loss of generality that $t \leq n$, otherwise we can artificially increase the input length to size $t$ (this would increase $s$ by an additive factor of $t$, and would therefore not influence the parameters of the construction).

Moreover, let $\tilde{c}, \tilde{c}_s, \tilde{c}_a > 0$ be the constants such that the secure implementation $\hat{C}$ satisfies $\hat{s} := |\hat{C}| \leq cm \cdot \hat{n}^2(1, t) \cdot \left(m \cdot s \cdot \log^{c_s}(m \cdot s) + (nkdc_m)^{c'}\right)$. Since $s \geq m, n, d_C, k$, and for the parity encoding, $\hat{n}(1, t) = t$, then $\hat{s} \leq c \cdot s^{5 + 2\tilde{c}_s + 4c'} \cdot t^2$. Finally, let $c_d, c'$ denote the constants for which Lemma 3.8.4 holds, then the lemma guarantees that $\text{Shallow}_2^\gamma(\log m + 5, c' \cdot \hat{n}^2(1, t) \cdot S \cdot m) \subseteq \text{UnBBool}(c_d(\log m + 5), c' \cdot c'' \cdot \hat{n}^2(1, t) \cdot S \cdot m)$. Specifically, to use Proposition 3.8.3 it suffices to find a linear-structured encoding scheme that resists leakage from $\mathcal{L} \circ \text{UnBBool}(c_d(\log m + 5), c' \cdot c'' \cdot \hat{n}^2(1, t) \cdot S \cdot m)$.

We apply $\mathcal{E}_{\|}$ with security parameter $\tilde{t} = \max\{t, s\}$, in which case $\hat{s} \leq c \cdot s^{5 + 2\tilde{c}_s + 4c'} \cdot t^2 \leq \max\{c \cdot s^{7 + 2\tilde{c}_s + 4c'}, c \cdot t^{7 + 2\tilde{c}_s + 4c'}\}$. Moreover, for a large enough $\tilde{t}$, $\hat{s} \leq \max\{s^{8 + 2\tilde{c}_s + 4c'}, t^{8 + 2\tilde{c}_s + 4c'}\} = t^{8 + 2\tilde{c}_s + 4c'}. \text{ We denote } c_{\|} := 8 + 2\tilde{c}_s + 4c', \text{ and the constant } \gamma \text{ guaranteed by the theorem statement is } \gamma = c_{\|}^{-1} < 1.$

Fix some family $C^{\text{leak}}$ of boolean circuits with depth $d$, size $s^{\text{leak}} = 2^{\gamma'(\gamma - \delta)/(\gamma d)}$ (for some constant $\gamma' \in (0, 1)$ whose value will be set later), and output length $\hat{s}^\delta$ over gates of unbounded fan-in and fan-out. Let
\[ \hat{d} = d + c_d \cdot (\log m + 5), \text{ and } \hat{\delta} = c_\oplus \cdot \delta. \]

Then Theorem 3.7 guarantees that \( \text{Enc}_{\circ} \left( \cdot, 1^t \right) \) is \( \left( \text{UnBBool} \left( \hat{t}, \hat{s}, \hat{d}, 2c_1 \hat{d}^{(1-\delta)/\hat{d}}, 2^{-2c_2 \hat{d}^{(1-\delta)/\hat{d}}} \right) \right) \)-leakage-indistinguishable, for some constants \( c_1, c_2 > 0 \). Consequently, \( \text{Enc}_{\circ} \left( \cdot, 1^t \right) \) is also \( \left( \text{UnBBool} \left( \hat{t}, \hat{s}, \hat{d}, 2c_1 \cdot \max\{t,s\} \hat{d}^{(1-\delta)/\hat{d}}, 2^{-2c_2 \cdot \max\{s,t\} \hat{d}^{(1-\delta)/\hat{d}}} \right) \right) \)-leakage-indistinguishable.

We choose \( \epsilon = \epsilon_\delta = 2c \cdot 2^{-(c_2/4) \cdot \hat{d}^{(1-\delta)/\hat{d}}} \), where \( c \) is a constant such that the error term \( \epsilon''(n,t) \) in Proposition 3.8.3 satisfies \( \epsilon''(n,t) \leq c \cdot \epsilon(t) \cdot n(1,t) \cdot S + 2^{-n+1} \). Notice that \( \epsilon_\delta \leq 2^{-\Omega((\gamma-\delta)/\gamma d)} \).

We set \( \gamma' \in (0,1) \) such that for a large enough \( N, 2^{c_1/2}N^{\frac{1-\delta}{1-\gamma'}} \geq 2^{N^\gamma' (\gamma-\delta)/\gamma d} \). Then for large enough \( t, s, 2^{c_1 \cdot \max\{s,t\} \hat{d}^{(1-\delta)/\hat{d}}} \geq s_{\text{leak}} + \max\{s_{\text{c}}, t_{\text{c}}\} \), which means that the composition of \( C_{\text{leak}} \) with any boolean circuit of size at most \( \max\{s_{\text{c}}, t_{\text{c}}\} \) and depth \( c_d \cdot (\log m + 5) \), obtains advantage at most \( 2^{-(c_2/2) \cdot \hat{d}^{(1-\delta)/\hat{d}}} \cdot 2^{-(c_2/4) \cdot \max\{s,t\} \hat{d}^{(1-\delta)/\hat{d}}} \cdot 2^{-(c_2/4) \cdot \hat{d}^{(1-\delta)/\hat{d}}} \cdot \hat{t} \cdot S + 2^{-n+1} \) statistically close. Since \( S = \text{poly} \{s\} \) (or \( \text{poly} \{s, t\} \) if we padded the input length to length \( t) \), then \( 2^{-(c_2/2) \cdot \hat{d}^{(1-\delta)/\hat{d}}} \cdot \hat{t} \leq 1 \). Finally, since \( n \geq t \) then for a large enough \( t, 2^{-n+1} \leq 2^{-t+1} \leq c \cdot 2^{-(c_2/4) \cdot \hat{d}^{(1-\delta)/\hat{d}}} \), so the wire values are \( \epsilon_\delta(t) \) statistically close.

### 3.8.2 Multiparty LRCCs Over Large Fields

In this section we use Theorem 3.34 to construct multiparty LRCCs that are secure over large fields. Intuitively, security is guaranteed only over large fields, since known additively-secure circuit compilers over \( \mathbb{F} \neq \mathbb{F}_2 \) only guarantee security when \( |\mathbb{F}| \) is large. More specifically, we instantiate Construction 3.33 with the AMD encoding scheme of Theorem 3.36, and the additively-secure circuit compiler [GIP+14, Theorem 1.1 in the full version]:

**Theorem 3.38 (Additively-secure circuits, full version of [GIP+14]).** Let \( \mathbb{F} \) be a finite field. Then there exists a constant \( c_{\text{add}} > 0 \), and a circuit compiler \( \text{Comp}^{\text{GIPST}} \), such that the following holds for every arithmetic circuit \( C \) over \( \mathbb{F} \). \( \hat{C} = \text{Comp}^{\text{GIPST}}(C) \) is an \( O \left( \frac{|C|}{|\mathbb{F}|} \right) \)-additively-secure implementation of \( C \), and \( |\hat{C}| \leq c_{\text{add}} |C| \).

**Remark 3.39.** The additively-secure circuit compiler of Theorem 3.38 has the following property: there exists an algorithm \( \text{Alg} \) that on input an arithmetic circuit \( C : \mathbb{F}^n \to \mathbb{F}^k \), and an additive attack \( A \) on \( C \), finds the corresponding attacks \( a_{\text{in}} \in \mathbb{F}^n, A_{\text{out}} \in \mathbb{F}^k \) (as specified by the additive-attack security property of Definition 3.8.2) in time \( O \left( |\mathbb{F}| \right) \).

By instantiating Construction 3.33 with the AMD encoding scheme of Theorem 3.36, and the additively-secure circuit compiler of Theorem 3.38, we obtain the following result for general fields.
Proposition 3.8.5. Let $\mathbb{F}$ be a finite field, $m \in \mathbb{N}$ denote the number of parties, $n \in \mathbb{N}$ be an input length parameter, $k \in \mathbb{N}$ be an output length parameter, $t \in \mathbb{N}$ be a security parameter, $\epsilon(t) : \mathbb{N} \to \mathbb{R}^+$ be an error function, $S(n) : \mathbb{N} \to \mathbb{N}$ be a size function, and $\mathcal{L}, \mathcal{L}_E$ be families of functions. Let $E^{in} = (\text{Enc}^{in}, \text{Dec}^{in})$ be an encoding scheme that on inputs of length $n$, and given security parameter $t$, outputs encodings of length $\hat{n}(n, t)$, is linear-structured over $\mathbb{F}$, and $(\mathcal{L}_E, \epsilon(t))$-leakage-indistinguishable, where $\mathcal{L}_E = \mathcal{L} \circ \text{Shallow}_F \left( \log m + 5, O \left( \hat{n}^2(1, t) \cdot S(n) \cdot m \right) \right)$. Then there exist constants $c, c' > 0$, and an $m$-party $(\mathcal{L}, S'(n, k, m), \epsilon'(n, t))$-LRCC over $\mathbb{F}$, where $S'(n, k, m) = c \cdot (S(n) - m(n + k))$, and $\epsilon'(n, t) = \epsilon(t) \cdot (\hat{n}(1, t) + 1) \cdot S(n) + c' \cdot \left( \frac{S(n)}{|\mathbb{F}|} + \frac{1}{|\mathbb{F}|^n} \right)$.

Proof. We instantiate Construction 3.33 with the compiler algorithm $\text{Comp}'$ of Construction 3.31, which is based on the encoding scheme $E^{in}$; the $\epsilon'(n)$-additively-secure circuit compiler $\text{Comp}^{GIPST}$ of Theorem 3.38; and the $(n, t, \epsilon^{\text{amd}}(t))$-AMD encoding scheme $(\text{Enc}^{\text{amd}}, \text{Dec}^{\text{amd}})$ of Theorem 3.36, with security parameter $t = n$. Let $c^{\text{add}}, c^{\text{amd}}$ be the constants whose existence is guaranteed by Theorems 3.38 and 3.36, respectively. Then for every circuit $C$, the circuit $C'$ obtained through Construction 3.33 has size $c^{\text{add}} \cdot c \cdot (|C| + c^{\text{amd}} \cdot m \cdot (n + k))$, for some constant $c > 0$. Set $S'(n, m, k) = \frac{S(n)}{c^{\text{add}}} - c^{\text{amd}} \cdot m \cdot (n + k)$, and notice that if $|C| \leq S'(n)$, then $|C'| \leq S(n)$, where $C'$ denotes the circuit obtained in Step (2) of Construction 3.33. Using Theorem 3.34, this implies that Construction 3.33 is an $m$-party $(\mathcal{L}, S'(n, k, m), \epsilon''(n))$-LRCC over $\mathbb{F}$, where $\epsilon''(n, t) = \epsilon'(n) + \epsilon^{\text{amd}}(n) + \epsilon(n, t)$ for $\epsilon(n, t) = \epsilon(t) \cdot (\hat{n}(1, t) + 1) \cdot S(n)$; $\epsilon'(n) = O \left( \frac{S(n)}{|\mathbb{F}|} \right)$ (by Theorem 3.38); and $c^{\text{amd}}(n) = |\mathbb{F}|^{-n}$ (by Theorem 3.36).
Chapter 4

Non-Adaptive Probabilistically Checkable Proofs with Zero-knowledge

In this chapter we construct non-adaptively verifiable probabilistically checkable proofs (PCPs) with zero-knowledge properties, and establish a connection between such proofs and leakage-resilient circuits. Before describing our main results, we first give a short overview of previous constructions of zero-knowledge PCPs (ZKPCPs).

4.1 Introduction

A ZKPCP is a PCP with a probabilistic choice of proof, and the additional guarantee that any $q \leq q^*$ proof symbols reveal no information about the NP witness (this is formalized in the simulation-based paradigm by showing that the view of every (possibly malicious) verifier making at most $q^*$ queries to its proof can be simulated from the input alone). Previous ZKPCP constructions [KPT97, IMS12] are obtained from standard (i.e., non-ZK) PCPs in two steps. First, the standard PCP is transformed into a PCP with a weaker “honest-verifier” ZK guarantee (which is much easier to achieve than full-fledged ZK). Then, this “honest-verifier” ZKPCP is combined with an unconditionally secure oracle-based commitment primitive called a “locking scheme” [KPT97, IMS12]. This transformation yields ZKPCPs for NP with statistical ZK against query-bounded malicious verifiers, namely ones who are only limited to asking at most $p(|x|)$ queries, for some fixed polynomial $p$ that is much smaller than the proof length, but can be much bigger than the (polylogarithmic) number of queries asked by the honest verifier.

As discussed in Chapter 1, a common limitation of all previous ZKPCP constructions (even ones based on non-adaptively verifiable PCPs) is that they require adaptive verification (namely, the queries of the verifier may depend on the oracle answers to previous queries). Thus, the “cost” of zero-knowledge in these constructions is the adaptivity of the honest verifier, since the celebrated PCP theorem [ALM+92, AS92,
Din06 guarantees that any NP language admits a PCP which can be verified (with soundness error 1/2) by non-adaptively reading only a constant number of proof bits. This state of affairs gives rise to the question of construction ZKPCPs that can be verified non-adaptively, as stated in Question 1 (Chapter 1).

We make a significant first step towards answering this question, and suggest a new approach for the construction of ZKPCPs. We apply leakage-resilient circuit compilers (LRCCs) to construct witness-indistinguishable PCPs (WIPCPs) for NP, a weaker variant of ZKPCPs in which the simulation is not required to be efficient. We then apply the so-called “FLS technique” [FLS90] to convert these WIPCPs into computational ZKPCPs (CZKPCPs) in the common random string model, based on the existence of one-way functions. In such a CZKPCP, the view of any query-bounded PPT verifier can be efficiently simulated, in a way which is computationally indistinguishable from the actual view.

Other than the theoretical interest in this question, our study of PCPs with ZK properties is motivated by their usefulness for cryptographic applications. For instance, ZKPCPs are the underlying combinatorial building blocks of succinct zero-knowledge arguments, which have been the subject of a large body of recent work (see, e.g., [BCG+13, BCTV14a, BCTV14b] and references therein). A more direct application of WIPCPs and ZKPCPs is for implementing efficiently verifiable zero-knowledge proofs in a distributed setting involving a prover, verifier, and multiple (potentially corrupted) servers. This application is described in more details in Chapter 7.

4.1.1 Our Results and Techniques

We now give a more detailed account of our results, and the underlying techniques.

From LRCCs and PCPs to WIPCPs

Let $L$ be an NP-language with a corresponding NP-relation $R_L$, and a boolean circuit $C$ verifying $R_L$. Recall that the prover $P$ in a PCP system for $R_L$ is given the input $x$ and a witness $y$ for the membership of $x$ in $L$, and outputs a proof $\pi$ that is obtained by applying some function $f_P$ to $x, y$. For our purposes, it would be more convenient to think of $f_P$ as a function of the entire wire values $w$ of $C$, when evaluated on $x, y$. In a ZKPCP, few bits in the output of $f_P$ should reveal essentially nothing about the wire values $w$, i.e., $C$ should withstand “leakage” from $f_P$. In general, we cannot assume that $C$ has this guarantee, but using an LRCC, $C$ can be compiled into a circuit $\hat{C}$ with this property. Informally, an LRCC is associated with a function class $\mathcal{L}$ (the leakage class) and a (randomized) input encoding scheme $E$, and compiles a deterministic circuit $C$ into a deterministic circuit $\hat{C}$, that emulates $C$, but operates on an encoded input. The new circuit $\hat{C}$ is leakage-resilient in the following sense: for any input $z$ for $C$, and any $\ell \in \mathcal{L}$, the output of $\ell$ on the wire values of $\hat{C}$, when evaluated on $E(z)$, reveals nothing other than $C(z)$. This is formalized in the simulation-based paradigm (i.e., the
wire-values of $\hat{C}$ can be efficiently simulated given only $C(z)$.

We establish a connection between ZKPCPs and LRCCs. Assume the existence of an LRCC associated with a leakage class $L$, such that any restriction $f_P^I$ of $f_P$ to a “small” subset $I$ of its outputs satisfies $f_P^I \in L$. Then the oracle answers to the queries of a query-bounded verifier $V$ correspond to functions in $L$, since for every possible set $I$ of oracle queries, the answers are $f_P^I(w)$. Therefore, if $w$ is the wire values of a leakage-resilient circuit then the system is ZK. This gives a general method of transforming standard PCPs into ZKPCPs: $P, V$ replace $C(x) = C(x, \cdot)$ (i.e., $C$ with $x$ hard-wired into it) with $\hat{C}_x$; and $P$ proves that $\hat{C}_x$ is satisfiable by generating the PCP $\pi$ from the wire values of $\hat{C}_x$.

This transformation crucially relies on the fact that $\hat{C}_x$ emulates $C_x$ (e.g., if $\hat{C}_x$ always outputs 1 then the resultant PCP system is not sound). However, as noted in Chapter 3, in previous LRCC constructions (e.g., [ISW03, FRR+14, MV13]) this holds only if the encoded input of $\hat{C}_x$ was honestly generated. Moreover, there always exists a choice of an ill-formed “encoding” that satisfies $\hat{C}_x$ (i.e., causes it to output 1). In our case the prover generates the encoded input of $\hat{C}_x$ (the verifier does not know this input), so a malicious prover is able to pick an ill-formed “encoding” that satisfies $\hat{C}_x$, causing the verifier to accept with probability 1. Therefore, soundness requires that if $C_x$ is not satisfiable, then there exists no satisfying input for $\hat{C}_x$ (either well- or ill-formed), namely the LRCC should be SAT-respecting (as in Chapter 3, Definition 3.2.11). Consequently, we base our construction on the SAT-respecting relaxed LRCC of Chapter 3. (The leakage-resilience property is relaxed in the sense that it holds with a possibly inefficient simulator.)

**WIPCPs and CZKPCPs for NP**

Recall that our general transformation described above relied on $f_P$ being in the function class $L$ that is associated with the SAT-respecting LRCC. Thus, constructing ZKPCPs for NP through the aforementioned methodology requires two building blocks: a PCP system for NP in which, roughly speaking, the prover algorithm can be implemented in a computationally-simple complexity class $L$; and an LRCC that resists leakage from $L$. We obtain the first building block by observing that the PCP system of Arora and Safra [AS92] has the property that every “small” subset of proof bits can be generated using a low-depth circuit of polynomial size over the operations $\land, \lor, \lnot, \oplus$, with “few” $\oplus$ gates. For the second building block, we use the SAT-respecting circuit compiler of Chapter 3, Section 3.6, that is relaxed leakage-resilient with respect to this function class. (This construction is based on recent correlation bounds of Lovett and Srivinasan [LS11].) Thus, we obtain the following result, where NA-WIPCP denotes the class of all NP-languages that have a PCP system with a negligible soundness error, polynomial-length proofs, a non-adaptive honest verifier that queries poly-logarithmically many proof bits, and guarantee WI against (adaptive) malicious verifiers querying a fixed polynomial
number of proof bits.

**Theorem 4.1** (NA-WIPCPs for NP). \( \text{NP} = \text{NA} - \text{WIPCP} \).

Using a general technique of Feige et al. [FLS90], and assuming the existence of One-way functions (OWFs), we transform our WIPCP into a CZKPCP in the common random string (CRS) model, in which the PCP prover and verifier both have access to a common random string. Concretely, we prove the following result, where NA-CZKPCP corresponds exactly to the class NA-WIPCP, except that the WI property is replaced with CZK in the CRS model.

**Corollary 4.2** (NA-CZKPCPs for NP). Assume that OWFs exist. Then \( \text{NP} = \text{NA} - \text{CZKPCP} \).

At the end of Section 4.3 we describe a simple alternative approach for constructing CZKPCPs, which applies a PCP on top of a standard non-interactive zero-knowledge proof. However, this alternative relies on stronger assumptions (e.g., the existence of trapdoor functions [FLS90]) than our construction which only relies on an OWF.

**Zero-Knowledge PCPs for NP?**

The zero-knowledge properties of our non-adaptively verifiable PCPs are somewhat weaker than the standard definition of zero-knowledge PCPs, which holds for computationally unbounded verifiers in the standard model, with a simulator whose running time is polynomial in the running time of the (possibly malicious) verifier. This raises the question of whether one can obtain ZKPCPs through this general transformation.

The simulator in our final PCP construction is inefficient because the general transformation from LRCCs to ZKPCPs is applied to a SAT-respecting relaxed LRCC, in which leakage-resilience holds with a possibly inefficient simulator. The PCP simulator emulates the leakage-resilience simulator, and consequently may also be inefficient. In particular, had we applied the transformation to a SAT-respecting non-relaxed LRCC, then the resultant PCP would be zero-knowledge (namely, the ZKPCP simulator would be efficient). However, we give evidence (Section 4.4) that unless \( \text{NP} \subseteq \text{BPP} \), known LRCCs withstanding global leakage [FRR\textsuperscript{+}14, MV13] cannot be transformed into SAT-respecting non-relaxed LRCCs. Intuitively, this is because these constructions admit a simulator which is universal in the sense that it simulates the wire values of the compiled circuit without knowing the leakage function, and the simulated values “fool” all functions in \( \mathcal{L} \). Combining such a SAT-respecting LRCC with PCPs for NP (through the transformation described above) would give a BPP algorithm of deciding any NP-language.

### 4.1.2 Chapter Organization

Section 4.2 contains some preliminaries used throughout the chapter. In Section 4.3 we describe our transformation from non-ZK PCPs, and SAT-respecting LRCCs, to
ZKPCPs, construct non-adaptively verifiable WIPCPs, and CZKPCPs in the CRS model, and prove Theorem 4.1 and Corollary 4.2. Then, in Section 4.4 we give evidence that SAT-respecting non-relaxed LRCCs for NP that withstand leakage from “useful” leakage classes exist only if NP \( \subseteq \) BPP. The analysis of the PCP of Arora and Safra [AS92] (which is used to construct the WIPCPs and CZKPCPs in the CRS model of Theorem 4.1 and Corollary 4.2, respectively) appears in Appendix A.

4.2 Preliminaries

We consider PCP systems with zero-knowledge properties that are somewhat weaker than the standard guarantee of ZKPCPs. The first notion that we consider is witness-indistinguishability (WI), which is similar to the standard notion of zero-knowledge, except that the verifier is not required to be efficient. An alternative (and equivalent) manner of formulating WI is that for every (possibly malicious, possibly adaptive) \( q^* \)-query bounded verifier \( V^* \), every \( x \in L_R \), and every pair \( w_1, w_2 \) of witnesses for \( x \), the views of \( V^* \), when verifying an honestly generated proof for \( (x, w_1) \), and an honestly generated proof for \( (x, w_2) \), are statistically close. Formally,

**Definition 4.2.1** (WIPCP). We say that a probabilistic proof system \((P, V)\) is a **witness-indistinguishable probabilistically checkable proof (WIPCP)** system for an NP-relation \( R = R(x, w) \), if the following holds.

- \((P, V)\) has the syntax, completeness, and soundness guarantees of Definition 2.2.3.
- The zero-knowledge property (Definition 2.2.6) is replaced with the following \((\epsilon_{ZK}, q^*)\)-**witness-indistinguishability (WI)** property. For every (possibly adaptive) \( q^* \)-query bounded verifier \( V^* \), every \( x \in L_R \), and every pair \( w_1, w_2 \) of witnesses for \( x \) (i.e., \((x, w_1), (x, w_2) \in R\)), \( SD(\text{Real}_{V^*}, P(x, w_1), \text{Real}_{V^*}(x, w_2)) \leq \epsilon_{ZK} \), where \( \text{Real}_{V^*}(x, w) \) denote the view of \( V^* \) on input \( \epsilon_S, \epsilon_{ZK}, q^*, x \), and given oracle access to an honest generated proof \( \pi \) for \((x, w)\).

We say that a WIPCP is a **non-adaptive WIPCP (NA-WIPCP)** system for a relation \( R = R(x, w) \), if the honest verifier is non-adaptive. That is,

**Definition 4.2.2** (Non-adaptive WIPCP). We say that a probabilistic proof system \((P, V)\) is a **non-adaptive WIPCP (NA-WIPCP)** system for an NP-relation \( R = R(x, w) \), if it is a WIPCP system as in Definition 4.2.1 in which the honest verifier is non adaptive, i.e., his queries are determined by his inputs and randomness.

**Notation 4.3.** We use \( \text{NA-WIPCP} \left[ r, q, q^*, \epsilon_S, \epsilon_{ZK}, \ell \right] \) to denote the class of NP-languages that admit an NP-relation \( R \) with a non-adaptive \((\epsilon_{ZK}, q^*)\)-WIPCP, in which the prover outputs proofs of length \( \ell \), the honest verifier tosses \( O(r) \) coins, queries \( O(q) \) proof bits, and rejects false claims except with probability
at most $\epsilon_S$. We use $\text{PCP}[r,q,\epsilon,\ell]$ to denote the class of NP-languages admitting a standard (i.e., non-WI) PCP system with the same properties, and write $\mathcal{R} \in \text{PCP}[r,q,\epsilon,\ell]$ to denote that $L_\mathcal{R} \in \text{PCP}[r,q,\epsilon,\ell]$. We denote $\text{NA} – \text{WIPCP} := \text{NA} – \text{WIPCP}[\text{polylog}n,\text{polylog}n,\text{poly}(n),\text{negl}(n),\text{negl}(n),\text{poly}(n)]$.

The second zero-knowledge notion that we consider is computational ZK in the common random string (CRS) model, which we define next. Roughly speaking, a probabilistic proof system $\text{CZKPCP}$ (computational ZKPCP in the CRS model)

**Definition 4.2.3 (Computational ZKPCP in the CRS model).** We say that a probabilistic proof system $(P,V)$ is a computational zero-knowledge probabilistically checkable proof (CZKPCP) system for an NP-relation $\mathcal{R} = \mathcal{R}(x,w)$ in the CRS model, if the following holds.

- **Syntax.** Let $\sigma \in \mathbb{N}$ be a security parameter. The prover $P$ has input $1^{|s|},q^*,1^\sigma,x,w$, and access to a common random string (CRS) $s \in \{0,1\}^\sigma$, and outputs a proof $\pi$ for $(x,w)$ (i.e., $P(1^{|s|},1^\sigma,1^\sigma,x,w,s)$ defines a distribution over proofs for $(x,w)$). The verifier $V$ has input $1^{|s|},q^*,1^\sigma,x$, access to $s$, and oracle access to $\pi$, and outputs either acc or rej.

We associate with $P,V$ as above the following efficiency measures. The alphabet $\Sigma = \Sigma(\epsilon_S,q^*,|x|,\sigma)$ over which $r$ is defined; The length $\ell = \ell(\epsilon_S,q^*,|x|,\sigma)$ of the proof $\pi$; the query complexity $q = q(\epsilon_S,q^*,|x|,\sigma)$ of $V$ (i.e., the number of queries that $V$ makes to his oracle); and the randomness complexity $r = r(\epsilon_S,q^*,|x|,\sigma)$ of $V$ (namely, the number of random bits that he uses).

- **Semantics.** $(P,V)$ has the following properties.

  - **Completeness.** For every $(x,w) \in \mathcal{R}$, every CRS $s \in \{0,1\}^\sigma$, and every proof $\pi \in P(1^{|s|},1^\sigma,1^\sigma,x,w,s)$, $\Pr[V^\pi(1^{|s|},q^*,1^\sigma,x,s) = \text{acc}] = 1$, where the probability is over the randomness of $V$.

  - **Soundness.** For every $x \notin L_\mathcal{R}$ and every $\pi^*$, if $s \in_\mathcal{R} \{0,1\}^\sigma$ then $\Pr[V^{\pi^*}(1^{|s|},q^*,1^\sigma,x,s) = \text{acc}] \leq \epsilon_S$, where the probability is over the choice of $s$, and the random coins used by $V$.

  - $q^*$-computational zero-knowledge (CZK). There exists a PPT simulator $\text{Sim}$ such that the following holds for every (possibly malicious) $q^*$-query-bounded PPT verifier $V^*$, and every $(x,w) \in \mathcal{R}$. The simulator $\text{Sim}$ on input $x,1^\sigma$
generates a simulated CRS $s_{\text{Sim}}$, and gives $s_{\text{Sim}}$ to $V^*$. Then, $V^*$ (adaptively) queries the proof, and $\text{Sim}$ generates simulated answers to these queries. At the end of this interaction, $\text{Sim}$ outputs a simulated view of $V^*$. Then $(x, s, \text{Real}_{V^*,p}(x, w, s)) \approx \text{Sim}(x, 1^{\sigma})$, where $\text{Real}_{V^*,p}(x, w, s)$ denotes the view of $V^*$ on input $1^{|x|}, q^*, x, 1^{\sigma}$, given access to $s \in_R \{0, 1\}^{\sigma}$, and oracle access to $\pi \in_R P(1^{|x|}, 1^{\sigma}, 1^{\sigma}, x, w, s)$; $\text{Sim}(x, 1^{\sigma})$ denotes the output of $\text{Sim}$ on input $x, 1^{\sigma}$; and $\approx$ denotes computational indistinguishability.

Remark. It would sometimes be useful to bound the computational distance between $(x, s, \text{Real}_{V^*,p}(x, w, s))$ and $\text{Sim}(x, 1^{\sigma})$ precisely. Therefore, we say that a CZKPCP system has $(\epsilon_{ZK}, q^*)$-CZK if the computational distance between $(x, s, \text{Real}_{V^*,p}(x, w, s))$ and $\text{Sim}(x, 1^{\sigma})$ is at most $\epsilon_{ZK}$. In this case, $\epsilon_{ZK}$ is given to both $P$ and $V$, and the parameters of the system (i.e., $\Sigma, \ell, q, r$) may depend on $\epsilon_{ZK}$.

Similar to WIPCPs, we consider non-adaptive CZKPCPs:

**Definition 4.2.4 (Non-adaptive CZKPCP).** We say that a probabilistic proof system $(P, V)$ is a non-adaptive CZKPCP (NA-CZKPCP) system for an NP-relation $R = R(x, w)$ in the CRS model, if it is a CZKPCP for $R$ in the CRS model (as in Definition 4.2.3), and the honest verifier is non-adaptive, i.e., his queries are determined by his inputs and randomness.

### 4.3 Non-Adaptive WIPCPs and CZKPCPs in the CRS Model

In this section, we use SAT-respecting relaxed LRCCs (see Section 3.2) to transform standard PCPs into WIPCPs, and CZKPCPs in the CRS model, with a non-adaptive honest verifier. We first describe (Section 4.3.1) a general transformation from PCPs to WIPCPs, and then (Section 4.3.2) apply this general transformation to the PCP of Arora and Safra [ALM+92], obtaining non-adaptive WIPCPs for NP, which proved Theorem 4.1.

Finally, in Section 4.3.3 we use techniques of Fiege et al. [FLS90] to construct CZKPCPs for NP in the CRS model from our WIPCPs for NP, thus proving Corollary 4.2.

#### 4.3.1 From PCPs and SAT-Respecting Relaxed LRCCs to WIPCPs

In this section we describe a transformation from PCPs for 3SAT to NA-WIPCPs for arbitrary NP-relations $R = R(x, w)$. This transformation can be applied to any PCP system for 3SAT in which the proof is obtained from the witness through an “easy” function $f_{3\text{SAT}}$ (we formalize this notion below). More specifically, we will prove the following.
Theorem 4.4. Let \( n \) be a length parameter, \( \epsilon_S, \epsilon_{ZK} \in [0, 1] \), \( S = S(n) \) be a size function, \( q^* = q^*(n) \) be a query function, and \( g(\cdot) \) be a polynomial. Let \( L \) be a family of leakage functions, such that:

- there is a SAT-respecting (\( L, S, \epsilon_{ZK} \))-relaxed LRCC (\( \text{Comp}, E \)) satisfying \( |\text{Comp}(C)| \leq g(|C|) \) for every circuit \( C \);

- there is a PCP \( [r(n), q(n), \epsilon_S, \ell(n)] \) system for 3SAT, such that for every \((\varphi, W) \in 3\text{SAT}\), every subset \( Q \) of \( q^* \) bits of an honestly-generated proof \( \pi = \pi(\varphi, W) \) is computable from \( W \) by a function \( f_{\varphi,Q} \in L \).

Then for every NP-relation \( R = R(x, w) \) with verification circuit \( C^R \) of size at most \( S \), we have that

\[
R \in \text{NA} - \text{WIPCP} \left[ r(m), q(m), q^*, \epsilon_S, O\left(\epsilon_{ZK} \cdot q^* \cdot (\ell(m))^{2q^*}\right) + e^{-\Omega(q^*(\ell(m))^{q^*})}, \ell(m) \right]
\]

where \( m = O\left(g\left(|C^R|\right)\right) \).

The high-level idea of the transformation is the following. If \((x, w) \in R \) then \( w \) satisfies \( C_R(x, \cdot) \) (i.e., \( C_R \) with \( x \) hard-wired into it). If every “small” subset of bits in the output of \( f_{3\text{SAT}} \) constitutes a function in a function class \( L \), then the system can be made WI as follows. The prover and verifier both compile \( C_R(x, \cdot) \) into a SAT-respecting circuit \( \hat{C}_R \) that is relaxed leakage-resilient against \( L \), and then generate a 3CNF \( \varphi \) that represents \( \hat{C}_R \) (see Definition 4.3.2 below). Notice that by the SAT-respecting property, \( \hat{C}_R \) (and consequently, also \( \varphi \)) is satisfiable if and only if \( x \in L_R \). The prover then samples a random encoding \( \hat{w} \) of \( w \) (notice that \([\hat{C}_R, \hat{w}]\) is a satisfying assignment for \( \varphi \)), and generates the PCP \( \pi = f_{3\text{SAT}}(\hat{C}_R, \hat{w}) \). The verifier probabilistically verifies that \( \varphi \) is satisfiable by reading few symbols of \( \pi \), which (if the verifier is non-adaptive) correspond to applying a leakage function from \( L \) to the wire values of \([\hat{C}_R, \hat{w}]\). This gives a WIPCP against non-adaptive verifiers. Formally:

Proposition 4.3.1. Let \( n \) be a length parameter, \( \epsilon_S, \epsilon_{ZK} \in [0, 1] \), \( S = S(n) \) be a size function, \( q^* = q^*(n) \) be a query function, and \( g(\cdot) \) be a polynomial. Let \( L \) be a family of leakage functions, such that:

- there is a SAT-respecting (\( L, \epsilon_{ZK}, S \))-relaxed LRCC (\( \text{Comp}, E \)) satisfying \( |\text{Comp}(C)| \leq g(|C|) \) for every circuit \( C \);

- there is a PCP \( [r(n), q(n), \epsilon_S, \ell(n)] \) system for 3SAT, such that for every \((\varphi, W) \in 3\text{SAT}\), every subset \( Q \) of \( q^* \) bits of an honestly-generated proof \( \pi = \pi(\varphi, W) \) is computable from \( W \) by a function \( f_{\varphi,Q} \in L \).

Then for every NP-relation \( R = R(x, w) \) with verification circuit \( C^R \) of size at most \( S \), we have that \( R \in \text{NA} - \text{WIPCP} \left[ r(m), q(m), q^*, \epsilon_S, 2\epsilon_{ZK}, \ell(m) \right] \), where \( m = O\left(g\left(|C^R|\right)\right) \), and WI holds against non-adaptive verifiers.
Theorem 4.4 will follow as a corollary from Proposition 4.3.1, by employing techniques of [CDD+01] (see Theorem 4.5 below) to extend the WI property from non-adaptive to adaptive verifiers.

Proof of Theorem 4.4. The proof follows by applying Theorem 4.5 to the NA-WIPCP system of Proposition 4.3.1, and by noting that the proofs in the NA-WIPCP system of Proposition 4.3.1 have length $\ell(m)$.

The proof of Theorem 4.4 used the following result of [CDD+01].

**Theorem 4.5 (Implicit in [CDD+01]).** Let $(P,V)$ be a PCP system which is $(\epsilon,q^*)$-WI against non-adaptive verifiers, and in which the prover generates proofs of length $\ell$. Then $(P,V)$ is also $\left(O(\epsilon \cdot q^* \cdot \ell^2 q^*), q^*\right)$-WI against adaptive verifiers.

Therefore, it remains to prove Proposition 4.3.1. We first explicitly define the WIPCP system, starting with the 3CNF-representation of circuits.

**Representing computations as 3CNFs**

We will use boolean formulas to represent computations of boolean circuits with no $\oplus$ gates.

**Definition 4.3.2 (Canonical 3CNFs representing boolean circuits).** A 3CNF is a conjunction of clauses, where each clause contains exactly 3 literals (a literal is a variable or its negation). Given a circuit $C$, we define the canonical 3CNF representing $C$, denoted $\varphi_C$, as follows.

- For every input gate $g_i$ of $C$ we introduce a variable $x_i$.
- For every gate $g$ of $C$, with input wires $a,b$ and output wire $c$:
  - We introduce a variable $x_c$. (Notice that if the gates are traversed from the input gates to the output gate, then the variables $x_a,x_b$ corresponding to $a,b$ have already been defined.)
  - We define a 3CNF $\varphi_g$ as follows:
    
    \[ * \text{gate, } c = a \land b : \varphi(x_a,x_b,x_c) = (x_c \lor \neg x_a \lor \neg x_b) \land (\neg x_c \lor x_a \lor \neg x_c) \land (\neg x_c \lor x_b \lor \neg x_c) \]  
    \[ * \text{gate, } c = a \lor b : \varphi(x_a,x_b,x_c) = (\neg x_c \lor x_a \lor x_b) \land (\neg x_c \lor \neg x_a \lor \neg x_c) \lor (\neg x_c \lor x_b \lor x_c). \]

\[ ^1 \text{Notice that some variables appear twice in the same clause. This is not needed for the functionality of the formula, but is required for } \varphi \text{ to be a 3CNF. Instead, we could have introduced new variables, where a clause of the form } a \lor b \text{ would have been replaced with the 3CNF } (a \lor b \lor z) \land (a \lor b \lor \neg z), \text{ where } z \text{ is a new variable. The alternative transformation has the advantage that each variable appears at most once in every clause, but increases the number of variables. As we will not require that every variable appears at most once in each clause, we have chosen the first transformation, which has the advantage that a wire assignment to } C \text{ is also an assignment to } \varphi_C.\]
\* g is a \text{\neg} gate, c = \neg a: \quad \varphi(x_a, x_c) = (\neg x_c \lor x_a \lor \neg x_c) \land (x_c \lor x_a \lor x_c).

- For the output gate \(g_o\) of \(C\), with output wire \(o\), we concatenate the clause \((x_o \lor x_o \lor x_0)\) to \(\varphi_{g_o}\), i.e., we obtain a new 3CNF \(\varphi_{g_o} \land (x_o \lor x_o \lor x_o)\).

- \(\varphi_C = \land_g \varphi_g\), where the conjunction is over all gates except input gates.

**Example 4.3.3.** Let \(C : \{0, 1\}^2 \rightarrow \{0, 1\}\), \(C(y, z) = (y \land z) \lor (\neg y)\). Let \(g_\land, g_\lor, g_\neg\) denote the \&, \lor, \neg gates of \(C\), and notice that \(g_\lor\) is also the output gate. Then:

- The variables of \(\varphi_C\) are \(x_y, x_z\) (corresponding to the input gates of \(C\)), and \(x_\land, x_\lor, x_\neg\) (corresponding to the output wires of \(g_\land, g_\lor, g_\neg\), respectively).
- \(\varphi_{g_\land}(x_y, x_z, x_\land) = (x_\land \lor \neg x_y \lor \neg x_z) \land (\neg x_\land \lor x_y \lor \neg x_\land) \land (\neg x_\land \lor x_z \lor \neg x_\land)\).
- \(\varphi_{g_\lor}(x_y, x_\lor) = (\neg x_\lor \lor x_\land \lor x_\neg) \land (x_\lor \lor \neg x_\land \lor x_\lor) \land (x_\lor \lor \neg x_\neg \lor x_\lor)\).
- \(\varphi_{g_\neg}(x_\land, x_\lor, x_\neg) = (\neg x_\lor \lor x_\land \lor x_\neg) \land (x_\lor \lor \neg x_\land \lor x_\lor) \land (x_\lor \lor \neg x_\neg \lor x_\lor)\).
- \(\varphi_C(x_y, x_z, x_\land, x_\lor, x_\neg, x_\lor) = \varphi_{g_\land} \land \varphi_{g_\lor} \land \varphi_{g_\neg}\).

Notice that the variables of \(\varphi_C\) correspond to the wires of \(C\). \(\varphi_C\) represents \(C\) in the sense that a wire assignment to \(C\) (which is also an assignment to the variables of \(\varphi_C\)) satisfies \(\varphi_C\) only if it corresponds to the evaluation of \(C\) on a satisfying input, as stated in the next fact.

**Fact 4.3.4.** Let \(C\) be a boolean circuit, and let \(\varphi_C\) be the canonical 3CNF representing \(C\) (as in Definition 4.3.2). Then a wire assignment \(W\) to \(C\), which is also an assignment to the variables of \(\varphi_C\), satisfies \(\varphi_C\) if and only if \(W\) is the assignment to the wires of \(C\) when evaluated on a satisfying input \(x\). Moreover, \(\varphi_C\) can be constructed from \(C\) in linear time, so \(|\varphi_C| = O(|C|)|\), where \(|\varphi|\) denotes the number of clauses in \(\varphi\).

Using the canonical 3CNF representation of boolean circuits, we can now formally describe the NA-WIPCP system.

**Construction 4.6.** Let \((P_{3SAT}, V_{3SAT})\) be a PCP system for 3SAT, and \((\text{Comp}, E = (\text{Enc}, \text{Dec}))\) be a SAT-respecting \((\mathcal{L}, S(n), \epsilon(n))\)-relaxed LRCC. Let \(\mathcal{R} = \mathcal{R}(x, w)\) be an NP-relation with verification circuit \(C^\mathcal{R}\). (More precisely, \(C^\mathcal{R}\) is a family \(\{C^\mathcal{R}_n\}\) of circuits, where \(C^\mathcal{R}_n\) is applied to inputs \(x\) of length \(n\). To simplify notations, we denote all circuits in the family by \(C^\mathcal{R}\).) The NA-WIPCP system consists of the prover \(P\) and the verifier \(V\).

**Prover algorithm.** On input \((x, w) \in \mathcal{R}\):

- Let \(C^\mathcal{R}(x, \cdot)\) denote the circuit \(C^\mathcal{R}\) with \(x\) hard-wired into it, then \(P\) computes \(\hat{C}_x(\cdot) = \text{Comp}(C^\mathcal{R}(x, \cdot))\).
- Samples a random encoding $\hat{w} \leftarrow \text{Enc}(w, 1^{\lceil |C^R| \rceil})$, and computes the internal wires of $\hat{C}_x(\hat{w})$. Let $\mathcal{W}$ denote this wire assignment.
- Construct the canonical 3CNF $\varphi_x$ representing $\hat{C}_x$.
- Outputs a proof $\pi \in R 3_{\text{SAT}}(\varphi_x, \mathcal{W})$ for the claim “$\varphi_x \in 3_{\text{SAT}}$”.

**Verifier algorithm.** On input $x$, and given oracle access to $\pi$, $V$ computes $\hat{C}_x(\cdot) = \text{Comp}(C^R(x, \cdot))$, and constructs the 3CNF formula $\varphi_x$. Then, $V$ runs $V_{3_{\text{SAT}}}^\pi(\varphi_x)$ (and accepts or rejects according to the output of $V_{3_{\text{SAT}}}$).

**Remark 4.7.** The prover and verifier of the PCP system for 3SAT may expect additional parameters (such as a soundness error parameter, etc.) as part of their input. In such cases, these parameters would also be given as input to the prover and verifier of the NA-WIPCP, who will pass them on to $P_{3_{\text{SAT}}}, V_{3_{\text{SAT}}}$. However, to make the construction clearer, we chose not to explicitly include these additional parameters in Construction 4.6.

Next, we prove Proposition 4.3.1.

**Proof of Proposition 4.3.1.** We show that the system of Construction 4.6, when using $(P_{3_{\text{SAT}}}, V_{3_{\text{SAT}}})$ and $(\text{Comp}, E)$ as the underlying components, has the required properties.

**Parameters.** The wire assignment $\mathcal{W}$ to $\hat{C}_x$ has size $|\mathcal{W}| = |\hat{C}_x| \leq g(|C^R(x, \cdot)|) \leq g(|C^R|)$. $V$ emulates the verification procedure of the inner verifier $V_{3_{\text{SAT}}}$, on input $\varphi_x$ of size $|\varphi_x| = O(|\hat{C}_x|) \leq O(g(|C^R|))$. Therefore, the query complexity of $V$ is $O(g(O(g(|C^R|))))$, and the randomness complexity is $O(r(O(g(|C^R|))))$. The proof is generated from the witness $\mathcal{W}$, that has size at most $O(g(|C^R|))$, so the proof has size $\ell(O(g(|C^R|)))$.

**Perfect completeness** follows directly from the perfect completeness of the underlying systems.

**Soundness.** Let $x \notin L_R$, then for every “witness” $w$, $C(x, w) = 0$, i.e., $C_x$ is not satisfiable. Since $(\text{Comp}, E)$ is SAT-respecting, $\hat{C}_x$ is not satisfiable, i.e., $\varphi_x \notin 3_{\text{SAT}}$.

(Notice that here perfect SAT-respecting is crucial, otherwise there may exist a satisfying input $\hat{w}^*$ for $\hat{C}_x$, even though $C(x, \cdot)$ is not satisfiable, and a malicious prover would use $\hat{w}^*$ to convince the verifier.) Therefore, the soundness of $(P_{3_{\text{SAT}}}, V_{3_{\text{SAT}}})$ guarantees that for every $\pi^*$, $\Pr[V_{3_{\text{SAT}}}^\pi^*(x) = \text{acc}] = \Pr[V_{3_{\text{SAT}}}^{\hat{w}^*}(\varphi_x) = \text{acc}] \leq \epsilon_S$.

**Witness-indistinguishability.** Let $x \in L_R$, $\varphi_x$ be the canonical 3CNF representing $\hat{C}_x$, and $w_1, w_2$ be two witnesses for $x$. Let $V^*$ be a non-adaptive $q^*\text{-query-bounded}$ verifier, and let $\pi_1, \pi_2$ be proofs that were randomly generated by the honest prover $P$ for $w_1, w_2$, respectively. The entire view of $V^*$ can be reconstructed from the oracle answers to his queries, and since applying a function to a pair of random variables does

---

$^2$Since $g(|C^R|)$ is an upper-bound on the size of the formula, we assume here that $q, r, \ell$ are non-decreasing, which is without loss of generality.

111
not increase the statistical distance between them, it suffices to show that for every set \( Q \) of \( q^* \) symbols of \( \pi_1, \pi_2 \), \( \text{SD} (\pi_1|_Q, \pi_2|_Q) \leq 2\epsilon_{\text{ZK}} \), where \( \pi_i|_Q \) denotes the restriction of \( \pi_i \) to the coordinates indexed by \( Q \).

Since \( P \) evaluates \( \hat{C}_x \) on random encodings \( \hat{w}_i \leftarrow \text{Enc}(w_i, 1^{\ell(x,i)}) \) of \( w_i, i = 1, 2 \), and \( |C[R]| \leq S \), then the relaxed leakage-resilience of \( (\text{Comp}, E) \) guarantees that for every \( f \in L \), \( \text{SD} \left( f \left[ \hat{C}_x, \hat{w}_1 \right], f \left[ \hat{C}_x, \hat{w}_2 \right] \right) \leq 2\epsilon_{\text{ZK}}. \)

Using the union bound, this holds because both \( \pi_1|_Q, \pi_2|_Q \) are \( \epsilon_{\text{ZK}} \)-statistically close to the output of the leakage function on the simulated wire values generated by the simulator of the relaxed LRCC. Moreover, \( f_{\phi,x}, Q \in L \), and \( \pi_i|_Q = f_{\phi,x}, Q \left[ \hat{C}_x, \hat{w}_i \right] \) (recall that \( f[C, x] \) denotes the output of \( f \), when given as input the wire assignment of the circuit \( C \) when evaluated on input \( x \), so by the \( (\epsilon_{\text{ZK}}, L) \)-relaxed leakage-resilience of \( \text{Comp}, \text{SD} (\pi_1|_Q, \pi_2|_Q) \leq 2\epsilon_{\text{ZK}}. \)

\[ \Box \]

### 4.3.2 WIPCPs for NP

In this section we use the general transformation of Section 4.3.1 to obtain WIPCPs for NP, and prove Theorem 4.1. We first use the SAT-respecting, relaxed-LRCC of Theorem 3.23 (Chapter 3) to formulate a specific restriction on the function transforming a satisfying assignment (for a 3CNF) into a corresponding PCP, such that NP-relations with “relatively small” verification circuits admit NA-WIPCP systems.

**Corollary 4.8.** Let \( n \) be a length parameter, \( \epsilon_S, \epsilon_{\text{ZK}} \in [0, 1] \), and \( q^* = q^*(n) = \text{poly}(n) \) be a query function. Assume that \( 3\text{SAT} \in \text{PCP} (r(n), q(n), \epsilon_S, \ell(n)) \) with the system \( (P_{3\text{SAT}}, V_{3\text{SAT}}) \), where \( \ell(n) = \text{poly}(n) \), and for every \( (\phi, W) \in 3\text{SAT} \), every bit of an honestly-generated proof \( \pi = \pi(\phi, W) \) is computable by \( L_{O(1), \text{poly}(n), \oplus 1} \) (i.e., by a constant-depth, poly-sized, boolean circuit with unbounded fan-in and fan-out \( \land, \lor, \neg, \oplus \) gates, out of which only one is an \( \oplus \) gate, see Definition 3.6.2). Then \( \text{NP} \subseteq \text{NA} - \text{WIPCP} (r(\text{poly}(n)), q(\text{poly}(n)), q^*, \epsilon_S, \text{negl}(n), \ell(\text{poly}(n))) \).

**Proof.** The corollary follows by applying Theorem 4.4 to the circuit compiler of Theorem 3.23. More specifically, let \( d, c \in \mathbb{N} \) be constants such that every proof bit generated by the honest \( P_{3\text{SAT}} \) is computable from the NP-witness in \( L_{d,n^c,\oplus 1} \), where \( n \) denotes the witness length. Moreover, proofs generated by the prover have length at most \( n^{ec''} \), for some constant \( c'' \).

Let \( R \) be an NP-relation with verification circuit \( C = C_R \), then \( |C| = n^{c'} \) for some constant \( c' \) (because \( R \) is polynomially bounded and efficiently computable). We use Theorem 3.23 with parameters \( d_{\text{Comp}} = d, s'_{\text{Comp}} = |C|, n_{\text{Comp}} = n, m_{\text{Comp}} = 1, k_{\text{Comp}} = nq^*, \) and take \( c_{\text{Comp}} \geq c \) to be a large enough constant whose value will be set later (the subscript \( \text{Comp} \) is used to denote the parameters in the statement of Theorem 3.23, and to differentiate them from the constants mentioned in the proof).

Notice that \( s'_{\text{Comp}}, m_{\text{Comp}}, k_{\text{Comp}} \) are all polynomial in \( n \). We apply the circuit compiler \( (\text{Comp}, E) \) obtained from Theorem 3.23 with these parameters, to \( C \), and obtain a circuit \( \hat{C} \) of size \( |\hat{C}| \leq S(n) \) which, because \( |C| \leq s'_{\text{Comp}} \), is \( \left( L_{d,n^c,\oplus 1}, 2^{-n^{ec_{\text{Comp}}}} \right) \).
relaxed leakage-resilient (where $S(n)$ is the polynomial whose existence is guaranteed by Theorem 3.23). Let $t \in \mathbb{N}$ be the constant such that $S(n) = n^t$.

Let $\varphi$ denote the canonical 3CNF representing $\hat{C}$ (see Definition 4.3.2). Then a wire assignment $W$ for $\hat{C}$, which is also an assignment to the variables of $\varphi$, has size $|W| = |\hat{C}| \leq S(n)$. Moreover, every proof bit of a proof generated by $P_{3\text{SAT}}$ for $\varphi$ can be generated from $W$ in $L_{d,S^∗(n)}∅1$. Therefore, every $q^∗$ proof bits are computable from $W$ in $L_{d,S^∗(n)}∅1$, and Theorem 4.4 guarantees that the system $(P,V)$ of Construction 4.6 is an NA-WIPCP system for $R$, with $(q^∗,O(q^∗\cdot t2q^∗\cdot 2^{−n^\text{comp}}) + e^{−Ω(q^∗\cdot t^∗)})$-WI (where $t$ denotes the proof length), perfect completeness, and soundness error $ε_S$. Since $g(\{|C|\}) = S(n) = n^t$, then the proof length is $t = (n^t) = n^{c^t}$.

We set $c_{\text{Comp}}$ to be large enough, such that the statistical error in the relaxed leakage-resilience property

$$O\left(q^∗ \cdot (n^{c^t} )^{2q^∗ \cdot 2^{−n^\text{comp}} } + e^{−Ω(q^∗\cdot t^∗)}\right) = \text{negl}(n).$$

(Such a choice of $c_{\text{Comp}}$ exists since

$$O\left(q^∗ \cdot (n^{c^t} )^{2q^∗ \cdot 2^{−n^\text{comp}} } + e^{−Ω(q^∗\cdot t^∗)}\right) \leq O\left((q^∗\cdot t^{2c^∗}q^∗ \cdot 2^{−n^\text{comp}}) + e^{−Ω(q^∗\cdot t^∗)}\right)$$

where the left hand side is equal to $2^{c^∗2c^∗q^∗k\log(nq^∗)} + 2^{−c^∗c^∗q^∗n^{c^∗q^∗}}$ for some constant $c^∗$, because $q^∗ = \text{poly}(n)$, and $t, c^∗, c_{\text{Comp}}$ are constants.) Finally, we note that the honest verifier is non adaptive, tosses $O(r(\text{poly}(nq^∗))) = O(r(\text{poly}(n)))$ coins and reads $O(q(\text{poly}(nq^∗))) = O(q(\text{poly}(n)))$ bits of the proof.

In Appendix A we show that the PCP system of Arora and Safra [AS92] for 3SAT has the property that every proof bit is computable from the witness by an $AC^0$ circuit, augmented with a single $⊕$ gate of unbounded fan-in and fan-out. Combining this PCP system with Corollary 4.8 yields Theorem 4.1. We first restate the theorem.

**Theorem (Theorem 4.1, restated).** $NP = NA − \text{WIPCP}$. 

**Proof of Theorem 4.1.** We show that $NP \subseteq NA − \text{WIPCP}$. Let $(P^{AS}, V^{AS})$ denote the PCP system of Arora and Safra [AS92, Theorem 1] (used to prove that $3\text{SAT} \in \text{PCP}[\log n, \log^2 n, \frac{1}{2}, \text{poly}(n)]$). In their system, there exists a constant $c > 0$ such that every proof bit is computable given the witness by a depth-3 boolean circuit, where the first layer contains $n^c$ constant gates, the second layer contains $n^c$ unbounded fan-in $\land$ gates, and the third layer consists of a single unbounded fan-in $⊕$ gate. (See Appendix A for a more detailed description of $(P^{AS}, V^{AS})$.) In particular, every proof bit is computable, given the witness, by an $AC^0$ circuit, augmented with a single $⊕$ gate of unbounded fan-in and fan-out. Applying Corollary 4.8 to $(P^{AS}, V^{AS})$ with $q^∗ = \text{poly}(n)$, and amplifying soundness via repetition, we get that $NP \subseteq \text{WIPCP}[\text{polylog}(n), \text{polylog}(n), \text{poly}(n), \text{negl}(n), \text{negl}(n), \text{poly}(n)]$. 

113
4.3.3 CZKPCPs for NP in the CRS Model

In this section we use NA-WIPCPs, and a technique of Fige et al. [FLS90], to construct NA-CZKPCPs in the CRS model, thus proving Corollary 4.2. (We note that though [FLS90] use this technique to construct non-interactive zero-knowledge proofs (NIZKs) from non-interactive WI proofs, the same transformation can also be applied to WIPCPs.)

The high-level idea of the construction is as follows. To construct a CZKPCP for an NP-relation \( R_L \), we use a WIPCP system for a “related” NP-relation, which admits a trapdoor used by the simulator in the simulation (this trapdoor is the reason that a CRS is needed). Formally,

Construction 4.9. Let \( \sigma \in \mathbb{N} \) be a security parameter, and \( G : \{0,1\}^{\sigma} \rightarrow \{0,1\}^{p(\sigma)} \) be a PRG with stretch function \( p : \mathbb{N} \rightarrow \mathbb{N} \). Let \( L \) be an NP-language with the corresponding NP-relation \( \mathcal{R} = \mathcal{R}(x,w) \), then we define an NP-language \( L_G \) as follows:

\[
L_G = \{ (x,y) : x \in L \lor y \in \text{Im}(G) \}.
\]

Let \( \mathcal{R}_G \) denote the corresponding NP-relation, and let \((P^{\text{in}},V^{\text{in}})\) be a WIPCP system for \( L_G \).

**Prover algorithm.** The prover \( P \), on input \((x,w) \in \mathcal{R}\), and given access to a CRS \( s \in \{0,1\}^{p(\sigma)} \), runs \( P^{\text{in}} \) on \((x,s),w\) and outputs the proof \( \pi \) that \( P^{\text{in}} \) outputs.

**Verifer algorithm.** The verifier \( V \), on input \( x \), given access to the CRS \( s \), and oracle access to a proof \( \pi \), runs \( V^{\text{in}} \) on input \((x,s)\), with oracle \( \pi \), and outputs whatever \( V^{\text{in}} \) outputs.

The next proposition summarizes the properties of Construction 4.9. We say that a PRG \( G \) has \( \epsilon(n) \)-pseudorandom output against non-uniform distinguishers, if for every \( n \), and every (possibly non-uniform) polynomial distinguisher \( D \),

\[
\Pr_{u \in U_n}[D(u) = 1] - \Pr_{s \in U_n}[D(G(s)) = 1] = O(\epsilon(n)),
\]

where \( U_n \) denotes the uniform distribution over \( \{0,1\}^n \).

**Proposition 4.3.5.** Let \( G : \{0,1\}^{\sigma} \rightarrow \{0,1\}^{p(\sigma)} \) be a PRG, whose output is \( \epsilon_G(\sigma) \)-pseudorandom against non-uniform distinguishers, where \( p : \mathbb{N} \rightarrow \mathbb{N} \) is a stretch function. Let \( L_G \) be as defined in Construction 4.9. If \( L_G \in \mathsf{NA-WIPCP} [r(n),q(n),q^*(n),\epsilon_S(n),\epsilon_{\text{ZK}}(n),\ell(n)] \) with the system \((P^{\text{in}},V^{\text{in}})\), then \( \mathcal{R} \in \mathsf{NA-CZKCP}[r(n+p(\sigma)),q(n+p(\sigma)),q^*(n+p(\sigma)),\epsilon_S(n+p(\sigma)) + 2^\sigma-\epsilon(\sigma),\epsilon_{\text{ZK}}(n+p(\sigma)) + \epsilon_G(\sigma),\ell(n+p(\sigma))] \) with the system of Construction 4.9.

**Proof.** We analyze the properties of Construction 4.9.

**Parameters.** Follows from the fact that the underlying WIPCP is run on instances of length \( n + p(\sigma) \).

**Perfect completeness.** Follows from the perfect completeness of the underlying WIPCP system, and the definition of \( L_G \).

---

\(^3\)It suffices for \((P^{\text{in}},V^{\text{in}})\) to be a WIPCP system for any NP-complete language \( L^m \), since the prover and verifier of Construction 4.9 can reduce instances in \( L_G \) to instances in \( L^m \) using a witness-preserving transformation. We choose to use a WIPCP system for \( L_G \) because it makes the presentation clearer.
Soundness. Let $x \notin L$, $\pi^*$ be the purported “proof” provided to $V$, and $s \in_R \{0,1\}^{p(\sigma)}$. If $s \notin \text{Im}(G)$ then $(x, s) \notin L_G$ (because $x \notin L$), so the soundness of $(P^{\text{in}}, V^{\text{in}})$ guarantees that $V^{\text{in}}$ (and consequently, $V$) accepts with probability at most $\epsilon_S(n + p(\sigma))$. Since $s \in_R \{0,1\}^{p(\sigma)}$ and $|\text{Im}(G)| \leq 2^N$ then $s \notin \text{IM}(G)$ except with probability at most $2^{\sigma-p(\sigma)}$. Therefore, $V$ accepts with probability at most $\epsilon_S(n + p(\sigma)) + 2^{\sigma-p(\sigma)}$.

CZK. Let $V^*$ be a $q^*$-query bounded PPT verifier, and we describe the simulator $\text{Sim}$. On input $x$, $\text{Sim}$ picks $z \in_R \{0,1\}^\sigma$ and computes $s = G(z)$. Then, it runs $P^{\text{in}}$ to generate a proof $\pi$ for $(x, s)$, providing $z$ as the witness to $P^{\text{in}}$. $\text{Sim}$ then emulates $V^*$ with input $x$, CRS $s$, and proof oracle $\pi$, and outputs the view of $V^*$ in this interaction (which includes the CRS $s$). We prove that $(x, s, \text{Real}_{V^*, P}(x, w, s)) \approx_{\epsilon_{\text{ZK}}} (\epsilon_{\text{CZK}}(n + p(\sigma))) + \epsilon_{\text{CZK}}(\sigma) \text{Sim}(x, 1^\sigma)$, where $\approx^\epsilon$ denotes computational distance of $\epsilon$.

We define a hybrid distribution $\mathcal{H}$, describing a mental experiment in which the simulator is given the witness $w$, and uses it to generate the proof $\pi$, but generates the CRS as the PRG image of a random value. Concretely, the hybrid $\mathcal{H}$ is defined as follows. $\text{Sim}$ generates $z \in_R \{0,1\}^\sigma$ and $s = G(z)$, runs $P^{\text{in}}$ with input $((x, s), w)$ to obtain a proof $\pi$, and emulates $V^*$ on input $x$, CRS $s$ and proof $\pi$.

Notice that $(x, s, \text{Real}_{V^*, P}(x, w, s)) \approx_{\epsilon_{\text{CZK}}} \mathcal{H}$. Indeed, if a PPT distinguisher $\mathcal{D}$ could distinguish between the two with advantage more than $\epsilon_{\text{CZK}}(\sigma)$, then we would obtain a non-uniform PPT distinguisher $\mathcal{D}_G$ between a random string and the PRG output, achieving distinguishing advantage at least $\epsilon_{\text{CZK}}(\sigma)$: $\mathcal{D}_G$ on input $s \in_R \{0,1\}^{p(\sigma)}$ would run $P^{\text{in}}$ on input $((x, s), w)$ to generate a proof $\pi$, then run $V^*$ on $x, s, \pi$, and feed $\mathcal{D}$ with $x, w$, and the view of $V^*$. (We note that $\mathcal{D}_G$ is non-uniform since $x, w$ are hard-wired into it.)

Second, the $(\epsilon_{\text{ZK}}, q^*)$-WI of the underlying WIPCP guarantees that $\text{Sim}(x, 1^\sigma) \approx_{\epsilon_{\text{ZK}}(n + p(\sigma))} \mathcal{H}$, since in both cases the verifier is run on an honestly-generated proof for the same instance, only using different witnesses. (We note that the instance is $(x, s)$ and therefore it is not fixed, but if the hybrids are not $\epsilon_{\text{ZK}}(n + p(\sigma))$-computationally close then using an averaging argument we can fix $s$.)

The proof of Corollary 4.2 now follows from Proposition 4.3.5 by instantiating Construction 4.9 with the NA-WIPCP system of Theorem 4.1. Recall that by OWF we mean a function which is one-way against non-uniform adversaries. The existence of such functions implies (by standard reductions) the existence of PRGs that are pseudorandom against non-uniform distinguishers. We first restate Corollary 4.2.

**Corollary (Corollary 4.2, restated).** Assume that OWFs exist. Then $\text{NP} = \text{NA - CZKPCP}$.

**Proof of Corollary 4.2.** Assuming OWFs exist, there exists a PRG $G : \{0,1\}^\sigma \rightarrow \{0,1\}^{2^\sigma}$ whose output is negl$(\sigma)$-pseudorandom against non-uniform distinguishers ($G$ can be constructed using standard reductions). We take $\sigma = n$, and $L = 3\text{SAT}$, then by
Theorem 4.1

\[ L_G \in \text{NA} - \text{WIPCP}[\text{polylog}_n, \text{polylog}_n, \text{poly}(n), \text{negl}(n), \text{negl}(n), \text{poly}(n)] \]
(because \( L_G \in \text{NP} \)). Therefore, by Proposition 4.3.5,

\[ \text{SAT} \in \text{NA} - \text{CZKPCP}[\text{polylog}_n, \text{polylog}_n, \text{poly}(n), \text{negl}(n), \text{negl}(n), \text{poly}(n)]. \]

Alternative CZKPCP constructions in the CRS model

We note that a simple alternative construction of CZKPCP for NP can be obtained by applying a standard PCP on top of a standard NIZK proof [BSMP91, Gol01]. Concretely, the CZKPCP prover generates a PCP for the NP-claim “there exists a NIZK for the claim \( x \in L_R \), relative to the CRS \( s \), that would cause the NIZK-verifier to accept”, where the witness is the NIZK proof string. Since the NIZK itself is CZK, the resultant PCP is also CZK. However, NIZK proofs for NP are not known to follow from the existence of one-way functions, and can currently be based only on much stronger assumptions such as the existence of trapdoor permutations [FLS90].

4.4 LRCC-Based ZKPCPs for NP Imply NP \( \subseteq \text{BPP} \)

In this section we give evidence that the techniques we use to construct WIPCPs cannot be used to construct ZKPCPs, unless \( \text{NP} \subseteq \text{BPP} \). At a high level, this is because the leakage-resilience guarantee of (non-relaxed) LRCCs withstanding global leakage is (in some sense) universal. Concretely, the LRCC simulator generates, in polynomial time, wire values for the entire computation of the circuit; and these wire values simultaneously fool every leakage function in the family of leakage functions. When the LRCC is used to construct PCPs with ZK guarantees, this gives a ZKPCP simulator that generates a “fake witness” that can be used to construct a “fake PCP”. This “fake PCP” is simultaneously “good” for every (small) set of verifier queries, in the sense that with high probability, the verifier is convinced. In effect, the simulator “commits” to a fake proof in advance, and this can be used to probabilistically decide the language, namely, the language is in \( \text{BPP} \).

We emphasize that the simulator constructed here is stronger than the simulator whose existence is guaranteed by the ZK property, since the simulator in the definition of ZK is first given the queries, and is then required to generate a partial fake proof, which should convince the verifier \emph{conditioned on the event that it queries the particular pre-determined set of queries}. (This is the case when the verifier is non-adaptive. For adaptive verifiers, for every set of queries of the verifier, the simulator adaptively constructs the simulated proof symbols queried by this set.) We first formally define this stronger simulation notion, which we call \emph{oblivious zero-knowledge}. 

116
**Definition 4.4.1** (Oblivious zero-knowledge). We say that a probabilistic proof system \((P, V)\) is an oblivious ZKPCP system for a relation \(R = R(x, w)\), if it has the syntax, completeness and soundness properties of a WIPCP system (as defined in Definition 4.2.1), and the following \((\epsilon, q^*-)\)-oblivious zero-knowledge property: for every \(q^*\)-query-bounded verifier \(V^*\) there exists a PPT simulator \(\text{Sim}\) such that the following holds for every \((x, w)\) in \(R\). \(\text{Sim}\) is given input \(x, 1^\sigma\), and outputs a (possibly ill-formed) proof \(\pi^*\), such that \(\text{SD}(\text{Real}_{V^*, P}(x, w), \text{Simulated}_{V^*, \pi^*}(x)) \leq \epsilon\), where \(\text{Real}_{V^*, P}(x, w)\) (\(\text{Simulated}_{V^*, \pi^*}(x)\)) denotes the view of \(V^*\) on input \(1^{|s|}, q^*, x, 1^{[w]}\), and given oracle access to \(\pi \in R P(1^{[s]}), 1^{q^*}, x, w)\) (to \(\pi^*\)).

The next remark states that if our transformation from SAT-respecting relaxed-LRCC to WIPCP used a SAT-respecting (non-relaxed) LRCC, then the resultant probabilistic proof system would have oblivious ZK.

**Remark 4.10.** We note that if the SAT-respecting relaxed-LRCC of Proposition 4.3.1 (and Theorem 4.4) was replaced with a SAT-respecting (non-relaxed) LRCC, then the resultant PCP system would be oblivious-ZK. Indeed, the PPT simulator of the LRCC would give a PPT simulator for the PCP system. Moreover, the LRCC simulator generates simulated wire values for the entire circuit, and these wire values constitute a complete “fake” witness, so the PCP simulator could generate an entire “fake” proof \(\pi^*\) (the simulator would first run the LRCC-simulator to obtain the fake witness, then run the honest PCP prover to obtain a fake proof from this fake witness). Moreover, every possible set of verifier queries to the proof corresponds to applying a leakage function (from the family of leakage functions against which the LRCC is leakage-resilient) on the wire values of the compiled circuit, so \(\pi^*\) is simultaneously convincing (with high probability) for every possible set of verifier queries.

**Definition 4.4.2** (BPP). The complexity class BPP (bounded-error probabilistic polynomial time) consists of all languages \(L \subseteq \{0, 1\}^*\) for which there exists a PPT algorithm \(A_L\) with the following properties (the probability is over the randomness of \(A_L\)):

- \(x \in L \Rightarrow \Pr[|A_L(x) = \text{accept}] \geq \frac{2}{3}\).
- \(x \notin L \Rightarrow \Pr[|A_L(x) = \text{accept}] \leq \frac{1}{3}\).

The next theorem states that oblivious ZKPCP systems exist only for languages in BPP. As we show below (Corollary 4.13), the PCP systems obtained from Construction 4.6, when it uses an LRCC with a PPT simulator, are oblivious-ZKPCPs. Therefore, it seems implausible that we could use this transformation to obtain the stronger notion of a ZKPCP for NP.

**Theorem 4.11.** Let \(R = R(x, w)\) be a polynomially-bounded relation. If \(L_R\) has an \([r, q^*, \epsilon_S, \epsilon_{ZK}, \ell]-ZKPCP\) system with oblivious ZK, constants \(\epsilon_S, \epsilon_{ZK} \leq \frac{1}{6}\), and \(r, q, \ell = \text{poly} (n)\) such that \(q \leq q^*\), then \(L \in \text{BPP}\).
Remark 4.12. We have defined probabilistic proofs systems as having perfect completeness. For Theorem 4.11 to hold, the probabilistic proof system should only have a noticeable gap between the probability of accepting $x \in L$ and $x \notin L$. Specifically, it suffices that every $x \in L$ is accepted with probability at least $\frac{5}{6}$ (when the verifier is given an honestly-generated proof), and that every $x \notin L$ is accepted with probability at most $\frac{1}{6}$ (for any purported proof given to the verifier). Moreover, we only need honest-verifier ZK, i.e., that an efficient simulator exists for the honest verifier, and it suffices to have a computational distance of at most $\frac{1}{6}$ between $\text{Real}_{V^*,P}(x, w)$ and $\text{Simulated}_{V^*,\pi^*}(x)$.

Proof of Theorem 4.11. Let $(P, V)$ be the oblivious ZKPCP system for $L = L_R$, and let $\text{Sim}$ denote the corresponding simulator. We assume that every $x \in L$ is accepted with probability at least $\frac{5}{6}$ (when the verifier is given an honestly-generated proof), that every $x \notin L$ is accepted with probability at most $\frac{1}{6}$ (for any purported proof given to the verifier), and that the computational distance between $\text{Real}_{V,P}(x, w)$, $\text{Simulated}_{V^*,\pi^*}(x)$ is at most $\frac{1}{6}$. (Notice that we only assume that the system has honest-verifier ZK, which follows from the ZK guarantee of the system because $q \leq q^*$.)

We show that the following probabilistic algorithm decides $L$:

- On input $x$, run $\text{Sim}$ on $x$ to obtain a simulated proof $\pi^*$.
- Run $V$ on input $x$, with oracle access to $\pi^*$.
- If $V$ accepts $x$ in this run, output $x \in L$, otherwise output $x \notin L$.

We first show that if $x \in L$ then the algorithm outputs $x \in L$ with probability at least $\frac{2}{3}$. Otherwise, the output of $V$ in the simulated view $\text{Simulated}_{V^*,\pi^*}(x)$ is accept with probability less than $\frac{2}{3}$. Note that the output of $V$ in $\text{Real}_{V,P}(x, w)$ (i.e., his output on $x$ given oracle access to an honestly-generated proof) is accept with probability at least $\frac{5}{6}$ by the completeness property. Therefore, the computational (and consequently, also statistical) distance between $\text{Simulated}_{V^*,\pi^*}(x)$, $\text{Real}_{V,P}(x, w)$ is more than $\frac{1}{6}$ (the corresponding distinguisher outputs 1 if and only if $V$ accepts), thus contradicting the $(\frac{1}{6}, q^*)$-oblivious ZK property.

Next, we show that if $x \notin L$ then the algorithm outputs “$x \in L$” with probability at most $\frac{1}{3}$. Otherwise, using an averaging argument there exists a “proof” $\pi^*$ for $x$, on which $V$ accepts $x$ with probability more than $\frac{1}{3}$. But this contradicts the soundness of the system.

Combining Theorem 4.11 with Remark 4.10, we now show that if there exists a SAT-respecting LRCC withstanding leakage from a leakage class $\mathcal{L}$, and in addition, there exists a PCP system $(P, V)$ for 3SAT such that the answers to the proof queries of $V$ constitute the output of a leakage function $\ell \in \mathcal{L}$ on the NP-witness, then $\textbf{NP} \subseteq \textbf{BPP}$. This is formalized in the next corollary.
Corollary 4.13. Let \( n \) be a length parameter, and \( \mathcal{L}_{O(1), \text{poly}(n), \oplus 1} \) be the class of functions computable by constant depth, poly-sized boolean circuits with unbounded fan-in \( \land, \lor, \neg \) gates, and a single \( \oplus \) gate of unbounded fan-in and fan-out. If for every constant \( c \) there exists a negligible function \( \epsilon : \mathbb{N} \to \mathbb{R}^+ \), and a SAT-respecting \( (\mathcal{L}_{O(1), \text{poly}(n), \oplus 1}, cn, \epsilon(n)) \)-LRCC, then \( \text{NP} \subseteq \text{BPP} \).

Proof. Notice that a 3CNF of size \( n \) can be verified by a boolean circuit \( C^{\text{SAT}} \) of size \( cn \), for some constant \( c \). We take the constant in the theorem statement to be this \( c \). Let \((\text{Comp}, E)\) denote the SAT-respecting \( (\mathcal{L}_{O(1), \text{poly}(n), \oplus 1}, cn, \epsilon(n)) \)-LRCC whose existence is guaranteed by the theorem statement, where \( \epsilon(n) = \text{negl}(n) \). Then for every circuit \( C : \{0,1\}^n \to \{0,1\} \) of size at most \( cn \), and every \( z_1, z_2 \in \{0,1\}^n \) that satisfy \( C \), the wire values \([\text{Comp}(C), \hat{z}_1], [\text{Comp}(C), \hat{z}_2] \) are \( (\mathcal{L}_{O(1), \text{poly}(n), \oplus 1}, \epsilon(n)) \)-leakage-indistinguishable, where \( \hat{z}_1 \) is a random encoding of \( z_2 \) according to \( E \). Using Lemma 3.6.13, \([\text{Comp}(C), \hat{z}_1], [\text{Comp}(C), \hat{z}_2] \) are also \( (\mathcal{L}_{O(1), \text{poly}(n), \oplus 1}, \epsilon(n) \cdot 2^{\text{polylog}(n)}) \)-leakage-indistinguishable, so \((\text{Comp}, E)\) is also an \( (\mathcal{L}_{O(1), \text{poly}(n), \oplus 1}, cn, \epsilon(n) \cdot 2^{\text{polylog}(n)}) \)-LRCC.

Let \((P^{\text{SAT}}, V^{\text{SAT}})\) denote the PCP system of [AS92, Theorem 1] (used to prove that \( 3\text{SAT} \in \text{PCP}[\log n, \log^2 n, 1/2, \text{poly}(n)] \)), amplified to have soundness error \( \epsilon_S < \frac{1}{4} \). We apply Construction 4.6 to \((\text{Comp}, E)\) and \((P^{\text{SAT}}, V^{\text{SAT}})\), and show that the resultant PCP system for \( 3\text{SAT} \) has oblivious honest-verifier ZK, and the view of the verifier, given oracle access to a simulated proof, is \( \epsilon(n) \cdot 2^{\text{polylog}(n)} \) statistically-close to his view when given oracle access to an honestly-generated proof. To that effect, we repeat the proof of Proposition 4.3.1, using the fact that \( |C^{\text{SAT}}| = cn \). The only difference from the proof of Proposition 4.3.1, is that we now need to prove ZK.

We define a simulator \( \text{Sim} \) as follows: on input \( x \), \( \text{Sim} \) runs the LRCC simulator \( \text{Sim}^{\text{in}} \) (whose existence is guaranteed by the leakage-resilience of the LRCC) to generate an assignment \( W^{S} \) to \( \varphi_x \) (which is also a wire assignment to the circuit \( C_x \), which is the output of \text{Comp} on the circuit “\( C^{\text{SAT}} \) with \( x \) hard-wired into it”); runs \( P^{\text{SAT}} \) on \( \varphi_x, W^{S} \) to generate a PCP \( \pi^S \) for the claim “\( \varphi_x \in 3\text{SAT} \)” and outputs \( \pi^S \).

Notice that every subset \( I \) of at most \( \text{polylog}(n) \) proof symbols constitutes a leakage function \( \ell_I \) on the proof oracle. If for every such set \( I \), \( \ell_I \in \mathcal{L}_{O(1), \text{polylog}(n)} \), and the honest verifier makes at most \( \text{polylog}(n) \) oracle queries, then the leakage-resilience of the LRCC guarantees that \( \text{Simulated}^{V, \pi^S}(x) \) and \( \text{Real}^{V, P}(x, w) \) are statistically close (where \( P, V \) are the prover and verifier of Construction 4.6). For our choice of the underlying PCP system \((P^{\text{SAT}}, V^{\text{SAT}})\), this condition on the subsets \( I \) of proof symbols is satisfied (due to the prover algorithm, and since the honest verifier makes only \( \text{polylog}(n) \) oracle queries, tosses only \( r = \text{polylog}(n) \) coins, and the proof has size \( \text{poly}(n) \)), and so the resultant system is an oblivious-ZKPCP system for \( 3\text{SAT} \) with soundness error \( \frac{1}{6} \). Since \( \epsilon = \text{negl}(n) \) then for a large enough \( n \), \( \epsilon(n) \cdot 2^{\text{polylog}(n)} \leq \frac{1}{6} \), so Theorem 4.11 implies that \( 3\text{SAT} \in \text{BPP} \), so \( \text{NP} \subseteq \text{BPP} \).
Chapter 5

Probabilistically Checkable Proofs of Proximity with Zero Knowledge

In this chapter we put forward and study the notion of a zero-knowledge PCPP (or ZKPCPP for short), which extends the previous notion of ZKPCP and makes it more useful.

5.1 Introduction

A ZKPCPP is a PCPP with a probabilistic choice of proof, and the additional guarantee that the view of any (possibly-malicious) verifier, making at most $q^*$ queries to its input and proof oracles, can be efficiently simulated by making the same number of queries to the input alone. ZKPCPPs are a natural extension of ZKPCPs (indeed, the existence of a ZKPCPP system implies the existence of a ZKPCP system with related parameters), and interesting objects on their own. As we explain next, they are also motivated by cryptographic applications that involve sublinear-communication zero-knowledge arguments on distributed or committed data.

To give the flavor of these applications, suppose that a database owner (prover) commits to a large sensitive database $D$ by robustly secret-sharing it among a large number of potentially unreliable servers. At a later point in time, a user (verifier) may want to learn the answer to a query $q(D)$ on the committed database. ZKPCPPs provide the natural tool for efficiently verifying that the answer $a$ provided by the prover is indeed consistent with the committed database, namely that $a = q(D)$, without revealing any additional information about the database to the verifier and a colluding set of servers. Concretely, the prover distributes between the servers a ZKPCPP asserting that the shares of $D$ (the input) are indeed valid shares of some database $D'$ such that $q(D') = a$. The verifier, by probing only few entries in the input and the proof string, is convinced that the shares held by the servers are indeed close to being consistent with valid shares.
of some database $D'$ such that $q(D') = a$. If not “too many” servers are corrupted, the robustness of the underlying secret-sharing scheme guarantees that $D' = D$. (This description is an over-simplification, see Chapter 7 for a detailed description of the problems that arise in this context.) The above approach can also be used for verifiable updates of a secret distributed database, where a ZKPPCPP is used to convince a verifier that the shares of the updated version of the database are consistent with the original shares with respect to the update relation.

A similar idea can be used to get a sublinear-communication implementation of a “Commit-and-Prove” functionality in the two-party setting. Here, the prover first succinctly commits, using a Merkle Hash Tree [Mer87], to the shares of $D$. To later prove that $q(D) = a$, the prover again uses a Merkle Hash Tree to succinctly commit to a ZKPCPP asserting that the values it committed to in the previous phase are valid shares of some database $D'$ such that $q(D') = a$. This gives the first sublinear-communication implementations of Commit-and-Prove which only make a black-box use of a collision-resistant hash function. Our applications of ZKPCPPs are described in more details in Chapter 7.

5.1.1 Our Results and Techniques

We introduce the notion of ZKPCPPs and construct query-efficient ZKPCPPs for NP. More specifically, our ZKPCPP system has the following properties:

- The prover, on input $x \in L$, a corresponding witness $w$, and a zero-knowledge parameter $q^*$, efficiently generates a proof string $\pi$ of length $\text{poly}(|x|, q^*)$ which is statistical zero-knowledge against (possibly malicious) verifiers that make at most $q^*$ queries to $(x, \pi)$.

- By making only $\text{polylog}(|x|, q^*)$ queries to $x, \pi$, the honest verifier can verify (except with negligible soundness error) that $x$ is at most $\delta$-far from $L$, where $\delta$ can be any positive constant (or even inverse polylogarithmic).

This construction is obtained by combining an (efficient) PCPP system without zero-knowledge, with a protocol for secure multiparty computation (MPC), inheriting the efficiency from the PCPP component, and the zero-knowledge from the MPC component. The transformation has two parts, as we now explain.

The first ingredient is a general transformation from a PCPP system $(P, V)$, and a secure MPC protocol, to a PCPP system $(P_H, V_H)$ with zero-knowledge against honest verifiers (HVZKPCPP, for honest-verifier ZKPCPP). The high level idea is to extend the proof composition technique of [KPT97] (used to transform PCPs into zero-knowledge PCPs) to PCPPs, and present a new (more modular, and arguably simpler) method of implementing the “inner” system. Intuitively, instead of verifying that the honest verifier $V$ would accept $x \in L$ given an honestly-generated proof $\pi$ for $x$, the HVZKPCPP verifier $V_H$ probabilistically verifies that a random test of $V$ would have passed. This
“inner” probabilistic verification requires a proof oracle of its own, so in addition to $\pi$, the HVZKPCPP includes, for every possible verification predicate $T$ of $V$, a PCPP $\pi_T$ for the claim "$V$ would have accepted $x$ with proof $\pi$, when performing the test $T$". This “inner” verification procedure is performed with input $(x, \pi)$, and proof $\pi_T$, so if $\pi_T$ is a ZKPCPP then the zero-knowledge property guarantees that a query-bounded verifier learns only few bits of $(x, \pi)$. However, $\pi$ itself is not zero-knowledge, and may therefore reveal global information about $x$, which cannot be simulated given only few bits of $x$. To overcome this, $\pi$ is first replaced with a proof $\tilde{\pi}$ in which every bit of $\pi$ is additively secret-shared, and the “inner” proofs are for the claim "$V$ would have accepted $x$ with the proof that is reconstructed from $\tilde{\pi}$, when performing the test $T$".

This approach raises the following additional issue. Think of $(x, \pi)$ as a global satisfying assignment to all the verification predicates $T$ of $V$. Then the soundness of $(P, V)$ guarantees that if $x$ is $\delta$-far from $L$, then $(x, \tilde{\pi})$ satisfies only “few” of the verification predicates of $V$, while making no guarantee regarding how many verification predicates one could satisfy by slightly “tweaking” $\tilde{\pi}$. Therefore, the soundness of the “inner” system, which holds only for $(x, \tilde{\pi})$ that are far from satisfying the corresponding predicate, does not suffice in this context. Indeed, if $(x, \tilde{\pi})$ is close to satisfying many verification predicates, then the soundness guarantee of the “inner” system in effect provides no soundness in the composed system. Consequently, we use “inner” ZKPCPPs with a stronger soundness guarantee that every non-satisfying $(x, \tilde{\pi})$ is rejected, except with some error probability (we call such systems exact PCPPs, see Section 5.2). In particular, the task of constructing the “inner” ZKPCPP now seems at least as hard as our original task of constructing ZKPCPPs for NP. We overcome this obstacle by restricting the claims verified by the “inner” ZKPCPP to statements in $P$, of size $O(\sigma)$ (where $\sigma$ is a security parameter). Thus, we can instantiate the “inner” system with an inefficient construction, and obtain an efficient HVZKPCPP through the transformation, as long as the (non-ZK) PCPP is efficient. (We note that the system obtained in this manner only guarantees zero-knowledge against the honest verifier, because a malicious verifier may query “too many” secret shares of $\tilde{\pi}$, thus breaking the secrecy of the secret-sharing scheme, and possibly learning through $\pi$ global information regarding the input $x$.)

We construct the “inner” ZKPCPP system $(P_{ZK}, V_{ZK})$ (with weak parameters) through the so-called “MPC in the head” technique, employing the methodology used by [IKOS07] to construct zero-knowledge interactive proofs from MPC protocols. Specifically, the ZKPCPP for an NP-language $L_{R}$ with corresponding NP-relation $R$ uses an MPC protocol $\Pi_{R}$ for the function $f(w, x_1, \ldots, x_n) = R(x_1 \ldots x_n, w)$. That is, $\Pi_{R}$ allows $n + 1$ mutually-distrusting parties $P_0, P_1, \ldots, P_n$ to securely compute $f$ while guaranteeing the privacy of their inputs $w, x_1, \ldots, x_n$, and the correctness of the outputs. At a high level, the system operates as follows. The prover $P_{ZK}$, on input $(x, w)$, “runs in its head” the protocol $\Pi_{R}$, with inputs $w, x_1, \ldots, x_n$, and outputs the proof consisting of the views of all parties except $P_0$, and the description of all messages
sent over the communication channels between every pair of parties. (Thus, the proof is generated over a large alphabet, containing a symbol for every possible value of a view, or communication channel.) To verify the proof, the verifier picks a random player \(i \in_R [n]\), and a random communication channel \(i \neq j \in_R \{0, 1, \ldots, n\}\), and verifies that: (1) the messages reported in the communication channel are consistent with the messages reported in the view, and with \(\Pi_R\); (2) the output of \(P_i\) (as reported in its view) is 1; and (3) the input of \(P_i\) (as reported in its view) is consistent with \(x\). Roughly speaking, the zero-knowledge property follows from the privacy of \(\Pi_R\), and the soundness follows from its security against malicious parties. (In fact, we can make due with a weaker property of \(\Pi_R\), as described in Section 5.3.)

Finally, we note that this transformation can also be applied to PCPs, yielding an HVZKPCP comparable to that of [DFK+92] that is conceptually simpler. Thus, our construction also simplifies the ZKPCP of Kilian et al. [KPT97] which uses the HVZKPCP of [DFK+92] as a building block.

The second ingredient strengthens the zero-knowledge property to hold against arbitrary (query-bounded) malicious verifiers, by forcing the queries of any such verifier to be distributed (almost) as the queries of the honest verifier of the HVZKPCPP system, thus reducing the task of designing ZKPCPPs to the (much simpler) task of designing HVZKPCPPs. This part follows the approach of [KPT97] of using a locking scheme, generalizing their construction (which was designed for PCPs) to the PCPP setting. Intuitively, a locking scheme allows one to “lock” a secret such that without the key, every “small” subset of lock bits information-theoretically hide the secret; but given the key the secret can be easily retrieved. Concretely, we use the combinatorial construction of locking schemes from [IMS12], except that to achieve negligible soundness error and negligible simulation error simultaneously we need to apply a natural amplification technique for reducing the error of the previous construction (this is described in Appendix B).

The (very) high-level idea of this part is to lock separately every bit of an HVZKPCPP \(\pi_H\), and to provide, for every verification predicate \(T\) of the HVZKPCPP verifier \(V_H\), the set of keys needed to unlock the bits of \(\pi_H\) that \(T\) depends on. Thus, the ZKPCPP \(\pi\) consists of the locks for the bits of \(\pi_H\), and for every predicate \(T\), the set of corresponding keys. This is in fact a gross over-simplification of the actual construction, since a malicious verifier can directly query the proof bits containing the keys for the locations it wishes to read from \(\pi_H\). To prevent such strategies, the verification predicates are permuted using a random permutation \(\text{perm}\), where the location corresponding to \(T\) in \(\pi\) contains the keys for the bits that \(\text{perm}(T)\) depends on. (We note that this “fix” gives rise to a possible cheating strategy of the prover, which is overcome by also locking the permutation as part of the proof. See Section 5.4.3 for additional details.) The transformation of this part is almost identical to the one given by [KPT97], with the natural generalization to the PCPP regime. Our main contribution is in the analysis, which in the context of PCPPs becomes more involved.
5.1.2 Chapter Organization

The necessary preliminaries are given in Section 5.2. In Section 5.3 we construct exact ZKPCPPs (with weak parameters) from MPC protocols, and in Section 5.4 we construct efficient ZKPCPP systems for NP. The amplification of locking schemes appears in Appendix B.

5.2 Preliminaries

In this chapter we use as building blocks PCPPs with strong soundness, and honest-verifier zero-knowledge guarantees, as described next.

Recall that the soundness property of a PCPP system holds in relation to some proximity parameter $\delta$. The special case in which $\delta = 0$ (i.e., soundness holds for every $x \notin L_R$) will be used in Section 5.3.

**Definition 5.2.1** (Exact PCPPs). An exact PCPP system is a PCPP system in which soundness holds with proximity parameter $\delta = 0$. That is, every $x \notin L_R$ is rejected (except with probability $\epsilon$).

As noted in Section 5.1.1, our ZKPCPP system is constructed from a PCPP system with the weaker guarantee of honest-verifier zero-knowledge via an amplification of the zero-knowledge property. However, this amplification will in fact require a somewhat stronger zero-knowledge guarantee, which we define next.

**Notation 5.1.** Let $k' \in \mathbb{N}$, and $(P,V)$ be a proof system for some relation $R$. We use $V^{k'}$ to denote the verifier that performs $k'$ random and independent invocations of the honest verifier $V$.

**Definition 5.2.2** (Strong HVZK). Let $k \in \mathbb{N}$, and $\epsilon$ be a soundness parameter $\epsilon$. We say that a proof system $(P,V)$ for some relation $R$ has $(\epsilon,k)$-strong honest-verifier zero-knowledge (strong HVZK) if there exists a straight-line simulator $\text{Sim}$ such that the following holds for every $k' \leq k$, and every $(x,w) \in R$. $\text{Sim}$ interacts with $V^{k'}$ (as defined in Notation 5.1), without rewinding the verifier. During the simulation, $\text{Sim}$ makes only $k'$ TTP queries, and generates a view which is statistically close, up to distance $\epsilon$, to the real-world view of $V^{k'}$, when it has oracle access to $x$, and a random honestly-generated proof for $x$.

5.2.1 Tools

Our constructions employ two main tools. The first is secure MPC protocols, used in Section 5.3 to construct ZKPCPPs from MPC protocols. The second is locking scheme, used in Section 5.4.3 to construct ZKPCPPs from HVZKPCPPs.
Secure MPC

We follow the terminology and notation of [IKOS07]. Let $P_1, \ldots, P_m$ be $m$ parties, where every party $P_i$ holds a private input $x_i$ (we allow $x_i$ to be empty, denoted by $x_i = \lambda$). We consider protocols for securely realizing an $m$-party boolean functionality $g$ mapping the tuple of inputs $(x_1, \ldots, x_m)$ to an output in $\{0, 1\}$ (all parties are expected to have the same output). The view of a party $P_i$, denoted $V_i$, includes his private input $x_i$, and a random input $r_i$, together with all the messages that $P_i$ received during the protocol execution. (The messages $P_i$ sends during the execution, as well as his local output, are determined by this view.)

**Definition 5.2.3.** Let $\Pi$ be an $m$-party protocol, $P_i, P_j$ be a pair of parties, with inputs $x_i, x_j$. We say that the pair $V_i, V_j$ of views are consistent with respect to $x_i, x_j$ and $\Pi$, if the outgoing messages (from $P_i$ to $P_j$) implicit in $V_i$ in an execution of $\Pi$ on inputs $x_i, x_j$, are identical to the incoming messages (from $P_i$ to $P_j$) reported in $V_j$, and vice versa. Consistency between a view and one of its incident communication channels is defined similarly.

We consider the execution of the protocol in the presence of an adversary $A$ who may corrupt up to $\tau$ parties. A passive (or semi-honest) adversary learns the entire view of corrupted parties, but does not modify their behavior, whereas an active (or malicious) adversary can arbitrarily modify the behavior of corrupted parties. A static adversary is restricted to picking the set of corrupted parties in advance, whereas an adaptive adversary may pick them one by one, choosing the next party to corrupt based on its view so far.

We require that the protocol have the following properties. First, it should be correct, namely correctly compute the functionality when all parties are honest. Formally:

**Definition 5.2.4 (Correctness).** A protocol $\Pi$ realizes a deterministic $m$-party functionality $g(x_1, \ldots, x_m)$ with perfect correctness if for all inputs $x_1, \ldots, x_m$, if no party is corrupted then all parties output $g(x_1, \ldots, x_m)$ (with probability 1).

Second, we require privacy, in the sense that an adversary corrupting a subset of at most $\tau$ parties learns nothing about the private inputs of the honest parties (except for what can be deduced from the inputs, and outputs, of corrupted parties).

**Definition 5.2.5 ($\tau$-Privacy).** Let $1 \leq \tau < m$ be a corruption threshold. A protocol $\Pi$ realizes $g$ with perfect $\tau$-privacy if for every passive adversary $A$ corrupting a subset $T \subseteq [m], |T| \leq \tau$ of parties, there exists a simulator $\text{Sim}$ such that for every tuple of inputs $(x_1, \ldots, x_m)$, the view $\text{View}_A(x_1, \ldots, x_m)$ is distributed identically to $\text{Sim}\left(\{x_i\}_{i \in T}, g_T(x_1, \ldots, x_m)\right)$.

A variant of privacy that applies to adaptive adversaries can be defined similarly. (In the adaptive case, we require the existence of a black-box, straight-line, PPT simulator.)
Thirdly, we require a weak form of security against active adversaries, which is implied by the standard (simulation-based) notion of security against active adversaries.

**Definition 5.2.6** (T-Robustness). Let $T \subseteq [m]$. We say that $\Pi$ realizes $g$ with perfect $T$-robustness if for every active adversary $A$ corrupting the parties in $T$, and for every tuple $x_T$ of inputs of uncorrupted parties, the following holds. If $g$ evaluates to 0 on all choices of inputs $x$ consistent with $x_T$, then all uncorrupted parties are guaranteed to output 0.\(^1\)

**Locking Schemes**

One of the main tools used in this chapter is locking schemes [KPT97, IMS12] which are, informally, an information-theoretically secure commitment primitive. The information-theoretic nature of a locking scheme means that the lock must be associated with a key, that allows one to honestly retrieve the locked secret while guaranteeing that even an unbounded adversary cannot retrieve the secret without the key. Thus, a locking scheme $(S, R)$ for message space $\mathcal{W}$ consists of a sender $S$, who generates the lock and key, and a receiver $R$, that is given oracle access to the lock, but should be unable to retrieve the secret without the key. The sender and receiver interact in two phases: during the **Commitment phase** $S$ sends a locking oracle $L_w$ to $R$, thus committing to some $w \in \mathcal{W}$; and in the **Decommitment phase**, $S$ reveals $w$ by sending $R$ a key $K_w$ that “opens” the lock.

**Definition 5.2.7** (Locking scheme (LS)). Let $\sigma$ be a security parameter. We say that a pair of probabilistic polynomial time algorithms $(S, R)$ is a **locking scheme (LS)** for the message space $\mathcal{W}$ with $(1 - \delta)$-binding, $(u, \epsilon)$-hiding and $(u, \epsilon)$-equivocation, if the following conditions hold for every $w \in \mathcal{W}$ of length $n$:

- **Syntax**: The interaction between the sender $S$ and the receiver $R$ is carried out in two phases. In the commitment phase, $S$ is given $1^\sigma, w$ as input, and generates a pair $(L_w, K_w)$, where $L_w$ is a locking oracle, and $K_w$ is a key. $R$ is given $1^\sigma, 1^n$ as input, and is also given oracle access to $L_w$. In the decommitment phase, $S$ sends the key $K_w$ to $R$, and $R$ uses $K_w$, and the oracle $L_w$, to determine his output, which is either some value $w' \in \mathcal{W}$ of length $n$, or $\perp$ (signaling that the decommitment failed).

- **Completeness**: If all parties are honest, then for every $(L_w, K_w) \in S(1^\sigma, w)$,

$$\Pr[R^{L_w}(1^\sigma, 1^n, K_w) = w] = 1.$$  

\(^1\)Notice that we only define robustness for the case that $g$ evaluates to 0, which suffices for our purposes since we only consider functions $g$ representing relations $R$. More specifically, robustness is used to construct sound proofs systems, where the (possibly) corrupted party is the party holding the witness (and the bits of the input are partitioned between the honest parties). As soundness concerns the case that $x \notin L_R$, i.e., $(x, w^*) \notin R$ for every “witness” $w^*$, then $g$ evaluates to 0 on every input of the party holding the witness.
• \((1-\delta)\)-binding: For every fixed oracle \(L\), there exists a \(w \in \mathcal{W}\) of size \(n\) such that for every key \(K\), \(\Pr [R^L (1^\sigma, 1^n, K) \notin \{w, \bot\}] < \delta\) (where the probability is over the randomness of \(R\)).

• \((u, \epsilon)\)-hiding: For every (possibly malicious, possibly unbounded) receiver \(R^*\) that makes at most \(u\) oracle queries, every pair of messages \(w_1, w_2 \in \mathcal{W}\) of length \(n\), and every pair of oracles \(L_{w_i} \in R(S(1^\sigma, 1^n, w_i)), i = 1, 2, SD(\text{View}_{R^{L_{w_1}}} (1^\sigma, 1^n), \text{View}_{R^{L_{w_2}}} (1^\sigma, 1^n)) \leq \epsilon\), where \(\text{View}_{R^{L_{w_i}}} (1^\sigma, 1^n)\) is defined similarly to the view of \(V^*\) in the real-life model (Section 2.2.3).

• \((u, \epsilon)\)-equivocal: For every (possibly malicious, possibly unbounded) receiver \(R^*\), there exists a simulator \(\text{Sim}\) that, given \(1^\sigma, 1^n\) as input, simulates the view of \(R^*\) in a real-world interaction with a locking oracle, where the interaction in the ideal process consists of the two following phases.

  - Phase (1). \(\text{Sim}\) is requested to answer up to \(u\) queries of \(R^*\).
    At an arbitrary point, \(\text{Sim}\) is given an arbitrary message \(w \in \mathcal{W}\) of length \(n\), and generates a simulated key \(K\). At this point, phase (1) ends and phase (2) begins.

  - Phase (2). \(\text{Sim}\) continues to answer the queries of \(R^*\) (up to a total of \(u\) queries).

\(\text{Sim} (1^\sigma, 1^n)\) consists of the queries of \(R^*\), and the simulated answers (from both phases of the interaction, i.e., \(\text{Sim}\) cannot change the simulated answers after receiving the message \(w\)). We say that the system is \((u, \epsilon)\)-equivocal, if for every message \(w \in \mathcal{W}\), \(SD((\text{Sim} (1^\sigma, 1^n), K), (\text{View}_{R^{L_{w_i}}} (1^\sigma, 1^n), K_{w_i}, K_w)) \leq \epsilon\), where \((L_{w_i}, K_{w_i}) \in R(S(1^\sigma, w))\). (The view of \(R^*\) in this case consists solely of his queries to, and the answers of, the locking oracle \(L_{w_i}\)).

### 5.3 Exact ZKPCPPs from MPC Protocols: Feasibility

In this section, we show a general connection between secure MPC protocols and exact ZKPCPPs, namely ZKPCPP systems with proximity parameter \(\delta = 0\), using “MPC-in-the-head” techniques a-la [IKOS07] to transform an MPC protocol for a specific function into an exact-ZKPCPP.

More specifically, we associate with an NP-relation \(\mathcal{R}\) a characteristic function as follows.

**Definition 5.3.1** (Characteristic function \(g_{\mathcal{R}_n}\) for relation \(\mathcal{R}\)). Let \(\mathcal{R}\) be an NP-relation. The characteristic function \(g_{\mathcal{R}_n} : \{0, 1\}^* \times \{0, 1\}^n \rightarrow \{0, 1\}\) of \(\mathcal{R}_n = \{(x, w) \in \mathcal{R} : |x| = n\}\) (or simply \(g\), when \(\mathcal{R}, n\) are clear from the context) is defined as follows. \(g_{\mathcal{R}_n}(w, x_1, \ldots, x_n) = 1\) if and only if \(((x_1, \ldots, x_n), w) \in \mathcal{R}_n\).
We use techniques similar to those used by Ishai et al. [IKOS07] to construct zero-knowledge proofs from MPC protocols. Concretely, we transform a protocol $\Pi$ that securely realizes $g_{\mathcal{R}_n}$ into an exact-ZKPCPP system for $\mathcal{R}_n$, with perfect zero-knowledge against malicious (query-bounded) verifiers. For any $\tau = \tau (n)$, if the underlying $m$-party protocol is $\tau$-private (for some $m = m (n, \tau)$), then the system has perfect zero-knowledge against $\tau$-bounded verifiers.

In Section 5.3.1, we use MPC protocols to construct exact ZKPCPPs over large alphabets, and in Section 5.3.2 we reduce the alphabet size, thus obtaining exact ZKPCPPs over the binary alphabet.

### 5.3.1 Exact ZKPCPP Over Large Alphabets

In this section we use MPC protocols to construct exact ZKPCPPs over large alphabets where, Roughly speaking, each symbol in the alphabet represents a possibly view of a party in the MPC protocol. Specifically, we obtain the following result.

**Theorem 5.2 (Exact ZKPCPPs for NP).** Let $\mathcal{R} = \mathcal{R} (x, w)$ be an NP-relation, and $t (n)$ be a zero-knowledge parameter. Then $\mathcal{R} \in \text{exact-} \text{ZKPCPP}_{\Sigma} [r, q, \epsilon_{ZK}, \delta, \epsilon_{S}, \ell]$ with a non-adaptive honest verifier, and $t (|x|)$-ZK with a straight-line, black-box simulator, where $\Sigma = 2^{\text{poly}(t, |x|)}$, $r = O (\log t + \log |x|)$, $q = 2$, $\epsilon_{ZK} = 0$, $\epsilon_{S} = 1 - \frac{1}{\text{poly}(t, |x|)}$, $\ell = \text{poly} (t, |x|)$.

The high level idea of the construction is as follows. The prover runs in its head the execution of a protocol $\Pi$ for $g_{\mathcal{R}_n}$, and generates the proof by concatenating the views of all parties (except the one holding the witness), and the messages sent over all point-to-point communication channels between every pair of parties. The verifier either checks that the views correspond to an honest execution of $\Pi$ (by checking the consistency of a view, and an adjacent communication channel), or that $\Pi$ was executed on input $x$ (by comparing the input used by a single party with the corresponding symbol of $x$). This is formalized in the following construction.

**Construction 5.3.** The system is parameterized by an input length parameter $n \in \mathbb{N}$, a zero-knowledge parameter $t = t (n)$, and employs a protocol $\Pi$ with parties $P_0, P_1, \ldots, P_n$, that realizes $g_{\mathcal{R}_n}$ with perfect adaptive $t$-privacy, and perfect static 1-robustness. We assume without loss of generality that $w$ is the input of $P_0$, and $x_1, \ldots, x_n$ are the inputs of $P_1, \ldots, P_n$ (respectively).

**Prover algorithm.** On input $(x, w)$, and $1^t$, the prover $P_E$ emulates “in his head” a random execution of $\Pi$ on inputs $(w, x_1, \ldots, x_m)$. Let $V_0, \ldots, V_n$ denote the views of $P_0, \ldots, P_n$ in this execution, and for every $0 \leq i < j \leq n$, let $C_{i,j}$ describe the
messages sent over the communication channel between $P_i, P_j$ during the execution. $P_E$ outputs the proof $\pi$ consisting of the concatenation of the views $V_1, \ldots, V_n$ and the communication channels $Ch_{i,j}$ for $0 \leq i < j \leq n$, where every view and communication channel constitutes a symbol of the proof. (Notice that the proof does not include the view $V_0$, since $V_0$ reveals the witness $w$.)

**Verifier algorithm.** On input $1^t, n$, and give oracle access to $x, \pi$, the verifier $V_E$ performs one of the following tests, each with probability half.

- Picks a random view $V_i, i \in R [n]$, and verifies that the input of $P_i$ in the protocol execution was $x_i$. (This test ensures that the protocol execution is consistent with $x$.)
- Picks $i \in R [n]$, and $j \in R \{0, 1, \ldots, n\}, i \neq j$, and verifies that $V_i$ is consistent with $Ch_{i,j}$. (This test ensures that the views correspond to an honest execution of $\Pi$.)

In both cases, $V_E$ verifies that $P_i$ outputs 1.

The next claim summarizes the properties of Construction 5.3.

**Claim 5.3.2** (From MPC to exact ZKPCPPs). Let $R = R(x, w)$ be an NP-relation. Then Construction 5.3 is an exact-PCPP for $R$, with perfect $t$-zero-knowledge, and soundness error $(1 - \Omega(\frac{1}{n^2}))$. Moreover, the prover $P_E$ generates proofs of length $n^2$, over an alphabet of size $O(2^{r+c})$, where $r, c$ denote the randomness and communication complexity of $\Pi$ (respectively); and the honest verifier $V_E$ makes only 2 (non-adaptive) oracle queries.

**Remark 5.4.** We note that if MPC protocol with which Construction 5.3 is instantiated is $t$-private with a black-box simulator, then Construction 5.3 is $t$-zero-knowledge with a straight-line, black-box simulator.

The proof follows from the one-to-one correspondence between consistent executions of $\Pi$ (see Definition 5.2.3), and tuples of views and communication channels, as specified in the next couple of Lemmas. More specifically, Lemma 5.3.3 states that an honest execution of $\Pi$ results in consistent views and communication channels. Since an honestly-generated proof for $(x, w) \in R$ corresponds to an honest execution of $\Pi$ on inputs $x, w$ (with some random inputs), Lemma 5.3.3, together with the perfect correctness of $\Pi$, guarantee that the proof consists of consistent views and communication channels, and all parties output 1, so the $V_E$ accepts $x$ with probability 1.

**Lemma 5.3.3** (Global to local consistency, [IKOS07], Lemma 2.3 (restated)). Let $\Pi$ be an $m$-party protocol, and let $x = (x_1, \ldots, x_m)$ be a tuple of inputs. Given random inputs $r_1, \ldots, r_m$, let $V_1, \ldots, V_m$ denote the $m$-tuple of views of $P_1, \ldots, P_m$ in the execution of $\Pi$ on inputs $x_1, \ldots, x_m$, and random inputs $r_1, \ldots, r_m$. Let $(Ch_{i,j})_{1 \leq i < j \leq m}$ denote the messages sent over the communication channel between $P_i, P_j$ during the
execution, for every $1 \leq i < j \leq m$. Then every view $V_i, i = 1, \ldots, m$ is consistent (as in Definition 5.2.3) with respect to $\Pi$, $x$ with all the communication channels indexed by $i$.

The soundness property of Construction 5.3 follows from the opposite direction of Lemma 5.3.3, which states that if every view and incident communication channel are locally consistent, then there exists an execution of $\Pi$ in which $P_1, \ldots, P_m$ are honest, that results in this specific tuple of views and communication channels. Thus, if no honest execution of $\Pi$ yields this specific tuple, then there is some local inconsistency, which is detected by $V_E$ with probability $O\left(\frac{1}{n^2}\right)$. (The proof of Lemma 5.3.4 is by induction on the number of rounds in $\Pi$, and is omitted.)

**Lemma 5.3.4** (Local to global consistency). Let $n \in \mathbb{N}$, $R$ be an NP-relation, and $g$ denote the characteristic function of $R_n$ of Definition 5.3.1. Let $\Pi$ be an $(n+1)$-party protocol realizing $g$. Let $V_1, \ldots, V_n$ denote an $n$-tuple of (possibly incorrect) views of $P_1, \ldots, P_n$, and let $(Ch_{i,j})_{0 \leq i < j \leq n}$ be an $(\frac{n+1}{2})$-tuple of (possibly incorrect) communication channels between the parties $P_0, \ldots, P_n$. Let $x_1, \ldots, x_m$ denotes the inputs reported in $V_1, \ldots, V_m$. If every view $V_i, i = 1, \ldots, n$ is consistent (with respect to $\Pi, (x_1, \ldots, x_m)$) with all its incident communication channels, then there exists an input $w \in \{0, 1\}^*$ for $P_0$ such that in an execution of $\Pi$ with honest $P_1, \ldots, P_n$, input $w$ for $P_0$, and inputs $x_1, \ldots, x_n$ for parties $P_1, \ldots, P_n$ (and random inputs as reported in $V_1, \ldots, V_n$), in which the views of $P_1, \ldots, P_n$ are $V_1, \ldots, V_n$ (respectively), and for every $0 \leq i < j \leq n$, $Ch_{i,j}$ describes the messages sent over the communication channel between $P_i, P_j$.

We now use the connection between honest executions of $\Pi$, and tuples of consistent views and communication channels, to prove Claim 5.3.2.

**Proof of Claim 5.3.2.** Set some $n \in \mathbb{N}$, and let $\Pi = \Pi_n$.

**Prefect completeness** follows directly from the perfect completeness of $\Pi$, and Lemma 5.3.3.

**Soundness.** Let $x \notin L_R$, and let $\pi^*$ be a purported proof for the claim “$x \in L_R$”. We distinguish between three possible cases relating to the views, and communication channels, as reported in $\pi^*$.

- **The input reported in the views is $x' \neq x$.** This is detected by $V_E$ with probability at least $\frac{1}{2n} \geq \frac{1}{n^2}$.

- **There exists a view $V_i$ which is inconsistent with an incident communication channel.** This case is detected with probability at least $\frac{1}{2n(n+1)}$.

- **All views and communication channels are consistent with $\Pi, x$.** Then Lemma 5.3.4 guarantees that there exists an execution of $\Pi$ on $x$, in which all parties (except possibly $P_0$) are honest, such that the view of $P_i$ is $V_i$, and the messages exchanged
between $P_i, P_j$ are according to $Ch_{i,j}$. Therefore, the $P_0$-robustness of $\Pi$ guarantees that the outputs in $V_1, \ldots, V_n$ are 0, so $V_E$ rejects with probability 1.

Soundness can be amplified (while preserving zero-knowledge) by repeating the verification procedure $\lceil \frac{t}{4} \rceil$ times.

**$t$-zero-knowledge.** We show a straight-line simulator $Sim$ that perfectly simulates the view of any $t$-bounded verifier $V^*$, using a simulator $Sim_{in}$ for a semi-honest adversary $\mathcal{A}_{MPC}$ operating in $\Pi$ who adaptively corrupts at most $t$ parties when the execution of $\Pi$ ends who. (The existence of $Sim_{in}$ follows from the $t$-privacy of $\Pi$.) Moreover, if $Sim_{in}$ is black-box, then so is $Sim$.

Let $(x, w) \in \mathcal{R}$, and denote $n = |x|$. $Sim(1^t, n)$, chooses a random string $r$ for $V^*$, and interacts with $V^*$ to extract his queries $a_1, \ldots, a_t$. (Notice that every $a_i$ is a query to $x$, to a view, or to a communication channel, and the queries may be made adaptively.) $Sim$ answers the queries of $V^*$ with answers as follows. A query $a$ to $x_i$ is forwarded to the TTP. If $a$ is to the view $V_i$ of a party $P_i$, or to a communication channel $Ch_{i,j}$, then $Sim$ requests $x_i$ from the TTP, and provides $x_i$ to $Sim_{in}$, asking it to corrupt party $P_i$ at the end of the execution of $\Pi$. (We note that if $P_i$ has no input in $\Pi$ then $Sim$ need not make any query to the TTP.) Let $\hat{V}_i$ denote the simulated view of $P_i$, as generated by $Sim_{in}$. If $a$ is a query to $Ch_{i,j}$ for some $j \in \{0, 1, \ldots, n\}$, then $Sim$ uses it to generate the message sent over the communication channel between $P_i, P_j$, and let $\hat{Ch}_{i,j}$ denote these messages.

We claim that for every $(x, w) \in \mathcal{R}$, $\text{Ideal}_{Sim} (t, x) \equiv \text{Real}_{V^*, P} (t, x, w)$. Indeed, the number of TTP-queries that $Sim$ makes during his simulation is bounded by the query complexity of $V^*$, so the perfect $t$-privacy of $\Pi$ guarantees that for every $t' \leq t$, the views $\hat{V}_{i_1}, \ldots, \hat{V}_{i_{t'}}$ are identically distributed to the views $V_{i_1}, \ldots, V_{i_{t'}}$ in an honest, real-world execution of $\Pi$ on $(x, w)$, with uniformly random random inputs. As the view $V_i$ of every party $P_i$ completely determines the value of all incident communication channels (namely, all communication channels between $P_i$ and $P_j$, $i \neq j \in \{0, 1, \ldots, n\}$), and queries to $x$ are answered according to the actual input oracle in both worlds, then the answers of $Sim$ are identically distributed to the real-world oracle answers.

**Remark 5.5.** It follows directly from the proof of Claim 5.3.2 that Construction 5.3 is $(0, t)$-strong HVZK (see Definition 5.2.2). More specifically, for every $t' \leq t$, the oracle answers to all the queries of $V'_E$ (see Notation 5.1) can be reconstructed from the oracle answers to the his queries to views alone. As the system is zero-knowledge against arbitrary $t$-bounded verifiers, the simulator for $(P_E, V'_E)$ (whose existence follows from Claim 5.3.2) can simulate the oracle answers to queries to $t'$ views, and these can be used to reconstruct the oracle answers to the queries of $V'_E$. Thus, the entire view of $V'_E$ can be simulated with only $t'$ TTP-queries. We use this strong HVZK property in Section 5.3.2 to reduce the alphabet size.

**Remark 5.6.** An additional property of the exact ZKPCPP system of Claim 5.3.2 (and consequently also that of Theorem 5.2, which uses the same system) is that the set $Q$
of verifier queries that the simulator is asked to simulate completely determines the identity of its TTP queries during the simulation (this follows directly from the proof of Claim 5.3.2).

We consider \((P_E, V_E)\) to be a feasibility result as it is only \textit{weakly-sound}, in the sense that its soundness error is large, and its alphabet size may be exponential in \(|x|\), and the zero-knowledge parameter \(t\). (The error can be reduced by repetition, but this increases the query complexity of the honest verifier.) In any case, this inefficiency will not pose a problem in later constructions. Indeed, the construction of honest-verifier ZKPCPPs (Construction 5.10) uses exact ZKPCPPs only for constant-sized claims, and the construction of a ZKPCPP (with zero-knowledge against malicious verifiers, Construction 5.23) uses exact ZKPCPPs for claims of size \(O(\sigma)\), where \(\sigma\) denotes the security parameter of the ZKPCPP. Moreover, we will only use exact ZKPCPPs for relations in P.

We note that if one only requires \textit{honest-verifier} zero-knowledge, then Construction 5.3 can be based on a \(1\)-\textit{private} protocol \(\Pi\) (however, if we amplify soundness by repeating the verification procedure, we again need \(\Pi\) to be private against larger coalitions of parties).

By a result of \cite{BGW88}, every \(g_n\) has a protocol \(\Pi_n\) realizing it with \(m_n \geq \max\{n, 3(t + 1)\}\) parties, with communication complexity at most \(O(m_n^c \cdot |C_{g_n}|)\) (for some constant \(c > 0\), where \(C_g\) is an arithmetic circuit that computes \(g_n\)) and randomness complexity \(\text{poly}(m_n)\). Therefore, \(|\Sigma_\Pi| \leq 2^{O(\max\{O((t')^c), n^c\} \cdot |C_{g_n}|)}\). Theorem 5.2 now follows from Claim 5.3.2 by instantiating Construction 5.3 with the protocol of \cite{BGW88}.

### 5.3.2 Exact ZKPCPPs over the Boolean Alphabet

Construction 5.3 produces proofs over a large alphabet, whose symbols correspond to all possible views and communication channels generated in \(\Pi\), and so its size may be exponential in \(|x|\), and the zero-knowledge parameter \(t\). In this section we reduce the alphabet size, obtaining an exact ZKPCPP over the boolean alphabet:

**Claim 5.3.5** (Binary exact ZKPCPPs for NP). Let \(\mathcal{R} = \mathcal{R}(x, w)\) be an NP-relation. Then for every zero-knowledge parameter \(t = t(|x|), \mathcal{R} \in \text{exact-ZKPCPP}[r, 3, \varepsilon_{ZK}, \delta, \varepsilon_S, \ell], \) where \(r = O(\log t + \log|x|), \varepsilon_{ZK} = 0, \varepsilon_S = 1 - \frac{1}{\text{poly}(t, |x|)}, \ell = \text{poly}(t, |x|), \) and the exact ZKPCPP system has perfect \(t\)-zero-knowledge with a straight-line, black-box simulator. Moreover, the honest verifier is non-adaptive, and the system has \((0, t)\)-strong honest-verifier zero-knowledge with a straight-line, black-box simulator.

The natural approach towards reducing the alphabet size is the following. The prover generates proofs over \(\{0, 1\}\) in which every view and communication channel is represented using bit-strings; and to query a view, or communication channel, the
verifier queries all the bits corresponding to that symbol. However, this naive approach does not preserve zero-knowledge. Indeed, it increases the query complexity of the honest verifier, and consequently a malicious (even query bounded) verifier may query many parts of views, thus potentially breaking the privacy of the underlying protocol, and consequently the zero-knowledge of the system.

Therefore, we use a different approach, due to Dwork et al. [DFK+92] (explicitly described in [KN95]), for reducing the alphabet size. We describe the reduction here for the sake of completeness. As in the natural approach, every proof symbol is represented using a bit string, but we avoid the increase in the query complexity of the honest verifier by having it probabilistically verify that the oracles satisfy the decision predicate of $V_E$. Thus, we can preserve the query complexity of $V_E$, where the new verifier locally translates the decision predicate of $V_E$ to ternary constraints over the binary alphabet.

More specifically, let $\Sigma_\Pi$ denote the alphabet of Construction 5.3, and let $V_E (r)$ denote the run of $V_E$ with random inputs $r$. The output of $V_E (r)$ is determined by a boolean circuit $C_r$ (of size $\text{polylog} |\Sigma_\Pi|$), whose inputs are the oracle answers to the queries of $V_E$. $C_r$ can be represented as a 3CNF formula of $\psi_r = \bigwedge_{j=1}^l \psi_{r,j}$, where $l = \text{polylog} |\Sigma_\Pi|$. The input $x$ constitutes an assignment to some of the variables of $\psi_r$. We call these variables input variable, and refer to all other variables as proof variables.

Let $x_r$ denote the restriction of $x$ to the input variables of $\psi_r$. For a proof oracle $\pi$ (in its representation over $\{0,1\}$), we say that $Y_r \in \{0,1\}^*$ is “consistent with $\pi$” if $Y_r$ is an assignment to the proof variables of $\psi_r$, and $Y_r, \pi$ assign the same values to these proof variables. Let $\pi$ be an honestly-generated proof for the claim “$x \in L_R$”, and $Y_r$ be an assignment to the proof variables of $\psi_r$ which is consistent with $\pi$, then we say that $Y_r$ is an assignment-extension of $x$ to $\psi_r$. For every clause $\psi_{r,j} = (t_{j,1}y_{j,1} \lor t_{j,2}y_{j,2} \lor t_{j,3}y_{j,3})$ (where $t_{j,k}$ denotes the presence or lack of negation), we define a variable $z_{r,j} \in \{1,2,3\}$, whose value is the smallest index such that $Y_r$ satisfies $t_{j,z_{r,j}}y_{j,z_{r,j}}$. (Notice that if $x \in L_R$ and $\pi$ is an honestly-generated proof for $x$ then such an index necessarily exists.) Notice that $|Y_r| = O (l)$, and consequently $|(z_{r,1}, \ldots, z_{r,l})| = O (l)$. We are now ready to describe the exact ZKPCPP system over $\{0,1\}^m$.

Construction 5.7. The exact ZKPCPP system $(P_{\text{bin}}, V_{\text{bin}})$ uses the system $(P_E, V_E)$ of Construction 5.3 as a building block.

**Prover algorithm.** On input $1^t, x = (x_1, \ldots, x_n)$ and $w$, $P_{\text{bin}}$ generates a proof $\pi \in P_E (1^t, x, w)$, and let $\tilde{\pi}$ denote the binary representation of $\pi$. For every possible randomness $r$ of the verifier $V_E$, $P_{\text{bin}}$ generates the 3CNF $\psi_r = \bigwedge_{j=1}^l \psi_{r,j}$ corresponding to the decision circuit $C_r$ of $V_E (r)$, and determines the assignment-extension $Y_r$ that is consistent with $\pi$. For every clause $\psi_{r,j} = (t_{j,1}y_{j,1} \lor t_{j,2}y_{j,2} \lor t_{j,3}y_{j,3})$ (where $t_{j,k}$ denotes the presence or lack of negation), let $z_{r,j} \in \{1,2,3\}$ denote the smallest index such that $t_{j,z_{r,j}}y_{j,z_{r,j}}$ is satisfied under $Y_r$, and let $Z_r := (z_{r,1}, \ldots, z_{r,l})$. Then $P_{\text{bin}}$ outputs the proof $\pi_{\text{bin}} = \tilde{\pi} \circ (Z_r)_r$.

---

3As noted above, if $x \in L_R$ then such an index necessarily exists.
Verifier algorithm. On input $1^t, m$, and given oracle access to $x = (x_1, ..., x_m)$ and $\pi_{\text{bin}}$, the verifier $V_{\text{bin}}$ chooses a random string $r$ for $V_E$, a clause $\psi_{r,j}$ in $\psi_r$, and a variable $y_{j,l}$ in $\psi_{r,j}$. Then, he queries $x, \pi_{\text{bin}}$ about $y_{j,l}$ and $z_{r,j}$ (the assignment to $y_{j,l}$ appears either in $x$ or in $\pi_{\text{bin}}$). $V_{\text{bin}}$ rejects if $z_{r,j} = l$ and $t_j, y_{j,l} = \text{false}$, and otherwise accepts.

Claim 5.3.5 now follows by instantiating Construction 5.7 with the exact ZKPCPP system of Theorem 5.2.

Remark 5.8. We note that the simulator of Construction 5.7 simply emulates the simulator of the exact ZKPCPP system of Theorem 5.2, so using Remark 5.6, the system has the additional guarantee that the set $Q$ of verifier queries that the simulator is asked to simulate completely determines the identity of its TTP queries during the simulation.

5.4 From Efficient PCPPs to Efficient ZKPCPPs

In this section we show a general transformation from PCPPs to ZKPCPPs, and construct an efficient ZKPCPP system for any NP-relation $\mathcal{R}$. (Using the same methods one can transform a PCP into a ZKPCP.) First, we use proof-composition techniques to transform a PCPP into an HVZKPCPP, using an exact ZKPCPP as the inner proof system. Then, we show a transformation from an HVZKPCPP and a locking scheme, into a ZKPCPP that guarantees zero-knowledge against malicious query-bounded verifiers. Finally, by applying the first transformation to an efficient PCPP, and the second to an efficient locking scheme and to the HVZKPCPP obtained through the first transformation, we get an efficient ZKPCPP.

5.4.1 From PCPPs to HVZKPCPPs

In this section we show a general transformation from PCPPs to HVZKPCPPs. The high-level idea is to use proof composition (see, e.g., [BGH+04, DR04, Din06]), which was used in the context of PCPs to reduce the query complexity of the PCP verifier.

More specifically, a verifier that makes $q$ queries, and either accepts of rejects according to a predicate $\text{Pred}$, is replaced with an “inner” verifier that probabilistically verifies that the oracle answers would satisfy $\text{Pred}$. Thus, the query complexity is reduced since the probabilistic “internal” verification requires less than $q$ queries. Notice that the final verifier in the “composed” system emulates the “internal” verification procedure, so to get zero-knowledge it suffices that the “inner” system is zero-knowledge, even if the “outer” (original) system has no zero-knowledge guarantees. However, this intuition is not entirely correct. Indeed, the “inner” verifier queries bits of the “outer” proof which, if the “outer” proof is not zero-knowledge, may reveal non-trivial information on the input.

To overcome this problem, we first secret-share the “outer” proof (see Section 5.4.1 below for a more detailed discussion). The advantage of using composition in this case
is similar to the advantage of employing composition towards constructing standard PCPs: as long as the “outer” system is efficient, so is the composed system, even if the “inner” system is not. We construct the final system in two steps: we first construct a weakly-sound HVZKPCPP; and then improve the soundness error to obtain the final system. More specifically, we prove the following:

Corollary 5.9 (HVZKPCPPs for NP). Let $\epsilon$ be a soundness parameter, and $\delta$ be a proximity parameter. Then every relation $R = R(x,w) \in \text{DTIME}(T(n))$ has an HVZKPCPP system $(P_H,V_H)$ with perfect completeness, perfect honest-verifier zero-knowledge with a straight-line, black-box simulator, and soundness error $\epsilon$ with proximity parameter $\delta$. On input $x$, $P_H$ generates a proof of size $\text{poly}(T(|x|),\log \frac{1}{\epsilon}, \frac{1}{\delta})$ and $V_H$ makes $O\left(\frac{1}{\delta} \log \frac{1}{\epsilon}\right)$ queries.

A weakly-sound HVZKPCPP

we use a standard, non-zero-knowledge PCPP system to construct a weakly-sound ZKPCPP system. By “weakly-sound” we mean that the soundness error of the ZKPCPP depends on the zero-knowledge parameter $q^*$, and the input length $n$, and tends to 1 as $q^*,n$ tend to infinity.

Let $R$ be a relation, and let $(P_{\text{out}}, V_{\text{out}})$ be a PCPP for $R$ with soundness error $\epsilon$ and proximity parameter $\delta$, where $V_{\text{out}}$ makes $q$ oracle queries and uses $r$ random bits. Then every random string $\text{rand}$ of $V_{\text{out}}$ corresponds to a set of $q$ queries, and a predicate $\varphi_{\text{rand}} : \{0,1\}^q \to \{0,1\}$ describing the decision of $V_{\text{out}}$. Let $(\varphi_1, \ldots, \varphi_{2^r})$ denote the vector of all $2^r$ predicates corresponding to all random strings of $V_{\text{out}}$. Then: if $x \in L_R$ and $\pi$ was honestly-generated by $P_{\text{out}}$ for $x$, then $(x, \pi)$ satisfies $\varphi_i$ for every $1 \leq i \leq 2^r$; and if $x$ is $\delta$-far from $L_R$ then for any “proof” $\pi^*$, $(x, \pi^*)$ satisfies at most an $\epsilon$-fraction of $\varphi_1, \ldots, \varphi_{2^r}$.

In standard proof-composition constructions, the prover concatenates $\pi$ with proofs $\pi_1^{\text{in}}, \ldots, \pi_{2^r}^{\text{in}}$, where $\pi_i^{\text{in}}$ should convince the inner verifier that $(x, \pi)$ satisfies $\varphi_i$. The verifier (of the composed system) then runs the outer verifier to generate $\text{rand}$ and $\varphi_{\text{rand}}$, and the inner verifier to check that $(x, \pi)$ satisfies $\varphi_{\text{rand}}$. However, since the inner verification procedure may query $\pi$, which may reveal non-trivial information about $x$, then the composed system may not be zero-knowledge even if the inner system is zero-knowledge. Therefore, we combine proof composition with secret-sharing that “hides” $\pi$. Concretely, we replace every proof bit $\pi_i$ (i.e., every predicate variable corresponding to a proof bit) with a set of bits $\{\pi_{ij}\}$ (i.e., with a set of new predicate variables) such that $\pi_i$ is reconstructible given all the new bits $\pi_{ij}$, but any subset of the new bits $\{\pi_{ij}\}$ reveals no information about $\pi_i$. We refer to a predicate obtained thorough this “secret-sharing” transformation as a private predicate, since a partial assignment to few predicate variables reveals no information about the proof $\pi$. Formally,

Definition 5.4.1 (Private predicates). Let $k \in \mathbb{N}$, and $\varphi : \{0,1\}^q \to \{0,1\}$ be a decision predicate over variables $v_1, \ldots, v_q$. We partition the variables of $\varphi$ to a set $V_{\text{inp}}$ of
input variables (i.e., variables corresponding to bits of the input oracle) and a set $V_{pf}$ of proof variables. The $k$-private form of $\varphi$, denoted $\varphi(k)$, is obtained from $\varphi$ by replacing every proof variable $v_i \in V_{pf}$ with the XOR of $k+1$ new variables $y_{i,1}, \ldots, y_{i,k+1}$, namely every appearance of $v_i$ is replaced with $\bigoplus_{j=1}^{k+1} y_{i,j}$. The private-predicates relation $R_{\text{priv}}$ consists of all pairs of private predicates, and satisfying assignments for them, i.e.,

$$R_{\text{priv}} = \{((w, \varphi(k)), \lambda) : w \text{ satisfies } \varphi(k)\}.$$ 

We now describe the transformation from PCPPs to weakly-sound HVZKPCPPs.

Construction 5.10. The basic HVZKPCPP system, denoted $(P_B, V_B)$, is obtained through the composition of an “outer” PCPP system $(P_{out}, V_{out})$ for $R$, with the “inner” exact ZKPCPP system $(P_{in}, V_{in})$ for the relation $R_{\text{priv}}$. The system is parameterized by $t \in \mathbb{N}$ which determines the zero-knowledge requirement from the inner system. (Without loss of generality, $t \geq 3$)

**Prover algorithm.** On input $1^t, x, w$ such that $(x, w) \in R$, $P_B$:

- Generates the verification predicates $\varphi_1, \ldots, \varphi_m$ of $V_{out}$, where $m := 2^r$, and $r$ denotes the length of the randomness of $V_{out}$; and a proof $\pi \in P_{out}(x, w)$.

- Generates the $t$-private form $\varphi_i(t)$ of every predicate $\varphi_i$, and replaces every proof variable $v_j \in V_{pf}$ with the XOR of $t+1$ new variables $y_{j,1}, \ldots, y_{j,t+1}$, such that $\bigoplus_{k=1}^{t+1} \pi(t)_{y_{j,k}} = \pi_{v_j}$, where $\pi(t)$ denotes the proof obtained after this step. (Recall that $(x, \pi)$ is interpreted as an assignment to the predicates $\varphi_1, \ldots, \varphi_m$, so this step transforms $(x, \pi)$ into an assignment $(x, \pi(t))$ to the private predicates $\varphi_1(t) \land \cdots \land \varphi_m(t)$.)

- For every $1 \leq i \leq m$, let $(x, \pi(t))_i$ denote the restriction of $(x, \pi(t))$ to the variables of the private predicate $\varphi_i(t)$. Then $P_B$ generates a proof $\pi_{in}^i \in P_{in}(1^t, ((x, \pi(t))_i, \varphi_i(t)), \lambda)$ for the claim $(((x, \pi(t))_i, \varphi_i(t)), \lambda) \in R_{\text{priv}}$.

- Outputs the proof $\pi_B = \pi_{in}^1 \circ \cdots \circ \pi_{in}^m \circ \pi(t)$.

**Verifier algorithm.** $V_B$ on input $1^t, |x|$, and given oracle access to $x, \pi_B = \pi_{in}^1 \circ \cdots \circ \pi_{in}^m \circ \pi(t)$:

- Picks an $i \in_R [m]$, uses $V_{out}$ to generate the predicate $\varphi_i$, and transforms it into the $t$-private predicate $\varphi_i(t)$.

- Runs $V_{in}$ to check that $(x, \pi(t))_i$ satisfies $\varphi_i(t)$, using $(x, \pi(t))_i$ as the input oracle of $V_{in}$, and $\pi_{in}^i$ as its proof oracle.

The next claim summarizes the connection between the properties of $(P_B, V_B)$, and the underlying (inner and outer) systems.
Claim 5.4.2. Let \( t \in \mathbb{N} \), and \( \mathcal{R} \in \text{PCPP} [r,q,\delta,\epsilon_{\text{out}},\ell] \) with the PCPP system \((P_{\text{out}},V_{\text{out}})\). Let \((P_{\text{in}},V_{\text{in}})\) be an exact ZKPCPP system for \( \mathcal{R}_{\text{priv}} \) with perfect \( t \)-zero-knowledge, and soundness error \( \epsilon_{\text{in}}(n,t) \) for a non-decreasing function \( \epsilon_{\text{in}} \), where \( n \) denotes the input length to the exact ZKPCPP system. Assume that \( P_{\text{in}} \) generates proofs of length \( \text{poly}(n,t) \), and \( V_{\text{in}} \) makes \( q_{\text{in}} \leq t \) queries, and tosses \( r_{\text{in}} \) random coins. Then Construction 5.10, when instantiated with \((P_{\text{out}},V_{\text{out}})\), and \((P_{\text{in}},V_{\text{in}})\), is a PCPP system for \( \mathcal{R} \) with perfect completeness, perfect honest-verifier zero-knowledge, and soundness error \( \epsilon_{\text{out}} \cdot (1 - \epsilon_{\text{in}}(q,t)) + \epsilon_{\text{in}}(q,t) \). Moreover, \( V_B \) makes only \( q_{\text{in}} \) queries, tosses \( r + r_{\text{in}} \) coins, and the prover generates proofs of length \( O_q(t \ell) + 2^\ell \cdot \text{poly}_q(t) \). Furthermore, if \((P_{\text{in}},V_{\text{in}})\) has a straight-line, black-box simulator, then so does \((P_B,V_B)\), and if \( V_{\text{in}} \) is non-adaptive, then \( V_B \) is non-adaptive.

Remark 5.11. When Construction 5.10 is instantiated with Construction 5.7 as the inner system, it has the additional property that the set \( Q \) of verifier queries that the simulator is asked to simulate completely determines the identity of its TTP queries during the simulation. Indeed, the simulator of Construction 5.10 emulates the simulator \( \text{Sim}_{\text{in}} \) of Construction 5.7, and all its TTP queries are due to TTP queries of \( \text{Sim}_{\text{in}} \), so this property follows directly from Remark 5.8.

Let \( \varphi_1, \ldots, \varphi_k \) be predicates, with assignments \( x, W \) to their input and proof variables, respectively. Let \( \varphi_1(t), \ldots, \varphi_k(t) \) be the \( t \)-private forms of \( \varphi_1, \ldots, \varphi_k \), respectively, and let \( W(t) \) be an assignment to the proof variables of \( \varphi_1(t), \ldots, \varphi_k(t) \). We say that \( W(t) \) is consistent with \( W \) if \( W(t) \) is a \( t+1 \) additive secret sharing of \( W \). That is, for every proof variable \( y_j \) of a predicate \( \varphi_j, j \in [k] \), we have \( \oplus_{i=1}^{t+1} W(t)_{y_{ji}} = W_{y_j} \). We will use the following fact.

Fact 5.4.3. Let \( \varphi : \{0,1\}^t \to \{0,1\} \) be a predicate, with assignments \( x, W \) to its input and proof variables, respectively. For a parameter \( k \in \mathbb{N} \), let \( \varphi(k) \) be the \( k \)-private form of \( \varphi \). Let \( W(k) \) be an assignment to the proof variables of \( \varphi(k) \) that is consistent with \( W \). Then \((x,W)\) satisfies \( \varphi \) if and only if \((x,W(k)) \) satisfies \( \varphi(k) \).

Proof of Claim 5.4.2. Let \((P_B,V_B)\) be the ZKPCPP system of Construction 5.10.

Complexity. \( V_B \) emulates the verifier \( V_{\text{in}} \), so it makes only \( q_{\text{in}} \) queries. Moreover, \( V_B \) tosses \( r \) coins to determine the random test of \( V \), and \( r_{\text{in}} \) coins to emulate \( V_{\text{in}} \). The prover \( P_B \) uses the PCPP prover \( P_{\text{out}} \) to generate a proof \( \pi \) of length \( \ell \). Then, for each of the \( 2^\ell \) random strings of the PCPP verifier \( V_{\text{out}} \), \( P_B \) generates a verification predicate \( \varphi \) of size \( O_q(1) \), generates the \( d \)-private form \( \varphi(t) \), where \( |\varphi(t)| = O_q(t) \), and generates an exact ZKPCPP for \( \varphi(t) \), which has size \( \text{poly}(t,|\varphi(t)|) = \text{poly}_q(t) \). The proof consists of the secret shared version of \( \pi \), whose length is \( O(t \ell) \), and the exact ZKPCPPs, and therefore has size \( O_q(t \ell) + 2^\ell \cdot \text{poly}_q(t) \). Finally, since \( V_B \) emulates \( V_{\text{in}} \), then if \( V_{\text{in}} \) is non-adaptive, then so is \( V_B \).

Perfect completeness holds due to the completeness of the underlying proof systems, and Fact 5.4.3.
Soundness. Let $x$ be $\delta$-far from $L_R$, then the soundness of $(P_{out}, V_{out})$ guarantees that for every "proof" $\pi^*_\text{out}$, $(x, \pi^*_\text{out})$ satisfies at most an $\epsilon_{out}$-fraction of the predicates $\varphi_1, \ldots, \varphi_m$, so Fact 5.4.3 guarantees that for every assignment extension $W^*(t)$ to the proof variables of the $t$-private predicates $\varphi_1(t), \ldots, \varphi_m(t)$, $(x, W^*(t))$ satisfies at most an $\epsilon_{out}$-fraction of $\varphi_1(t), \ldots, \varphi_m(t)$. Consequently, for every "proof" $\pi^*(t) = \pi^*_{in} \circ \ldots \circ \pi^*_{in} \circ W^*(t)$, $(x, \pi^*(t))$ satisfies at most an $\epsilon_{out}$-fraction of $\varphi_1(t), \ldots, \varphi_m(t)$. If $V_B$ chooses to verify a predicate $\varphi(t)$, that is not satisfied by $x \circ \pi^*(t)$, then the soundness of $(P_{in}, V_{in})$ guarantees that for any proof $\pi^i$,

$$\Pr \left[ V^2_{in}(x \circ W^*(t))_{i}, \pi^i (1^t, |\varphi_i(t)|, \varphi_i(t)) = \text{acc} \right] \leq \epsilon_{in} (|\varphi(t)|, t) \leq \epsilon_{in} (q(t + 1), t)$$

(where the rightmost inequality holds since $\epsilon_{in}$ is non-increasing, and the minimal input length is $q$, i.e., a private predicate containing only input variables).

Therefore, conditioned on the event that $V_B$ chose a private predicate that is not satisfied under $\pi^i$, then it accepts with probability at most $\epsilon_{in}(q,t)$. Consequently, if $\beta \leq \epsilon_{out}$ denotes the fraction of private predicates that is satisfied, then $V_B$ accepts a false claim with probability at most $\beta + (1 - \beta) \epsilon_{in}(q,t) = \beta (1 - \epsilon_{in}(q,t)) + \epsilon_{in}(q,t) \leq \epsilon_{out} \cdot (1 - \epsilon_{in}(q,t)) + \epsilon_{in}(q,t)$.

Honest-verifier zero-knowledge. We describe a simulator $\text{Sim}$ for $V_B$ that employs a simulator $\text{Sim}_{in}$ that perfectly simulates the view of $V_{in}$. (The existence of $\text{Sim}_{in}$ follows from the zero-knowledge of the inner exact ZKPCPP system, since $V_{in}$ is $t$-query bounded.) On input $1^t$, $|x|$, $\text{Sim}$ operates as follows:

- Randomly samples a random string $r$ for $V_B$, and extracts from it the index $i \in [m]$ of the predicate that $V_B$ chooses to check, and the queries $a, b, c$ that $V_B$ makes when given oracle access to $x, \pi^1 \circ \ldots \circ \pi^m \circ W(t)$. (Notice that these are the queries of the honest $V_{in}$, when given input oracle $(x \circ W(t))_i$ and proof oracle $\pi^i$.)

- Constructs the private predicate $\varphi_i(t)$.

- Sends $a, b, c$ to $\text{Sim}_{in}$ as the queries of $V_{in}$, and let $A, B, C$ denote the answers of $\text{Sim}_{in}$. Every TTP-query that $\text{Sim}_{in}$ makes to $x_i$ during his simulation is forwarded to the TTP of $\text{Sim}$, and TTP-queries to bits of $W(t)$ are answered with random (and independent) bits.\(^4\)

- When $\text{Sim}_{in}$ terminates, $\text{Sim}$ outputs $(r, a, b, c, A, B, C)$.

We claim that $\text{Ideal}_{\text{Sim}}(t, x) \equiv \text{Real}_{V_B, P_B}(t, x, w)$. $i, a, b, c$ are completely determined by the randomness of $V_{in}$, and therefore equally distributed in both worlds. Moreover, the simulated oracle answers that $\text{Sim}$ gives to $\text{Sim}_{in}$ are identically distributed to the answers of a TTP of $\text{Sim}_{in}$, since every 3 bits of $W(t)$ are uniformly distributed.

\(^4\)Notice that the "input oracle" of $V_{in}$ is of the form $(x, W(t))_i$ for some $i \in [m]$, i.e., $\text{Sim}_{in}$ may query bits of $W(t)$.
Indeed, in a random proof $\pi_B \in_R P_B(1^t, x, w)$, $\pi(t)$ is a random sharing of $\pi$, so every set of bits $\pi_{j,1}, \ldots, \pi_{j,t+1}$ that correspond to a bit $\pi_j$ of $\pi$, is random such that $\pi_{j,1} \oplus \cdots \oplus \pi_{j,t+1} = \pi_j$ (and $t \geq 3$). Conditioned on the identity of the “input” oracle $W(d)$ of $V_{in}$, indistinguishability follows from the zero-knowledge of $(P_{in}, V_{in})$. 

**Remark 5.12.** We note that the soundness property of Construction 5.10 crucially relied on the fact that the inner system was an exact-ZKPCPP. Indeed, the soundness property of the outer system guarantees that for input $x$ which is far from $L$, and given any proof oracle $\pi^*$, a constant fraction of the tests of the outer verifier would fail, namely the restriction of $(x, \pi^*)$ to the bits queried by the verifier does not satisfy the verification predicate. However, the soundness property does not rule out the possibility that this restriction is very close to satisfying the verification predicate, and so if the inner system was a (standard, non-exact) PCPP, then the inner verifier might have accepted in this case. The stronger soundness property of an exact-PCPP guarantees that the inner verifier rejects any restriction that does not satisfy the verification predicate, regardless of how “close” it is to satisfying the predicate. Alternatively, we could have used a robust PCPP as the outer system, and a (standard) ZKPCPP as the inner system. (A robust PCPP guarantees that for a constant fraction of the tests performed by the outer verifier, the restriction of $(x, \pi^*)$ to the bits queried by the verifier are far from satisfying the verification predicate.)

**Remark 5.13.** If $(P_{out}, V_{out})$ has strong soundness with rejection ratio $\beta$, then the proof of Claim 5.4.2 shows that $(P_B, V_B)$ has strong soundness with rejection ratio $\beta \cdot (1 - \epsilon_{in}(q,t))$. Moreover, since the verifier of Construction 5.10 simply emulates the “inner” verifier $V_{in}$, then Construction 5.10 has the stronger guarantee of perfect zero-knowledge against malicious $d$-bounded verifiers.

The soundness error of Construction 5.10 depends both on the soundness error of the outer PCPP system, and on the soundness error of the inner exact ZKPCPP system. Consequently, the soundness error (as stated in Claim 5.4.2) degrades through this transformation, so our next goal is to reduce the soundness error.

**Sound HVZKPCPPs for NP**

We amplify the soundness of Construction 5.10 by repeating the verification procedure (of $V_B$). Notice that this requires modifying the ZKPCPP itself, since repetition does not necessarily preserve zero-knowledge. (That is, if the verifier simply repeats the verification procedure, then its queries may exceed the upper bound for which zero-knowledge is guaranteed.)

Intuitively, to overcome this we have the prover generate several independent “copies” of the proof generated by $P_B$, which we call basic proofs, and the verifier can repeatedly emulate $V_B$, using a “fresh” copy each time. This assumption that each copy is used at most once is the reason that we only get honest-verifier zero-knowledge. Indeed,
we increase the query complexity of the verifier without increasing the zero-knowledge
guarantee of the basic system (since increasing the zero-knowledge guarantee of the
“inner” system will also increase the soundness error). Therefore, a malicious verifier can
potentially break the zero-knowledge by using the same proof in several iterations.

Next, we describe the amplified system.

Construction 5.14. The modified HVZKPCPP system \((P_H, V_H)\) uses the system
\((P_B, V_B)\) of Construction 5.10 as a building block, and is parameterized by \(l\), the
number of basic proofs in a proof generated by \(P_H\); \(k\), the number of runs (of \(V_B\)) that
\(V_H\) emulates; and \(t\), to be passed on to the underlying HVZKPCPP system. We assume
without loss of generality that \(l \geq k\).

Prover algorithm. \(P_H\) on input \(1^l, 1^k, 1^t\) and \((x, w) \in \mathcal{R}\), uses \(P_B\) to generate \(l\)
independent (basic) proofs \(\pi_B^1, \ldots, \pi_B^l\) for the claim \((x, w) \in \mathcal{R}\), and outputs the proof
\(\pi_H = \pi_B^1 \circ \ldots \circ \pi_B^l\).

Verifier algorithm. \(V_H\) on input \(l, 1^k, 1^t, |x|\), and given access to oracles \(x, \pi_H\), picks
at random \(k\) different basic proofs \(\pi_B^1, \ldots, \pi_B^k\), and for every \(1 \leq i \leq k\), performs the
following basic verification step: \(V_H\) runs \(V_B\) with parameter \(t\), and oracles \(x, \pi_B^i\). (We
note that the basic verification steps are performed in parallel.) \(V_H\) accepts if \(V_B\)
accepted in all \(k\) iterations, otherwise it rejects.

The next lemma summarizes the properties of \((P_H, V_H)\).

Lemma 5.4.4. Construction 5.14 has perfect completeness, and perfect honest-verifier
zero-knowledge, such that if the underlying system \((P_B, V_B)\) has a straight-line, black-box
simulator, then so does Construction 5.14. Moreover, let \(\delta \in (0, 1)\) denote the proximity
parameter of \((P_B, V_B)\), and \(\epsilon (t)\) denote the soundness error of \((P_B, V_B)\) with parameter
\(t\). Then the soundness error of \((P_H, V_H)\) is \((\epsilon (t))^k\). Furthermore, if \(V_B\) is non-adaptive,
then so is \(V_H\).

Remark 5.15. The soundness guarantee of \((P_H, V_H)\) is in fact somewhat stronger than
“standard” soundness, as will be evident from the proof. Specifically, for every \(x\) that is
\(\delta\)-far from \(L_R\), for every “proof” \(\pi^*\) for \(x\), and for every \(1 \leq i \leq l\), the probability that a
random basic verification step of \(V_H^{x, \pi^*}\) in which he queries \(\pi^{*i}\) succeeds is at most \(\epsilon (t)\).
Moreover, if \((P_B, V_B)\) has strong soundness with soundness error \(\epsilon (t, \delta)\), then \((P_H, V_H)\)
has strong soundness with soundness error \((\epsilon (t, \delta))^k\).

Proof of Lemma 5.4.4. Let \((P_H, V_H)\) denote the system of Construction 5.14.

Soundness. Let \(x\) be \(\delta\)-far from \(L_R\), and set some proof oracle \(\pi^* = \pi^1 \circ \ldots \circ \pi^l\) for \(V_H\).
The soundness property of \((P_B, V_B)\) (Claim 5.4.2) guarantees that for every basic proof
\(\pi^i\), \(Pr [V_B^{x, \pi^*}(1^i, |x|) = \text{acc}] \leq \epsilon (t, \delta)\), which bounds the probability that a single basic
verification step of \(V_H\) succeeds. As the basic verification steps of \(V_H\) are independent,
the probability that all the basic verification steps succeed, and \(V_H\) accepts, is at most
\((\epsilon (t, \delta))^k\).
Honest-Verifier zero-knowledge. We describe a simulator \( \text{Sim} \) that uses the simulator \( \text{Sim}_B \) of \((P_B, V_B)\). On input \( l, 1^k, 1^t, |x| \) (where \( x \in L_R \) with witness \( w \)), \( \text{Sim} \) operates as follows:

- Randomly samples a random string \( r \) for \( V_H^* \), and extracts from it the indices \( i_1, \ldots, i_k \) of basic proofs that \( V_H^* \) queries when using randomness \( r \). Let \( Q_1, \ldots, Q_k \) denote the sets of queries \( V_H^* \) that makes in the \( k \) basic verification steps (notice that these queries are also determined by \( r \)).
- Runs \( \text{Sim}_B \) independently on each of \( Q_1, \ldots, Q_k \). Denote the simulated answers of \( \text{Sim}_B \) by \( A_1, \ldots, A_k \).
- During his simulations, \( \text{Sim}_B \) makes TTP-queries to bits of \( x \). \( \text{Sim} \) forwards these queries to his own TTP, and sends the TTP-answers back to \( \text{Sim}_B \).
- When the simulation of \( \text{Sim}_B \) ends, \( \text{Sim} \) outputs \((r, i_1, \ldots, i_k, Q_1, \ldots, Q_k, A_1, \ldots, A_k)\).

We show that \( \text{Ideal}_{\text{Sim}} (l, k, t, x) \equiv \text{Real}_{V_H^*, P_H} (l, k, t, x, w) \). Since \( r \) determines \( i_1, \ldots, i_k \), and the sets of queries \( Q_1, \ldots, Q_k \), these are equally distributed in the real-world and in the simulation. The TTP-queries of \( \text{Sim}_B \) are answered by the real TTP of \( \text{Sim} \), so by the zero-knowledge property of \((P_B, V_B)\), the simulated answers in every basic verification step are identically distributed to the answers of a real-world random proof oracle generated by \( P_B \left( 1^t, x, w \right) \). Moreover, if \( \text{Sim}_B \) is asked to simulate the answers to \( q' \) queries, then he makes \( q' \) TTP-queries. Therefore, \( \text{Sim} \) makes exactly \( \sum_{i=1}^{k} |Q_i| \) TTP-queries (the same as \( V_H^* \)), and \( A_1, \ldots, A_k \) are indistinguishable from the answers of \( x \), and a real-world random proof oracle \( \pi \), to the queries \( Q_1, \ldots, Q_k \) (since \( \pi \) consists of random and independent basic proofs).

For an appropriate choice of the parameters \( l, k \), Lemma 5.4.4 implies the following general transformation from PCPPs to HVZKPCPPs.

**Theorem 5.16 (HVZKPCPPs from PCPPs).** Let \( \sigma \) be a security parameter. Then for any query parameter \( q \in \mathbb{N} \), any soundness error parameter \( \epsilon_S = \epsilon_S (\sigma, |x|) \), any proximity parameter \( \delta = \delta (\sigma, |x|) \), any randomness complexity parameter \( r = r (\sigma, |x|) \) and any proof length parameter \( \ell = \ell (\sigma, |x|) \),

\[
\text{PCPP} \left[ r, q, \delta, \frac{1}{2}, \ell \right] \subseteq \text{HVZKPCPP} \left[ r', q', \epsilon'_{ZK} = 0, \delta' = \delta, \epsilon_S, \ell' \right]
\]

where \( r' = O_q (r \cdot \text{polylog} \frac{1}{\epsilon_S}) \), \( q' = O_q \left( \log \frac{1}{\epsilon_S} \right) \) and \( \ell' = O_q \left( (\ell + 2^t) \cdot \log \frac{1}{\epsilon_S} \right) \), and the zero-knowledge property holds with a straight-line, black-box simulator. Moreover, the honest HVZKPCPP verifier is non-adaptive.

**Remark 5.17.** When the HVZKPCPP system of Theorem 5.16 is based on Construction 5.10, which in turn uses Construction 5.7 as the inner system, then it has the
additional guarantee that the set $Q$ of verifier queries that the simulator is asked to simulate completely determines the identity of its TTP queries during the simulation. Indeed, the HVZKPCPP is based on Construction 5.14, whose simulator simply emulates multiple independent copies of the simulator for the underlying system (Construction 5.10). Therefore, the property follows directly from Remark 5.11.

**Proof of Theorem 5.16.** We instantiate Construction 5.14 with the parameters $t = O(1)$, and $k = l = O_q\left(\log\frac{1}{\epsilon_S}\right)$. Then by Claim 5.4.2, the underlying HVZKPCPP system $(P_B, V_B)$ has query complexity $q_B = q_{in}$, randomness complexity $r_B = r_{in} + r$, and proof length $\ell_B = O_q\left(\log\frac{1}{\epsilon_S}\right)$, $\ell' = l \cdot \ell_B = O_q\left((\ell + 2^r) \log\frac{1}{\epsilon_S}\right)$, and $r' = k (\log l + r_B) = O_q\left(r \cdot \text{polylog}\frac{1}{\epsilon_S}\right)$.

Completeness follows from the completeness of the basic HVZKPCPP system, and perfect honest-verifier zero-knowledge follows directly from Lemma 5.4.4. Regarding soundness, the soundness error of $(P_B, V_B)$ is $\epsilon_{out} \cdot (1 - \epsilon_{in}) + \epsilon_{in} = \frac{1}{2} (1 + \epsilon_{in})$ (where $\epsilon_{out} = \frac{1}{2}$ is the soundness error of the original PCPP, and $\epsilon_{in} < 1$ is the soundness error of the exact ZKPCPP which, since $t$ is constant, depends only on $q$), so by Lemma 5.4.4 $V_H$ accepts an $x$ that is $\delta$-far from $L_R$ with probability at most $\left(\frac{1}{2} (1 + \epsilon_{in})\right)^k = \epsilon_S$ (for an appropriate choice of the constant defining $k$).

**Remark 5.18.** Since the verifier in both Construction 5.14 and Construction 5.10 emulate the verifier of Construction 5.3, then the strong honest-verifier zero-knowledge feature of Construction 5.3 (see Section 5.3, Remark 5.5) guarantees that both systems have strong honest-verifier zero-knowledge properties. (See below for a more detailed analysis.)

At a high level, Theorem 5.16 states that Construction 5.14 inherits many of its properties from the underlying (outer) PCPP system. Therefore, efficient PCPPs yield efficient HVZKPCPPs, as stated in Corollary 5.9. The proof of Corollary 5.9 is obtained by applying Theorem 5.16 to the following PCPP system due to Dinur [Din06].

**Theorem 5.19** (PCPP, implicit in [Din06]). Let $\mathcal{R} = \mathcal{R}(x, w) \subseteq \text{DTIME}(T(n))$, then $\mathcal{R}$ has a strong PCPP system $(P, V)$ with constant rejection ratio in which $P$ outputs proofs of length $\text{poly}(T(n))$, and $V$ on inputs of length $n$ tosses $O(\log T(n))$ coins, and reads $O(1)$ bits from his oracles.

**Proof of Corollary 5.9.** Let $\epsilon, \delta \in (0, 1)$, and let $\mathcal{R} = \mathcal{R}(x, w) \subseteq \text{DTIME}(T(n))$. Since the PCPP system $(P, V)$ of Theorem 5.19 has strong soundness, then (by Claim 5.4.2 and Remark 5.13) the system $(P_B, V_B)$ of Construction 5.10, when instantiated with $(P, V)$ also has strong soundness with rejection ratio $\beta$, for some $\beta > 0$. We take $t = 3$ and $k = l = \frac{\log \frac{1}{\epsilon_S}}{\beta}$. Since $\delta = \frac{1}{\beta}$, in Construction 5.14, and notice that for this choice of $t$, $\beta$ is constant.

Let $(x, w) \in \mathcal{R}$, and denote $|x| = n$. Then every proof $\pi \in P_H\left(1^t, 1^l, 1^l, x, w\right)$ has size

$$|\pi| = l \cdot O_q(t (\ell + 2^r)) = \frac{1}{\delta} \log \frac{1}{\epsilon} \cdot \text{poly}(T(n)) = \text{poly}\left(T(n), \frac{1}{\delta}, \log \frac{1}{\epsilon}\right).$$
reads $k \cdot O(1) = O(\frac{1}{3} \log \frac{1}{\delta})$ oracle bits, and in every basic verification step he uses $O(\log l + \log T(n) + \log t) = O(\log \frac{1}{\delta} + \log \log \frac{1}{\epsilon} + \log T(n))$ random bits. Completeness, honest-verifier zero-knowledge and soundness follow from Lemma 5.4.4 (and Remark 5.15).

5.4.2 HVZKPCPPs with Stronger Honest-Verifier Zero-Knowledge Properties

In this section we show that Constructions 5.10 and 5.14 possess stronger (than standard) honest-verifier zero-knowledge properties, which will be used in Section 5.4.3 to construct ZKPCPPs for NP.

More specifically, Construction 5.14 is zero-knowledge against verifiers whose queries are distributed as the queries in “not too many” basic verification steps of the honest verifier $V_H$. (Notice that unlike strong honest-verifier zero-knowledge, in which for some parameter $k'$, zero-knowledge holds against the verifier $V^k_H$ of Notation 5.1, here zero-knowledge holds against a verifier emulating $k'$ basic verification steps of $V_H$, where every basic verification step emulates a single run of $V_B$.) Our goal is to prove that Construction 5.14 possesses the aforementioned stronger honest-verifier zero-knowledge guarantee. It would be useful to first show that Construction 5.10 has strong HVZK.

Strong Honest-Verifier Zero-Knowledge of Construction 5.10

We show that Construction 5.10 has perfect strong honest-verifier zero-knowledge against $t$ emulations of the honest verifier $V_B$, where $t$ is the zero-knowledge parameter of the system. That is,

**Lemma 5.4.5** (Strong HVZK, Construction 5.10). Let $t \in \mathbb{N}$. Then $(P_B, V_B)$, when instantiated with parameter $t$, has $(0, t)$-strong honest-verifier zero-knowledge. Moreover, if the underlying exact ZKPCPP system $(P_{in}, V_{in})$ has a straight-line, black-box simulator, then so does $(P_B, V_B)$.

**Proof.** Let $V_B^t$ denote the verifier of Notation 5.1, and $V_B^t(r)$ denote the run of $V_B^t$ when it uses randomness $r$. Remember that an honestly-generated proof for $V_B$ is of the form $\pi = \pi^1 \circ \ldots \circ \pi^l \circ W(t) \in P_B(1^t , x, w)$ (for some $l \in \mathbb{N}$), where every $\pi^i$ is an exact ZKPCPP and $W(t)$ is an assignment extension of $x$.

The simulator $\text{Sim}$ on input $1^t, |x|$ operates in the following way:

- Randomly samples a random string $r$ for $V_B^t$. We interpret $r$ as $r = (r^1, \ldots, r^t)$, where every $r^i$ is a random string for $V_B$.

- Uses $r$ to determine sets $A^i_1, \ldots, A^i_t$ of queries that $V_B^t(r)$ makes to $\pi^1, \ldots, \pi^t$, respectively. (That is, $V_B^t(r)$ makes $t$ sets $Q_1, \ldots, Q_t$ of queries, where every $Q_i$ queries some exact ZKPCPP $\pi^i$, and $A^i_t$ consists of the union over all $Q_i$'s in which $\pi^i$ is queried.)
• Using \( t \) independent instantiations of the simulator \( \text{Sim}_{in} \) of the underlying exact ZKPCPP system, simulates oracle answers to the queries \( A_1, \ldots, A_l \). We denote these instantiations by \( \text{Sim}_1, \ldots, \text{Sim}_t \). (More specifically, the queries of \( A_i \) are sent to \( \text{Sim}_i \), and its answers are returned to \( V_B^t \).)

• During its simulation, \( \text{Sim}_t \) may send TTP queries to \( (x \circ W (t))_l \). Queries to \( x \) are answered using the TTP of \( \text{Sim} \). To answer queries to \( W (t) \), \( \text{Sim} \) recursively constructs, and maintains, a set \( \text{Ans} \) of pairs of variables and assignments, initialized to \( \emptyset \) (That is, \( \text{Ans} \) consists of elements \((z, c)\), where \( z \) is a variable of the \( t \)-private form \( \varphi (t) \) of the 3CNF representing the corresponding decision predicate of the original PCPP verifier; and \( c \in \{0, 1\} \) is an assignment to \( z \).) For every TTP-query \( z \) of \( \text{Sim}_t \), if there exists a \( c \in \{0, 1\} \) such that \((z, c) \in \text{Ans}\) then \( \text{Sim} \) sends \( c \) to \( \text{Sim}_t \) as the TTP answer. Otherwise, \( \text{Sim} \) chooses \( c \in \mathcal{R} \) \{0, 1\}, adds \((z, c)\) to \( \text{Ans} \), and sends \( c \) as the answer of the TTP.

• At the end of the simulation, \( \text{Sim} \) outputs \((r, \text{View}_{\text{Sim}_1}, \ldots, \text{View}_{\text{Sim}_t}) \) (where \( \text{View}_{\text{Sim}_i} \) denotes the output of \( \text{Sim}_i \)).

We claim that for \((x, w) \in \mathcal{R} \), the simulation of \( V_B^t \) is perfect, and \( \text{Sim} \) makes only \( t \) TTP queries. As the randomness \( r \) determines the sets \( A_1, \ldots, A_l \) of queries of \( V_B^t \), we prove the claim conditioned on the values of \( A_1, \ldots, A_l \). Let \( \pi = \pi_1 \circ \ldots \circ \pi_t \circ W (t) \) denote the proof oracle of \( V_B^t \), where for every \( i \in [l] \), \( \pi_i \in P_m (1^l, (x \circ W (t))_l, \varphi_l (t), \lambda) \). Moreover, for a random honestly-generated \( \pi, W (t) \) is a random assignment-extension of \( x \) to \( \varphi_1 (t) \wedge \ldots \wedge \varphi_l (t) \) that is consistent with \( w \); and the exact ZKPCPPs \( \pi_1, \ldots, \pi_t \) are random and independent, conditioned on \( W (t) \). In particular, conditioned on \( W (t) \) the views \( \text{View}_{V_B, 1}, \ldots, \text{View}_{V_B, l} \) in the \( l \) runs of \( V_B \) are random and independent.

In every run, \( V_B \) queries a single exact ZKPCPP, so the queries of \( V_B^t \) to each exact ZKPCPP constitute the queries \( t_i \leq t \) runs of the honest exact ZKPCPP verifier \( V_{in} \), and let \( q_1, \ldots, q_t \) be such that for every \( 1 \leq i \leq l \), \( A_i \) are the queries that \( V_{in} \) makes in \( t_i \) emulations (notice that \( \sum_{i=1}^{l} q_i = t \)). Then the \((0, t)-\)strong honest-verifier zero-knowledge of \((P_{in}, V_{in}) \) (Claim 5.3.5) guarantees that for every \( i \in [l] \), and every implicit input \((x \circ W (t))_i \), \( \text{Sim}_i \) can perfectly simulate the view of \( V_{in}^t \), with only \( t_i \) TTP queries. Consequently, \( \text{Sim} \) perfectly simulates \( V_B^t \), if the simulated TTP answers that \( \text{Sim} \) gives to \( \text{Sim}_1, \ldots, \text{Sim}_t \) are indistinguishable from the answers of a “real” TTP who answers according to \( W (t) \).

To complete the proof, we show that the real-world and simulated TTP answers are identically distributed. The answers of the TTP are consistent with \((x \circ W (t))_1, \ldots, (x \circ W (t))_l \), where \((x \circ W (t))_1, \ldots, (x \circ W (t))_l \) are the restrictions of \( x \circ W (t) \) to the variables of \( \varphi_1 (t), \ldots, \varphi_l (t) \); and \( W (t) \) is a random assignment-extension of \( x \) to \( \varphi_1 (t) \wedge \ldots \wedge \varphi_l (t) \), that is consistent with \( w \). \( \text{Sim} \) uses his TTP to answer queries to \( x \), so these answers are identical to the real-world answers. Furthermore, for every \( i \in [l] \), \( \text{Sim}_i \) makes \( q_i \) TTP-queries, so \( \text{Sim} \) simulates a total of \( \sum_{i=1}^{l} q_i = t \).
TTP-answers (to queries to $W(t)$) during the entire simulation. Since every $t$ bits of a random assignment-extension $W(t)$ to $\varphi_1(t) \land \ldots \land \varphi_l(t)$ (that is consistent with $w$) are independent and uniformly distributed, these answers are identically distributed to the real-world.

A Stronger Honest-Verifier Zero-Knowledge Guarantee of Construction 5.14

In this section we use Lemma 5.4.5 to prove that Construction 5.14 has the following a stronger honest-verifier zero-knowledge guarantee. There exists a bound $K = K(k, l)$, and a straight-line, black-box simulator $\text{Sim}$, such that $\text{Sim}$ can simulate the view of every verifier $V$, whose queries are distributed as the queries that $V_H$ makes in random and independent basic verification steps. Moreover, during the simulation $\text{Sim}$ makes only $k'$ TTP queries. Notice that the queries of $V^{k'}$ are not distributed as the queries of $V_H$, when it uses parameter $k = k'$, since $V_H$ is guaranteed to use different basic proofs in each basic verification step, whereas $V^{k'}$ is not. We note that this property is similar to strong HVZK, except that (in each invocation) $V_H$ performs a single basic verification step (rather than performing $k$ verification steps).

Notice that for $k' \leq k$, if $V^{k'}$ is restricted to using every basic proof in at most $t$ basic verification steps, where $t$ denotes the zero-knowledge parameter of the underlying system $(P_B, V_B)$, then the aforementioned property follows directly from the independence of the basic proofs (given $x, W(t)$), and from Lemma 5.4.5. By choosing $l$ to be sufficiently larger than $k, K$, we can guarantee that the probability that any basic proof is queried in more than $t$ basic verification steps is negligible (in $t$). We now formalize this intuition.

The following lemma establishes the connection between the parameters which guarantees that (except with negligible probability) every basic proof is queried in at most $t$ basic verification steps.

Lemma 5.4.6. Let $2 \leq k' \in \mathbb{N}$. If $k'^4 \leq l$. Then for every input oracle $x$, and proof oracle $\pi$, the probability that a single basic proof in $\pi$ is queried in more than $t$ basic verification steps in a run of $(V^{k'})^{x, \pi}(1^k, l, 1^t, |x|)$ is at most $2^{-t-1}$.

Proof. $(V^{k'})^{x, \pi}(1^k, l, 1^t, |x|)$ chooses $k'$ basic proofs $\pi^{i_1}, \ldots, \pi^{i_{k'}} \in_R \{\pi^1, \ldots, \pi^l\}$, so the probability that any one of $\pi^1, \ldots, \pi^l$ is chosen more than $t$ times is at most

$$\frac{l(t+1)}{t+1} < \frac{k^{t+1}}{l^{t+1}} \leq \frac{k^{t+1}}{k^{4t}} = k'^{-3t+1} \leq k'^{-t-2+1} = k'^{-t-1} \leq 2^{-t-1}.$$  

Using Lemma 5.4.6, we can now show that for an appropriate choice of $t, l$, the view of $V^{k'}$ can be statistically simulated with only $k'$ TTP queries.

Lemma 5.4.7. Let $2 \leq k' \in \mathbb{N}$ such that $k'^4 \leq l$. Then there exists a simulator $\text{Sim}$ that can simulate the view of every verifier $V^{k'}$ whose queries are distributed as the queries in $k'$ random and independent basic verification steps of $V_H$ (where in each
step the proof oracle for $V_B$ is randomly selected from the $l$ basic proofs). During the simulation, Sim makes at most $k'$ TTP queries. Moreover, there exists a “bad event” $B$ such that: $\Pr[B] \leq 2^{-l-1}$; the probability that $B$ occurs depends only on the queries of $V^k'$ (and is independent of the identity of his oracles); and conditioned on $B$, the simulation is perfect. Moreover, if the underlying system $(P_B, V_B)$ has a straight-line, black-box simulator for $V_B^t$, then Sim is straight-line, and black-box.

Proof. Let $V^k_r$ denote the run of $V^k$ when it uses randomness $r$. Remember that an honestly-generated proof for $V_H$ is of the form $\pi = \pi^1 \circ \ldots \circ \pi^l$, where every $\pi^i$ is a basic proof. Let $Sim_B$ denote the (straight-line, black-box) simulator that can perfectly simulate the view of $V_B^t$ while making only $t$ TTP queries (and whose existence follows from Lemma 5.4.5).

The simulator Sim on input $1^k, l, 1^t, |x|$ (where $x \in L_R$) operates in the following way:

- Samples a random string $r$ for $V^k$. We interpret $r$ as $r = (r^1, \ldots, r^{k'})$, where $r^i$ is a random string for a single basic verification step of $V_H$ (consisting of a random string for $V_B$, and the index of a basic proof to be used).

- Uses $r$ to determine sets $A^1_r, \ldots, A^l_r$ of queries that $V^k_r$ makes to $\pi^1, \ldots, \pi^l$, respectively. That is, $V^k_r$ makes $k'$ sets $Q_1, \ldots, Q_{k'}$ of queries, where every $Q_i$ is to some basic proof $\pi^i$, and $A^i_r$ consists of the union over all $Q_i$’s which query $\pi^i$. For every $1 \leq i \leq l$, let $n_i$ denote the number of $Q_i$’s participating in the union defining $A^i_r$. If $n_i > t$ for some $1 \leq i \leq l$ (i.e., the same basic proof is used in more than $t$ basic verification steps) then Sim halts with output \bot.

- Using $l$ independent instantiations of $Sim_B$, which we denote by $Sim_1, \ldots, Sim_l$, Sim simulates the answers to the queries $A^1_r, \ldots, A^l_r$. The TTP queries of $Sim_1, \ldots, Sim_l$ are answered using the TTP of Sim.

- When all the simulations terminate, Sim outputs $r$, together with the views View$_{Sim_1}, \ldots, View_{Sim_l}$ of $Sim_1, \ldots, Sim_l$.

We define the “bad event” $B$ to be the event that some basic proof was queried in more than $t$ basic verification steps. We show that conditioned on $B$, the simulation is perfect. The random string $r$ determines the sets $A^1_r, \ldots, A^l_r$, so we prove indistinguishability conditioned on the identity of these sets. Since we have conditioned on $B$ then for every $1 \leq i \leq l$, $A^i_r$ is the queries that $V_B^{n_i}$ makes with the random string induced by $r$, where $n_i \leq t$. Therefore, Lemma 5.4.5 guarantees that each $Sim_1, \ldots, Sim_l$ perfectly simulates the view of $V_B^{n_i}$. Moreover, as the emulations of $V_B$, and the proof oracles used for them, are independent (because an honestly-generated proof consists of random and independent basic proofs), the concatenated simulated views are identically distributed to the real-world view. Moreover, every $Sim_i$ makes only $n_i$ TTP-queries, so Sim makes $\sum_{i=1}^l n_i = k'$ TTP-queries.
Using Lemma 5.4.6, and the fact that $2 \leq k' \leq \sqrt{l}$, $\Pr[B] \leq 2^{-l-1}$, and whether or not $B$ occurs is a function of the queries of $V^{k'}$. (The queries of $V^{k'}$ are independent of the identity of his oracles, so $B$ is independent of the identity of the oracles.) \qed

The following corollary gives a more accurate characterization of the HVZKPCPP system $(P_H, V_H)$ of Construction 5.14.

**Corollary 5.20.** Let $\epsilon > 0$ be a soundness parameter, $\delta > 0$ be a proximity parameter, and $q^* \in \mathbb{N}$ be a zero-knowledge parameter. Then every relation $R = R(x, w) \in \text{DTIME}(T(n))$ has an HVZKPCPP system $(P_H, V_H)$ with the following properties:

- The systems have perfect completeness, perfect honest-verifier zero-knowledge with a straight-line, black-box simulator, proximity parameter $\delta$, and soundness error $\epsilon$.
- $P_H(1^[\epsilon], 1^[\delta], t^{q^*}, x, w)$ generates proofs of length $\text{poly}(T(n), q^*, \log \frac{1}{\delta}, \frac{1}{\epsilon})$, and every proof can be divided into $\text{poly}(q^*, \log \frac{1}{\delta}, \frac{1}{\epsilon})$ sections called “basic proofs”.
- $V^{x,\pi}_H(1^[\epsilon], 1^[\delta], q^*, |x|)$ is non adaptive, and the verification consists of $\text{poly}(\log \frac{1}{\delta}, \frac{1}{\epsilon})$ “basic verification steps”, where each step consists of emulating $V_B$ on a basic proof that has not been used before. In every basic verification step $V_H$ tosses $O(\log T(n) + \log q^* + \log \log \frac{1}{\epsilon} + \log \frac{1}{\delta})$ random coins, and reads 3 bits from the basic proof chosen for the step. Moreover, If $x$ is $\delta$-far from $L_R$ then for every proof $\pi^*$, and every basic proof $\pi^i$ in $\pi^*$, a single basic verification step of $V_H$ in which he uses the oracle $\pi^i$ succeeds with probability at most $1 - \frac{4}{\text{polylog}\frac{1}{\delta}}$.
- There exists a straightforward, black-box simulator $\text{Sim}$ that can simulate the view of every verifier $V^{k'}$, $k' \leq q^*$ that emulates $k'$ random and independent basic verification steps of $V_B$. When simulating $V^{k'}$, $\text{Sim}$ makes only $k'$ TTP-queries. and the simulated and actual views are $\frac{\epsilon}{2}$-statistically close.

**Remark 5.21.** As will be evident from the proof, the simulation of $V^{k'}$ has the following property: unless the queries of $V^{k'}$ fall in a “bad” set, then $\text{Sim}$ can perfectly simulate the view of $V^{k'}$, and the probability that the queries of $V^{k'}$ fall in the “bad” set is at most $\frac{\epsilon}{2}$.

**Proof.** Let $\epsilon, \delta \in (0, 1)$, and $q^* \in \mathbb{N}$ (we assume without loss of generality that $q^* \geq 2$), and let $R = R(x, w) \in \text{DTIME}(T(n))$. The proof is similar to the proof of Corollary 5.9, when Construction 5.14 uses parameters $t = \log \frac{1}{\epsilon}, k = \frac{1}{3} \cdot \text{polylog}\frac{1}{\delta}$ and $l = \max\left\{q^*, k\right\}$; and is based on the basic HVZKPCPP system of Claim 5.4.2, which is instantiated with the exact ZKPCPP system of Claim 5.3.5 (based on the PCPP system of Theorem 5.19).

Let $(x, w) \in R$ such that $|x| = n$, then by Claim 5.4.2, every proof $\pi \in P_H(1^[1], 1^[1], 1^[1], x, w)$ has size $|\pi| = \log q^* (t \ell) + 2^t \cdot \text{poly}_q(t) = \text{poly}(T(n), q^*, \log \frac{1}{\epsilon}, \frac{1}{\delta})$, where $r$ denotes the number of random bits used by the PCPP verifier (of Theorem 5.19).
In every basic verification step \( V_H \) reads 3 bits from his oracles and tosses \( O \left( \log l + \log T(n) + \log t \right) = O \left( \log T(n) + \log q^* + \log \log \frac{1}{\delta} + \log \frac{1}{\epsilon} \right) \) random coins. (Indeed, \( V \) uses \( O(\log l) \) bits to determine which basic proof to use, and \( O(\log T(n) + \log t) \) bits to emulate \( V_B \).)

Completeness follows directly from the perfect completeness of all underlying systems. Honest-verifier zero-knowledge follows from Claim 5.4.2 (because the queries sent by the honest verifier \( V_H \) to every basic proof do not exceed the zero-knowledge guarantee of the exact ZKPCPP system). Moreover, since the underlying PCPP system has strong soundness, and every basic verification step of \( V_H \) emulates \( V_B \), then by Claim 5.4.2, Claim 5.3.5, and Remark 5.15, for any \( \delta \in (0,1) \), any \( x \) that is \( \delta \)-far from \( L_R \), and any basic proof \( \pi_B^* \), a single basic verification step of \( V_H \) with oracles \( x, \pi_B^* \) succeeds with probability at most \( 1 - \delta \). The basic verification steps of \( V_H \) are random and independent, so for an appropriate choice of the powers in the definition of \( k \), the soundness error is at most \( \left( 1 - \frac{\delta}{\text{polylog} \frac{1}{\epsilon}} \right)^k \leq \epsilon \). As for the stronger zero-knowledge guarantee, since \( 2 \leq q^* \leq 4 \sqrt{\log l} \), Lemma 5.4.7 guarantees that there exists a straight-line, black-box simulator \( \text{Sim} \) that can perfectly simulate the view of \( V^{k'} \) for every \( k' \leq k \), unless a “bad” event \( B \) occurs. (Here, we also use the guarantee of Claim 5.3.5 that the exact ZKPCPP system has a straight-line, black-box simulator.) Moreover, the probability that \( B \) occurs is a function of the queries of \( V^{k'} \) (namely, \( B \) occurs if the queries of \( V^{k'} \) fall in a “hard set”), and \( \Pr [B] \leq 2^{-t-1} \leq \frac{\epsilon}{2} \).

### 5.4.3 From HVZKPCPPs and Locking Schemes to ZKPCPPs

In this section we use locking schemes, and a PCPP system with a (somewhat stronger than standard) honest-verifier zero-knowledge guarantee, to construct ZKPCPPs for NP with zero-knowledge against arbitrary query-bounded verifiers:

**Theorem 5.22 (ZKPCPPs for NP).** Let \( \epsilon > 0 \) be a soundness parameter, \( \delta > 0 \) be a proximity parameter, and \( q^* \in \mathbb{N} \) be a zero-knowledge parameter. Then every relation \( R(x,w) \in \text{DTIME}(T(n)) \) has a ZKPCPP system \( (P,V) \) with soundness error \( \epsilon \), proximity parameter \( \delta \), and straight-line, black-box \( (\epsilon, q^*) \)-zero-knowledge. \( P \) on input \( x \) generates proofs of length \( \text{poly} \left( T(|x|), q^*, \log \frac{1}{\delta}, \log \frac{1}{\epsilon} \right) \), and \( V \) on input \( |x| \) makes \( \text{poly} \left( \log T(|x|), q^*, \log \frac{1}{\epsilon} \right) \) queries.

We first describe the high-level idea of the transformation. For a query bound \( q^* \in \mathbb{N} \), let \( (P_H, V_H) \) denote an HVZKPCPP that has the \( q^* \)-stronger zero-knowledge guarantee of Lemma 5.4.7 (e.g., the system of Construction 5.14, see Remark 5.18). That is, the system is zero-knowledge against any verifier whose queries are distributed as in \( q^* \) random and independent emulations of the basic verification step of \( V_H \).

Intuitively, to achieve zero-knowledge against arbitrary \( (q^* \)-bounded) verifiers, it suffices to “force” the queries of every (possibly malicious) verifier to be distributed

149
identically to the queries in $q^*$ random and independent basic steps of $V_H$. Following Kilian et al. [KPT97], we achieve this by employing a locking scheme.

Hiding several technical details, the proof consists of three sections: the PCPP section in which the prover $P$ locks (using the locking scheme) every bit of the HVZKPCPP; the PERM section which contains a locked permutation of the random strings used by the honest verifier $V_H$ in a single basic step (namely, $P$ picks a random permutation $\tau$ over the space of random strings, and for every such random string $r$, $P$ lock the image $\tau(r)$ in the PERM section); and the MIX section, where the location indexed by $\tau(r)$ contains $r$ and the collection of keys for the locks holding the HVZKPCPP bits that $V_H$ (with randomness $r$) queries.

To verify the proof, $V$ picks a random string $r'$, queries MIX$_{r'}$, and retrieves some (other) random string $r$ and a set of keys, which it uses to unlock the corresponding locks. Then, $V$ verifies that the lock PERM$_r$ holds the string $r'$, and that $V_H$ would accept (given the HVZKPCPP bits locked in the PCPP section of the proof). This is formalized in the next construction.

Construction 5.23. The system $(P, V)$ employs a locking scheme $(S, R)$, and an HVZKPCPP $(P_H, V_H)$. We assume that the proofs generates by $P_H$ consist of several smaller components called basic proofs, and that the verification procedure consists of several basic verification steps, where in every basic verification step $V_H$ non-adaptively queries 3 bits from the input and a fresh basic proof.\(^5\) The system is parameterized by a soundness parameter $\epsilon$, a proximity parameter $\delta$, and a zero-knowledge parameter $q^*$. in the following, we use rand to denote the space of random strings that $V_H$ uses in a single basic verification step, and $V_H(r), r \in \text{rand}$ to denote a basic verification step of $V_H$ in which it uses the random string $r$.

Prover algorithm. $P$ on input $1^{|\epsilon|}, 1^{|\delta|}, 1^{q^*}$ and $(x, w) \in \mathcal{R}$:

- Generates a proof $\pi_H \in P_H(1^{|\epsilon|}, 1^{|\delta|}, 1^{q^*}, x, w)$ for the claim “$(x, w) \in \mathcal{R}$”.
- Uses $S$ to creates a lock-key pair $(L_i, K_i)$ for every bit $b_i$ of $\pi_H$, and sets PCPP$_i = L_i$. (That is, PCPP$_i$ consists of $|L_i|$ bits.)
- Generates a random permutation $\tau$ on rand. Then, for every $r \in \text{rand}$, $P$ uses $S$ to generate a lock-key pair $(L_r, K_r)$ for $\tau(r)$, and sets PERM$_r = L_r$.
- For every $r \in \text{rand}$, let $a, b, c$ denote the bits that $V_H(r)$ queries. Without loss of generality, assume that $a$ is an input query and that $b, c$ are proof queries. Then $P$ sets MIX$_{\tau(r)} = (r, K_b, K_c, K_r)$.
- $P$ outputs the proof $\pi = \text{PCPP} \circ \text{PERM} \circ \text{MIX}$.

\(^5\)This is consistent with the system of Corollary 5.20, which is the system with which $(P, V)$ will be instantiated.
Verifier algorithm. On input $|\epsilon|, 1^{[6]}, q^*, n$, input oracle $x = (x_1, \ldots, x_n)$ and proof oracle $\pi = \text{PCPP} \circ \text{PERM} \circ \text{MIX}$, $V$ performs $t = \frac{1}{\delta} \text{polylog} \frac{1}{\epsilon}$ independent iterations of the following:

- Chooses a random $r' \in_R \text{rand}$ and queries the entry $\text{MIX}_{r'}$. Without loss of generality, assume that $\text{MIX}_{r'} = (r, K_b, K_c, K_r)$, where $V_H(r)$ makes the input query $a$, and proof queries $b, c$.
- Decommits $\text{PERM}_r$ (using $K_r$) and verifies that it locks the value $r'$.
- Reads $x_a$, decommits $\text{PCPP}_b, \text{PCPP}_c$ (using $K_b, K_c$), and obtains the bits $\pi_{H,b}, \pi_{H,c}$.
- Verifies that $V_H(r)$, given oracle answers $x_a, \pi_{H,b}, \pi_{H,c}$, accepts.
- If at least one of the decommitment operations failed, or $V_H$ rejected in at least one iteration, then $V$ rejects. Otherwise, $V$ accepts.

The next lemma summarizes the connection between the error probability of a single iteration of $V$, the binding of the locking scheme, and the error probability of a single basic verification step of $V_H$.

Lemma 5.4.8 (Soundness error, single iteration). Let $\epsilon_H, \epsilon_{LS} > 0$ be error parameters, and $(S, R)$ be a $(1 - \epsilon_{LS})$-binding locking scheme. Let $\delta$ be a proximity parameter, and $\mathcal{R}$ be an NP-relation with PCPP system $(P_H, V_H)$ with proximity parameter $\delta$. Assume that $P_H$ generates proofs consisting of $t$ sections (called “basic proofs”), and the verification procedure of $V_H$ consists of $t$ steps (called “basic verification steps”), each using a different basic proof. Assume that for any $x$ that is $\delta$-far from $L_R$, any proof oracle $\pi^*_H$ for $V_H$, and any basic proof $\pi^I$ in $\pi^*$, a single basic verification step of $V_H^{x,\pi^*_H}$ in which it uses the oracle $\pi^I$ succeeds with probability at most $1 - \epsilon_H$. Then for every $x$ that is $\delta$-far from $L_R$, and every $\pi^*$, a single iteration of the verifier $V_H^{x,\pi^*}$ of Construction 5.23 succeeds with probability at most $1 - \epsilon_H (1 - 3\epsilon_{LS})$.

Proof. Set some $x$ that is $\delta$-far from $L_R$, and let $\pi^*$ be the proof oracle of $V$. We can assume without loss of generality that each lock in PCPP and PERM has a well defined content. Indeed, the $(1 - \epsilon_{LS})$-binding of the underlying locking scheme guarantees that for every lock there is a value $w$ such that the lock decommits to a $w' \notin \{w, \bot\}$ with probability at most $\epsilon_{LS}$. Denote the proof obtained from the values locked in PCPP by $\pi^I_H$, and let $\tau(r)$ denote the value locked in $\text{PERM}_r$ (notice that $\tau$ may not be a permutation). We say that $r' \in \text{rand}$ is good if $r' = \tau(r)$ for some $r \in \text{rand}$, and $V_H(r)^{x,\pi^*_H}$ succeeds (otherwise we call $r'$ bad). Notice that at least $\epsilon_H |\text{rand}|$ of the random strings $r' \in \text{rand}$ are bad, since for every $1 \leq i \leq l$ a random verification step of $V_H^{x,\pi^*_H}$ in which he queries $\pi^I_H$ fails with probability at least $\epsilon_H$.

We claim that if $V$ chooses a bad $r'$, it rejects with probability at least $1 - 3\epsilon_{LS}$, i.e. overall $V$ rejects with probability at least $\epsilon_H (1 - 3\epsilon_{LS})$. Let $r$ be the randomness
reported in MIX\(_r\)'s view, then we consider two possible cases. First, if \(r'\) is locked in PERM\(_r\), then \(V_H(r)^{x,\pi_H} \) fails (since \(r'\) is bad). Let \(a,b,c\) be the bits that \(V_H(r)\) queries. For every input query among \(a,b,c\), \(V\) sends the query to \(x\), and for every proof query he tries to unlock the corresponding lock. If all the locks decommit to the corresponding bits of \(\pi_H\) (or at least one lock decommits to \(\bot\)), then the basic verification step of \(V\) fails (since \(V_H(r)^{x,\pi_H} \) rejects). Therefore, if the basic verification step of \(V\) succeeds then at least one of the locks succeed in decommitting to a different (legal) value, which happens (for every single lock) with probability at most \(\epsilon_{LS}\). Using the union bound, and the fact that every basic verification step unlocks at most three locks, the basic verification step of \(V\) fails except with probability \(3\epsilon_{LS}\).

Second, if the value locked in PERM\(_r\) is not \(r'\), then the basic verification step succeeds only if \(L_r\) successfully decommitted to \(r'\) (i.e. to a value different than the one it was locked with), which happens with probability at most \(\epsilon_{LS}\). Therefore, every basic verification step of \(V\) fails with probability at least \(\epsilon_H \cdot \min \{1 - 3\epsilon_{LS}, 1 - \epsilon_{LS}\} = \epsilon_H (1 - 3\epsilon_{LS})\). \(\square\)

Next, we show that Construction 5.23 is zero-knowledge against \(q^*\)-bounded verifiers.

**Lemma 5.4.9 (Zero-knowledge).** Let \(\epsilon_H, \epsilon_{LS} > 0\) be error parameters, \(q^* \in \mathbb{N}\) be a zero-knowledge parameter, \((S,R)\) be a \((q^*, \epsilon_{LS})\)-equivocal locking scheme, and \((P_H,V_H)\) denote a PCPP system in which the verification procedure consists of repeated iterations (called “basic verification steps”). Assume that \((P_H,V_H)\) has a straight-line, black-box simulator that can simulate, with statistical error \(\epsilon_H\), the view of every verifier \(V_H\) whose queries are distributed as in at most \(q^*\) random and independent basic verification steps, while making only \(q^*\) TTP queries. Then the PCPP system of Construction 5.23, when based on \((S,R)\), and \((P_H,V_H)\), is \((\epsilon, q^*)\)-zero-knowledge with a black-box, straight-line simulator, where \(\epsilon_{ZK} = \epsilon_H + q^* \epsilon_{LS}\).

**Proof.** Let \(\text{Sim}_H\) denote the simulator for \((P_H,V_H)\), and \(\text{Sim}_{LS}\) the simulator for \((S,R)\). The simulator \(\text{Sim}\) for Construction 5.23, on input \(|e|, 1^{|\delta|}, q^* , |x|\), samples a random string \(r^*\) for \(V^*\), and emulates \(V^*\) until \(V^*\) either halts with some output, or tries to make more than \(q^*\) queries. During the simulation, \(\text{Sim}\) simulates the oracle answers to the queries of \(V^*\) as follows.

- **Queries to MIX\(_r^\prime\).** If this is not the first query to MIX\(_r^\prime\), then \(\text{Sim}\) uses the simulated entry (which was already generated during the simulation). Otherwise, \(\text{Sim}\):
  - Randomly selects a random string \(r \in_R \text{rand}\) that has not been used before in the simulation, and sets \(\tau (r) = r'\).
  - Extracts from \(r\) the queries \(a,b,c\) that \(V_H(r)\) makes, and uses \(\text{Sim}_H\) to generate simulated answers \(A,B,C\) to these queries. (TTP queries that \(\text{Sim}_H\) makes during the simulation are forwarded to the TTP of \(\text{Sim}\).)
- For every proof query \( z \) among \( a, b, c \), \( \text{Sim} \) provides (an independent instantiation of) \( \text{Sim}_{LS} \) with the corresponding simulated answer \( A, B, C \), and uses \( \text{Sim}_{LS} \) to simulate the key \( K_z \) that corresponds to the pair \((L_z, K_z)\). (For every key \( K \) already determined during the simulation, \( \text{Sim} \) uses the simulated key \( K \) instead of invoking \( \text{Sim}_{LS} \).)

- Constructs the simulated entry \( \text{MIX}_r \), using \( A, B, C, r \), and the simulated keys generated by \( \text{Sim}_{LS} \), extracts from it the bit that \( V^* \) queried, and sends it to \( V^* \).

- **Queries to a lock \( L \) in PCPP or PERM.** If the value of the corresponding bit has already been determined during the simulation, then \( \text{Sim} \) provides this value to \( V^* \). Otherwise, \( \text{Sim} \) uses \( \text{Sim}_{LS} \) to simulate the bit. The simulation of \( \text{Sim}_{LS} \) is based on the value locked in \( L \) (if it has already been determined), and simulated bits of \( L \) already determined during the simulation. The simulated bit is sent to \( V^* \) as the answer of the oracle.

If during the simulation \( V^* \) tried to make more than \( q^* \) queries, then \( \text{Sim} \) outputs \( \bot \). Otherwise, its output consists of \( r^* \), the queries of \( V^* \), and the simulated answers.

Since \( \text{Sim}_H \) is black-box and straight-line (by assumption), and \( \text{Sim}_{LS} \) is black-box and straight-line (by the definition of equivocation), then so is \( \text{Sim} \).

We claim that \( \text{SD} \left( \text{Ideal}_{\text{Sim}}(\epsilon, \delta, q^*, x), \text{Real}_{V^*,P}(\epsilon, \delta, q^*, x, w) \right) \leq \epsilon_{ZK} \) for \( \epsilon_{ZK} = \epsilon_H + q^* \epsilon_{LS} \). Since the partial permutation \( \tau \) chosen by \( \text{Sim} \) is distributed identically to the real world (namely, random such that it is a permutation), we bound the statistical distance conditioned on every possible \( \tau \).

Our key observation is that the simulation of \( \text{Sim}_H \) is essentially independent of the simulations of \( \text{Sim}_{LS} \). Indeed, conditioned on the random \( \tau \) (and since \( V^* \) makes at most \( q^* \) queries), \( \text{Sim}_H \) is asked to adaptively simulate the answers to the queries in (at most) \( q^* \) random and independent basic verification steps of \( V_H \).

In particular, by the stronger honest-verifier zero-knowledge property of \((P_H, V_H)\)

\[
\text{SD} \left( \text{Ideal}_{\text{Sim}}(\epsilon, \delta, q^*, x), \text{Real}_{V^*,P}(\epsilon, \delta, q^*, x, w) \right) \leq \epsilon_H, \quad \text{and during the simulation} \quad \text{Sim} \text{ makes at most } q^* \text{ TTP queries.} \quad \text{(Specifically, the number of TTP queries is the same as the number of basic verification steps that Sim is asked to simulate.)}
\]

Next, we analyze the simulation quality for the locks, conditioned on the answers to the queries of \( \text{Sim}_H \), and show that conditioned on the values being the same in both worlds, the statistical distance is at most \( q^* \epsilon_{LS} \), concluding that

\[
\text{SD} \left( \text{Ideal}_{\text{Sim}}(\epsilon, \delta, q^*, x), \text{Real}_{V^*,P}(\epsilon, \delta, q^*, x, w) \right) \leq \epsilon_H + q^* \epsilon_{LS}.
\]

Fix the oracle proof \( \pi_H \) for \( V_H \). (It suffices to fix the bits determined during the simulation of \( \text{Sim}_H \), but as the other bits of \( \pi_H \) are never queried, this is the same as conditioning on the entire proof.) Let \( L_1, \ldots, L_{q'}, q' \leq q^* \) denote the locks that \( V^* \) queried. Let \( \text{Real}_{LS} \) denote the random variable describing the queries of \( V^* \) to the \( L_1, \ldots, L_{q'} \) and their answers. Similarly, let \( \text{Ideal}_{LS} \) denote the random variable describing the queries of \( V^* \) to \( L_1, \ldots, L_{q'} \), and the simulated answers provided by
Sim_{LS,1}, \ldots, Sim_{LS,q'}$, where Sim_{LS,i} denotes the emulation of Sim_{LS} for L_i. (Notice that both random variables are conditioned on the proof π_H, so for every lock whose value was determined during the simulation, the value provided to Sim_{LS} by Sim is consisted with the real-world value locked in the lock.)

Assume towards contradiction that \( \Delta (\text{Real}_{LS}, \text{Ideal}_{LS}) > q^* \epsilon_{LS} \geq q' \epsilon_{LS} \), i.e., there exists a distinguisher D_{mult} whose advantage in distinguishing Real_{LS} from Ideal_{LS} is more than \( q' \epsilon_{LS} \). We define a sequence of hybrids \( H_0, \ldots, H_{q'} \), where in \( H_i \), the queries to, and answers of, \( L_1, \ldots, L_i \) are as they appear in Real_{LS}; and the queries to, and answers of, \( L_{i+1}, \ldots, L_{q'} \) are as they appear in Ideal_{LS}. Then \( H_0 = \text{Ideal}_{LS}, H_{q'} = \text{Real}_{LS} \), and there exists an \( 1 \leq i \leq q' \) such that \(|H_i - H_{i-1}| > \epsilon_{LS}\).

Using an averaging argument, we can fix the locks \( L_1, \ldots, L_{i-1}, L_{i+1}, \ldots, L_{q'} \), and the randomness used for the emulations of Sim_{LS,1}, \ldots, Sim_{LS,q'}, while preserving the statistical distance. Then the advantage of the following distinguisher D_{LS} in distinguishing the real-world, and simulated, interaction with \( L_i \) is more than \( \epsilon_{LS} \), which contradicts the equivocation property. The fixed queries to, and answers of, \( L_1, \ldots, L_{i-1}, L_{i+1}, \ldots, L_{q'} \), as well as the randomness of Sim_{LS,1}, \ldots, Sim_{LS,q'}, are hardwired into D_{LS}. Given (either a real-world or a simulated) transcript \( T \) of the receiver \( R \) with his lock, D_{LS} uses \( T \) (and the hard-wired values) to construct a communication transcript \( T_{mult} \) of \( V^* \) with \( L_1, \ldots, L_{q'} \), runs D_{mult} on \( T_{mult} \), and answers as D_{mult} does. Then if \( T \) is the real-world transcript then \( T_{mult} \equiv H_i \), otherwise \( T_{mult} \equiv H_{i-1} \).

**Remark 5.24.** The proof of Lemma 5.4.9 crucially relied on the stronger honest-verifier guarantee of the underlying HVZKPCPP (see Section 5.4.2 for a more detailed description of this property). More specifically, the stronger property guarantees that (for a “not too large” \( q' \)) the view of a verifier emulating \( q' \) random and independent basic verification steps of the honest HVZKPCPP verifier \( V_H \) can be simulated with only \( q' \) queries.

This property is needed due to the method used to prove the zero-knowledge property: the answer to a single bit of a MIX is simulated by generating the entire simulated entry. This requires simulating an entire random basic verification step of \( V_H \), which consists of 3 queries, by emulating the simulator Sim_{H} of the HVZKPCPP. During the emulation, Sim_{H} may send TTP queries to \( x \), which are answered by the TTP of Sim. However, since Sim cannot make more queries than the verifier \( V^* \), the emulation of each basic verification step of \( V_H \) must incur at most a single TTP query. We note that this stronger property is not required for the construction of ZKPCPs [KPT97], since in that case the entire input \( x \) is given to the simulator.

Applying Construction 5.23 to the HVZKPCPP of Corollary 5.20, and the locking scheme of Corollary B.6, give Theorem 5.22.

**Proof of Theorem 5.22.** We instantiate Construction 5.23 with:

- The HVZKPCPP \( (P_H, V_H) \) of Corollary 5.20, with parameters \( \epsilon, \delta, q^* \).
- The locking scheme \((S, R)\) of Corollary B.6 for the message space \(\{0, 1\}\), with security parameter \(\sigma = \max\{\log \frac{1}{\epsilon}, 2\}\). For this choice of parameters, the locking scheme is \((1 - \epsilon)\)-binding, and \((q^*, \frac{1}{q^*} \cdot \frac{1}{\epsilon})\)-equivocal (this follows from the strong equivocation property), with locks of size \(\text{poly}(q^*, \log \frac{1}{\epsilon})\), and keys of size \(\text{polylog}(q^*, \frac{1}{\epsilon})\). Moreover, to decommit the lock \(R\) reads \(\text{polylog}(q^*, \frac{1}{\epsilon})\) bits from the lock.

- The locking scheme \((S_r, R_r)\) of Remark B.7 for the message space \(\{0, 1\}^r\), \(r = O(\log t(n) + \log q^* + \log \log \frac{1}{\epsilon} + \log \frac{1}{\delta})\), with security parameter \(\sigma = \max\{\log \frac{1}{\epsilon}, 2\}\). (More specifically, \(r\) is taken to be the number of random bits that \(V_H\) uses in a single basic verification step, see Corollary 5.20.) For this choice of parameters, the locking scheme is \((1 - \epsilon)\)-binding, and \((q^*, \frac{1}{q^*} \cdot \frac{1}{\epsilon})\)-equivocal, with locks of size \(\text{poly}(q^*, \log \frac{1}{\epsilon}, \log t(n), \log \frac{1}{\delta})\), and keys of size \(\text{poly}(\log q^*, \log \frac{1}{\epsilon}, \log t(n), \log \frac{1}{\delta})\). Moreover, to decommit the lock \(R\) reads \(\text{polylog}(t(n), q^*, \frac{1}{\epsilon}, \frac{1}{\delta})\) bits from the lock.

By Corollary 5.20, there exists a constant \(c > 0\) such that for every input \(x\) that is \(\delta\)-far from \(L_R\), and every proof oracle \(\pi_H\) for \(V_H\), every basic verification step of \(V_H^{x, \pi_H}\) succeeds with probability at most \(1 - \frac{\delta}{\log^c \frac{1}{\epsilon}}\). We set \(t = 4\frac{1}{\delta} \log^{c+1} \frac{1}{\epsilon}\) in Construction 5.23 (i.e. \(V\) performs \(4\frac{1}{\delta} \log^{c+1} \frac{1}{\epsilon}\) basic verification steps).

**Complexity.** Let \((x, w) \in \mathcal{R}, \text{ and } |x| = n\). Then every proof \(\pi_H \in P_H(1^{|\epsilon|}, 1^{|\delta|}, 1^{q^*}, x, w)\) has length \(|\pi_H| = \text{poly}\left(t(n), q^*, \log \frac{1}{\epsilon}, \frac{1}{\delta}\right)\). Therefore, every proof \(\pi \in P(1^{|\epsilon|}, 1^{|\delta|}, 1^{q^*}, x, w)\) has size \(\text{poly}\left(t(n), q^*, \log \frac{1}{\epsilon}, \frac{1}{\delta}\right)\) (since every basic verification step of \(V_H\) uses only \(r\) random coins.) Moreover, the honest verifier \(V\) performs \(\text{poly}(\frac{1}{\epsilon}, \log \frac{1}{\delta})\) iterations, in each reading a single \(\text{MIX}\) entry (consisting of a constant number of keys, of size \(\text{polylog}(t(n), q^*, \frac{1}{\epsilon}, \frac{1}{\delta})\) each), and unlocking at most 4 locks, which requires reading at most \(\text{polylog}(t(n), q^*, \frac{1}{\epsilon}, \frac{1}{\delta})\) proof bits. Therefore, the total query complexity is \(\text{poly}(\log t(n), \log q^*, \log \frac{1}{\epsilon}, \log \frac{1}{\delta})\).

**Perfect completeness.** Follows directly from the perfect completeness of the underly- ing systems.

**Soundness.** Lemma 5.4.8 guarantees that for every \(x\) that is \(\delta\)-far from \(L_R\), and for every proof oracle \(\pi^*\), a single basic verification step of \(V_H^{x, \pi^*}\) succeeds with probability at most \(1 - \frac{\delta}{\log^c \frac{1}{\epsilon}} (1 - 3 \cdot 2^{-\sigma}) \leq 1 - \frac{\delta}{\log^c \frac{1}{\epsilon}} (1 - 3 \cdot 2^{-2}) \leq 1 - \frac{\delta}{4 \log^c \frac{1}{\epsilon}}\). \(V\) performs \(t = 4\frac{1}{\delta} \log^{c+1} \frac{1}{\epsilon}\) independent iterations, so

\[
\Pr \left[V_H^{x, \pi^*}(|\ell|, 1^{|\delta|}, q^*, |x|) = \text{acc} \right] \leq \left(1 - \frac{\delta}{4 \log^c \frac{1}{\epsilon}}\right)^{4\frac{1}{\delta} \log^{c+1} \frac{1}{\epsilon}} \leq \epsilon.
\]

**Zero-knowledge.** By Lemma 5.4.9 the statistical distance of the simulation is at most \(\epsilon_H + q^* \epsilon_{LS}\), where for our choice of parameters, \(\epsilon_H = \frac{\epsilon}{2}\), and \(\epsilon_{LS} = \frac{1}{2} \cdot \frac{\epsilon}{2}\). \(\square\)

**The adaptivity of the honest verifier.** Unlike our HVZKPCPP systems (Section 5.4.1), the verifier \(V\) of Theorem 5.22 is inherently adaptive. Indeed, to commit
the locks, $V$ must first retrieve the corresponding keys from the appropriate MIX entry (the binding property of the locking schemes means that without the keys it would be practically impossible to retrieve the locked value). However, all iterations of the verification procedure may be executed in parallel (namely, making all MIX-queries simultaneously, then unlocking all the locks simultaneously etc.). Thus, the adaptivity of $V$ (i.e., the number of query rounds it makes) is $k_{lock} + 1$, where $k_{lock}$ is the adaptivity of the locking scheme receiver $R$. Concretely, if Construction 5.23 employs the locking schemes of Corollary B.6, and Remark B.7, then $V$ has adaptivity 3, namely it makes only 3 rounds of queries to the proof.
Chapter 6

Locally Testable Codes with Zero-knowledge

In this chapter we introduce the notion, and construct the first example of, locally testable codes (LTCs) with zero-knowledge properties.

6.1 Introduction

An LTC is a code with a randomized tester algorithm that has oracle access to a purported codeword $w$. The tester reads few symbols of $w$, and based on this “local view” decides whether $w$ is in the code. The tester should always accept codewords (this is called completeness), and it should reject (with high probability) words that are far from the code (this is called soundness).\(^1\)

Zero-knowledge LTCs (ZKLTCs) are LTCs equipped with a randomized encoding function which ensures that the encoded data remain hidden against any (possibly malicious) tester who observes a bounded number of codeword symbols. Such objects naturally arise in cryptographic settings, such as distributed storage, which require efficient verification while preserving the secrecy of the distributed data. For example, suppose that a user wishes to reliably store her data by distributing it among a large number of potentially unreliable, or even malicious, servers. To ensure the integrity of the data, the user can apply a good error correcting code before distributing the data. However, traditional codes lack two useful properties. First, they do not admit an efficient procedure for checking whether the stored data has been tampered with to an extent which may compromise its integrity. Second, malicious servers may gather a significant amount of information about the data, which is problematic for sensitive data. Using ZKLTCs to encode the data solves both problems. Note that the standard notion of testing is insufficient in this scenario, since malicious servers may adaptively

\(^1\)There are two kinds of LTCs: weak and strong (see Section 2.3.2). In a weak LTC the tester is only guaranteed to reject words that are far from the code, whereas in a strong LTC the rejection probability is proportional to the distance from the code.
determine their answers after seeing the queries of the user. This calls for a notion of stability - a stronger form of testability which we introduce and study. A more detailed discussion of the stability and zero-knowledge properties of LTCs follows.

**Stable LTCs**

As noted above, the standard testability property of LTCs is insufficient when an adversary can adaptively control few symbols of a purported codeword, and determine the value of these symbols after seeing (and possibly based on) the queries of the tester. We construct LTCs resisting such tampering attacks, which we call stable LTCs. More specifically, in a stable LTC $C$, the tester $D$ tests whether a purported codeword $w$ is in $C$, in the presence of an adversary $A$ who selects in advance a subset $T$ of at most $\tau$ symbols of $w$ that it can adaptively modify after seeing the queries of $D$. (In particular, the answers to the queries of $D$ are according to $w$, and the symbols that $A$ chose to alter.) The goal of $A$ is to cause $D$ to reject $w$ when $w \in C$, and to accept $w$ when $w$ is far from $C$. Informally, a code $C$ is a $(\tau, \epsilon, \delta)$-stable LTC with respect to the tester $D$, if $D$ correctly determines (except with probability at most $\epsilon$) whether a purported codeword is in the code, or $\delta$-far from it, even in the presence of an adversary $A$ as above.

**ZKLTCs**

We initiate the study of ZKLTCs, that combine the notions of ZK codes, and local testability. Informally, a code $C$, associated with a randomized encoding function $\text{Enc}_C$, is $\tau$-ZK if any adversary querying $\tau' \leq \tau$ symbols of a codeword $c = \text{Enc}_C(x)$ learns no information about the message $x$. $\text{Enc}_C$ is $(\tau, \epsilon)$-ZK if such adversaries achieve advantage at most $\epsilon$ in distinguishing between two messages. A ZKLTC is an LTC, associated with a ZK encoding function. We note that for cryptographic applications of ZKLTCs, we will typically be interested in codes with $\tau = \Omega(\hat{n})$, where $\hat{n}$ denotes the codeword length, and membership in the code can be tested by probing only a sublinear number of symbols. Such ZKLTCs will be used in Chapter 7 to obtain efficient cryptographic protocols tolerating a constant fraction of corrupted parties.

**Related Work**

Zero-knowledge codes were considered by Decatur et al. [DGR97, DGR99], who explicitly construct a family of linear zero-knowledge encoding functions with constant rate, together with a family of decoding algorithms that can correct a constant fraction of errors. More specifically, [DGR97, DGR99] describe a “good” matrix property, such that any matrix possessing the property gives rise to a zero-knowledge encoding. (Given such a matrix, a message is encoded by applying the matrix to the message padded by randomness.) Decatur et al. then use a code of Justesen [Jus72] to construct a matrix possessing this “good” property. Feldman et al. [FMSS04] show that any linear code
with sufficiently good parameters gives rise to a matrix with the “good” property, where the matrix is constructed probabilistically.

We note that ZKPCPs, which can be thought of as the PCP analogue of ZKLTCs, do not seem to imply good ZKLTCs, or any of the cryptographic applications of ZKLTCs described in Chapter 7, and their construction is considerably more involved.

6.1.1 Our Results and Techniques

Our contributions in this chapter are twofold. First, we construct efficient ZKLTCs by applying and extending previous general transformations from codes to ZK-codes. More specifically, using a probabilistic transformation from linear codes to ZK-codes due to Feldman et al. [FMSS04], we obtain constant rate ZKLTCs with sublinear query complexity, and a constant fractional ZK parameter. We also give a fully explicit general transformation from linear codes to ZK-codes, as well as a non-explicit transformation of arbitrary codes to ZK-codes. Second, we construct good LTCs and ZKLTCs with the additional stability guarantee. (The usefulness of stable ZKLTCs is demonstrated in Chapter 7, where we apply such codes towards the design of efficient protocols for verifiable secret sharing, and distributed coin-flipping.) Our results are described in more detail below.

Constructing ZKLTCs

Feldman et al. [FMSS04, Section 3.3] gave a probabilistic construction of ZK-codes with good parameters from linear codes. We slightly generalize their result, and formulate (Section 6.4) a common property, valid for both large and small fields, for the existence of ZK-codes over the field. We use this condition to construct (Section 6.4.2) asymptotically good strong LTCs with a constant fractional ZK parameter \( \tau \).

The main drawback of the result of [FMSS04], and the ZKLTCs obtained from it, is that the ZK-encoding is constructed probabilistically. Therefore, applications which employ such encodings may need to rely on a trusted setup in which the generator matrix \( G' \) describing the encoding \( \text{Enc}_C \) is picked at random. To avoid such a setup, we present (Section 6.2) a fully explicit transformation of linear codes to ZK-codes. The main price we pay is that the zero-knowledge property of the encoding becomes statistical rather than perfect. The idea is to first encode the message by a randomized encoding which has the property that the statistical advantage of any linear function \( L : F^k \rightarrow F \) in distinguishing between encodings of two different messages is small, and then apply the given linear encoding to the result.

While all of the LTCs we will rely on in this thesis are linear, there could potentially be non-linear LTCs with better, or incomparable, parameters (an example being the Long Code used in efficient PCP constructions, cf. [Din06]). We observe that for a general (not necessarily linear) code \( C \), a completely random choice of an encoding function has good ZK parameters with high probability. Using this observation, we
describe (Section 6.3) a transformation from general codes to ZK codes. Note that this construction is less explicit than the transformation of Section 6.2 in that the encoding function is inefficient.

**Stable LTCs**

We construct (Section 6.5) a good family of stable linear LTCs, which we then transform into stable ZKLTCs. (These stable ZKLTCs are used in Chapter 7 to design efficient protocols for verifiable secret sharing and distributed coin-flipping.)

Our construction of stable LTCs is based on the testability of the tensor products of codes studied in [DSW06, BV08, BV09, BS04, Vid12]. Intuitively, the codewords in such codes are multidimensional tensors, whose restriction to every axis-parallel line belongs to the fixed base code. Given a purported codeword, the tester picks a random 2-dimensional hyperplane, and reads all symbols of the purported codeword indexed by the hyperplane. Based on this “local view”, the tester either accepts or rejects. Such codes are linear strong LTCs [BS04, Vid12].

Our contribution in this aspect is in showing that tensor products of codes can yield *stable* LTCs. Concretely, we show that if a base code is an efficiently decodable linear code, then its tensor products are stable LTCs with respect to a new tester.

### 6.1.2 Chapter Organization

We describe our explicit construction of ZK-codes from linear codes, our construction of ZK-codes from general codes, and our probabilistic construction of ZK-codes from linear codes, in Sections 6.2, 6.3, and 6.4, respectively. In Section 6.5 we define and construct stable ZKLTCs.

### 6.2 Explicit ZK Codes

In this section we present a fully explicit transformation of linear codes to ZK-codes. The high-level idea is to first encode the message using a randomized encoding with the property that any linear function \( L : \mathbb{F}^n \to \mathbb{F} \) has only a small statistical advantage in distinguishing between encodings of two different messages, and then apply the given linear encoding to the result. This approach yields the following theorem.

**Theorem 6.1** (Linear codes to ZK-codes, explicitly). Let \( n \in \mathbb{N} \) be a dimension parameter, \( \hat{n} \in \mathbb{N} \) be a codeword length parameter, and \( \mathbb{F} = \mathbb{F}_2 \) denote the binary field. For any linear code \( C \subseteq \mathbb{F}^n \) of dimension \( n \), there exists a randomized encoding function \( \text{Enc}_C : \mathbb{F}^n \to \mathbb{F}^{\hat{n}} \), whose image is contained in \( C \), such that \( n = \Omega(\hat{n}) \) and \( \text{Enc}_C \) is \((\tau, \epsilon)-ZK\) for \( \tau = \Omega(\hat{n}) \), and \( \epsilon = 2^{-\Omega(\hat{n})} \). Furthermore, a randomized circuit for computing \( \text{Enc}_C \) (resp., a circuit for inverting \( \text{Enc}_C \)) can be computed in deterministic polynomial time given any generator matrix for \( C \).
We first formalize the notion of randomized encodings that “fool” linear functions that were described above.

**Definition 6.2.1 (Linear-secure encoding).** Let $F$ be a finite field, $n, \hat{n} \in \mathbb{N}$, and $\epsilon > 0$ be a statistical distance parameter. A randomized encoding $\text{Enc}: F^n \rightarrow F^{\hat{n}}$ is $\epsilon$-secure against linear distinguishers if for every $x, x' \in F^n$, and every linear function $L: F^{\hat{n}} \rightarrow F$,

$$\text{SD} (L(\text{Enc}(x)), L(\text{Enc}(x'))) \leq \epsilon.$$ 

We will need the following result, which is implicit in [IKOS09].

**Lemma 6.2.2 (Constant-rate linear-secure encoding).** Let $n \in \mathbb{N}$ be an input length parameter, and $F = F_2$ denote the binary field. There exists a probabilistic polynomial-time algorithm computing a randomized encoding $\text{Enc}: F^n \rightarrow F^{\hat{n}(n)}$ such that $\hat{n}(n) = O(n)$, and $\text{Enc}$ is $\epsilon$-secure against linear distinguishers for $\epsilon(n') = 2^{-\Omega(n)}$. Furthermore, there exists a polynomial-time algorithm $\text{Dec}$ such that for all $x \in F^n$, $\Pr[\text{Dec}(\text{Enc}(x)) = x] = 1$.

The next lemma states that applying a linear encoding function on top of a linear-secure encoding with sufficiently small $\epsilon$ yields a good $(\tau, \epsilon)$-ZK code.

**Lemma 6.2.3.** Let $n \in \mathbb{N}$ be a dimension parameter, $\hat{n} \in \mathbb{N}$ be a codeword length parameter, and $F$ be a finite field. Let $C \subseteq F^{\hat{n}}$ be a linear code with $\dim (C) = n$, and generator matrix $G$. Let $\text{Enc}: F^n \rightarrow F^{\hat{n}}$ be a randomized encoding that is $\epsilon$-secure against linear distinguishers. Then for every $\tau \in \mathbb{N}$ the randomized encoding $\text{Enc}_C: F^n \rightarrow F^{\hat{n}}$ defined by $\text{Enc}_C(x) = G \circ \text{Enc}(x)$ is $(\tau, 2^{\tau/2} \cdot \epsilon)$-ZK.

**Proof.** Suppose towards contradiction that there are $x, x' \in F^n$, and a subset $I \subseteq [\hat{n}]$ of size at most $\tau$, such that $\text{SD} (\text{Enc}_C(x)|_I, \text{Enc}_C(x')|_I) > 2^{\tau/2} \cdot \epsilon$. By Lemma 2.1.1, there exists an $\alpha \in F^\tau$ such that $\text{SD} (\alpha^T \cdot \text{Enc}_C(x)|_I, \alpha^T \cdot \text{Enc}_C(x')|_I) > \epsilon$. Therefore, by the definition of $\text{Enc}_C$ there exists a $\beta \in F^{\hat{n}}$ such that $\text{SD}(\beta^T \cdot G \circ \text{Enc}(x), \beta^T \cdot G \circ \text{Enc}(x')) > \epsilon$, contradicting the $\epsilon$-security of $\text{Enc}$.

Theorem 6.1 now follows by applying Lemma 6.2.3 to the efficient linear-secure encoding of Lemma 6.2.2.

**Proof of Theorem 6.1.** Let $\text{Enc}: F^n \rightarrow F^{\hat{n}(n)}$ denote the randomized encoding of Lemma 6.2.2, where $\hat{n}(n) = \alpha n$, and $\epsilon(n) = 2^{-\beta n}$ for some constants $\alpha, \beta > 0$. Let $\tau(n) = \frac{2n}{\beta}$, then the following holds for every $n$. The code $C': F^n \rightarrow F^{\hat{n}}$ defined in Lemma 6.2.3, with encoding function $G'$, maps $\alpha n$-symbol messages to $\hat{n}(n) = O(n)$-symbol codewords, with $(\frac{2n}{\beta}, 2^{-\frac{\beta n}{2}})$-ZK. Moreover, the circuit computing $G \circ \text{Enc}$ can be computed efficiently given the generator matrix $G$. \[\square\]
6.3 General ZK Codes

In this section we show a transformation from any (not necessarily linear) code $C$ into a ZK code. The transformation is probabilistic, namely with high probability the encoding function will have good ZK parameters. Informally, this holds because a completely random choice of an encoding function for $C$ will (with high probability) induce a ZK code. More specifically, we will prove the following.

**Theorem 6.2** (General codes to ZK-codes, non-explicitly). Let $n \in \mathbb{N}$ be an input length parameter, $\hat{n} \in \mathbb{N}$ be a codeword length parameter such that $\hat{n} = o(2^n)$, and $\mathbb{F} = \mathbb{F}_2$ denote the binary field. For any code $C \subseteq \mathbb{F}^n$ of size $2^n$, there exists a randomized encoding function $\text{Enc}_C : \mathbb{F}^{n'} \rightarrow \mathbb{F}^\hat{n}$, whose image is contained in $C$, such that $n' = \Omega(n)$, and $\text{Enc}_C$ is $(\tau, \epsilon)$-ZK for $\tau = \Omega(n)$, and $\epsilon = 2^{-\Omega(n)}$.

We note that the construction of Theorem 6.2 is less explicit than the one given by Theorem 6.3 below since the encoding function is inefficient.

The proof of Theorem 6.2 uses the probabilistic method, along with the XOR lemma (Lemma 2.1.1). Intuitively, a code $C \subseteq \{0,1\}^{\hat{n}}$ of size $2^n$ can be associated with a 1:1 mapping $\text{Enc} : \{0,1\}^n \rightarrow C$. For an $n' \leq n$, a randomized encoding function $\text{Enc}_C : \{0,1\}^{n'} \rightarrow \{0,1\}^{\hat{n}}$ can be defined as follows. Partition $\{0,1\}^n$ into $2^{n'}$ subsets of size $2^{n-n'}$, where each subset is associated with an element of $\{0,1\}^{n'}$. To encode an $x \in \{0,1\}^{n'}$, first randomly map $x$ to an element $y$ in the corresponding subset of $\{0,1\}^n$, and then compute $\text{Enc}(y)$. Lemma 2.1.1 states that to prove that $\text{Enc}_C$ is a $\tau$-ZK encoding, it suffices to bound the statistical distances between the XOR of at most $\tau$ coordinates of $\text{Enc}_C(x), \text{Enc}_C(x')$, for arbitrary $x \neq x' \in \{0,1\}^{n'}$. Using the probabilistic method, we can show that this statistical distance is negligible.

**Proof of Theorem 6.2.** Given a code $C \subseteq \{0,1\}^{\hat{n}}$ of size $2^n$, we define an arbitrary ordering $\{c_1, \ldots, c_{2^n}\}$ on $C$, and let $\text{Enc} : \{0,1\}^n \rightarrow \{0,1\}^{\hat{n}}$ map $x \in \{0,1\}^n$ to $c_x$ (here, $x$ denotes both the vector in $\{0,1\}^n$, and the value $i$ such that $x$ is the binary representation of $i$). Let $n' = cn, \tau = c_{\tau}n$ where $c, c_{\tau} \in (0,1)$ are constants (whose value will be set later). Denote

$$\mathcal{F} := \{F = \text{XOR} \circ \text{Enc} : \text{XOR is over } \leq \tau \text{ coordinates in the output of } \text{Enc}\}$$

then $|\mathcal{F}| \leq 2^{\tau \cdot \binom{n}{\tau}} \leq (2\hat{n})^\tau$. Let $P = \{A_1, \ldots, A_m\}$ denote a random partition of $\{0,1\}^n$ into $m := 2^{n'}$ equal-sized subsets of size $s := 2^{n-n'}$. (Notice that each $A_i$ consists of $s$ elements chosen independently at random without replacements.)

Set some $F \in \mathcal{F}$, and let $P = \{A_1, \ldots, A_m\}$ be a random partition of $\{0,1\}^n$ as defined above. Let $U_n$ denote the uniform distribution over $\{0,1\}^n$, and $\mathcal{D}_i$ denote the uniform distribution over $A_i$. We show first that for every $i \in [m], \Delta_F := |\Pr[F(U_n) = 1] - \Pr[F(\mathcal{D}_i) = 1]| < 2^{-n'}$ except with negligible probability. Denote $p_F := \Pr[F(U_n) = 1]$, and we assume that $1 - p_F > 2^{-n'}$. (This is without loss
of generality, since if $p_F \leq 1 - 2^{-n'}$ then even if all the elements on which $F$ outputs 0 are in $A_i$, still $\Delta_F \leq 1 - 2^{-n'} - \frac{2^{n-n'}2^{n'}}{2n-n'} = 2^{2n-n} - 2^{-n'} = 2^{-n'}2^{3n-n-1} \leq 2^{-n'}$.

Define an ordering $x_1, \ldots, x_{2^n}$ on $\{0, 1\}^n$ in which for every $i$, all elements in $A_i$ appear before all elements in $A_{i+1}$. For every $x_j \in \{0, 1\}^n$, we define a random variable $X_j = F(x_j)$. Then $X_j \in \{0, 1\}$, and $\Pr_{j \in \{0, 1\}^n}[X_j = 1] = p_F$.

We claim that for every $1 \leq i \leq m$, except with negligible probability, $\Pr_{x \in \mathcal{A}_i}[F(x) = 1] = \frac{\sum_{j \in \mathcal{A}_i} X_j}{s}$ is “close” to $p_F$. If the indicators $X_j$ were independent, this would follow directly from the Chernoff inequality, but in this case the indicators are dependent since the elements in $A_i$ were chosen without replacement. Therefore, we use a result of Hoeffding [Hoe63, Section 6] who showed that the Chernoff bound holds even if elements are chosen without replacement. We say that $F$ is bad for $A_i$, if $\left| \frac{\sum_{j \in \mathcal{A}_i} X_j}{s} - p_F \right| > 2^{-n'}$ (otherwise we say that $F$ is good). Then for every $1 \leq i \leq m$:

$$\Pr[F \text{ is bad for } A_i] = \Pr\left[\left| \frac{\sum_{j \in \mathcal{A}_i} X_j}{s} - p_F \right| > 2^{-n'} \right]$$

By [Hoe63], this expression can be upper-bounded by

$$2 \cdot e^{-2s(2^{-n'})^2} = 2 \cdot e^{-2^{-2n-n'-2n'}} < 2^{-2n-3n'}$$

Let $p := 2^{-2n-3n'}$. Using the union bound, except with probability at most $m \cdot p$, $F$ is good for all $A_i$, in which case we say that $F$ is good for the partition $P = \{A_1, \ldots, A_m\}$.

To sum up, a fixed $F$ is bad for a random partition $P$ with probability at most $m \cdot p$, so by the union bound, the probability that at least one $F \in \mathcal{F}$ is bad for a random partition is at most $|\mathcal{F}| \cdot m \cdot p \leq (2\hat{n})^7 \cdot m \cdot p$. For a large enough $n$, there exist constants $c, c_\tau$ such that $(2\hat{n})^7 \cdot m \cdot p < 1$, meaning that there exists a partition for which all $F \in \mathcal{F}$ are good.

Let $P$ be such a partition. Define $\text{Enc}_C$ by $\text{Enc}_C(x) = \text{Enc}(y)$ for $y \in R A_x$. Then for every pair of messages $x \neq x' \in \{0, 1\}^n$, and every $\alpha \in \{0, 1\}^n$ such that $|\{i : \alpha_i = 1\}| \leq \tau$, we can upper-bound the statistical distance $\text{SD}\left(\alpha^T \cdot \text{Enc}_C(x), \alpha^T \cdot \text{Enc}_C(x')\right)$ by

$$\text{SD}\left(\alpha^T \cdot \text{Enc}_C(x), \alpha^T \cdot \text{Enc}_C(U_n)\right) + \text{SD}\left(\alpha^T \cdot \text{Enc}_C(U_n), \alpha^T \cdot \text{Enc}_C(x')\right) \leq 2 \cdot 2^{-n'}.$$ 

Therefore, Lemma 2.1.1 guarantees that for every $I \subset \mathcal{A}$, $|I| \leq \tau$,

$$\text{SD}\left(\text{Enc}_C(x) | I, \text{Enc}_C(x') | I\right) \leq 2^\tau \cdot 2^{-n'}$$

which, for an appropriate choice of $c, c_\tau$ (for which it also holds that $(2\hat{n})^7 \cdot m \cdot p < 1$ for a large enough $n$) is at most $2^{-\Omega(n)}$. \qed
6.4 Probabilistic Construction of ZK Codes from Linear Codes

In this section, we describe a probabilistic construction of ZK codes from linear codes. More specifically, we generalize a result of Feldman et al. [FMSS04, Section 3.3], who gave a probabilistic construction of ZK-codes (with good parameters) from linear codes, formulating a common requirement which is valid for both large and small fields:

**Theorem 6.3** (Linear codes to ZK-codes, probabilistically). Let $n \in \mathbb{N}$ be a dimension parameter, $\hat{n}$ be a codeword length parameter, $n' \in \mathbb{N}$ such that $n' < n$, and $F$ be a finite field. Let $C \subseteq F^{\hat{n}}$ be a linear code of dimension $n$. Assume that for some $0 < \tau < \hat{n}$ it holds that
\[
\log (|F|) \cdot (n - n' - \tau) > \hat{n} \cdot H \left( \frac{\tau}{\hat{n}} \right) - \tau.
\]
Then there exists a generator matrix $G'$ for $C$ such that $C$, under the randomized encoding defined by $\text{Enc}_C(x; r) = G' \cdot (x; r)$, is a $\tau$-ZK code for messages in $F^{n'}$. Furthermore, such a $G'$ can be constructed in probabilistic polynomial time (except with negligible failure probability) given $n'$, $\tau$, and any generator matrix for $C$.

Then, we apply Theorem 6.3 to known LTCs, obtaining the following construction of ZKLTCs, where rate($\text{Enc}_C$) denotes the rate of the randomized encoding function $\text{Enc}_C$ (i.e., the ratio between the length of the original message and the length of the encoded message).

**Corollary 6.4** (Probabilistic construction of ZKLTCs). For every $\gamma \in (0, 1)$, there exists a finite field $F$ of constant size (depending only on $\gamma$) such that the following holds for every $\beta > 0$. For every $n' \in \mathbb{N}$ there exists a dimension parameter $n$, and a codeword length parameter $\hat{n}$, such that there exists a linear code $C \subseteq F^{\hat{n}}$ of dimension $n$, and a generator matrix $G'$ for $C$, such that:

- $C$ is an $(\hat{n}^\beta, \frac{1}{2})$-strong LTC with relative distance $\delta(C) = \Omega_\beta(1)$.
- The randomized encoding $\text{Enc}_C(x; r) = G' \cdot (x; r)$, where $x \in F^{n'}$, and $r \in F^{n-n'}$, has rate $\text{rate}(\text{Enc}_C) = n'/\hat{n} = \Omega_\beta(1)$, and is $\tau$-ZK for $\tau = \lfloor \gamma \hat{n} \rfloor$.

Furthermore, given $n'$, such a $G'$ can be constructed in probabilistic poly($n'$) time (except with negligible failure probability).

6.4.1 A Proof of Theorem 6.3

In this section we prove Theorem 6.3. We first discuss the relation of the entropy function and the generator matrix of a code.

**Lemma 6.4.1.** Let $n \in \mathbb{N}$ be an input length parameter, $\hat{n} \in \mathbb{N}$ be a codeword length parameter, and $\tau \in \mathbb{N}$ be a zero-knowledge parameter. Let $n_1, n_2 \in \mathbb{N}^+$ such that
\( n = n_1 + n_2 \), and \( G \in \mathbb{F}^{n \times n} \) be a generator matrix for the code \( C \). Assume that any non-trivial linear combination of at most \( \tau \) rows of \( G \) does not result in a \( w \in \mathbb{F}^n \) such that \( w|_{[n]|n_1} = 0^{n_2} \). Then \( C \), when encoding is done by \( G \), is a \( \tau \)-ZK code for messages of length \( n_1 \).

**Proof.** Let \( \text{Enc}_C \) denote the encoding algorithm which operates as follows: on input a message \( m \in \mathbb{F}^{n_1} \), we append a random string \( r \in_R \mathbb{F}^{n_2} \), and output the codeword \( G \cdot (m, r) \). We claim that for any messages \( x_1, x_2 \in \mathbb{F}^{n_1} \), subset \( I \subseteq \{\hat{n}\} \) of size at most \( \tau \), and codeword \( c \in C \),

\[
\Pr_r[\text{Enc}_C(x_1)|I = c|z] = \Pr_r[\text{Enc}_C(x_2)|I = c|z],
\]

where the probability is taken over the randomness of \( \text{Enc}_C \).

Indeed, \( R = G|_{[X \times \{0, ..., n_1\}} \in \mathbb{F}^{[I] \times n_2} \) has full rank \( (|I|) \), so equations of the form \( R \cdot r = a \), where \( r \in \mathbb{F}^{n_2} \), have the same number of solutions for every \( a \in \mathbb{F}^{[I]} \).

In what follows, let \( \text{span}_r(G) \subseteq \mathbb{F}^n \) (for \( G \in \mathbb{F}^{n \times n} \)) denote a subset of \( \mathbb{F}^n \) that includes all strings that could be obtained as a linear combination of at most \( \tau \) rows of \( G \). In particular, all rows of \( G \), and \( 0^n \), belong to \( \text{span}_r(G) \).

**Lemma 6.4.2.** If \( G \in \mathbb{F}^{\hat{n} \times \hat{n}} \) then for every \( 0 < \tau < \hat{n} \)

\[
|\text{span}_r(G)| \leq 2^{\hat{n} \cdot \left(H\left(\frac{\tau}{\hat{n}}\right) + \frac{\tau}{\hat{n}} \cdot \log(|\mathbb{F}|-1)\right)}.
\]

**Proof.** We know that \( |\text{span}_r(G)| \leq 1 + \binom{\hat{n}}{\tau} \cdot (|\mathbb{F}|-1)^1 + \binom{\hat{n}}{\tau} \cdot (|\mathbb{F}|-1)^2 + \cdots + \binom{\hat{n}}{\tau} \cdot (|\mathbb{F}|-1)^\tau \).

This bound is equal to the number of strings in \( \mathbb{F}^\hat{n} \) of weight at most \( \tau \). Assume that we pick a random string \( X_1, X_2, ..., X_{\hat{n}} \) of weight at most \( \tau \) in \( \mathbb{F}^\hat{n} \), where \( X_i \) denotes the \( i \)th symbol of the string. Then, \( H(X_1, X_2, ..., X_{\hat{n}}) = \log \left( \sum_{i=0}^{\tau} \binom{\hat{n}}{i} \cdot (|\mathbb{F}|-1)^i \right) \).

On the other hand,

\[
H(X_1, X_2, ..., X_{\hat{n}}) \leq H(X_1) + \ldots + H(X_{\hat{n}}) \leq \hat{n} \cdot \left( H\left(\frac{\tau}{\hat{n}}\right) + \frac{\tau}{\hat{n}} \cdot \log(|\mathbb{F}|-1) \right)
\]

because

\[
H(X_i) \leq H\left(\frac{\tau}{\hat{n}}\right) + \frac{\tau}{\hat{n}} \cdot H\left(\frac{1}{|\mathbb{F}|-1}, \frac{1}{|\mathbb{F}|-1}, \ldots, \frac{1}{|\mathbb{F}|-1}\right) \leq H\left(\frac{\tau}{\hat{n}}\right) + \frac{\tau}{\hat{n}} \cdot \log(|\mathbb{F}|-1).^2
\]

Thus, \( \log \left( \sum_{i=0}^{\tau} \binom{\hat{n}}{i} \cdot (|\mathbb{F}|-1)^i \right) \leq \hat{n} \cdot \left( H\left(\frac{\tau}{\hat{n}}\right) + \frac{\tau}{\hat{n}} \cdot \log(|\mathbb{F}|-1) \right) \) and this implies the claim.

We are ready to prove Theorem 6.3. We first restate the theorem.

\(^2\)The inequality \( H(X_i) \leq H\left(\frac{\tau}{\hat{n}}\right) + \frac{\tau}{\hat{n}} \cdot H\left(\frac{1}{|\mathbb{F}|-1}, \frac{1}{|\mathbb{F}|-1}, \ldots, \frac{1}{|\mathbb{F}|-1}\right) \) holds since with probability at most \( \frac{\tau}{\hat{n}} \), the variable \( X_i \) takes non-zero value, and given that it is non-zero it has any value in \( \mathbb{F} \setminus \{0\} \) with equal probability.
Theorem (Theorem 6.3, restated). Let \( n \in \mathbb{N} \) be a dimension parameter, \( \hat{n} \) be a codeword length parameter, \( n' \in \mathbb{N} \) such that \( n' < n \), and \( \mathbb{F} \) be a finite field. Let \( C \subseteq \mathbb{F}^{\hat{n}} \) be a linear code of dimension \( n \). Assume that for some \( 0 < \tau < \hat{n} \) it holds that

\[
\log (|\mathbb{F}|) \cdot (n - n' - \tau) > \hat{n} \cdot H \left( \frac{\tau}{\hat{n}} \right) - \tau.
\]

Then there exists a generator matrix \( G' \) for \( C \) such that, under the randomized encoding defined by \( \text{Enc}_C(x; r) = G' \cdot (x; r) \), is a \( \tau \)-ZK code for messages in \( \mathbb{F}^{n'} \). Furthermore, such a \( G' \) can be constructed in probabilistic polynomial time (except with negligible failure probability) given \( n' \), \( \tau \), and any generator matrix for \( C \).

Proof of Theorem 6.3. Let \( G \in \mathbb{F}^{n \times \hat{n}} \) be the generating matrix for \( C \). By Lemma 6.4.2, it holds that \( |\text{span}_\tau(G)| \leq 2^{\hat{n} \cdot H \left( \frac{\tau}{\hat{n}} \right) + \frac{\tau}{\hat{n}} \log(|\mathbb{F}|-1)} \).

We argue that there exists a matrix \( M \in \mathbb{F}^{n \times \hat{n}} \) such that, letting its \( n \) rows be \( M_1, M_2, \ldots, M_n \), we have

- rank\((M) = n \) and
- For every \( i \in [n_1] \) it holds that \( M_i \notin \text{span}(M_1, M_2, \ldots, M_{i-1}) + \text{span}_\tau(G) \).

The matrix \( M \) can be obtained by greedy search row by row. Indeed, every row \( i \in [n_1] \) that we are looking for belongs to \( \mathbb{F}^n \), i.e., there are \( |\mathbb{F}|^n \) potential options, while the number of forbidden rows is at most

\[
|\text{span}(M_1, M_2, \ldots, M_{i-1}) + \text{span}_\tau(G)| \leq |\text{span}(M_1, M_2, \ldots, M_{i-1})| \cdot |\text{span}_\tau(G)| \leq |\mathbb{F}|^{n_1} \cdot 2^{\hat{n} \cdot (H \left( \frac{\tau}{\hat{n}} \right) + \frac{\tau}{\hat{n}} \log(|\mathbb{F}|-1))}
\]

where the last inequality follows from Lemma 6.4.2. For every \( i \in [n] \) such that \( n_1 + 1 \leq i \leq n \) we choose the row \( M_i \in \mathbb{F}^n \) to satisfy \( M_i \notin \text{span}(M_1, M_2, \ldots, M_{i-1}) \). Thus rank\((M) = n \). To show that such a choice is possible, we need to show that

\[
|\mathbb{F}|^n > |\mathbb{F}|^{n_1} \cdot 2^{\hat{n} \cdot (H \left( \frac{\tau}{\hat{n}} \right) + \frac{\tau}{\hat{n}} \log(|\mathbb{F}|-1))}
\]

By the theorem assumption we have

\[
\log(|\mathbb{F}|) \cdot (n - n_1 - \tau) > \hat{n} \cdot H \left( \frac{\tau}{\hat{n}} \right) - \tau
\]

so

\[
\log(|\mathbb{F}|) \cdot (n - n_1) > \hat{n} \cdot H \left( \frac{\tau}{\hat{n}} \right) + \tau \cdot (\log(|\mathbb{F}|) - 1)
\]

which implies that

\[
|\mathbb{F}|^{n-n_1} > 2^{\hat{n} \cdot H \left( \frac{\tau}{\hat{n}} \right) + \tau \cdot (\log(|\mathbb{F}|) - 1)} > 2^{\hat{n} \cdot H \left( \frac{\tau}{\hat{n}} \right) + \frac{\tau}{\hat{n}} \cdot (\log(|\mathbb{F}|) - 1)}
\]

166
namely
\[ |\mathbb{F}|^n > |\mathbb{F}|^{n_1} \cdot 2^{\hat{n} \cdot (\frac{H(\gamma)}{\gamma} + \frac{\gamma}{\pi} \log(|\mathbb{F}| - 1))}. \]

Thus, every time we do succeed in finding an additional row for \( M \). We note that probabilistically picking the matrix \( M \) will, with high probability, result in a matrix with the two required properties discussed above. Therefore, for the rest of the proof we assume that such a matrix \( M \) has been fixed.

Since the matrix \( M \) has full rank, there exists an inverted matrix \( M^{-1} \). Let
\[ G' = G \cdot M^{-1} \]
be a new generating matrix. Notice that this matrix generates the same code \( C \), i.e.,
\[ C = \{ G' \cdot x : x \in \mathbb{F}^n \} = \{ G \cdot x : x \in \mathbb{F}^n \}. \]

We argue now that any nontrivial linear combination of up to \( \tau \) rows of \( G' \) does not result in \( w \in \mathbb{F}^n \) such that \( w|_{[n]\setminus[n_1]} = 0^{n_2} \). Assume towards contradiction that this does not hold, then there exists a \( 0^n \neq y \in \mathbb{F}^{\hat{n}} \) such that \( |y| \leq \tau \), and \( y \cdot G' = w \) where \( w \in \mathbb{F}^n \) such that \( w|_{[n]\setminus[n_1]} = 0^{n_2} \). It follows that
\[ y \cdot G' \cdot M^{-1} = w \]
\[ y \cdot G \cdot M^{-1} \cdot M = w \cdot M \]
\[ y \cdot G = w \cdot M. \]

By definition, \( y \cdot G \in \text{span}_\tau(G) \). Moreover, \( w \cdot M \) is a linear combination of only the \( n_1 \) top rows of \( M \), and is equal to an element of \( \text{span}_\tau(G) \). Therefore, there exists an \( i \in [n_1] \) such that \( M_i \in \text{span}(M_1, M_2, \ldots, M_{i-1}) + \text{span}_\tau(G) \), contradicting the construction of \( M \).

This completes the proof since Lemma 6.4.1 guarantees that the code \( C \), when encoding is done by \( G' \), is a \( \tau\)-ZK code for messages of length \( n_1 \). \( \square \)

### 6.4.2 A Proof of Corollary 6.4

In this Section, we use Theorem 6.3 to prove Corollary 6.4. We first restate the corollary.

**Corollary** (Corollary 6.4, restated). **For every \( \gamma \in (0, 1) \), there exists a finite field \( \mathbb{F} \) of constant size (depending only on \( \gamma \)) such that the following holds for every \( \beta > 0 \). For every \( n' \in \mathbb{N} \), there exists a dimension parameter \( n \), and a codeword length parameter \( \hat{n} \), such that there exists a linear code \( C \subseteq \mathbb{F}^{\hat{n}} \) of dimension \( n \), and a generator matrix \( G' \) for \( C \), such that:**

- \( C \) is an \( (\hat{n}\beta, \frac{1}{2}) \)-strong LTC with relative distance \( \delta(C) = \Omega_\beta(1) \).
Furthermore, given \( n' \), such a \( G' \) can be constructed in probabilistic \( \text{poly}(n') \) time (except with negligible failure probability).

Proof of Corollary 6.4. Let \( n' \in \mathbb{N} \), and \( n = 2n' \). Let \( n_\ast \in \mathbb{N} \) such that \( n_\ast^{1/\beta} \geq n \), but \( n_\ast^{1/\beta} - 1 < n \). Then \( n_\ast^{1/\beta} = O_\beta(n) \). For a code length parameter \( \hat{n} \), let \( \tau = \lceil \gamma \hat{n} \rceil \).

Then there exist \( \hat{n}_\ast = \hat{n}_\ast(n) \), \( F = \mathbb{F}(\beta) \), and a linear code \( C_\ast \subseteq \mathbb{F}^{\hat{n}_\ast} \) such that \( \dim(C_\ast) = n_\ast \); \( \delta(C_\ast) = \Omega_\beta(1) \); \( \text{rate}(C_\ast)^{1/\beta} = n_\ast/\hat{n}_\ast \), and the following inequality is satisfied:

\[
\log (|F|) \cdot (n - n' - \tau) > \hat{n} \cdot H\left(\frac{\tau}{\hat{n}}\right) - \tau.
\]

Let \( C = C_\ast^{1/\beta} \). Then Viderman [Vid12] shows that \( C \) is an \( (\hat{n}_\ast^{\beta}, \frac{1}{2}) \)-strong LTC.

Since

\[
\log (|F|) \cdot (n - n' - \tau) > \hat{n} \cdot H\left(\frac{\tau}{\hat{n}}\right) - \tau
\]

then Theorem 6.3 implies the existence of an encoding function \( \text{Enc}_C(x; r) = G' \cdot (x; r) \) for the code \( C \), where \( x \in \mathbb{F}^n \), and \( r \in \mathbb{F}^{n' - n} \), such that \( \text{Enc}_C \) is a \( \tau \)-ZK encoding function. That is, \( \text{rate}_{\text{ZK}}(\text{Enc}_C) = \frac{n'}{\hat{n}} = \Omega_\beta(1) \).

Furthermore, given \( n' \), such a \( G' \) can be constructed in probabilistic \( \text{poly}(n') \) time (except with negligible failure probability). \( \square \)

### 6.5 Stable ZKLTCs

In this section construct stable linear ZKLTCs. We first formally define stable LTCs.

Recall that standard correctness and soundness are insufficient for cryptographic applications of LTCs in a distributed setting, since standard testing is defined only in relation to a pre-determined purported codeword, whereas in a distributed setting the parties holding the codeword symbols can determine their answers after seeing the queries. (See Chapter 7 for a more detailed discussion.) The distributed setting corresponds to the case in which a \( \tau \)-adversary (namely, an adversary that can alter at most \( \tau \) codeword symbols) can adaptively change few codeword symbols after seeing the queries of the tester. A \( \tau \) adversary can have one of two goals: to cause the tester to reject codewords; or to cause the tester to accept words that are far from the code. We call codes that withstand such adversaries stable codes. We first define the notion of a stability game between a tester and a \( \tau \) adversary.

**Definition 6.5.1** (Stability game). Let \( \tau \in \mathbb{N} \), and \( C \subseteq \mathbb{F}^{\hat{n}} \) be a code with tester \( D \). The stability game \( G_w^{\text{stab}}_{D, A_\tau} \) on \( w \in \mathbb{F}^{\hat{n}} \) between \( D \) and a \( \tau \)-adversary \( A_\tau \) is defined as follows.

- \( A_\tau \) selects a subset \( T \subseteq [\hat{n}] \) of size at most \( \tau \).
• \( \mathcal{D} \) non-adaptively picks a subset of coordinates \( I \subseteq [\hat{n}] \) to query.

• \( A_\tau \) is given \( I \), and can modify any symbol \( w_i \) such that \( i \in T \). Let \( w' \in \mathbb{F}^{\hat{n}} \) denote the codeword obtained after this step. (Notice that \( \text{supp}(w' - w) \subseteq T \).)

• \( \mathcal{D} \) receives \( w'|_I \) (i.e., the queried symbols after the above modifications), and outputs either \( \text{acc} \) or \( \text{rej} \).

• The output of the game is defined to be the output of \( \mathcal{D} \).

We now use the notion of stable games to define stable LTCs.

**Definition 6.5.2** (Stable LTCs). Let \( \tau \in \mathbb{N} \) be a corruption parameter, \( \epsilon > 0 \) be an error parameter, and \( \delta > 0 \) be a proximity parameter. Let \( C \subseteq \mathbb{F}^{\hat{n}} \) be a code with tester \( \mathcal{D} \).

• We say that \( \mathcal{D} \) has \((\tau, \epsilon)\)-completeness if for every \( \tau \)-adversary \( A_\tau \), and every codeword \( c \in C \), \( \Pr \left[ G_{c,\mathcal{D},A_\tau}^{\text{stab}} = \text{rej} \right] \leq \epsilon \).

• We say that \( \mathcal{D} \) has \((\tau, \epsilon, \delta)\)-soundness if for every \( \tau \)-adversary \( A_\tau \), and every word \( w \in \mathbb{F}^{\hat{n}} \) such that \( \delta(w, C) \geq \delta \), \( \Pr \left[ G_{w,\mathcal{D},A_\tau}^{\text{stab}} = \text{acc} \right] \leq \epsilon \).

We say that \( C \) is a \((t, \epsilon, \delta)\)-stable LTC if it has \((t, \epsilon)\)-completeness and \((t, \epsilon, \delta)\)-soundness.

Our construction of stable ZKLTCs is based on the testability of the tensor products of codes, studied in [DSW06, BV08, BV09, BS04, Vid12]. More specifically, we show that tensor products of codes can yield stable LTCs, namely if a base code is an efficiently decodable linear code, then its tensor products are stable LTCs with respect to a new tester. The parameters of our construction are summarized in the next theorem.

**Theorem 6.5** (Stable ZKLTCs). Let \( \hat{n} \in \mathbb{N} \) be a codeword length parameter. Then there exists a constant \( \alpha \in (0, 1) \), such that for any constants \( \beta > 0 \), and \( \delta \in (0, \alpha) \), there is a family of binary linear codes \( C \subseteq \mathbb{F}_2^{\hat{n}} \) with a linear randomized encoding function \( \text{Enc}_C \), such that:

• \( C \) is efficiently decodable from \( \alpha \hat{n} \) errors.

• \( C \) is a \((\tau, \frac{1}{4}, \delta)\)-stable LTC with respect to a \( \text{poly}(\hat{n}) \)-time tester that makes \( \hat{n}^3 \) queries, and uses \( O(\log(\hat{n})) \) random bits, where \( \tau = \Omega_\beta(\hat{n}) \).

• \( \text{Enc}_C \) is \( \tau \)-ZK.

• \( \text{rate}(\text{Enc}_C) = \Omega_\beta(1) \), and \( \delta(C) = \Omega_\beta(1) \).

The proof (see Section 6.5.2) is based on the robust testability of tensor products [Vid12], and the efficiently decodable linear codes of [Spi95]. We first define stable tensor products in Section 6.5.1.
6.5.1 Tensor Products of Codes

The definitions presented here are standard in the literature on tensor-based LTCs (e.g., [DSW06, BS04, Mei08, BV08]).

Let $I$ and $J$ be sets of coordinates. For $x \in \mathbb{F}^I$, and $y \in \mathbb{F}^J$, we let $x \otimes y$ denote the tensor product of $x$ and $y$ (i.e., the matrix $M$ with entries $M_{i,j} = x_i \cdot y_j$, where $(i, j) \in I \times J$). Given linear codes $R \subseteq \mathbb{F}^I$, and $C \subseteq \mathbb{F}^J$, we define the tensor product code $R \otimes C$ as the linear space spanned by words $r \otimes c \in \mathbb{F}^{I \times J}$, where $r \in R$, and $c \in C$. Tensor products have the following properties (see e.g., [DSW06]):

- The code $R \otimes C$ consists of all $|I| \times |J|$ matrices over $\mathbb{F}$, whose rows belong to $R$, and whose columns belong to $C$,
- $\dim(R \otimes C) = \dim(R) \cdot \dim(C)$,
- $\text{rate}(R \otimes C) = \text{rate}(R) \cdot \text{rate}(C)$,
- $\delta(R \otimes C) = \delta(R) \cdot \delta(C)$,
- The tensor product operation is associative, i.e., for any linear codes $C_1, C_2$ and $C_3$ it holds that $(C_1 \otimes C_2) \otimes C_3 = C_1 \otimes (C_2 \otimes C_3)$.

We let $C^1 = C$, and $C^k = C^{k-1} \otimes C$ for $k > 1$. Note that for this definition, $C^{0} = C$, and $C^2 = C^{2k-1} \otimes C^{2k-1}$ for $k > 0$. We also note that for a code $C \subseteq \mathbb{F}^k$, and $k \geq 1$, it holds that $\text{rate}(C^k) = (\text{rate}(C))^k$, $\delta(C^k) = (\delta(C))^k$, and the blocklength of $C^k$ is $\hat{n}^k$.

Testers for $C^k$. Next, we define testers for tensor product codes. A point in an $k$-dimensional cube can be associated with an $k$-tuple $(i_1, i_2, \ldots, i_k)$ such that $i_j \in [\hat{n}]$. We say that $\mathcal{H}$ is a $(k-1)$-dimensional $(b, i)$-hyperplane (or simply a hyperplane, in short) if

$$\mathcal{H} = \{(i_1, i_2, \ldots, i_k) \mid i_b = i, \text{ and for all } j \in [k] \setminus \{b\} \text{ we have } i_j \in [\hat{n}]\}.$$

**Definition 6.5.3** (Hyperplane Tester). Let $k \geq 3$. Let $M \in \mathbb{F}^k$ be an input word, where we wish to test whether $M \in C^k$. The $(k-1)$-dimensional hyperplane tester $\mathcal{D}$ picks (non-adaptively) a random $b \in [k]$, and a random $i \in [\hat{n}]$, and returns the $(b, i)$-hyperplane (the corresponding local view is $M|_{(b,i)}$). It can be shown that if $M \in C^k$ then $M|_{(b,i)} \in C^{k-1}$.

The hyperplane testers of Definition 6.5.3 can be composed to yield the following tester. Given a purported codeword $M \in \mathbb{F}_2^k$, the tester picks a random $(k-1)$-dimensional hyperplane $\mathcal{H}$, and considers $M' = M|_{\mathcal{H}}$, which (if $M$ is a codeword) should be a codeword in $C^{k-1}$. Next, the tester can pick a random $(k-2)$-dimensional hyperplane $\mathcal{H}_1$, and consider $M'' = M'|_{\mathcal{H}_1}$, and so on. We obtain a tester with query complexity $\hat{n}^2$, by picking a random 2-dimensional hyperplane when the blocklength of the code $C^k$ is $\hat{n}^k$. 

170
Definition 6.5.4 (2-dimensional hyperplane tester). Let $k \geq 3$. Let $M \in \mathbb{F}^{\hat{n}k}$ be a purported codeword, and we would like to test whether $M \in C^k$. The 2-dimensional hyperplane tester $D$ picks random $b_1, b_2, \ldots, b_{k-2} \in [k]$ such that $b_{i_1} \neq b_{i_2}$ for $i_1 \neq i_2$, and random $j_0, j_2, \ldots, j_{k-2} \in [\hat{n}]$, and outputs $M|_H$, where $B = \{b_1, b_2, \ldots, b_{k-2}\}$, and $H = \{(i_1, i_2, \ldots, i_k) \in [\hat{n}]^k : \forall g \in B : i_g = j_g\}$. Note that for any such choice, we have $C^k|_H = C^2$.

Remark 6.6. A 2-dimensional hyperplane tester for the code $C^k$ uses at most $\log\left(\binom{k}{2} \cdot \hat{n}^{(k-2)}\right) \leq 2k \log(\hat{n})$ random bits.

The following connection between the testability of a code and its tensor products was proved in [Vid12].

Proposition 6.5.5 ([Vid12]). Let $\alpha > 0$, and $k \in \mathbb{N}$ be a constant. Let $\hat{n} \in \mathbb{N}$ be a codeword length parameter, and $C \subseteq \mathbb{F}^{\hat{n}}$ be a linear code that is linear-time encodable, and linear-time decodable from $\alpha \cdot \hat{n}$ errors. Then $C^k$ is a linear code that is linear-time encodable, and linear-time decodable from $\alpha^k \cdot \hat{n}^k$ errors.

Robust Testing

We now define the notion of robustness, as was introduced in [BS04]. Informally, we say that a tester is robust if its view on any word that is far from the code, is far on average from any consistent view. (This notion was defined for LTCs following an analogous definition for PCPs [BGH04, Din06].) Formally,

Definition 6.5.6 (Robustness). Let $C \subseteq \mathbb{F}^{\hat{n}}$ be a code. Given a tester (i.e., a distribution) $D$ for the code $C \subseteq \mathbb{F}^{\hat{n}}$, we let

$$\rho^D(w) = \frac{\mathbb{E}_{I \sim D} [\delta(w|_I, C|_I)]}{\mathbb{E}_{I \sim D} [\delta(w|_I)]}$$

be the expected relative local distance of input $w$.

We say that the tester $D$ has robustness $\rho^D(C)$ on the code $C$ if for every $w \in \mathbb{F}^{\hat{n}}$, $\rho^D(w) \geq \rho^D(C) \cdot \delta(w, C)$.

Let $\{C_n\}_n$ be a family of codes where, $C_n$ has blocklength $\hat{n}$, and $D_n$ is a tester for $C_n$. We say that $\{C_n\}_n$ is robustly testable with respect to testers $\{D_n\}_n$ if there exists a constant $\alpha > 0$ such that for all $n$ we have $\rho^{D_n}(C_n) \geq \alpha$.

Viderman [Vid12] proved the following:

Theorem 6.7 ([Vid12]). Let $\hat{n} \in \mathbb{N}$ be a codeword length parameter, $C \subseteq \mathbb{F}^{\hat{n}}$ be a linear code, and $k \geq 3$ be a constant integer. Let $D$ be the 2-dimensional hyperplane tester for $C^k$. Then

$$\rho^D\left(C^k\right) \geq \left(\frac{\delta(C)}{18^k \log 1.5^k}\right)^{3k}.$$

Moreover, the query complexity of $D$ is $n^2$ and $C^m$ is smooth with respect to $D$, i.e., every coordinate is queried with the same probability. Note that the blocklength of $C^m$ is $n^m$.  

171
Definition 6.5.7 (Smoothness). We say that a code $C \subseteq \mathbb{F}^\hat{n}$ is smooth with respect to its tester $D$ if every coordinate of $[\hat{n}]$ is queried by $D$ with the same probability, i.e., for all $i, j \in [\hat{n}]$, $\Pr_{I \sim D}[i \in I] = \Pr_{I \sim D}[j \in I]$.

6.5.2 Stable LTCs via Tensor Product Codes

In this section we construct stable LTCs, thus proving Theorem 6.5. We will shorten notations by omitting the prefix “2-dimensional” before “hyperplane” and “hyperplane tester”. (For example, when we say “hyperplane tester” we mean 2-dimensional hyperplane tester.)

We will use a repetitive tester, defined as follows.

Definition 6.5.8 (Repetitive tester). Let $\hat{n} \in \mathbb{N}$ be a codeword length parameter, and $k, k' \in \mathbb{N}$. Let $C \subseteq \mathbb{F}^{\hat{n}}$ be efficiently decodable from $\alpha \hat{n}$ errors. Let $D$ be a 2-dimensional hyperplane tester for $C^k$. Let $\beta, \rho \in [0, 1]$ such that $\beta \leq \alpha^2$. We say that $D_{\text{rep}}$ is a $(k', \beta, \rho)$-repetitive tester if on an input word $M \in \mathbb{F}^{\hat{n}}_k$, $D_{\text{rep}}$:

- Invokes $D$ $k'$ times, and obtains $k'$ hyperplanes $H_1, \ldots, H_{k'}$.
- For every $i \in [k']$, uses a decoder for $C^2$ to determine (in polynomial time) $\delta(M|H_i, C^2)$, where if the decoding fails (i.e., $\delta(M|H_i, C^2) > \alpha^2$), then the corresponding estimate is defined as 1.
- If the fraction of hyperplanes (among $H_1, \ldots, H_{k'}$) that are $\beta$-far from $C^2$ is at least $\rho$, then $D_{\text{rep}}$ rejects. Otherwise, it accept.

Remark 6.8. The repetitive tester of Definition 6.5.8 uses at most $k' \cdot 2k \log(\hat{n})$ random bits, since it invokes $k'$ times a 2-dimensional hyperplane tester that uses at most $2k \log(\hat{n})$ random bits (see Remark 6.6).

We will need the following result of Spielman [Spi95].

Theorem 6.9 ([Spi95]). Let $\hat{n} \in \mathbb{N}$ be a codeword length parameter, and $\alpha > 0$ be constant. Then there exists an explicit construction of linear codes $C \subseteq \mathbb{F}_2^{\hat{n}}$ such that $\text{rate}(C) = \Omega(1)$, $\delta(C) = \Omega(1)$, and $C$ is encodable in linear time, and decodable in linear time from $\alpha \hat{n}$ errors.

Theorem 6.5 follows as a corollary from the following result:

Theorem 6.10. Let $\hat{n} \in \mathbb{N}$ be a codeword length parameter, $\tau \in \mathbb{N}$ be a corruption parameter, $\delta > 0$ be a proximity parameter, and $k \geq 3$ be a constant integer. Let $C \subseteq \mathbb{F}_2^{\hat{n}}$ be the code of Theorem 6.9, which is efficiently decodable from $\alpha \hat{n}$ errors for $\alpha < \delta(C)/2$. Let $\gamma = \frac{(\delta(C))^{3k}}{18^k \log_{1.5} k}$. If $\tau, \delta$ satisfy:

- $\delta \leq \alpha^2$, and

$^3$Recall that by Proposition 6.5.5, $C^2$ is efficiently decodable from $\alpha^2 \hat{n}^2$ errors.
\[
\tau \leq \min \left\{ \text{rate}(C^k) \cdot 0.0001, \frac{1}{20} \cdot \gamma \cdot \delta \right\} \cdot \hat{n}^k = \Omega(\hat{n}^k).
\]

Then, letting \(D_{\text{rep}}\) be a \((10^2 \cdot 20^2, \frac{1}{4} \gamma \delta, \frac{1}{4})\)-repetitive tester for \(C^k\), it holds that:

- \(\text{rate}(C^k) = (\text{rate}(C))^k = \Omega_k(1), \delta(C^k) = (\delta(C))^k = \Omega_k(1),\) and the query complexity of \(D_{\text{rep}}\) is \(O(\hat{n}^2) = O((\text{blocklength}(C^k))^2)\).
- \(C^k\) has a \(\tau\)-ZK encoding function \(\text{Enc}_C\) with rate \((\text{Enc}_C) = \Omega_k(1)\),
- \(C^k\) is \((\tau, \frac{1}{4} \cdot \delta)\)-stable with respect to \(D_{\text{rep}}\).

**Remark 6.11.** The repetitive tester of Theorem 6.10 uses at most \((10^2 \cdot 20^2) \cdot 2k \log(\hat{n}) = O(k \cdot \log \hat{n}) = O(\log(\text{blocklength}(C^k)))\) random bits.

The following simple observation states the conditions that allows one to recognize, given a purported codeword in which a small fraction of symbols were modified, whether or not the original word was far from the code.

**Observation 6.5.9.** Let \(\hat{n} \in \mathbb{N}\) be a codeword length parameter, \(k \in \mathbb{N}\), \(T \subseteq [\hat{n}]^k\), and \(\rho > 0\). Let \(C \subseteq \mathbb{F}^\hat{n}\) be a linear code, \(\mathcal{H}\) be a hyperplane, and \(M \in \mathbb{F}^\hat{n}\) be a purported codeword of \(C^k\). Assume that \(M' \in \mathbb{F}^\hat{n}\) such that \(\text{supp}(M - M') \subseteq T\), i.e., \(M'\) was obtained from \(M\) by modifying some symbols indexed by \(T\). Then:

- If \(M|_T \in C^2\), then \(\delta(M'|_T, C^k) \leq \frac{|T \cap \mathcal{H}|}{\hat{n}^2}\)
- If \(\delta(M|_T, C^2) \geq \rho + \frac{|T \cap \mathcal{H}|}{\hat{n}^2}\), then \(\delta(M'|_T, C^2) \geq \rho\).

Next, we prove Theorem 6.10.

**Proof of Theorem 6.10.** The first bullet of the Theorem follows from the definition of tensor products, and the definition of the repetitive tester. Theorem 6.3 guaranteed that \(C^k\) is \(\tau\)-ZK under encoding by the appropriate generator matrix.

Let \(D\) be the hyperplane tester and \(D_{\text{rep}}\) be the \((k', \beta, \rho)\)-repetitive tester, where \(k' = 10^2 \cdot 20^2, \beta = \frac{1}{4} \gamma \delta,\) and \(\rho = \frac{1}{4}\). We prove that \(C^k\) is a \((\tau, \frac{1}{4}, \delta)\)-stable with respect to \(D_{\text{rep}}\). Let \(M \in \mathbb{F}^\hat{n}\), and \(T \subseteq [\hat{n}]^k, |T| \leq \tau\), be the subset chosen by the \(\tau\)-adversary.

For the completeness part, assume that \(M \in C^k\). In this case, for every hyperplane \(\mathcal{H}\), we have \(M|_T \in C^2\). As every point is queried by \(D\) with probability \(\frac{\hat{n}^2}{n^2}\), then the expected size of \(\mathcal{H} \cap T\) for a random hyperplane \(\mathcal{H}\) is \(\frac{\hat{n}^2}{n^2} \cdot \tau = \frac{\tau}{n^2} \cdot \hat{n}^2\). By the Markov inequality, the probability that for a random \(\mathcal{H}\) we have \(|\mathcal{H} \cap T| \geq \frac{10}{\hat{n}^2} \cdot \hat{n}^2\) is at most \(\frac{1}{10}\). The Chernoff bound (see Claim 6.5.10 below) Guarantees that the probability that the fraction of \(\frac{10}{\hat{n}^2}\)-far hyperplanes obtained by \(D_{\text{rep}}\) is at least \(\frac{1}{2}\), is at most \(\exp(-\frac{k'}{310})\).

By the definition of \(\tau, \frac{10}{\hat{n}^2} \leq \beta,\) and \(\beta = \frac{1}{4} \gamma \delta \leq \delta \leq \alpha^2 \leq \alpha,\) so \(D_{\text{rep}}\) rejects \(M\) with probability at most \(\exp(-\frac{k'}{310}) \leq \frac{1}{2}\).

For the soundness part, assume that \(\delta(M, C^k) \geq \delta\), and \(\text{supp}(M - M') \subseteq T\). Theorem 6.7 Guarantees that \(E_{\mathcal{H}}[\delta(M|_T, C^2)] \geq \gamma \cdot \delta\). Since \(0 \leq \delta(M|_T, C^2) \leq 1\), it holds that

\[
\Pr_{\mathcal{H}}\left[\delta(M|_T, C^2) \leq \frac{1}{2} \cdot \gamma \cdot \delta\right] \leq \frac{1}{2} \cdot \gamma \cdot \delta.
\]

173
Moreover, we know that \( E_{\mathcal{H}}[|\mathcal{H} \cap T|] = \frac{\tau}{\pi} \cdot \hat{n}^2 \), and as noted above, the Markov inequality guarantees that for a random \( \mathcal{H} \), the probability that \(|\mathcal{H} \cap T| \geq \frac{10 \tau}{\pi} \cdot \hat{n}^2 \) is at most \( \frac{1}{10} \). We say that a hyperplane \( \mathcal{H} \) is bad if either \( \delta(M_{\mathcal{H}}, C^2) \leq \frac{1}{2} \cdot \gamma \cdot \delta \), or \(|\mathcal{H} \cap T| \geq \frac{10 \tau}{\pi} \cdot \hat{n}^2 \), otherwise \( \mathcal{H} \) is good. Thus, a random \( \mathcal{H} \) is bad with probability at most \( \frac{1}{2} \cdot \gamma \cdot \delta + \frac{1}{10} \leq \frac{1}{5} \). Note that if \( \mathcal{H} \) is good, then

\[
\delta(M'_{\mathcal{H}}, C^2) \geq \frac{1}{2} \cdot \gamma \cdot \delta - \frac{10 \tau}{\hat{n}^k} \geq \beta
\]

where the right-most inequality holds because \( \frac{10 \tau}{\hat{n}^k} \leq \frac{1}{4} \cdot \gamma \cdot \delta \).

By the Chernoff bound (see Claim 6.5.10 below), with probability at most \( \exp\left(-\frac{k^2 \gamma^2}{380^2}\right) \), the fraction of bad hyperplanes (among the ones obtained by \( \mathcal{D}_{\text{rep}} \)) is less than \( \frac{1}{5} \). Since \( \mathcal{D}_{\text{rep}} \) accepts only if at least \( \frac{3}{5} \) of the chosen hyperplanes are bad, then in this case \( \mathcal{D}_{\text{rep}} \) accepts with probability at most \( \exp\left(-\frac{3k}{2}\right) \leq \frac{1}{4} \). \( \Box \)

The proof of Theorem 6.10 used the following Chernoff bound.

**Claim 6.5.10 (Chernoff Bound).** If \( X = \sum_{i=1}^{k} X_i \) is a sum of independent \( \{0,1\} \)-valued random variables, where \( \Pr[X_i = 1] = \gamma \), then

\[
\Pr[X < (1 - \sigma) \gamma k] \leq \exp\left(-\frac{\sigma^2 \gamma k}{3}\right) \quad \text{and} \quad \Pr[X > (1 + \sigma) \gamma k] \leq \exp\left(-\frac{\sigma^2 \gamma k}{3}\right).
\]

**Proof of Theorem 6.5.** We choose \( k \) in Theorem 6.10 to be a large enough constant, such that \( \beta \geq 10^2 \cdot 20^2 \cdot \frac{2}{k} \). Define the constant \( \alpha \) of Theorem 6.5 as \( \alpha = \alpha_{\text{spi}}^k \), where \( \alpha_{\text{spi}} \) is the constant whose existence is guaranteed by theorem 6.9. Notice that for every \( \delta \in (0, \alpha) \) it holds that \( \delta \leq \alpha_{\text{spi}}^k \leq \alpha_{\text{spi}}^2 \). Moreover, notice that for our choice of \( k \), \( \gamma \) is constant. Let \( \hat{n}_{\text{spi}} \) denote the codeword length for the code of Theorem 6.9, then \( \hat{n} = \hat{n}_{\text{spi}}^k \). Moreover, the repetitive tester \( \mathcal{D}_{\text{rep}} \) uses \( k \log \hat{n} = O(\hat{n}) \) random bits, and tests a constant number \( (10^2 \cdot 20^2) \) of hyperplanes, each consisting of \( \hat{n}_{\text{spi}}^k = \hat{n}^2 \) points. Therefore, in total \( \mathcal{D}_{\text{rep}} \) makes at most \( \hat{n}^2 \) queries. Moreover, \( \tau = \Omega_{\beta, \delta}(\hat{n}) \), rate \( (\text{Enc}_C) = \Omega_{\beta}(1) \), and \( \delta(C) = \Omega_{\beta}(1) \). \( \Box \)
Chapter 7

Cryptographic Applications

In this chapter we use the zero-knowledge techniques described in previous chapters to improving the efficiency of various cryptographic tasks, such as verifying a shared secret; flipping a common random coin; and proving NP statements in zero-knowledge in a distributed setting. We exhibit applications both in the two party, and the multiparty, setting.

7.1 Introduction

We use ZKPCPs, ZKPCPPs, and ZKLTCs, to improve the communication complexity, round complexity, and underlying assumptions, of cryptographic tasks in the distributed setting, such as Verifiable Secret Sharing (VSS); coin-flipping; and zero-knowledge proofs, as well as for Commit-and-Prove style functionalities both in the 2-party and in the multiparty setting. Our motivation for focusing on these tasks is the possibility of employing them for more general tasks that involve sublinear-communication zero-knowledge arguments on distributed or committed data.

More specifically, as explained in Chapters 5 and 6, the efficient verification feature of our codes and proofs is useful in the contexts of distributed storage and cloud computing. For example, a database owner can use stable ZKLTCs to distribute a large sensitive database between many potentially unreliable, or even malicious, servers. The ZK feature of the code guarantees that a small subset of corrupted servers learn essentially nothing about the database, whereas the stable testability of the code allows the database owner to verify, at any given moment, the integrity of her stored data, by probing only a sublinear number of servers. By replacing the ZKTC with ZKPCPPs, the database can be used to answer user queries. ZKPCPPs can also be used to construct updatable databases that allow users to securely perform read and write operations on a sensitive database. Using Merkle Hash Trees [Mer87], ZKPCPPs can also be used to implement a sublinear-communication “Commit-and-Prove” functionality in the 2-party setting, which only makes a black-box use of a collision-resistant hash function.
7.1.1 Our Results and Techniques

We now give a more detailed description of our results, and the underlying techniques.

VSS and Certifiable VSS

Motivated by applications that require verification with no information leakage, we focus on reducing the communication complexity of verifying the shares in a verifiable secret sharing (VSS) protocol [CGMA85, Fel87, BGW88, CCD88, BR89]. Roughly speaking, VSS allows a dealer $D$ to distribute a secret $x$ among $m$ servers in a way that prevents a coalition of up to $\tau$ servers from learning or modifying the secret, while on the other hand guaranteeing unique reconstruction, even if $D$ and up to $\tau$ servers can collude.

We consider the following designated receiver variant of VSS which involves, in addition to $D$ and the $m$ servers, a designated receiver $R$ who may assist in the verification. Such a VSS protocol consists of three phases. In the sharing phase, the dealer $D$ randomly distributes $x$ between the servers by privately sending a share $x_i$ to each server. In the verification phase, the receiver $R$ can freely interact with the servers, possibly using a broadcast channel. Finally, in the reconstruction phase, each server sends a single message to $R$, and $R$ reconstructs the secret.

We assume that the participating parties can interact over a synchronous network of secure point-to-point channels. The parties also have access to a broadcast channel, where a message sent over this channel is received by all other parties. When measuring communication complexity, we count a message sent over a broadcast channel only once towards the total communication. Alternatively, our protocols can be implemented with similar communication complexity using a public bulletin board, where every time a message is written to or read from the board is counted towards the communication complexity.

The security of protocols is defined by considering their execution in the presence of an active adversary who may corrupt and control a subset of the parties. We assume adversaries to be static, in the sense that they choose the set of corrupted parties at the onset of the protocol, but they are capable of rushing, namely sending their messages only after receiving all messages sent by honest parties in the same round.

Roughly speaking, a protocol as above is said to be $(\tau, \epsilon)$-secure if it satisfies the following requirements:

- **Correctness.** If the adversary corrupts $\tau$ servers, the receiver reconstructs the correct secret $x$ except with at most $\epsilon$ probability.

- **Secrecy.** Any adversary who corrupts $R$ and $\tau$ servers gets at most an $\epsilon$-advantage in distinguishing between two secrets.

- **Binding.** For any adversary who corrupts $D$ and $\tau$ servers, the following holds except with at most $\epsilon$ failure probability over the randomness of the sharing and
verification phases. At the end of the verification phase there is a unique secret $x^*$ (determined by the messages exchanged up to this point), such that $R$ will output $x^*$ regardless of the messages sent by the adversary during the reconstruction phase.

**VSS from ZKLTCs.** The application of ZKLTCs to VSS is conceptually simple. The protocol uses a *stable* ZKLTC $C$ with encoding function $\text{Enc}_C$ and tester $D$. (For the protocol to be computationally efficient, we need $\text{Enc}_C$ to be efficiently encodable and decodable.) In the sharing phase, the dealer $D$ randomly encodes the secret $x$ into a codeword $c = \text{Enc}_C(x) \in C$, and partitions the symbols of $c$ between the servers. In the verification phase, the receiver $R$ verifies that the messages received by the servers are close to a valid codeword by executing a random test of $D$, where $R$ broadcasts the queries of $D$ and the servers answer. (The use of broadcast prevents $R$ from contacting too many servers, which would violate the secrecy requirement.) If the test fails, $R$ outputs $x^* = 0$ (or some other default value), and ignores further messages. For reconstruction, the servers send their shares to $R$, who decodes the secret $x$. Secrecy follows from the zero-knowledge property of $\text{Enc}_C$, whereas the stability of $D$ is useful for both binding and correctness. First, it ensures that if a corrupted $D$ distributes $c^*$ which is far from $C$ then $R$ notices this with high probability, even if $D$ and up to $\tau$ servers collude. Second, if $D$ is honest and $c \in C$, the tester $D$ will accept, and the decoder of $\text{Enc}_C$ will output $x^* = x$ at the end of the reconstruction phase (except with small probability), even if there are $\tau$ malicious servers. Finally, if a corrupted $D$ distributes $c^*$ which is not a codeword, but is not far from $C$, then regardless of whether $D$ accepts or rejects, a unique secret $x^*$ is determined at the end of the verification phase.

**Sublinear-verification VSS.** Note that by using good stable ZKLTCs, the communication complexity during the verification phase of the above protocol becomes sublinear in the secrecy threshold $\tau$. Sublinear verification is motivated by situations in which verification forms an efficiency bottleneck. Consider, for example, a situation in which the secret is reconstructed long after the shares are distributed. As more and more servers may become corrupted over time, the receiver might want to run the verification procedure periodically, whereas the shares are distributed (and the secret is reconstructed) only once. The efficient verification feature is captured by the following theorem (see Corollary 7.13 for the formal statement).

**Theorem 7.1** (Sublinear-communication VSS, informal). *For every constant $\gamma > 0$ there exists a constant-round, $n$-server, designated receiver VSS protocol for secrets of length $\Omega(n)$ with verification phase which uses $O(n^\gamma)$ bits of communication. The protocol is $(\tau, \epsilon)$-secure for $\tau = \Omega(n)$, and $\epsilon = n^{-\omega(1)}$.*

**Linear-communication VSS.** Another feature of the designated receiver VSS protocols we obtain via stable ZKLTCs is that they can be implemented with only $O(n)$ bits
of communication, \( \tau = \Omega(n) \), and statistical error \( \epsilon \) that vanishes almost exponentially with \( n \). Previous VSS protocols (e.g., those from \([\text{BGW88, CCD88, BR89, FM02, GIKR01, DI06a, FGG}^{+}06, \text{PCRR09, KPR10, Pen11, Agr12}]\)) require nearly quadratic communication (or more) to achieve similar guarantees even when the secret is just a single bit, though they can offer a higher fractional resilience, and are not restricted to a designated receiver. The linear communication feature is captured by the following theorem (see Theorem 7.12 for the formal statement).

**Theorem 7.2 (Linear-communication VSS, informal).** For every constant \( \gamma > 0 \) there exists a constant-round, \( n \)-server, designated receiver VSS protocol for secrets of length \( \Omega(n) \) with total communication complexity \( O(n) \). The protocol is \((\tau, \epsilon)\)-secure for \( \tau = \Omega(n) \), and \( \epsilon = 2^{-n^{1-\gamma}} \).

### Certifiable VSS

We also study a certifiable variant of VSS (which we call certifiable VSS) which generalizes traditional VSS by providing the additional guarantee that the shared secret satisfies some NP predicate, namely, that \( x \) is in some NP-language \( L_R \). We say that a certifiable VSS protocol is \((\tau, \epsilon)\)-secure if it satisfies the following requirements:

- **Correctness.** If the adversary corrupts \( \tau \) servers, the receiver reconstructs the correct secret \( x \in L_R \) except with at most \( \epsilon \) probability.

- **Secrecy.** The view of any adversary who corrupts \( R \) and \( \tau \) servers can be efficiently simulated (with at most \( \epsilon \) statistical distance) from \( |x| \) alone.

- **Binding.** For any adversary who corrupts \( D \) and \( \tau \) servers, the following holds except with at most \( \epsilon \) failure probability over the randomness of the sharing and verification phases. At the end of the verification phase there is a unique secret \( x^* \in L_R \) (determined by the messages exchanged up to this point), such that if \( R \) did not abort at the end of the verification phase, then \( R \) will output \( x^* \) regardless of the messages sent by the adversary during the reconstruction phase.

### Certifiable VSS from ZKPCPPs

We use ZKPCPPs for designing certifiable VSS protocols for NP with polylogarithmic communication during the verification phase. The protocol uses a ZKPCPP system \((P, V)\), and a robust secret sharing scheme. (We note that for the protocol to be efficient, \( P, V \) should be efficient, as well as the sharing and reconstruction algorithms of the secret sharing scheme.)

In the sharing phase, the dealer \( D \) secret shares \( x \in L_R \) using the secret sharing scheme, generates a ZKPCPP for the claim ‘the secret shares are “close” to a vector of “legal” secret shares of \( x \), and \( x \in L_R \)’, and distributes the shares and proof between the servers. In the verification phase, the receiver \( R \) verifies that the shares are close to a sharing of some \( x' \in L_R \) by emulating a the verifier \( V \), where \( R \) broadcasts the queries of \( V \) and the servers answer. If the verification fails, \( R \) aborts. For reconstruction, the servers holding the secret shares of \( x \) send them to \( R \), who reconstructs the secret \( x \). The
security analysis of the protocol is similar to the security analysis of the ZKLT-based VSS protocol described above, where the “certifiable” property follows from the stronger soundness guarantee of the ZKPCPP (namely, that such proofs can be used to prove general NP-statements, whereas LTCs only guarantee membership in the code). Thus, we obtain the following:

**Theorem 7.3** (Verification-efficient certifiable VSS). For every NP-relation \( R = \mathcal{R}(x,w) \), every corruption threshold \( \tau \), and every soundness parameter \( \epsilon > 0 \), there exists a poly \((n,\tau,\log \frac{1}{\epsilon})\)-server, \((\tau,\epsilon)\)-secure certifiable VSS protocol for \( n \)-bit messages. The protocol has total communication complexity \( \text{poly}(n,\tau,\log \frac{1}{\epsilon}) \), and a verification phase that uses \( \text{polylog}(n,\tau,\frac{1}{\epsilon}) \) bits of communication.

Dealing with corrupted servers. We note that the overview of our certifiable VSS protocol described above is in fact an over-simplification of the construction: the verification procedure of the ZKPCPP cannot be used as-is, since it only guarantees soundness when the verification is performed with a proof oracle, whereas corrupted servers can determine their answers after seeing the queries of the receiver. We solve this problem by designing (Section 7.3) a transformation from probabilistic proof systems into stable systems, in which the influence of corrupted servers is restricted, and is guaranteed to not harm the soundness of the system.

Distributed Coin-Flipping

We consider a distributed model for coin-flipping in which two clients want to agree on a common random bit with the help of \( m \) servers. The clients can interact with the servers via synchronous, secure point-to-point channels, and a broadcast channel, where at the end of the interaction each client outputs a single bit. The protocol is said to be \((\tau,\epsilon)\)-secure if the following requirements are met.

- **Correctness.** If an adversary can corrupt \( \tau \) servers at the onset of the protocol, the joint outputs of the clients will be \( \epsilon \)-close (in statistical distance) to a pair of identical random bits.

- **Agreement.** An adversary who corrupts one client, and \( \tau \) servers, at the onset of the protocol can bias the output of the other client by at most \( \epsilon \).

The above distributed model for coin-flipping is motivated by the impossibility of achieving a similar fairness guarantee via direct interaction between the clients. This impossibility holds even if one settles for security against computationally bounded clients [Cle86].

Our coin-flipping protocols are obtained from designated-receiver VSS in the natural way: each client picks a random \( b \in \{0,1\} \) and distributes \( b \) between the servers using the VSS protocol, where the other client acts as the receiver. The two secret bits are then reconstructed, and the common coin is defined as their exclusive-or. (To save
The communication complexity of the protocols obtained via this approach is captured by the following theorem (see Theorem 7.18 for the exact statement).

**Theorem 7.4** (Distributed coin-flipping, informal). For every constant \( \gamma > 0 \) there exists a constant-round \( n \)-server \( (\tau, \epsilon) \)-secure distributed coin-flipping protocol, where \( \tau = \Omega(n) \), and \( \epsilon = 2^{-n^{1-\gamma}} \), with total communication complexity \( O(n) \).

We note that coin-flipping protocols which are based on VSS protocols from the literature require nearly quadratic communication to achieve a similar security guarantee.

### 3-Round Distributed Proofs with Zero-Knowledge Properties

We use the non-adaptively verifiable WIPCPs and CZKPCPs of Chapter 4 to construct 3-round distributed WI and CZK proofs (respectively) for NP in what we call \( m \)-distributed proof systems, in which the prover and verifier are aided by \( m \) servers.

Our motivation for studying proofs in a distributed setting is to minimize the round complexity, and underlying assumptions, of sublinear ZK proofs. Concretely, it is known that assuming the existence of collision resistant hash functions, there exist 2-party 4-round sublinear ZK arguments for NP [Kil92, IMS12]. (Arguments guarantee soundness only against bounded malicious provers.) We show that in the distributed setting, there exist 3-round sublinear CZK (respectively, WI) proofs for NP, assuming the existence of OWFs (respectively, unconditional). Thus, the distributed setting allows us to improve previous results in terms of round complexity, underlying assumptions, and soundness type.

We say that an \( m \)-distributed proof system is a \( (\tau, m) \)-distributed ZK proof system for an NP-relation \( \mathcal{R} \) if it satisfies the following properties.

- **Correctness.** If all parties are honest, and \((x, w) \in \mathcal{R}\), then \( V \) accepts \( x \) with probability 1.
- **Soundness.** If \( x \notin L_\mathcal{R} \) then \( V \) rejects \( x \) except with negligible probability, even if the prover is corrupted, and colludes with \( \tau' \leq \tau \) corrupted servers.
- **Zero-knowledge.** For every adversary \( \mathcal{A} \) corrupting \( V \), and \( \tau' \leq \tau \) servers, there exists a PPT simulator \( \text{Sim} \) such that for every \( x \in L_\mathcal{R} \), \( \text{Sim}(x) \) is computationally indistinguishable from the view of \( \mathcal{A} \) in the protocol execution, when \( \mathcal{A} \) has input \( x \).

We note that the aforementioned zero-knowledge property can be naturally relaxed to WI, or CZK in the CRS model.

We use WIPCPs (respectively, CZKPCPs) to construct 3-round distributed-WI proof systems (respectively, CZK proof system in the CRS model) for NP which, at a high level, operate as follows. In the first round the prover distributes a WIPCP...
(respectively, a CZKPCP) between the servers, and in the second and third rounds the
verifier and servers emulate the WIPCP (respectively, CZKPCP) verification procedure.
(That is, the verifier broadcasts the proof queries of the WIPCP or CZKPCP verifier,
and the servers provide the corresponding proof bits. As in the construction of certifiable
VSS, we need to use stable probabilistic proof system, as described in Section 7.3.)
Thus, we obtain the following results, where we say that an $m$-distributed proof system
is sublinear if the total communication involving the verifier is sublinear in the input
length. (In fact, in both systems the communication complexity involving the verifier is
polylogarithmic in the input length.)

**Theorem 7.5** (Sublinear distributed WI proofs). For every NP-relation $R$, and poly-
omial $\tau(n)$, there exists a polynomial $m(n) > \tau(n)$ such that $R$ has a 3-round sublinear
$(\tau(n), m(n))$-distributed WI proof system, where $n$ is the input length.

**Theorem 7.6** (Sublinear distributed CZK proofs in the CRS model). Assume that
OWFs exist. Then for every NP-relation $R$, and polynomial $\tau(n)$, there exists a
polynomial $m(n) > \tau(n)$ such that $R$ has a 3-round sublinear $(\tau(n), m(n))$-distributed
CZK proof system in the CRS model, where $n$ is the input length.

These constructions crucially rely on the non-adaptivity of the honest WIPCP
(respectively, CZKPCP) verifier (otherwise we would need at least 4 rounds, since
rounds cannot be compressed). Moreover, the verifier may collude with a subset of
servers, so the PCP should be WI (respectively, CZK) against malicious verifiers.

**Two-Party Commit-and-Prove**

We use ZKPCPPs to construct Commit-and-Prove protocols, a “certifiable” generaliza-
tion of a commitment scheme (alternatively, the “2-party analog” of certifiable VSS). A
commitment scheme is a two-phase protocol between a sender $S$ and a receiver $R$. In
the first phase, called the commit phase, the server on input $x$ freely interacts with $R$,
and the messages exchanged during the phase constitute the commitment on $x$. In the
second phase, called the reveal phase, $S$ sends $x$, together with a decommitment string
dec to $R$, and $R$ either accepts or rejects $x$. Informally, a commitment scheme should
hide $x$ until the reveal phase, and bind $S$ to $x$ after the commit phase (in the sense that
at the end of the commit phase $S$ cannot find two different $x, x'$ that $R$ would accept
during the reveal phase).

A Commit-and-Prove protocol is certifiable in the sense that $S$ not only commits to
$x$, but also proves it satisfies some predicate. As $S$ is a PPT algorithm, a witness $w$ for
$x$ is given to $S$, since it cannot generally be expected to find one on its own. Informally,
we say that a Commit-and-Prove protocol for a relation $R$ is secure if the following holds:

- **Correctness.** For every input $x \in L_R$ of $S$, if $S, R$ are honest then $R$ accepts $x$. 181
• **Binding.** At the end of the commit phase, no efficient (possibly malicious) sender algorithm $S^*$ can find (except with negligible probability) $x, x'$ such that $R$ would accept both $x, x'$, and $x \neq x'$ or $x \notin L_R$.

• **Hiding.** The view of every (possibly malicious) PPT receiver $R^*$ during the commit phase can be efficiently simulated (up to $\text{negl}(|x|)$ computational distance) given $|x|$ alone.

• **Zero-knowledge after reveal.** The view of every (possibly malicious) PPT receiver $R^*$ (during the entire interaction) can be efficiently simulated (up to $\text{negl}(|x|)$ computational distance) given $x$ alone.

Using techniques similar to those of [IMS12] (which were used to construct sublinear ZK arguments), we use HVZKPCPPs, and exponentially-hard collision-resistant hash functions (see Section 7.7 for the formal definition), to construct Commit-and-Prove protocols for NP that make only black-box use of the hash function, and require polylogarithmic communication during the commit phase. (By allowing sublinear communication during the commit phase, we can base the protocol on a super-polynomially hard hash function.)

The high-level idea of the construction is to use a robust secret sharing scheme to hide the input, and an HVZKPCPP system to prove that $x \in L_R$ (concretely, that the secret-sharing of $x$ is “close” to the sharing of some $x^* \in L_R$). During the commit phase, $S$ commits to the secret sharing and the proof (using a standard commitment scheme), compresses these commitments using a “Merkle Hash Tree” (MHT) [Mer87], and sends the root of the tree to $R$. The parties then emulate the HVZKPCPP verifier, where $S$ answers every query by opening the commitment at the leaves of the MHT, and also providing the nodes on the path from that leaf to the root. To decommit to $x$, $S$ sends the commitments to the secret sharing of $x$, together with the corresponding sub-tree of the MHT. We note that (unlike the distributed setting) honest-verifier zero-knowledge suffices in this case, since the sender can refuse to answer dishonest queries. See Section 7.7 for more details of this construction, and an exact formulation of our results.

We also describe generalizations of Commit-and-Prove to the reactive, and the multiparty, settings, and show an application of these generalizations to design updatable databases. More specifically, updatable databases allow multiple clients to perform read from, and write to, a sensitive database, such that the clients are guaranteed that their requested operations where executed correctly, whereas the database owner is guaranteed that these operations reveal no additional information, other than what follows from the natural output of the operations, e.g., in a read operation. (These databases also support client-specific access patterns which determine which locations can be read from, or written to, by each client.) See Sections 7.7.3 and 7.7.4 for additional details.
7.1.2 Chapter Organization

After giving the necessary preliminaries in Section 7.2, we describe in Section 7.3 probabilistic proof systems with stronger completeness and soundness guarantees (which we call stable proofs) that are useful for cryptographic applications. Then, we describe our VSS and certifiable VSS protocols in Section 7.4, and use these in Section 7.5 to design efficient coin-flipping protocols in a distributed setting. In Section 7.6 we construct 3-round distributed proof systems with zero-knowledge guarantees. Finally, in Section 7.7 we describe our Commit-and-Prove protocols in the 2-party setting, their generalizations to the reactive and multiparty settings, and the application of updatable databases.

7.2 Preliminaries

We assume that standard cryptographic primitives (e.g., OWFs) are secure against non-uniform adversaries.

Idealized Primitives. When describing our cryptographic applications for ZKPCPPs, we first present the protocols and analyze their security, in a hybrid model, in which the parties can use an idealized version of primitives (such as coin-flipping). This gives a more modular analysis. We note that the hybrid model is used only for convenience, and all our protocols are secure in the plain model (which we prove using general composition theorems).

Secret Sharing Schemes

A secret sharing schemes is a method of distributing a secret \( s \) between several parties, such that the shares completely determine \( s \), while “few” shares reveal no information about \( s \). Formally,

**Definition 7.2.1** (Secret sharing). Let \( \{ \mathbb{F}_n \}_{n \in \mathbb{N}} \) be a family of finite fields. A \((t(n), \epsilon)\)-private, \( \tau(n) \)-robust secret sharing scheme is a pair of algorithms \((\text{Share}, \text{Rec})\), such that the following holds.

- **Syntax.** \( \text{Share} : \mathbb{F}_n \times \mathbb{F}_t^{d(n)} \rightarrow \mathbb{F}_m^{(n)} \) is called the sharing algorithm and \( \text{Rec} : \mathbb{F}_m^{(n)} \rightarrow \mathbb{F}_n \times \mathbb{F}_t^{(m)} \) is called the reconstruction algorithm.

- **\( t(m) \)-privacy.** For any pair \( x, x' \in \mathbb{F}_n^h \) of messages, and any subset \( I \subseteq [m(n)], |I| \leq t(n) \), \( SD(\text{Share}(x, r) | _I, \text{Share}(x', r') | _I) \leq \epsilon \), where \( r, r' \in_R \mathbb{F}_n^{d(n)} \).

- **\( \tau(n) \)-robustness.** Let \( x \in \mathbb{F}_n^n, r \in \mathbb{F}_t^{d(n)} \), and \( y = \text{Share}(x, r) \). Let \( y' \in \mathbb{F}_m^{(n)} \) such that \( \Delta(y, y') \leq \tau(n) \). Then \( \text{Rec}(y') = (x, r) \).

We say that a secret sharing scheme is \( t(n) \)-private, if it is \((t(n), 0)\)-private.
7.3 Stable Probabilistic Proof Systems

In this section we define the notion of stable proof systems, which are proof systems in which every proof symbol is queried by the verifier with (roughly) the same (small) probability. The stability property is essential for the implementation of a proof system in a distributed setting. Indeed, the soundness guarantee of any oracle-based proof system holds only when the oracle answers are independent of the queries. However, when the oracle symbols are held by servers, then the emulated oracle answers provided by malicious servers may depend on the queries of the verifier. We refer to systems in which “corrupted” oracle symbols have limited influence on the verification procedure as “stable” systems.

We describe a method of transforming any probabilistic proof system into a stable system. More specifically, we add an additional layer of randomization on top of the verification procedure. Remember that the goal is to restrict the influence of a malicious prover (alternatively, of malicious servers) that can adaptively control symbols of the input and proof oracles. This can be achieved by duplicating the input and proof oracles, where the answer to every oracle query is reconstructed using a majority vote over the corresponding symbol in copies that the verifier randomly chooses. Thus, every single oracle symbol is queried with low probability, since the verifier chooses which copies to use at random (so a query to symbol $i$ in the original oracles is replaced with almost random queries to the new oracles). We note that since this “error-correction” procedure of reconstructing oracle symbols is applied also to the symbols of the input oracle, the verifier in effect verifies that the input defined by the majority vote is close to the language. This will not be problematic when we use stable proof systems in our cryptographic applications.

We note that since we are interested in stable probabilistic proof systems with zero-knowledge, we cannot use a more sophisticated method (e.g., a better error-correcting code than the repetition code) to obtain stability in a more sophisticated method, since the symbols of the new oracles must have low locality (i.e., depend on few symbols in the original oracles).

**Construction 7.7.** Let $R = R(x, w)$ be an NP relation, and $(P, V)$ be a probabilistic proof system for $R$. The stable probabilistic proof system $(P_{\text{stable}}, V_{\text{stable}})$ for $R$ is parameterized by $l$, the number of oracle copies generated by the prover; and $k$, the number of symbols used to reconstruct a single oracle answer.

**Prover algorithm.** $P_{\text{stable}}$, on input $1^l, 1^k, x, w$, generates a proof $\pi \leftarrow P(x, w)$. $P$ duplicates the proof $l$ times: $\pi^1, \ldots, \pi^l$, and also generate $l - 1$ copies $x^2, \ldots, x^l$ of $x$, and let $x^1 := x$. $P_{\text{stable}}$ outputs $\pi_{\text{stable}} = (x^2, \ldots, x^l, \pi^1, \ldots, \pi^l)$.

**Verifier algorithm.** $V_{\text{stable}}$ on input $1^k$, and given oracle access to $x, \pi_{\text{stable}}$, operates as follows.

- Emulates $V$ to obtain the queries $i_1, \ldots, i_q$ that $V$ makes to its oracles.
• For every $1 \leq j \leq q$:
  
  - Picks $k$ random indices $s_{j1}^1, \ldots, s_{jk}^k \in [l]$.
  
  - For every $1 \leq z \leq k$, if $i_{jz}$ is an input query, then $V_{\text{stable}}$ reads the symbols $x_{i_{jz}}^{s_{jz}}$, otherwise $i_{jz}$ is a proof query, and $V_{\text{stable}}$ reads the symbol $\pi_{i_{jz}}^{s_{jz}}$.
  
  - Reconstructs the queried symbol as follows. If there exists a symbol $b$ such that at least $\frac{7k}{8}$ of the queries symbols of $x_{i_{j1}}^{s_{j1}}, \ldots, x_{i_{jk}}^{s_{jk}}$ (or $\pi_{i_{j1}}^{s_{j1}}, \ldots, \pi_{i_{jk}}^{s_{jk}}$) are equal to $b$, then $V_{\text{stable}}$ sets the corresponding bit to $b$. Otherwise, $V_{\text{stable}}$ rejects.

• If $V_{\text{stable}}$ did not reject in the previous step, then the oracle answers to the queries of $V$ have been determined. These are given to $V$ as the oracle answers, and $V_{\text{stable}}$ either accepts, or rejects, according to the output of $V$.

**Remark 7.8.** We will also use a variant of Construction 7.7 in which the prover only duplicates the proof. This version will be useful for applications of ZKPCPs (in which the input oracle is given to the verifier in its entirety, so the input symbols cannot be tampered with) described in Section 7.6.

We show that the verification procedure of Construction 7.7, when carried out in the presence of a malicious adversary that can adaptively determine the oracle answers of “few”, pre-determined locations, emulates (with high probability) the verification procedure of the original system $(P, V)$ in the standard setting (in which oracle answers are independent of the queries). We will use the following terminology.

**Notation 7.9.** For an odd $l$, and a set $s_1^1, \ldots, s_l^l$ of bit strings (of equal length), we define the string $s^*$ that is consistent with $s_1^1, \ldots, s_l^l$, to be the string such that every bit $s^*_i$ is the majority vote over $s_{i1}^1, \ldots, s_{il}^l$.

**Lemma 7.3.1.** Let $l, k, \tau \in \mathbb{N}$ such that $l \geq 2^6 \cdot \tau$ and $k \geq 16$, and let $s = s_1^1 \circ \ldots \circ s_l^l$ be a bit-string such that $|s_1^1| = \ldots = |s_l^l|$. Let $s^*$ be a string that is consistent with $s_1^1, \ldots, s_l^l$ (as in Notation 7.9), and $A$ be an adversary who chooses in advance a subset $B$ of at most $\tau$ bits of $s$ which it can adaptively control. Then for every bit $1 \leq j \leq |s^*|$, $A$ wins the following game with probability at most $2^{1 - \frac{k}{8}}$.

- $k$ indices $i_1, \ldots, i_k$ are picked at random, and $s_{j1}^{i_1}, \ldots, s_{jk}^{i_k}$ are queried. For every bit $b \in \{s_{j1}^{i_1}, \ldots, s_{jk}^{i_k}\}$ such that $b \in B$, $A$ determines the answer to the query (for every bit $b \notin B$, the answer to the query is the corresponding bit in $s$).

- If there exists a bit $b$ such that at least $\frac{7k}{8}$ of the answers are equal to $b$, then the output of the game is $b$. Otherwise, the output is $\perp$.

- The adversary wins if the output of the game is $b \in \{0, 1\}$ such that $b \neq s_{j}^{*}$.
Intuitively, the lemma guarantees that if $P_{\text{stable}}$ is honest, then (except with probability at most $2^{1-\frac{k}{8}}$) every single bit that $V_{\text{stable}}$ reconstructs in Construction 7.7 is consistent with the corresponding bit of $x \circ \pi$. This holds even if $\tau' \leq \tau$ bits are “corrupted”, in the sense that they are arbitrarily modified after $V_{\text{stable}}$ determines its queries (as long as the identity of these indices is determined before $V_{\text{stable}}$ chooses its queries). Moreover, even if a corrupted $P_{\text{stable}}$ chooses a subset of $\tau' \leq \tau$ oracle bits that it can adaptively control, then except with probability at most $2^{1-\frac{k}{8}}$, the bit that $V_{\text{stable}}$ reconstructs is consistent with $x^* \circ \pi^*$, where $x^* \circ \pi^*$ is the (possibly corrupted) “majority vote” input and proof oracles (as in Notation 7.9) defined by the input $x$, and the input and proof copies that $P_{\text{stable}}$ generated.

**Proof of Lemma 7.3.1.** Let $s^*_j$ be the bit that should be reconstructed in the game (determined according to the majority vote over $s$). Then in the game, the bit $b = \bar{s}^*_j$ is reconstructed, when the $k$ (random and independent) indices chosen for the reconstruction are $i_1, \ldots, i_k$, only if at least $\frac{7k}{8}$ of the bits in $\{s^1_j, \ldots, s^k_j\}$ were equal to $b$. Let $E_{\text{maj}}$ denote the event that at most $\frac{3k}{4}$ of the bits in $\{s^1_j, \ldots, s^k_j\}$ were equal to $b$. Conditioned on $E_{\text{maj}}$, $A$ wins only if it was able to flip (by flipping bits in $B$) at least $\frac{7k}{8} - \frac{3k}{4} = \frac{k}{8}$ of the bits in $\{s^1_j, \ldots, s^k_j\}$. Let $I := \{i \in [l] : s^i_j \in B\}$, then $|I| \leq |B| \leq \tau$. Consequently, conditioned on $E_{\text{maj}}$, $A$ wins with probability at most $2^{-\frac{k}{8}}$, since

$$\Pr_{i_1, \ldots, i_k \in [l]} \left[\{i_1, \ldots, i_k\} \cap I \geq \frac{k}{8}\right] \leq \left(\frac{k}{8}\right) \cdot \left(\frac{\tau}{l}\right)^k \leq \left(k \cdot e \frac{k}{8}\right) \left(\frac{\tau}{l}\right)^k = \left(\frac{8e\tau}{l}\right)^k \leq 2^{-\frac{k}{8}}.$$ 

By the definition of $s^*$, less than half the bits $\{s^1_j, \ldots, s^l_j\}$ are equal to $b$. As $i_1, \ldots, i_k$ are random and independent, then (using Hoeffding’s bound) $Pr[E_{\text{maj}}] \leq 2^{-\frac{k}{8}}$. Consequently, $A$ wins the game with probability at most $2 \cdot 2^{-\frac{k}{8}} = 2^{1-\frac{k}{8}}$. 

7.4 Quasilinear Communication VSS and Generalizations of VSS

In this section we use ZKLCs to construct VSS protocols with quasilinear communication, which allow a dealer $D$ to distribute a secret $x$ among $m$ servers in a way that prevents a coalition of up to $\tau$ servers from learning or modifying the secret, while on the other hand guaranteeing unique reconstruction even if $D$ and up to $\tau$ servers can collude. We also use ZKPCPPs to implement a generalized certifiable version of VSS, which we call certifiable VSS, which has the additional guarantee that the shared secret satisfied some predicate. We first describe the model and VSS functionality in more detail.
The Model

We consider systems with a dealer $D$, a receiver $R$, and $m$ servers $S_1,\ldots,S_m$ (we refer to protocols in such systems as $m$-server protocols). The parties share a common broadcast channel, and every pair of parties share a private, authenticated, point-to-point channel.

The system also includes a $\tau$-adversary $A_\tau$ (where $\tau$ is a corruption threshold), who can non-adaptively corrupt a subset of (any) $\tau' \leq \tau$ servers, and either one of $D,R$ (or none of them, but not both). The non-corrupt parties (i.e. those who follow the protocol instructions) are called honest. The honest parties in our protocols are efficient, but the adversary is computationally unbounded, and assumes full control of corrupt parties. The adversary is also rushing, namely in every round he may send his messages after receiving all messages from the honest parties. Our protocols are parameterized by an error parameter $\epsilon$, which determines the error probability of the protocol. Particularly, we allow the properties of the protocol to hold statistically, except with probability at most $\epsilon$.

We consider the following designated receiver variant of VSS which (unlike the setting of “standard” VSS) involves, in addition to $D$ and the $m$ servers, the designated receiver $R$ who assists in the verification.

**Definition 7.4.1** (Designated-receiver Verifiable Secret Sharing). A designated-receiver VSS (VSS) scheme consists of three phases: Sharing, Verification and Reconstruction. (Each phase may consist of several rounds.)

- **Sharing**: initially, $D$ holds an input $x \in \mathcal{D}$ ($x$ is the secret, and $\mathcal{D}$ is a domain of secrets), and each party $D, S_1,\ldots,S_m, R$ holds a private random input $r_D,r_1,\ldots,r_m,r_R$. In the sharing phase $D$ distributes $x$ between the servers by sending a single message $s_i$ to each server $S_i$.

- **Verification**: $R$ interacts with the servers. At the end of the phase, $R$ holds some private information $V_R$.

- **Reconstruction**: Every server $S_i$ sends a single message $V_i$ to $R$, who applies a reconstruction function $\text{rec}$ to $V_R, V_1,\ldots,V_m$, and outputs $\text{rec}(V_R, V_1,\ldots,V_m)$.

**Remark 7.10.** In the standard definition of VSS, the primitive consists of only two phases: sharing and reconstruction (while the verification phase is considered as part of the sharing phase, and consists also of interaction between the servers, and $R$). To capture the efficient verification feature of our VSS protocols, we distinguish between the sharing and verification procedures.

Next, we formulate the notion of secure designated-receiver VSS protocols. We note that any $\tau$-adversary $A_\tau$ in a VSS protocol can be divided into a pair $A_{\text{share}}^\tau,A_{\text{rec}}^\tau$ of algorithms that determine its operation in each of the sharing and verification, and reconstruction phases, respectively. ($A_{\text{rec}}^\tau$ may also depend on the internal state of $A_\tau$ at the end of the sharing phase.)
Definition 7.4.2 \((\tau, \epsilon)\text{-VSS}\). Let \(m \in \mathbb{N}\) denote the number of servers, \(\tau \in [m]\) be a corruption threshold, and \(\epsilon \in (0, 1)\) be an error parameter. We say that an \(m\)-server three-phase protocol (as in Definition 7.4.1) is a \((\text{statistical}) (\tau, \epsilon)\text{-designated-receiver VSS}\) or \((\tau, \epsilon)\text{-VSS}\) in short) if the following holds for every \(\tau\)-adversary \(A^\tau = (A^\tau_{\text{share}}, A^\tau_{\text{rec}})\).

- **Correctness**: if \(A^\tau\) corrupts neither \(D\) nor \(R\), then for every input \(x\) of \(D\), \(R\) outputs \(x\) at the end of the reconstruction phase except with probability at most \(\epsilon\).

- **Secrecy**: if \(A^\tau\) does not corrupt \(D\), then for every pair of secrets \(x', x''\), \(\text{SD}(V^\text{share}_{A^\tau}(x'), V^\text{share}_{A^\tau}(x'')) \leq \epsilon\), where \(V^\text{share}_{A^\tau}(x)\) denotes the view of \(A^\tau\) at the end of the verification phase, when \(D\) has input \(x\).

- **Binding**: for any \(A^\tau_{\text{share}}\) that does not corrupt \(R\), the following holds except with at most \(\epsilon\) failure probability over the randomness of the sharing and verification phases. In the end of the verification phase there is a unique secret \(x^*\) (determined by the messages exchanged up to this point between \(R, A^\tau_{\text{share}}\) and the honest servers), such that if \(A^\tau_{\text{rec}}\) does not corrupt \(R\), then \(R\) will output \(x^*\) regardless of the messages sent by the adversary during the reconstruction phase.

It is instructive to note that if the binding requirement is relaxed so that \(R\) is only guaranteed to either output \(x^*\) or reject (in a way that may depend on the adversary’s messages during reconstruction) then the problem becomes much easier to solve \([BR89, FGG+06]\). This weaker variant, sometimes referred to as weak VSS or distributed commitment, does not suffice for several applications of VSS including the coin-flipping protocols we present in Section 7.5. On the other hand, traditional VSS is stronger than our designated receiver variant in that the verification phase does not involve the receiver \(R\). Thus, when there are multiple receivers, traditional VSS can guarantee that the same secret \(x^*\) be reconstructed by all receivers, whereas applying our VSS verification with each receiver separately does not. Still, designated receiver VSS is as good as traditional VSS in situations where agreement between different receivers is not required, as in the coin-flipping application described in Section 7.5. We are not aware of any simpler or better solutions to the problem of designated receiver VSS using previous VSS techniques from the literature. Moreover, our generalization of VSS described below (Definition 7.4.4) has the additional guarantee that the reconstructed secret satisfies some NP predicate, whereas traditional VSS protocols do not offer this guarantee.

Next, we describe a designated-receiver certifiable variant of VSS, in which the dealer \(D\) not only commits to some secret \(x\), but also proves that it satisfies some predicate. (For example, the predicate could be “\(x \in L_\mathcal{R}\)” for a relation \(\mathcal{R}\).) Concretely,

**Definition 7.4.3** (Certifiable VSS). Let \(\mathcal{R} = \mathcal{R}(x, w)\) be an NP-relation. We say that a designated-receiver VSS scheme (as in Definition 7.4.1) is a certifiable VSS if it is...
carried out in 3 phases between the dealer $D$, the receiver $R$, and $m$ servers $S_1, \ldots, S_m$ (as described in Definition 7.4.1), except that the dealer has input $(x, w) \in \mathcal{R}$, and not only a secret $x$. (We note that the goal is still that $R$ would reconstruct $x$, and not $(x, w)$.)

Next, we define the notion of secure certifiable VSS protocols.

**Definition 7.4.4** ($((t, \epsilon)$-secure certifiable VSS). Let $m \in \mathbb{N}$ denote the number of servers, $\tau \in [m]$ be a corruption threshold, and $\epsilon \in (0, 1)$ be an error parameter. We say that an $m$-server three-phase protocol (as in Definition 7.4.4) is a (statistical) $(\tau, \epsilon)$-certifiable VSS if the following holds for every $\tau$-adversary $A^\tau = (A^\tau_{\text{share}}, A^\tau_{\text{rec}})$.

- **Correctness**: if $A^\tau$ corrupts neither $D$ nor $R$, then for every input $(x, w) \in \mathcal{R}$ of $D$, $R$ reconstructs $x$ except with probability at most $\epsilon$.

- **Secrecy**: if $A^\tau$ does not corrupt $D$, then there exists a simulator $\text{Sim}$ such that for every $(x, w) \in \mathcal{R}$, $\text{SD}(\text{Sim}(|x|), V^\text{share}_{A^\tau}(x, w)) \leq \epsilon$, where $V^\text{share}_{A^\tau}(x, w)$ denotes the view of $A^\tau$ at the end of the verification phase when $D$ has input $(x, w)$.

- **Binding**: for any $A^\tau_{\text{share}}$ that does not corrupt $R$, the following holds except with at most $\epsilon$ failure probability over the randomness of the sharing and verification phases. At the end of the verification phase there is a unique secret $x^* \in L^R$ (determined by the messages exchanged up to this point between $R, A^\tau_{\text{share}}$ and the honest servers), such that if $R$ did not abort at the end of the verification phase, then after the reconstruction phase $R$ will output $x^*$ regardless of the messages sent by the adversary during the reconstruction phase.

**Notation 7.11.** We say that a (certifiable) VSS protocol is sublinear, if the communication complexity during the verification phase is $\text{polylog}(\epsilon, \tau, n)$, where $n$ denotes the input length.

### 7.4.1 Quasilinear Communication VSS from ZKLTCs

In this section we construct statistically-secure sublinear designated-receiver VSS protocols which tolerate a constant fraction of corrupted servers, and have error that vanishes almost exponentially with the number of servers $m$, while using only $O(m)$ bits of communication. Concretely, we prove the following.

**Theorem 7.12** (Efficient $(\tau, \epsilon)$-VSS). For every error parameter $\epsilon \in (0, 1)$, every constant $\gamma \in (0, 1)$, and every $m \in \mathbb{N}$, there exists a 4-round $m$-server designated-receiver $(\tau, \epsilon)$-VSS for secrets in $\{0, 1\}^n$, $n = \Omega_\gamma(m)$, with total communication complexity $O(m) + m^\gamma \cdot \text{polylog} \frac{1}{\epsilon}$, and verification communication complexity $n^\gamma \cdot \text{polylog} \frac{1}{\epsilon}$, where $\tau = \Omega_\gamma(n)$. 

189
Using Theorem 7.12 we can obtain a VSS protocol in which the communication complexity in the verification phase is sublinear in the secrecy threshold \( \tau \). The efficient verification feature is captured by the following corollary.

**Corollary 7.13 (Sublinear-communication VSS).** For every constant \( \gamma > 0 \) there exists a constant-round, \( m \)-server, designated receiver \( (\tau, \epsilon) \)-VSS protocol for secrets of length \( \Omega_\gamma(m) \), where \( \tau = \Omega_\gamma(m) \), and \( \epsilon = m^{-\omega(1)} \). Moreover, the verification phase uses \( O(m^\gamma) \) bits of communication, and the protocol has perfect secrecy.

We begin by describing the VSS protocol which employs a stable ZKLTC. Then, we instantiate our protocol with the stable ZKLTC of Theorem 6.5 to obtain a protocol with the properties stated in Theorem 7.12. We note that since the ZKLTCs that we use are stable, then there is no need to employ the techniques of Section 7.3.

**Protocol 7.14.** The protocol is parameterized by the number of servers \( m \), and an error parameter \( \epsilon \in (0, 1) \) (we assume without loss of generality that \( \log \frac{1}{\epsilon} \in \mathbb{N} \)). Let \( C \) be a ZKLTC with the encoding function \( \text{Enc}_C : \mathbb{F}^n \times \mathbb{F}^{n'} \rightarrow \mathbb{F}^m \), tester \( T \), and decoding algorithm \( \text{Dec}_C \). The VSS protocol consists of the following phases.

1. **Sharing:** the dealer \( D \) on input \( x \in \mathbb{F}^n \), chooses \( r \in_R \mathbb{F}^{n'} \) and generates the codeword \( c_{\text{xor}} = \text{Enc}_C(x \circ r) \). \( D \) distributes the symbols of \( c_{\text{xor}} \) between the servers \( S_1, \ldots, S_m \) (i.e., the server \( S_i \) receives \( c_i \in \mathbb{F} \)).

2. **Verification:** \( R \) and the servers emulate in parallel \( k := 8 \log \frac{1}{\epsilon} \) random and independent runs of \( T \), where every run is executed as follows.
   - **Generating the queries:** \( R \) broadcasts a random string \( r' \) for \( T \). Let \( I \subseteq [m] \) denote the set of queries that \( T \) makes when using randomness \( r' \).
   - **Determining the answers:** for every \( i \in I \), \( S_i \) (privately) sends \( c_i \) to \( R \). (If \( r' \) is not a legal random string of \( T \), the servers do not answer.) For every server \( S_i \) such that \( i \in I \), but \( S_i \) did not send \( c_i \) to \( R \), \( R \) sets the answer of \( S_i \) to 0.
   - **Determining simulation output:** for every run of \( T \), \( R \) verifies the run was successful (i.e., \( T \) accepted). If at least \( \frac{k}{2} \) of the runs were unsuccessful, then \( R \) outputs 0 and halts.

3. **Reconstruction:** every \( S_i \) sends \( c_i \) to \( R \), and let \( y_1, \ldots, y_m \) be the messages received from the servers (if a server \( S_i \) did not send a message to \( R \), then \( R \) sets \( y_i = 0 \)). Define \( c' \) as follows. \( c'_i = y_i \) for every \( i \notin I \), otherwise \( c'_i \) is set to the message that \( S_i \) sent to \( R \) during the sharing phase. \( R \) computes \( c_{x' \circ r'} = \text{Dec}_C(c') \), and outputs \( x' \).

The following claim summarizes the connection between the security of Protocol 7.14, and the properties of the underlying ZKLTC.
Claim 7.4.5. Let $\mathbb{F}$ be a finite field, $\epsilon \in (0,1)$ be an error parameter, $m \in \mathbb{N}$ be a party number parameter, $q \in \mathbb{N}$ be a query parameter, and $\tau, \tau_{VSS} \in \mathbb{N}$ be corruption thresholds such that $\tau_{VSS} \leq \tau \leq m$. Let $C$ be a $q$-query LTC with tester $T$, and encoding function $\text{Enc}_C : \mathbb{F}^m \times \mathbb{F}^m \rightarrow \mathbb{F}^m$ (for some $k_1, k_2 \in \mathbb{N}$), such that:

- $C$ is $\tau$-ZK with respect to $\text{Enc}_C$.
- $C$ is efficiently decodable from $\alpha m$ errors, for some constant $\alpha \in (0,1)$.
- $C$ is $(\tau_{VSS}, \frac{1}{4}, \delta)$-stable for some constant $\delta \in (0,1)$ (when $T$ is used as the tester).
- $\delta \leq \alpha - \frac{\tau_{VSS}}{m}$ and $\tau \geq \tau_{VSS} + 8 \log \frac{1}{c} \cdot q$.

Then Protocol 7.14, when instantiated with the code $C$, is an $m$-server $(\tau_{VSS}, \epsilon)$-VSS for secrets in $\mathbb{F}^m$, with total communication complexity $O\left(m + \log \frac{1}{\epsilon} \cdot (r + q)\right)$, and verification communication complexity $O\left(\log \frac{1}{\epsilon} \cdot (r + q)\right)$, where $r$ denotes the randomness complexity of $T$. Moreover, the protocol has perfect secrecy.

Before proving Claim 7.4.5, we first use it to prove Theorem 7.12, and Corollary 7.13.

Proof of Theorem 7.12. Theorem 6.5 guarantees that there exists a constant $\alpha$ such that for every constants $\delta \in (0, \alpha)$, and $\beta > 0$, there exists a ZKLTC $C \subseteq \mathbb{F}^m$ with encoding function $\text{Enc}_C$, that is $\tau$-ZK (when encoding is done using $\text{Enc}_C$) for $\tau = \frac{m}{c}$ (where $c = c(\beta)$ is a constant, and $m$ is the codeword length), message length $n = \Omega_\beta(m)$, randomness complexity $O(\log m)$, and query complexity $q = m^\beta = (c\tau)^\beta = c'(m^\beta)$ (for $c' := c^\beta$). Moreover, the code can efficiently correct up to $\alpha m$ errors and it is $(\tau, \frac{1}{4}, \delta)$-stable.

Let $k := 8 \log \frac{1}{c}$, and $\tau_{VSS} = \alpha \tau - (16k) \frac{1}{m^\gamma}$ for $\beta < \min \{\frac{1}{2}, \gamma\}$. Then for a large enough $m$, $\tau \geq \tau_{VSS} + 8 \log \frac{1}{c} \cdot q$. Let $\delta > 0$ to be a sufficiently small constant such that $\delta \leq \alpha - \frac{\tau_{VSS}}{m}$.

Then Claim 7.4.5 guarantees that when Protocol 7.14 employs $C$ as the underlying code, then for every large enough $m$ (namely, every $m$ such that $\tau_{VSS} \geq 0$), the protocol is an $m$-server $(\tau_{VSS}, \epsilon)$-VSS with total communication complexity $O\left(m + k \cdot (\log m + m^\beta)\right) \leq O(m) + \log \frac{1}{c} \cdot m^\gamma$ (for every large enough $m$ such that $\log m \leq m^\gamma$, verification communication complexity $O\left(k \cdot (\log m + m^\beta)\right) \leq O\left(m^\gamma \cdot \log \frac{1}{c}\right)$ (again, if $\log m \leq m^\gamma$), and perfect secrecy. Moreover, by our choice of $\tau_{VSS}$, $m = O(\tau) = O\left(\tau_{VSS} + k \frac{1}{m^\gamma}\right)$ which, since $\beta < \min \{\frac{1}{2}, \gamma\}$, implies that $\tau_{VSS} = \Omega_\beta(m) = \Omega_\gamma(m)$.

Next, we prove Corollary 7.13.

Proof of Corollary 7.13. We assume without loss of generality that $\gamma < 1$ (otherwise Corollary 7.13 follows from known VSS protocols, e.g. the protocol of [DI06a]). Let $\epsilon = 2^{-m^\beta} = m^{-\omega(1)}$, then we instantiate Theorem 7.12 with the constants $\frac{1}{2}, \epsilon$. Then Theorem 7.12 guarantees the existence of a 3-round, $m$-server designated-receiver...
(τ, ε)-VSS for secrets of length Ω2 (m) = Ωγ (m), where τ = Ω2 (m) = Ωγ (m), with verification communication complexity O \left( m^2 \cdot \log \frac{1}{ε} \right) = O (m^2), and perfect secrecy. □

To prove Claim 7.4.5, we use the following lemma which, roughly, states that the stability of the ZKLTC code is amplified through the repeated testing performed in Protocol 7.14. We emphasize that the lemma discusses only the case in which R is honest during the execution.

**Lemma 7.4.6.** Let ε ∈ (0, 1) be an error parameter, m ∈ N be a codeword length parameter, and τ_{VSS} ∈ N be a corruption threshold. Let α ∈ (0, 1) be a constant, and C be a (τ_{VSS}, 1/4, δ)-stable code, where δ ≤ α − τ_{VSS} / m. Let c be the purported codeword that D partitioned among the servers during the sharing phase of Protocol 7.14. Then:

- If c ∈ C, then R outputs 0 at the end of the sharing phase with probability at most ε.
- If c is δ-far from C, then R outputs 0 at the end of the sharing phase, except with probability at most ε.

**Proof.** Let k := 8 log \frac{1}{ε}, and assume without loss of generality the adversary is deterministic during the sharing phase. If c = c_{xor} ∈ C, then since at most τ_{VSS} servers are corrupt, and every corrupted server “controls” the value of exactly one coordinate of c_{xor}, the (τ_{VSS}, 1/4)-correctness of X guarantees every emulation of T is successful, except with probability at most \frac{1}{4}.

Let X_i be indicator of the event that the i'th emulation failed, then Pr[X_i = 1] = p ≤ \frac{1}{4}. Moreover, the X_i's are random and independent, because R is honest. Using Hoeffding’s inequality (Theorem 7.15 below), R outputs 0 at the end of the sharing phase with probability at most by

\[ \Pr \left[ \sum_{i=1}^{k} X_i \geq \frac{k}{2} \right] \leq \Pr \left[ \sum_{i=1}^{k} X_i \geq \left( p + \frac{1}{4} \right) \cdot k \right] \leq e^{-2 \left( \frac{1}{4} \right)^2 \cdot k} = 2^{-\log \frac{1}{ε}} = ε. \]

Second, if c is δ-far from C, then a similar argument (using the (τ_{VSS}, 1/4, δ)-soundness of C) shows that every emulation of T fails, except with probability at most \frac{1}{4}. Letting X_i be the indicator of the event that the i'th emulation succeeded, Pr[X_i = 1] = p ≤ \frac{1}{4} where the X_i's are random and independent. We conclude that R outputs 0 at the end of the sharing phase, except with probability at most

\[ \Pr \left[ \sum_{i=1}^{k} X_i \geq \frac{k}{2} \right] \leq ε. \]

The proof of Lemma 7.4.6 used the following version of the Hoeffding inequality.

**Theorem 7.15 (Hoeffding inequality).** Let X_1, X_2, ..., X_k ∈ {0, 1} be independent, identically distributed random variables, and let p = E[X_i]. Then for every ε > 0,

\[ \Pr \left[ \sum_{i=1}^{k} X_i \geq (p + ε) \cdot k \right] \leq e^{-2ε^2 k}. \]
Finally, we prove Claim 7.4.5.

Proof of Claim 7.4.5. Denote $k := 8 \log \frac{1}{\epsilon}$. We prove that Protocol 7.14 has correctness, secrecy and binding.

Correctness: Let $x \in \mathbb{F}^n$ be the input of $D$, and let $A$ be an adversary corrupting at most $\tau_{VSS}$ servers in Protocol 7.14. $D$ is honest, so the views $V_1, \ldots, V_m$ of $S_1, \ldots, S_m$ at the end of the sharing phase, correspond to a codeword $c_{xor} = \text{Enc}_C(x \circ r) \in C$, where $r \in_R \mathbb{F}^{n'}$.

Therefore, Lemma 7.4.6 guarantees that the sharing phase passes (i.e., $R$ does not halt with output 0 at the end of the sharing phase), except with probability at most $\epsilon$. Next, we show that conditioned on the event that the sharing phase passed, $R$ outputs $x$ (with probability 1). Let $c''$ be the (possibly corrupted) codeword defined by the answers of the servers during the sharing and reconstruction phases (i.e., $c''$ is obtained from $c_{xor}$, and the answers of corrupted servers). Throughout the protocol, at most $\tau_{VSS}$ servers are corrupted, so $\Delta(e'', c_{xor}) \leq \tau_{VSS} \leq \alpha m$, i.e. $\text{Dec}_C(c'') = c_{xor}$ (because the code $C$ can correct up to an $\alpha$-fraction of errors), so $R$ outputs $x$.

Secrecy: Let $A$ be a $\tau_{VSS}$-adversary that does not corrupt $D$, and let $B \subseteq \{S_1, \ldots, S_m\}$ be the subset of $\tau'$ \leq $\tau_{VSS}$ corrupted servers. Denote the input of $D$ by $x \in \mathbb{F}^n$.

The view of $A$ at the end of the verification phase, consists of the views $\{V_i\}_{S_i \in B}$ (and the view $V_R$ of $R$, if $A$ also corrupts $R$). Every view $V_i$ contains the message $c_i$ that $D$ sent to $S_i$ during the sharing phase, and the random strings $R = (r_1, \ldots, r_k)$ that $R$ broadcasted during the verification phase. Moreover, $D$ is honest so there exists a codeword $c = c_{xor} = \text{Enc}_C(x \circ r)$ (where $r \in_R \mathbb{F}^{n'}$) such that every $c_i, i \in [n]$ is a coordinate of $c$. The view of $R$ consists of his broadcasted messages, and the answers of the servers. Particularly, if $A$ corrupts $R$ then the view of $A$ consists of $c_i$ for every $S_i \in B$, and for every $S_i$ who answered $R$. Otherwise, the view of $A$ does not contain the answers that $R$ got. Let $I_R \subseteq [n]$ denote the codeword coordinates that the honest tester $T$ reads when it uses one of the random strings $r_1, \ldots, r_k$ that $R$ chose for the execution.

Let $T := \{c_i : S_i \in B \lor i \in I_R\}$. As $|T| \leq |B| + |I_R| \leq \tau_{VSS} + 8q \cdot \frac{1}{\epsilon} \leq \tau$, the $\tau$-zero-knowledge of $C$ (with encoding function $\text{Enc}_C$) guarantees that for every $x \in \mathbb{F}^n$, $y|_T \equiv y_0|_T$, where $y = \text{Enc}_C(x, r')$, and $y_0 = \text{Enc}_C(0^k, r_0)$ for $r', r_0 \in_R \mathbb{F}^{n'}$. Since $D$ encodes $x$ using a random $r \in_R \mathbb{F}^n$, we conclude that for every pair $x', x''$ of secrets, $y'|_T \equiv y_0|_T \equiv y''|_T$, where $y' = \text{Enc}_C(x', r')$, $y'' = \text{Enc}_C(x'', r'')$, and $y_0 = \text{Enc}_C(0^k, r_0)$ for $r', r'' , r_0 \in_R \mathbb{F}^{n'}$. Consequently, secrecy holds because the view of $A$ can be fully reconstructed from $y|_T$.

Binding: Let $A$ be an adversary corrupting $D$ and at most $\tau_{VSS}$ servers. Let $c_1, \ldots, c_m$ denote the field elements that $D$ sent to $S_1, \ldots, S_m$ during the sharing phase (if $D$ did not send a message to $S_i$, we take $c_i = 0$). Then $c_1, \ldots, c_m$ define a (possibly corrupt) codeword $c$, and we consider two possible cases.
First, if $c$ is $\delta'$-far from $C$ for $\delta' = \alpha - \frac{\tau_{VSS}}{m}$, then Lemma 7.4.6 guarantees that $R$ outputs 0 at the end of the sharing phase, except with probability at most $\epsilon$ (i.e., binding holds for $x^* = 0$).

Second, if $\delta(c, c^*) \leq \delta'$ for some $c^* = \text{Enc}_C(s^* \circ r^*) \in C$, then we assume that $R$ did not halt with output 0 at the end of the sharing phase (this is without loss of generality, otherwise binding holds for $x^* = 0$). Let $r_1, \ldots, r_k$ be the messages that $R$ broadcasted, and denote by $I_R \subseteq [m]$ the set of the codeword coordinates that the honest tester $T$ reads when using one of the random strings $r_1, \ldots, r_k$.

Let $c''$ be the (possibly corrupt) codeword defined as follows. For every $i \in I_R$ such that $S_i$ sent $R$ a message during the sharing, $c''_i$ is set to that answer; otherwise $c''_i$ is the message that $S_i$ sent to $R$ during the reconstruction phase ($c''_i = 0$ if $S_i$ sent no message). During the entire protocol, at most $\tau_{VSS}$ servers are corrupt, so $\Delta(c, c'') \leq \tau_{VSS}$ which implies that $\Delta(c^*, c'') \leq \delta'm + \tau_{VSS} \leq \alpha m$. Therefore, $\text{Dec}_C(c'') = c^* = \text{Enc}_C(x^* \circ r^*)$, and $R$ outputs $x^*$, regardless of the messages sent by corrupted servers during the reconstruction phase.

### 7.4.2 Verification-Efficient Certifiable VSS from ZKPCPPs

In this section we use ZKPCPPs to construct certifiable VSS protocols for NP in which the verification phase involves sublinear communication, and prove Theorem 7.3.

Let $(\text{Share}, \text{Rec})$ be a robust, and private, secret sharing scheme. Given an NP relation $\mathcal{R}$, let $\text{Share}(L_{\mathcal{R}})$ denote the set of all tuples of secret shares of $x \in L_{\mathcal{R}}$. Roughly speaking, the certifiable VSS protocol operates as follows. $D$ secret shares its input $x \in L_{\mathcal{R}}$ using $\text{Share}$, obtaining a secret sharing $c$, and distributes $c$, together with a ZKPCPP for the claim “$c \in \text{Share}(L_{\mathcal{R}})$”, between the servers. The receiver then interacts with the servers as in Protocol 7.14 to emulate the ZKPCPP verifier. If verification succeeded, then during reconstruction all servers send their shares to the receiver, who uses them to reconstruct the secret $x$. As noted in Section 7.3, we need to use a “smooth” ZKPCPP system to guarantee that correctness and soundness is preserved even in the presence of malicious servers. We first describe the robust secret sharing scheme which we employ.

### A Robust Secret Sharing Scheme

We describe a $t$-private, and $\tau$-robust, secret sharing scheme, which is based on the Reed-Solomon code, which we define next.

**Notation 7.16.** We use $\text{ECC}_\Sigma[n, \hat{n}, d_C]$ to denote the family of all error-correcting codes $C \subseteq \mathbb{F}^{\hat{n}}$ with dimension $n$, and distance $d_C$.

**Definition 7.4.7** (Reed-Solomon error-correcting code). Let $n \in \mathbb{N}$ be an input length parameter, and $\mathbb{F}$ be a finite field of size $\hat{n}$. The Reed-Solomon code $\text{RS}(n, \mathbb{F})$ is defined
by the following encoding function: \( \text{Enc}_{RS} : \mathbb{F}^n \rightarrow \mathbb{F}^\hat{n}, \)

\[
\text{Enc}_{RS} (m_0, m_1, \ldots, m_{n-1}) = \left( \sum_{i=0}^{n-1} m_i x^i \right)_{x \in \mathbb{F}}
\]

We denote the reconstruction algorithm of the code by \( \text{Dec}_{RS} \). We will need the following known facts regarding Reed-Solomon codes.

**Fact 7.4.8.** Let \( n \in \mathbb{N} \), and \( \mathbb{F} \) be a finite field such that \( |\mathbb{F}| = \hat{n} \geq n \). Then \( \text{RS} (n, \mathbb{F}) \in \text{ECC}_{\mathbb{F}} [n, \hat{n}, \hat{n} - n + 1] \). Moreover, \( \text{Dec}_{RS} \) can efficiently correct up to \( \frac{\hat{n} - n}{2} \) errors.

**Fact 7.4.9.** Let \( \mathbb{F} \) be a finite field of size \( \hat{n} \). For \( n, t \in \mathbb{N}, n + t \leq \hat{n} \), let \( \text{Enc}_{RS} \) be the generating function of the code \( \text{RS} (n + t, \mathbb{F}) \). The following holds for every \( x \in \mathbb{F}^n \), and every subset \( \{i_1, \ldots, i_{t'} \} \) of \( t' < t \) distinct elements in \([\hat{n}]\). Let \( y = \text{Enc}_{RS} (x, r) \), where \( r \in R \mathbb{F}^t \), then \( \{y_{i_1}, \ldots, y_{i_{t'}}\} \subseteq R \mathbb{F}^{t'} \).

We now use the Reed-Solomon code to construct a robust and private secret sharing scheme.

**Definition 7.4.10.** Let \( n \in \mathbb{N} \) be an input length parameters, and \( \hat{n}(n), t(n) : \mathbb{N} \rightarrow \mathbb{R}^+ \). Let \( \mathbb{F}_n \) be a finite field of size \( \hat{n}(n) \), and denote \( \text{RS}_n := \text{RS}(t(n) + n, \mathbb{F}_n) \) (namely, \( \text{RS}_n \) is the Reed-Solomon code for messages of length \( t(n) + n \) over the \( \mathbb{F}_n \)). The secret sharing scheme \((\text{Share}, \text{Rec})\) is defined as follows. \( \text{Share}(x, r) = \text{Enc}_{RS_n}, \) namely the secret shares are taken to be the codeword symbols. (More specifically, the dealer in the secret sharing scheme interprets the secret \( x \in \{0, 1\}^n \) as an element of \( \mathbb{F}_n^n \).) \( \text{Rec}(c) \) evaluate \( \text{Dec}_{RS_n}(n) \), and disregard the last \( t(n) \) symbols.

**Observation 7.4.11.** Since each share can be represented using \( \log \hat{n}(n) \) bits, then the total length of all shares is \( \ell = \hat{n}(n) \log \hat{n}(n) \). Moreover, by Fact 7.4.9 every \( t' \leq t(n) - 1 \) shares (i.e., \( t' \) symbols of the codeword) are randomly distributed in \( \mathbb{F}_n^{t'} \), and the error correction (Fact 7.4.8) guarantees that the scheme is \( \tau(n) \)-robust for \( \tau(n) = \frac{\hat{n}(n) - (t(n) - n)}{2} \).

**A Certifiable VSS Protocol**

The certifiable VSS protocol will use a robust and private secret sharing scheme, and a stable ZKPCPP system (as described in Section 7.3). Intuitively, the robustness of the secret sharing scheme, together with the stability of the ZKPCPP system, will guarantee that (except with negligible probability) corrupted servers cannot influence the verification procedure of the ZKPCPP system. The privacy of the encoding scheme will guarantee that the secret remains *entirely* hidden until the reconstruction phase, even though the ZKPCPP verifier may query its input oracle. Indeed, in our protocol the secret shares constitute the input oracle of the verifier, so the ZK property of the
ZKPCPP guarantees that the verification reveals only few secret shares which, by the privacy of the secret sharing scheme, reveal nothing about the hidden secret.

In the following, for \( y \in \mathbb{F}^n \) and \( I \subseteq [n] \), we denote \( y|_I := (y_i)_{i \in I} \). Given a \( t(n) \)-private secret sharing scheme \((\text{Share}, \text{Rec})\) in which on input a secret of length \( n \), \text{Share} outputs a tuple of \( \hat{n}(n) \) secret shares, we denote \( \text{supp}(\text{Share}_n) := \{ y \in \mathbb{F}^{\hat{n}(n)} : \exists x \in \mathbb{F}^n, r \in \mathbb{F}^{\hat{n}(n)} \text{ s.t. } y = \text{Share}(x, r) \} \).

**Protocol 7.17.** Let \( \mathcal{R} = \mathcal{R}(x, w) \) be an NP relation, and \((\text{Share}, \text{Rec})\) be a secret sharing scheme that is \( \tau(n) \)-robust, and \( t(n) \)-private. Let\(^1\)

\[
\mathcal{R}_{\text{Share}} = \{ ((y, n), w) : y \in \text{supp}(\text{Share}_n), \text{Rec}(y)|_{[n]} \in \{0, 1\}^n, (\text{Rec}(y)|_{[n]}, w) \in \mathcal{R} \}.
\]

Let \((P_{\text{ZK}}, V_{\text{ZK}})\) be a ZKPCPP system for \( \mathcal{R}_{\text{Share}} \),\(^2\) and \((P_{\text{stable}}, V_{\text{stable}})\) be the stable system obtained from \((P_{\text{ZK}}, V_{\text{ZK}})\) through Construction 7.7.

The certifiable VSS protocol is parameterized by an error parameter \( \epsilon \), and a parameter \( k \in \mathbb{N} \) that determines how many emulations of \( V_{\text{stable}} \) are performed during the verification phase. Let \( q \) denote the query complexity of \( V_{\text{stable}} \). The protocol consists of the following phases.

- **Sharing.** The dealer \( D \) on input \( (x = (x_1, \ldots, x_n), w) \in \mathcal{R} \) operates as follows.
  - Interprets every bit \( x_i \) as an element in \( \mathbb{F} := \mathbb{F}_n \), chooses \( r \in_R \mathbb{F}^{\hat{n}(n)} \), and generates a secret sharing \( c = \text{Share}(x, r) \).
  - Generated a proof \( \pi \in P_{\text{stable}}(1^k, 1^{[\epsilon]}, 1^{[\delta(n)]}, q^*, (c, n), w) \) for the claim 
    
    \[ \langle c, n \rangle \in L_{\mathcal{R}_{\text{Share}}} \]  
    
    where \( \delta = \frac{1}{\log n} \), \( q^* = t \), and \( l, k \) are parameters for the underlying stable proof system, whose value will be set later. Recall that \( \pi = (c^2, \ldots, c^l, \pi_{\text{ZK}}^2, \ldots, \pi_{\text{ZK}}^l) \) where \( c^2, \ldots, c^l \) are copies of \( c \), and \( \pi_{\text{ZK}}^2, \ldots, \pi_{\text{ZK}}^l \) are copies of a proof \( \pi_{\text{ZK}} \) generated by \( P_{\text{ZK}} \).
  - Distributes the symbols of \( c, \pi \) between the servers \( S_1, \ldots, S_n \) (\( c \) is represented in its binary representation, and every server receives a distinct bit of \( c \) or of \( \pi \)).\(^3\)

- **Verification.** \( R \) and the servers emulate the verification procedure of \( V_{\text{stable}}(1^k, [\epsilon], 1^{[\delta(n)]}, q^*, \hat{n}(n)) \) with oracles \( c, \pi \), as follows:
  - \( R \) picks a random string \( r \) for \( V_{\text{stable}} \), and broadcasts it.
  - Each server uses \( r \) to determine the first round of queries that \( V_{\text{stable}} \) makes when using random string \( r \). The servers holding the corresponding bits broadcast their value.

\(^1\) We note that \( \mathcal{R}_{\text{Share}} \) is defined over \( \{0, 1\}^* \times \{0, 1\}^* \), that is, strings over larger alphabets are interpreted using their binary representation. This will be of importance later on when we discuss distances from \( \mathcal{R}_{\text{Share}} \).

\(^2\) Notice that \( \mathcal{R}_{\text{Share}} \in \mathcal{P} \), and therefore has a ZKPCPP system as in Theorem 5.22.

\(^3\) In principle, \( n \) should also be divided between the servers, since the verifier \( V_{\text{stable}} \) may query this part of his input. However, the input length \( n \) is known to all parties, so \( R \) can locally answers queries of \( V_{\text{stable}} \) to \( n \).
Then Protocol 7.17 is $\epsilon$-construction 7.7, with parameters $(\text{error function})$. Let $q$, $n$, and $\tau$ be an adversary corrupting at most $4 \epsilon N$ servers (but not the dealer!) in Protocol 7.17. Moreover, let $\pi$ denote the combined length (in binary representation) of a secret sharing $m$ for the claim $"(c,n) \in L_{\text{Share}}"$. Let $\mathcal{A}$ be an adversary corrupting at most $\tau$ servers (but not the dealer!) in Protocol 7.17.

Since $D$ is honest, the views $V_1, \ldots, V_m$ of $S_1, \ldots, S_m$ at the end of the sharing phase correspond to a sharing $c$ of $x$, and an honestly-generated proof $\pi$ for $c$. Moreover, by the definition of $P_{\text{stable}}$, $\pi = (c^2, \ldots, c^l, \pi^1, \ldots, \pi^{l})$, where $c^2, \ldots, c^l$ are copies of $c$, and $\pi^1, \ldots, \pi^{l}$ are copies of an honestly-generated proof $\pi_{\text{ZK}}$ for the claim $"(c,n) \in L_{\text{Share}}"$. During the verification phase, $R$ and the servers emulate $V_{\text{stable}}$ on oracles $c', \pi' = (c'^2, \ldots, c'^{l}, \pi'^1, \ldots, \pi'^{l})$, where $c', \pi'$ are obtained from $c, \pi$ through the answers for every bit that was queried during the emulations, but the corresponding server did not answer, $R$ sets that bit to 0. After all queries have been answered, $R$ verifies that $V_{\text{stable}}$ accepted in all emulations, otherwise he halts with output $\perp$.

**Reconstruction.** All the servers holding the bits of $c$, and the bits of the copies $c^2, \ldots, c^l$ of $c$, send $R$ the bits they received from $D$ during the sharing phase, and let $y_1, \ldots, y_{\hat{n}(n)} \log \hat{n}(n), y_1^2, \ldots, y_{\hat{n}(n)}^2 \log \hat{n}(n), \ldots, y_1^L, \ldots, y_{\hat{n}(n)}^L \log \hat{n}(n)$ denote these messages (every $y_i, y_i^j$ that was not received during this phase is set to 0). Define $c = c^1, c^2, \ldots, c^l$ as follows: if $c_i^j$ was queried during the verification phase, then $c_i^j$ is taken to be that value; otherwise, $c_i^j := y_i^j$. Let $c^* \in \{0, 1\}^{\hat{n}(n) \log \hat{n}(n)}$. Let $e^* \in \{0, 1\}^n \log \hat{n}(n)$ denote the string in which every bit $e^*_j$ is defined as the majority vote over the bits $c_j^1, \ldots, c_j^l$. $R$ interprets $e^*$ as an element of $\mathbb{F}^{\hat{n}(n)}$, and computes $(x^*, r^*) = \text{Rec}(c^*)$. If $x^* \in \{0, 1\}^n$ then $R$ outputs $x^*$, otherwise he outputs $\perp$.

In the following sequence of lemmas, we analyze the correctness, binding, and secrecy, of Protocol 7.17.

**Lemma 7.4.12 (Correctness).** Let $\mathcal{R}$ be an NP relation, $n \in \mathbb{N}$ be an input length parameter, $\tau : \mathbb{N} \to \mathbb{N}$ be a security threshold parameter, and $\epsilon : \mathbb{N} \to [0, 1)$ be an error function. Let $(\text{Share}, \text{Rec})$ be a secret-sharing scheme, $(F_{\text{ZK}}, V_{\text{ZK}})$ be a ZKPCPP system for $\mathcal{R}_{\text{Share}}$ with perfect completeness, where the honest verifier $V_{\text{ZK}}$ has query complexity $q$. Let $(P_{\text{stable}}, V_{\text{stable}})$ be the stable system obtained from $(F_{\text{ZK}}, V_{\text{ZK}})$ through Construction 7.7, with parameters $k = \max\{16, 8 \log 2 \epsilon^{-1}\}$, and $l = \max\{2^6 \cdot \tau, k\}$. Then Protocol 7.17 is $\epsilon$-correct, even if $\tau' \leq \tau$ servers are corrupted.

**Proof.** Let $(x, w) \in \mathcal{R}$ be the input of $D$, where $n = |x|$, and denote $\hat{n} = \hat{n}(n)$. Let $m$ denote the combined length (in binary representation) of a secret sharing $c$ for $x$, together with an honestly generated proof of $P_{\text{stable}}$ for the claim $"(c,n) \in L_{\text{Share}}"$. Let $\mathcal{A}$ be an adversary corrupting at most $\tau$ servers (but not the dealer!) in Protocol 7.17.

Since $D$ is honest, the views $V_1, \ldots, V_m$ of $S_1, \ldots, S_m$ at the end of the sharing phase correspond to a sharing $c$ of $x$, and an honestly-generated proof $\pi$ for $c$. Moreover, by the definition of $P_{\text{stable}}$, $\pi = (c^2, \ldots, c^l, \pi^1, \ldots, \pi^{l})$, where $c^2, \ldots, c^l$ are copies of $c$, and $\pi^1, \ldots, \pi^{l}$ are copies of an honestly-generated proof $\pi_{\text{ZK}}$ for the claim $"(c,n) \in L_{\text{Share}}"$. During the verification phase, $R$ and the servers emulate $V_{\text{stable}}$ on oracles $c', \pi' = (c'^2, \ldots, c'^{l}, \pi'^1, \ldots, \pi'^{l})$, where $c', \pi'$ are obtained from $c, \pi$ through the answers for every bit that was queried during the emulations, but the corresponding server did not answer, $R$ sets that bit to 0. After all queries have been answered, $R$ verifies that $V_{\text{stable}}$ accepted in all emulations, otherwise he halts with output $\perp$.

---

If $(F_{\text{ZK}}, V_{\text{ZK}})$ is the ZKPCPP system of Theorem 5.22, then $V_{\text{stable}}$ makes 3 rounds of queries, so the verification phase consists of 6 rounds, where in odd rounds $R$ broadcasts messages, and in even rounds the servers answer.
of the corrupted servers during the verification phase. Let \( c^*, \pi^*_{\text{ZK}} \) denote the (possibly corrupted) secret sharing, and proof, in which every bit is determined by the majority vote over \( c', c^2, \ldots, c^l, \) and \( \pi'^1, \ldots, \pi'^l \), respectively. Since \( l > 2\tau \), and at most \( \tau \) servers are corrupted, then \( c^* = c \), and \( \pi^*_{\text{ZK}} = \pi_{\text{ZK}} \).

Since \( k \), and \( l \), satisfy the conditions of Lemma 7.3.1, then the lemma guarantees that every single bit in the emulation of \( V_{\text{stable}} \) is consistent with \( c^*, \pi^*_{\text{ZK}} \) (and consequently, with \( c, \pi_{\text{ZK}} \)) except with probability at most \( 2^{1-\frac{1}{\delta}} \). Since every query of \( V_{\text{stable}} \) incurs \( k \) queries of \( V_{\text{stable}} \), then \( V_{\text{stable}} \) makes at most \( kq \) queries, so using the union bound, except with probability at most \( 8kq \cdot 2^{1-\frac{1}{\delta}} \leq \epsilon \), all the query answers are consistent with \( c, \pi_{\text{ZK}} \), so the perfect completeness of \( (P_{\text{ZK}}, V_{\text{ZK}}) \) guarantees that the verification phase succeeds (with probability 1).

Next, we show that conditioned on the event that the verification phase passed, \( R \) outputs \( x \) (with probability 1). Let \( c'', c^2, \ldots, c^m \) denote the copies of secret shares obtained during the reconstruction phase (i.e., obtained from \( c, c^2, \ldots, c^l \), and the answers of the corrupted servers throughout the execution). Let \( \tilde{c} \) denote the purported secret sharing obtained by taking the majority vote over \( c'', c^2, \ldots, c^m \), then \( \tilde{c} = c \) because \( \Delta_F((c, c^2, \ldots, c^l), (c'', c^2, \ldots, c^m)) \leq \Delta((c, c^2, \ldots, c^l), (c'', c^2, \ldots, c^m)) \leq \tau \), and \( l > 2\tau \). (Here, \( \Delta_F \) denotes Hamming distance over \( \mathbb{F} \), and \( \Delta \) denotes Hamming distance over \( \{0,1\} \).

Therefore, \( R \) reconstructs \( x \), and \( x \in \{0,1\}^n \) (because \( D \) is honest), so \( R \) outputs \( x \).

\[ \square \]

**Lemma 7.4.13 (Binding).** Let \( \mathcal{R} \) be an NP relation, \( n \in \mathbb{N} \) be an input length parameter, \( \tau \in \mathbb{N} \) be a security threshold parameter, \( \delta : \mathbb{N} \rightarrow (0,1) \) be a proximity parameter, and \( \epsilon, \epsilon_{\text{ZK}} > 0 \) be soundness error parameters. Let \((\text{Share, Rec})\) be a \( \tau(n) \)-robust secret-sharing scheme that on inputs in \( \mathbb{F}_n \) outputs tuples of \( n'(n) \) secret shares, where \( \tau(n) \geq \tau + \delta (n(n) \log n'(n)) \cdot n(n) \log n'(n) \). Let \((P_{\text{ZK}}, V_{\text{ZK}})\) be a ZKPCPP system for \( \mathcal{R}_{\text{Share}} \) with perfect completeness, proximity parameter \( \delta \), and soundness error \( \epsilon_{\text{ZK}} \), where the honest verifier \( V_{\text{ZK}} \) has query complexity \( q \). Let \((P_{\text{stable}}, V_{\text{stable}})\) be the stable system obtained from \((P_{\text{ZK}}, V_{\text{ZK}})\) through Construction 7.7, with parameters \( k = \max \left\{ 16, 8 \log \frac{2k}{\epsilon} \right\} \), and \( l = \max \left\{ 2^k \cdot \tau, k \right\} \). Then Protocol 7.17 has \( (\epsilon + \epsilon_{\text{ZK}}) \)-binding.

**Proof.** Let \( A \) be an adversary that corrupts \( D \) and a subset of at most \( \tau \) servers, and denote the bits that \( D \) sent to \( S_1, \ldots, S_m \) during the sharing phase by \( b_1, \ldots, b_m \) (if \( D \) does not send a message to \( S_i \) then we set \( b_i = 0 \)), where \( m \) denotes the combined length (in binary representation) of the (possibly corrupted) secret shares, and purported proof. Then \( b_1, \ldots, b_m \) define a (possibly corrupted) secret sharing \( c' \) and a (possibly corrupted) proof \( \pi' = (c'^2, \ldots, c'^l, \pi'^1, \ldots, \pi'^l) \). Let \( c^*, \pi^*_{\text{ZK}} \) denote the (possibly corrupted) secret sharing, and purported proof, in which every bit is determined by the majority vote over \( c', c^2, \ldots, c^l, \) and \( \pi'^1, \ldots, \pi'^l \), respectively.

Since \( k \), and \( l \), satisfy the conditions of Lemma 7.3.1, then the lemma guarantees that every single bit in the emulation of \( V_{\text{stable}} \) is consistent with \( c^*, \pi^*_{\text{ZK}} \) except with
probability at most $2^{1-\frac{2}{k}}$, so using the union bound, except with probability at most $qk \cdot 2^{1-\frac{2}{k}} \leq \epsilon$, all the query answers are consistent with $c^*, \pi_{ZK}^p$.

Next, we show that conditioned on the event $E_{\text{good}}$ that all the query answers were consistent with $c^*, \pi_{ZK}^p$, then except with probability at most $\epsilon_{ZK}$, if $R$ does not abort at the end of the verification phase then there is a unique secret $x^* \in L_R$ which will be reconstructed during the reconstruction phase. Notice that conditioned on $E_{\text{good}}$, we can use the soundness of the ZKPCPP system. We consider two possible cases.

First, if $c^*$ (more accurately, $(c^*, n)$) is $\delta$-far (over $F$) from $L_{R_{\text{Share}}}$, then the soundness of $(P_{ZK}, V_{ZK})$ guarantees that the emulation of $V_{\text{stable}}$ succeeds (namely, $V_{ZK}$ accepts) with probability at most $\epsilon_{ZK}$. So in this case the verification phase fails (and $R$ halts with output $\bot$) except with probability at most $\epsilon_{ZK}$.

Second, if $c^*$ is $\delta$-close (over $F$) to $L_{R_{\text{Share}}}$, then there exists a $\tilde{c} \in L_{R_{\text{Share}}}$ such that $\Delta (c^*, \tilde{c}) \leq \delta \cdot \hat{n} (n) \log \hat{n} (n)$ (here, $\delta = \delta (\hat{n} (n) \log \hat{n} (n))$, and $\Delta (\cdot, \cdot)$ denotes hamming distance over $\{0,1\}$), where $\tilde{c}$ is a valid secret-sharing of some $x' \in \{0,1\}^n \cap L_R$. If the verification phase fails, then $R$ outputs $\bot$ (with probability 1). Otherwise, let $c''', c'^2, \ldots, c'^m$ denote the copies of secret shares obtained during the reconstruction phase (i.e., obtained from $c', c'^2, \ldots, c'^m$, and the answers of the corrupted servers throughout the execution). Let $\hat{c}$ denote the purported secret sharing obtained by taking the majority vote over $c''', c'^2, \ldots, c'^m$, then $\Delta (\hat{c}, c^*) \leq \tau$ because at most $\tau$ servers are corrupted. Consequently,

$$\Delta_{\hat{n}} (\hat{c}, \tilde{c}) \leq \Delta (\hat{c}, \tilde{c}) \leq \Delta (\hat{c}, c^*) + \Delta (c^*, \tilde{c}) \leq \tau + \delta \cdot \hat{n} (n) \log \hat{n} (n) \leq \tau (n)$$

so the $\tau (n)$-robustness of the secret sharing scheme guarantees that $R$ reconstructs (and outputs) $x' \in \{0,1\}^n \cap L_R$ (with probability 1). \qed

**Lemma 7.4.14 (Secrecy).** Let $\epsilon \in (0,1)$ be a statistical distance parameter, $\tau \in \mathbb{N}$ be a corruption threshold, $t : \mathbb{N} \rightarrow \mathbb{N}$ be a privacy function, and $k \in \mathbb{N}$. Let $(P_{ZK}, V_{ZK})$ be a ZKPCPP system with straight-line $(\epsilon, t (n))$-zero-knowledge, where the honest verifier has query complexity $q$. Let $(\text{Share}, \text{Rec})$ be a $t (n)$-private secret-sharing scheme. If $t (n) \geq \tau + qk$ then Protocol 7.17 is $\epsilon$-private against $\tau$-adversaries.

**Proof.** Let $A$ be a $\tau$-adversary that does not corrupt $D$, and let $\text{Sim}_{ZK}$ be the straight-line simulator for the system $(P_{ZK}, V_{ZK})$. Denote the input of $D$ by $(x, w)$, where $(x, w) \in \mathcal{R}$, $n := |x|$. Denote $q^* = t (n)$.

Let $T \subseteq \{S_1, \ldots, S_m\}$ be the subset of $\tau' \leq \tau$ servers that $A$ corrupts. Then the view of $A$ at the end of the verification phase consists of the views $\{V_i\}_{S_i \in T}$ (where every view $V_i$ contains the bit $b_i$ that $S_i$ received from $D$ during the sharing phase), and the answers of the servers to the messages $M_1, M_2, \ldots, M_k$ that the honest verifier $V_{\text{stable}}$ would have made. Since $D$ is honest, then the views of $S_1, \ldots, S_m$ are consistent with a sharing $c$ of $x$, and a proof $\pi$ for $c$ which was honestly generated by $P_{\text{stable}}$.

An honest run of $V_{\text{stable}}$ consists of $k$ random and independent emulations of $V_{ZK}$,
where the oracle answers to the queries of \( V_{ZK} \) are reconstructed from the oracles of \( V_{st} \) using the majority vote over a randomly-selected subset of the corresponding bits. Since \( D \) is honest, the view of \( V_{st} \) in every emulation can be reconstructed from the view of \( V_{ZK} \). Indeed, if \( V_{st} \) queries multiple bits that correspond to the same bit of the oracles of \( V_{ZK} \), then these multiple bits are simply copies of the same bit. For the remainder of the proof, we set some randomness \( r_{st} \) for \( V_{st} \), and this defines random strings \( r^1, \ldots, r^k \) for \( k \) random and independent emulations of \( V_{ZK} \) (Notice that it suffices to bound the statistical distance of the views conditioned on every such \( r_{st} \).

Let \( V_{ZK} (r^j) \) denote the emulation of \( V_{ZK} \) in which it uses random string \( r^j \). Then \( M_1, M_2, \ldots, M_{kq} \) are the \( qk \) queries that \( V_{ZK} (r^1), V_{ZK} (r^k) \) make, when given oracle access to \( c', \pi' \) that are obtained from \( c, \pi \) through the answers of the corrupted servers, and the (majority-based) reconstruction procedure that \( V_{st} \) performs. As \( \tau' + kq \leq t(n) \) then every subset of \( \tau' + kq \) shares of \( c \) are distributed as random shares of \( 0^m \).

The corruption pattern of \( A \) corresponds to a verifier who queries the oracles (perhaps adaptively) about \( \{b_i\}_{S_i \in T} \). Let \( V^* \) be a verifier who queries the oracles about \( \{b_i\}_{S_i \in T} \), and in addition makes the queries that \( V_{ZK} (r^1), V_{ZK} (r^k) \) makes to oracles \( c', \pi' \). Since \( V^* \) makes at most \( q^* \) queries, then the zero-knowledge property of \( (P_{ZK}, V_{ZK}) \) guarantees that for every \( ((y, n), w) \in \mathcal{R}_C \),
\[
\text{SD}\left( \text{Ideal}_{\text{Sim}_{ZK}} (q^*, y, n), \text{Real}_{V^*, P_{ZK}} (q^*, (y, n), w) \right) \leq \epsilon \quad \text{(here both random variables are conditioned on } r^1, \ldots, r^k) \text{.}
\]
In particular, \( \text{Sim}_{ZK} \) makes exactly \( \tau' + kq \leq t(n) \) TTP-queries during the simulation.

Remember that for every \( x \in \{0, 1\}^n \), the TTP-answers for a randomized secret sharing \( y \) of \( x \) are distributed as the corresponding bits in a random encoding of \( 0^m \). Therefore, these bits can be emulated by generating a random sharing of \( 0^m \), and simulating the TTP answers with the values \( r_1, \ldots, r_{\tau' + kq} \in \{0, 1\} \) of the corresponding bits. Consequently,
\[
(\text{Sim}_{ZK} (q^*, r_1, \ldots, r_{\tau' + kq}, n), \tau' + kq) = \text{Ideal}_{\text{Sim}_{ZK}} (q^*, r_1, \ldots, r_{\tau' + kq}, n)
\]
\[
\equiv \text{Ideal}_{\text{Sim}_{ZK}} (q^*, y, n)
\]
and
\[
\text{Real}_{V^*, P_{ZK}} (q^*, (y, n), w) = (M_1, M_2, \ldots, M_{kq}, (y, n, \pi)_{|i \in T \cup M}, \tau' + kq)
\]

---

\(^{5}\)As the corrupted servers may send \( R \) answers that are inconsistent with \( c, \pi \) (moreover, inconsistent with any legal sharing and any valid proof), \( M \) may differ significantly from any set of queries the honest \( R \) (namely, the honest \( V_{st} \) and \( V_{ZK} \)) makes in an honest execution. However, these queries can be simulated given the algorithms of \( V_{st} \) and \( V_{ZK} \) (which are known since \( R \) is honest), and the straight-line nature of the zero-knowledge simulation.
(where all random variables are conditioned on \(r^1, \ldots, r^k\)). We conclude that

\[
\text{SD} (\text{Sim}_{\text{ZK}} (q^*, r^1, \ldots, r^{\tau+q^*}, n), (M^1, M^2, \ldots, M^{q^*}, (y, n, \pi) \mid i \in T \cup M)) \leq \epsilon.
\]

To prove Theorem 7.3, we use Protocol 7.17 with an appropriate choice of parameters. We first restate the theorem.

**Theorem (Theorem 7.3, restated).** For every NP-relation \( \mathcal{R} = \mathcal{R}(x, w) \), every corruption threshold \( \tau \), and every soundness parameter \( \epsilon > 0 \), there exists a \( \text{poly} (n, \tau, \log \frac{1}{\epsilon}) \)-server, (\( \tau, \epsilon \))-secure certifiable VSS protocol for \( n \)-bit messages. The protocol has total communication complexity \( \text{poly} (n, \tau, \log \frac{1}{\epsilon}) \), and a verification phase that uses \( \text{polylog} (n, \tau, \frac{1}{\epsilon}) \) bits of communication.

**Proof of Theorem 7.3.** Let \((P_{\text{ZK}}, V_{\text{ZK}})\) be the ZKPCPP system of Theorem 5.22 for the relation \( \mathcal{R}_{\text{Share}} \), where the underlying secret sharing scheme in the definition of \( \mathcal{R}_{\text{Share}} \) is the secret sharing scheme of Definition 7.4.10. Let \( c \) be the constant in the expression bounding the query complexity of the honest \( V_{\text{ZK}} \), namely the query complexity of \( V_{\text{ZK}} \) on inputs of length \( n \) is at most \( q := \delta^{-c} \cdot \log^c \frac{nq^*}{\epsilon} \) (here, we replaced \( T(n) \) with \( \text{poly} (n) \), which is without loss of generality because \( \mathcal{R} \), and consequently also \( \mathcal{R}_{\text{Share}} \), are NP relations). (We note that \( c \) is well-defined, since it is independent of the identity of the proximity parameter, soundness parameter, and zero-knowledge parameter, of \((P_{\text{ZK}}, V_{\text{ZK}})\).)

We instantiate \((P_{\text{ZK}}, V_{\text{ZK}})\) with soundness parameter \( \epsilon_{\text{ZK}} = \frac{\epsilon}{2} \) (here, we use the subscript “ZK” to distinguish the soundness error of Theorem 5.22 from the soundness error of Theorem 7.3), proximity function \( \delta : \mathbb{N} \rightarrow (0, 1) \) such that \( \delta(n) = \frac{1}{\log n} \), and zero-knowledge parameter \( q^* = t(n) = c' \cdot 2(\tau + n) \log^{6c+2} \frac{1}{\epsilon} \), where \( c' \) is a constant such that \( c' \geq \max \{n_0, n_0', 64\} \), and \( n_0, n_0' \in \mathbb{N} \) are defined as follows. \( n_0 \in \mathbb{N} \) such that for every \( n \geq n_0 \), \( 1 - \frac{1}{\log (n \log n)} \leq \frac{1}{100} \); and \( n_0' \in \mathbb{N} \) such that for every \( n \geq n_0' \), \( 32 \cdot \log^{6c+2} \left( \frac{\log (n \log n)}{6n} \right) \leq \frac{n}{4} \).

Let \((\text{Share}, \text{Rec})\) be the secret sharing scheme of Definition 7.4.10, with functions \((t(n) + 1), \hat{n}(n) = 6t(n)\). We instantiate Protocol 7.17 with the system \((P_{\text{stable}}, V_{\text{stable}})\) with parameters \( k = \max \left\{ 16, 8 \log \frac{24}{\epsilon} \right\} \), and \( l = \max \{26 \cdot \tau, k\} \). We show that the protocol satisfies the conditions of Theorem 7.3.

**Parameters.** Notice that \((P_{\text{ZK}}, V_{\text{ZK}})\) is applied to inputs of length \( \hat{n}(n) \cdot \log (\hat{n}(n)) \) (since it is applied to a tuple of \( \hat{n}(n) \) secret shares, each represented using \( \log (\hat{n}(n)) \) bits). Moreover, the zero-knowledge parameter of the system is \( q^* = t(n) = c' \cdot 2(\tau + n) \log^{6c+2} \frac{1}{\epsilon} \), and the proximity parameter is \( \delta(n) = \frac{1}{\log^c n} \). Therefore, the query complexity of the honest verifier \( V_{\text{ZK}} \) is

\[
q = \delta^{-c} \cdot \log^c \frac{\hat{n}(n) \cdot \log (\hat{n}(n)) \cdot q^*}{\epsilon}
\]

\[
= \log^{2c} (\hat{n}(n) \log (\hat{n}(n))) \cdot \log^c \frac{\hat{n}(n) \cdot \log (\hat{n}(n)) \cdot c' \cdot 2(\tau + n) \log^{6c+2} \frac{1}{\epsilon}}{\epsilon}
\]
Therefore, the conditions of Lemma 7.4.13 hold with soundness parameter $\epsilon$ with soundness error $\epsilon$ which can be upper-bounded by $\log \epsilon$. The verification phase is poly-verification time is $\log \epsilon$. We claim that $\epsilon$.

Binding. By Fact 7.4.8, the secret sharing scheme $\text{Rec}$ is $\tau$-robust for $\tau(n) = \frac{n(n-\lceil t(n) \rceil -1)}{2} = \frac{6t(n) - 6t(n) - 1}{2} \geq 2t(n)$, where the right-most inequality holds because $t \geq n+1$. We claim that $\tau(n) \geq \tau + \delta(\hat{n}(n) \log \hat{n}(n)) \cdot \hat{n}(n) \cdot \log \hat{n}(n)$. Indeed, by the definition of $\delta$,

$$\delta(\hat{n}(n) \log \hat{n}(n)) = \frac{\hat{n}(n) \cdot \log \hat{n}(n) \cdot \log \hat{n}(n)}{\log^2 \hat{n}(n) \cdot \log \hat{n}(n)} \leq \frac{\hat{n}(n)}{\log \hat{n}(n) \cdot \log \hat{n}(n)}$$

Since $\epsilon \geq n_0$, it suffices to show that $\tau(n) \geq \tau + \frac{\hat{n}(n)}{100}$. Recall that $\tau(n) \geq 2t(n)$, and $\hat{n}(n) = 6t(n)$, so it suffices to show that $2t(n) \geq \tau + \frac{6t(n)}{100}$, which holds because $\tau \leq \frac{1}{3}$. Therefore, the conditions of Lemma 7.4.13 hold with soundness parameter $\epsilon' = \frac{\epsilon}{2}$, so the protocol has $\epsilon$-binding against $\tau$-adversaries. (Here, we also used the fact that $(P_{\text{ZK}}, V_{\text{ZK}})$ has soundness error $\frac{\epsilon}{2}$.)

Secrecy. We claim that $t(n) \geq \tau + qk$. It suffices to show that $\frac{t(n)}{2} \geq \tau$ (which follows directly from the definition of $t(n)$, since $\epsilon \in [0, 1]$), and $\frac{t(n)}{2} \geq qk$. Recall that

$$q = \log^{3c} (\hat{n}(n) \log (\hat{n}(n))) \cdot \log^c \frac{\hat{n}(n) \cdot \log (\hat{n}(n))}{\epsilon} \cdot \frac{t(n)}{\epsilon}$$

which can we can upper-bound by

$$\log^{3c} \frac{(6t)^2 \log (6t)}{\epsilon}.$$ 

Moreover,

$$k = 8 \log \frac{4q}{\epsilon} \leq 8 \log \left( \frac{4}{\epsilon} \cdot \log^{3c} \frac{(6t)^2 \log (6t)}{\epsilon} \right) \leq 32 \log^{3c+1} \frac{(6t)^2 \log (6t)}{\epsilon}$$

202
\[ kq \leq 32 \log^{6c+2} \frac{(6t)^2 \log (6t)}{\epsilon} = 32 \log^{6c+2} \frac{1}{\epsilon} + 32 \log^{6c+2} \left( (6t)^2 \log (6t) \right) \leq \frac{t(n)}{2} \]

where the right-most inequality holds because \( 32 \log^{6c+2} \frac{1}{\epsilon} \leq \frac{t(n)}{t} \), and
\[ 32 \log^{6c+2} \left( (6t)^2 \log (6t) \right) \leq \frac{t(n)}{n} \]

Moreover, using Fact 7.4.9, \((\text{Share}, \text{Rec})\) is \(t(n)\)-private, and Theorem 5.22 guarantees that the zero-knowledge property of \((P_{ZK}, V_{ZK})\) holds with a straight-line simulator. Therefore, the conditions of Lemma 7.4.14 are satisfied, and so the protocol has \(\epsilon\)-secrecy against \(\tau\)-adversaries.

\[ \square \]

### 7.5 Distributed Coin-Flipping From ZKLTCs

We consider a distributed model for coin-flipping in which two clients want to agree on a common random bit with the help of \(m\) servers. The clients can interact with the servers via synchronous, secure point-to-point channels, and a broadcast channel, where at the end of the interaction each client outputs a single bit. Intuitively, we say that the protocol is \((\tau, \epsilon)\)-secure if it is correct in the sense that even in the presence of an adversary who corrupts \(\tau\) servers, the joint outputs of the clients will be \(\epsilon\)-statistically close to a pair of identical random bits; and satisfies agreement in the sense that even an adversary who corrupts a client, and \(\tau\) servers, can bias the output of the honest client by at most \(\epsilon\). This is formalized in the next definition.

**Definition 7.5.1 \((\tau, \epsilon)\)-distributed coin-flipping.** We say that an \(m\)-server protocol \(\Pi\) between the parties \(A, B, S_1, \ldots, S_m\) is a \((\tau, \epsilon)\)-distributed coin-flipping protocol, if the following holds for every \(\tau\)-adversary \(A^\tau\).

- **Syntax:** the protocol consists of several rounds. In every round each party may send a message to every other party, and also broadcast a message. The parties have no input, the servers have no output, and each of \(A, B\) outputs a bit \(y_A, y_B\) (resp.).
- **Correctness:** if \(A^\tau\) corrupts neither \(A\) nor \(B\), then except with probability at most \(\epsilon\), \(y_A = y_B\) is a uniformly distributed bit.
- **Agreement:** if \(A^\tau\) does not corrupt \(A\) (resp. \(B\)), then \(y_A\) (resp. \(y_B\)) is \(\epsilon\)-close to uniform.

We use designated-receiver VSS protocols to construct coin-flipping protocols in the natural way, thus obtaining the following result.

**Theorem 7.18 \((\tau, \epsilon)\)-distributed coin-flipping.** For every error parameter \(\epsilon \in (0, 1)\), every constant \(\gamma \in (0, 1)\), and every large enough \(m \in \mathbb{N}\), there exists a 5-round \(m\)-server \((\tau, \epsilon)\)-distributed coin-flipping protocol, with total communication complexity \(O(n) + n^\gamma \cdot \text{poly log } \frac{1}{\epsilon}\), where \(\tau = \Omega_{\gamma}(n)\).
Protocol 7.19. The protocol is parameterized by an error parameter $\epsilon \in (0, 1)$, and the number of servers $m$, and employs an $m$-server designated-receiver VSS protocol $\Pi_{VSS}$.

- $B$ picks a random bit $b_B \in_R \{0, 1\}$, and initiates the sharing, and verification, phases of the VSS protocol $\Pi_{VSS}$ to share $b_B$ (we denote the execution by $\Pi^B_{VSS}$). That is, $B$ plays the role of the dealer $D$, and $A$ plays the role of the receiver $R$, in $\Pi^B_{VSS}$.
- $A$ picks a random bit $b_A$ and sends it to $B$.
- The parties run the reconstruction phase of $\Pi^B_{VSS}$. Let $b'_B$ denote the bit that $A$ reconstructs.
- $A$ outputs $y_A = b_A \oplus b'_B$, and $B$ outputs $y_B = b_A \oplus b_B$.

**The reason that distributed coin-flipping requires VSS.** The reason that we need to use VSS protocols is that the adversary is rushing. To demonstrate why VSS is crucial for the security of the distributed coin-flipping protocol, consider the following simpler protocol. Every client sends the same random bit to all servers, and then the servers compute the exclusive-or of both bits and send the outcome to both clients. A rushing adversary corrupting one client and a single server can cause the other client to output any bit (with probability 1!). Indeed, the adversary waits until the corrupt server receives the random bit from the honest client, and only then decides which bit the corrupt client will send. The VSS prevents such strategies, since it requires the adversary to effectively commit to its bit *before* obtaining any knowledge regarding the bit of the honest client.

Next, we show that Protocol 7.19 is an $m$-server $$(\tau, \epsilon)$$-distributed coin-flipping protocol, provided that the underlying VSS protocol $\Pi_{VSS}$ is an $m$-server $$(\tau, \epsilon_{VSS})$$-VSS for a slightly smaller error parameter $\epsilon_{VSS} = \Omega(\epsilon)$. Since Protocols 7.19 and 7.14 have the same query complexity (up to a single bit), this proves Theorem 7.18.

**Proof of Theorem 7.18.** Let $\epsilon$ be a soundness parameter. Then Theorem 7.12 (with parameters $\epsilon, \gamma$) guarantees that there exists a $\tau = \Omega_{\gamma}(m)$ such that $\Pi_{VSS}$ is a $$(\tau, \epsilon)$$-VSS, with total communication complexity $O(n) + n^\gamma \cdot \text{poly log} \frac{1}{\epsilon}$. We show when Protocol 7.19 is instantiated with the VSS of Theorem 7.12, then it satisfies the properties of Theorem 7.18 with the same parameter $\tau$. (We note that since the sharing and verification phases of $\Pi_{VSS}$ take a total of 3-rounds, then Protocol 7.19 has 5 rounds.)

**Correctness.** Let $\mathcal{A}$ be a $\tau$-adversary that corrupts neither $A$ nor $B$. Denote the random bits that $A, B$ pick by $b_A, b_B$, respectively. ($b_A, b_B$ are uniformly distributed since $A, B$ are honest.) By the correctness of $\Pi_{VSS}$, $A$ reconstructs $b_B$ except with

---

6We can save one round by having both clients commit to their bits using VSS, but this increases the communication complexity.
probability at most $\epsilon$, regardless of the behavior of the corrupted servers. Therefore, except with probability at most $\epsilon$, both parties output the randomly distributed bit $b_A \oplus b_B$.

**Agreement.** We consider two possible cases.

The adversary corrupts $A$. Let $\mathcal{A}$ be an adversary that corrupts $A$ and at most $\tau$ servers. We show that the bias of $y_B$ (in the execution with $\mathcal{A}$ and the honest servers) is at most $\epsilon$. Assume towards contradiction that there exists a $b \in \{0, 1\}$ which $\mathcal{A}$ can cause $B$ to output with probability more than $\frac{1}{2} + \epsilon$. We derive a contradiction to the privacy of $\Pi_{VSS}$, by constructing a distinguisher $\mathcal{D}$ that can distinguish between $V_{A}(0), V_{A}(1)$ with advantage greater than $\epsilon$ (where $V_{A}(b')$ is the view of $\mathcal{A}$ at the end of the sharing phase of $\Pi_{VSS}$, when $B$ uses input $b_B = b'$).

$\mathcal{D}$ on input $z$ (distributed according to either $V_{A}(0)$ or $V_{A}(1)$) provides $\mathcal{A}$ with $z$ as the views of the corrupted parties in the execution of $\Pi_{VSS}^B$, and let $b'_A$ be the bit that $\mathcal{A}$ sends to $B$ (as the bit from $A$), then $\mathcal{D}$ outputs $b'_A \oplus b$. Since $B$ is honest, then $B$ outputs $b'_A \oplus b_B$, where $b_B$ is the random bit that $B$ used for $\Pi_{VSS}^B$. By the negation assumption, $B$ outputs $b$ with probability more than $\frac{1}{2} + \epsilon$. Therefore, $\mathcal{D}$ guesses $b_B$ correctly with probability more than $\frac{1}{2} + \epsilon$, i.e.,

$$|\Pr[\mathcal{D}(V_{A}(1)) = 1] - \Pr[\mathcal{D}(V_{A}(0)) = 1]| > \frac{1}{2} + \epsilon - \left(1 - \left(\frac{1}{2} + \epsilon\right)\right) = 2\epsilon$$

which contradicts the privacy of the VSS protocol.

The adversary corrupts $B$. Let $\mathcal{A}$ be an adversary that corrupts $B$ and at most $\tau$ servers. We show that the bias of $y_A$ (in the execution with $\mathcal{A}$ and the honest servers) is at most $\epsilon$. The binding of $\Pi_{VSS}$ guarantees that except with probability at most $\epsilon$, the execution of the sharing and verification phases of $\Pi_{VSS}$ determines a unique value $b_B^*$ which $A$ reconstructs during the reconstruction phase. During the sharing phase of $\Pi_{VSS}$, $B$ has no information about $b_A$ (the random bit that $A$ chose), so $b_B^*$ is independent of $b_A$. Therefore, conditioned on the event that the sharing phase determines such a value $b_B^*$, $y_A$ is uniformly distributed. Therefore, $y_A$ is $\epsilon$ close to uniform.

7.6 Distributed Zero-Knowledge and Witness-Indistinguishable Proofs

We construct 3-round distributed WI and CZK proofs (respectively) for NP in a distributed setting, in which the PPT prover $P$ and verifier $V$ are aided by $m$ polynomial-time servers $S_1, \ldots, S_m$. We call such systems $m$-distributed proof systems. These distributed proof systems are built based on the non-adaptive WIPCPs and CZKPCPs of Chapter 4. The non-adaptivity of the honest WIPCP (respectively, CZKPCP) verifier is crucial for obtaining 3-round proofs, and the fact that WI (respectively, CZK) holds against malicious verifiers is crucial for obtaining witness-indistinguishability.
(respectively, zero-knowledge) in the distributed setting.

The setting

At a high level, an \( m \)-distributed proof system allows a prover to convince a verifier of the validity of an NP statement, where the parties are aided by several servers. The proof system should have the standard completeness property (guaranteeing that when all parties are honest, the verifier accepts true claims); and a strong soundness property, guaranteeing that a corrupted prover cooperating with a small subset of servers cannot convince the verifier of false claims (except with small probability). We are interested in systems with an additional ZK or WI property, which should hold against a corrupted receiver that cooperates with a small subset of the servers.

**Definition 7.6.1** \((m\text{-distributed proof system})\). An \( m \)-distributed proof system \((P, S_1, \ldots, S_m, V)\) for an NP-relation \( \mathcal{R} = \mathcal{R}(x, w) \) is a protocol executed between a PPT prover \( P \) that has input \((x, w)\), \( m \) polynomial-time servers \( S_1, \ldots, S_m \) that have no input, and a PPT verifier \( V \) that has input \( x \) and outputs either accept or reject. The protocol is executed in rounds, where in each round every party may send a message to every other party (over a secure, authenticated, point-to-point channel), and may also send a message over a broadcast channel. We denote an execution of the protocol, in which \( P \) has input \((x, w)\) and \( V \) has input \( x \), by \((P(x, w), S_1, \ldots, S_m, V(x))\).

For a corruption threshold \( \tau \in \mathbb{N} \), a \( \tau \)-adversary \( \mathcal{A} \) non-adaptively corrupts \( 0 \leq \tau' \leq \tau \) servers, and possibly also \( P \) or \( V \). The execution of a protocol in the presence of a \( \tau \)-adversary \( \mathcal{A} \) is carried out as follows. The honest parties follow the protocol, but the corrupted parties may arbitrarily deviate from the protocol (by sending arbitrary messages). The corrupted parties are also rushing (namely, in each round they first receive the messages sent to them, before sending their own messages). We consider both unbounded adversaries (in which case the corrupted parties are computationally unbounded), and bounded adversaries (in which case even corrupted parties run in polynomial time). We denote an execution of the protocol in the presence of a \( \tau \)-adversary \( \mathcal{A} \), by \((P(y), S_1, \ldots, S_m, V(x))_\mathcal{A}\), where \( y \) is either \((x, w) \in \mathcal{R} \) (if \( P \) is honest) or \( x \notin L_\mathcal{R} \) (if \( P \) is corrupted).

We say that an \( m \)-distributed proof system is sublinear if the total communication involving the verifier \( V \) is sublinear in the input length \(|x|\).

In Sections 7.6.1 and 7.6.2, we use WIPCPs (respectively, CZKPCPs) to construct a 3-round distributed-WI proof system (respectively, CZK proof system in the CRS model) which, at a high level, operates as follows. In the first round the prover distributes a WIPCP (respectively, a CZKPCP) between the servers, and in the second and third rounds the verifier and servers emulate the WIPCP (respectively, CZKPCP) verification procedure (the verifier sends the proof queries of the WIPCP or CZKPCP verifier, and the servers provide the corresponding proof bits). This overview is an over-simplification.
of the construction: the verification procedure of the WIPCP (respectively, CZKPCP) cannot be used as-is since it only guarantees soundness when the verification is performed with a proof oracle, whereas corrupted servers can determine their answers after seeing the queries of the verifier. We overcome this by using the stable system of Section 7.3.

Our distributed proof systems use the following WIPCP system.

Remark 7.20. For \( n \in \mathbb{N} \), and a polynomial corruption threshold \( \tau = \tau(n) \), let \((P_{\text{PCP}}, V_{\text{PCP}})\) be the proof system of Corollary 4.8, applied to the PCP for 3SAT of [AS92, Theorem 1] (more specifically, applied to the proof system used to prove that 3SAT \( \in \text{PCP} \left[ \log n, \log^2 n, \frac{1}{2}, \text{poly}(n) \right] \)), with length parameter \( n \), and zero-knowledge parameter \( q^* \) (see the proof of Theorem 4.1 for details). We amplify the soundness error of the PCP from \( \frac{1}{2} \) to \( \frac{1}{4} \) by repeating the verification procedure twice. Then by Corollary 4.8, there exist constants \( c_q, c_f \) such that the honest verifier \( V_{\text{PCP}} \) makes \( \log^{c_q}(nq^*) \) queries, and the proof has length \( (nq^*)^{c_f} \). Moreover, this system is WI against adaptive verifiers.

### 7.6.1 Distributed Witness-Indistinguishable Proof Systems

In this section we construct a **witness-indistinguishable** distributed proof system for \( \text{NP} \), and using WIPCPs with a **non-adaptive** honest verifier, prove Theorem 7.5. We first formally define such systems.

**Definition 7.6.2** \((\tau, m)\)-distributed WI proof system). Let \( \tau = \tau(n) \) be a corruption threshold function, and \( m = m(n) \). We say that an \( m \)-distributed proof system as in Definition 7.6.1 is a \((\tau, m)\)-distributed WI proof system for \( \mathcal{R} \) if it has the following properties.

- **Completeness.** For every \((x, w) \in \mathcal{R}, V(x) \) outputs accept (with probability 1) in the execution \((P(x, w), S_1, \ldots, S_m, V(x))\).

- **Soundness.** For every \( x \notin L_\mathcal{R} \), and every (possibly malicious, possibly unbounded) \( \tau \)-adversary \( A \) corrupting the prover \( P \), and a subset of \( 0 \leq \tau' \leq \tau \) servers, \( V(x) \), in an execution \((P(x), S_1, \ldots, S_m, V(x))_A\), outputs reject except with \( \text{negl}(n) \) probability.

- **Witness indistinguishability (WI).** Let \( A \) be a (possibly unbounded) \( \tau \)-adversary corrupting \( V \), and a subset \( I \subseteq [m] \) of \( \tau' \leq \tau \) servers. Then for every \( x \in L_\mathcal{R}, \) and every pair \( w_1, w_2 \) of witnesses for \( x, SD(V, A, P;\{S_i; i \in [m] \setminus I\}(x, w_1), V, A, P;\{S_i; i \in [m] \setminus I\}(x, w_2)) = \text{negl}(n) \), where \( V, A, P;\{S_i; i \in [m] \setminus I\}(x, w) \) denotes the view of \( A \) (consisting of the views of the verifier and the servers \( \{S_i : i \in I\} \)), in the execution \((P(x, w), S_1, \ldots, S_m, V(x))_A\).

We first give a high-level idea of the construction used to prove Theorem 7.5. The system employs a WIPCP system \((P_{\text{PCP}}, V_{\text{PCP}})\), and a stable proof system \((P_{\text{stable}}, V_{\text{stable}})\)
obtained from \((P_{\text{PCP}}, V_{\text{PCP}})\) (through the construction of Section 7.3). On input \((x, w)\), the prover generates a WIPCP for \((x, w)\), and distributes the proof bits between the servers. Then, the verifier emulates the PCP verifier \(V_{\text{stable}}(x)\), broadcasting the proof bits queried by \(V_{\text{stable}}\).\(^7\) The servers holding these bits send them to the verifier, who verifies that \(V_{\text{stable}}\) accepts.

**Protocol 7.21.** The distributed WI system consists of the prover \(P\), \(m\) servers \(S_1, \ldots, S_m\), and the verifier \(V\). The system uses a WIPCP system \((P_{\text{PCP}}, V_{\text{PCP}})\) parameterized by the ZK parameter \(q^*\), and where the honest verifier \(V_{\text{PCP}}\) is non-adaptive and makes \(q(n, q^*)\) queries to its proof oracle (where \(n\) denotes the input length). The system also uses the system \((P_{\text{stable}}, V_{\text{stable}})\) obtained from \((P_{\text{PCP}}, V_{\text{PCP}})\) through the transformation of Construction 7.7, with parameters \(l, k\). (We use the variant of Remark 7.8, in which \(P_{\text{stable}}\) does not duplicate the input.) The protocol is executed as follows.

- **Round 1.** \(P\), on input \(x, w\), generates a proof \(\pi \leftarrow P_{\text{stable}}(1^{q^*}, x, w)\) for \(q^* = O(\tau + |x|)\) (see the proof of Lemma 7.6.3 below for the exact constants). \(P\) distributes \(\pi\) between the servers (each server is given a single bit).

- **Round 2.** \(V\) on input \(x\) picks a random string \(r_{\text{stable}}\) for \(V_{\text{stable}}\), and broadcasts it.

- **Round 3.** Each server uses \(x, r_{\text{stable}}\) to determine the \(qk, q \leq (|x|, q^*)\) queries that \(V_{\text{stable}}\) makes when using random string \(r_{\text{stable}}\), and the servers holding the corresponding bits broadcast their value. \(V\) passes these bits to \(V_{\text{stable}}\) as the oracle answers, and either accepts or rejects \(x\), according to the output of \(V_{\text{stable}}\). (For every proof query \(\pi_i\) that \(V_{\text{stable}}\) makes when using random string \(r_{\text{stable}}\), but \(\pi_i\) was not received during this round, \(V\) sets \(\pi_i = 0\).)

Next, we analyze the properties of Protocol 7.21.

**Lemma 7.6.3.** Let \(R = R(x, w)\) be an NP-relation, and \(\tau(n)\) be a polynomial. Then there exists a polynomial \(m(n) > \tau(n)\) such that Protocol 7.21 is a \((\tau(n), m(n))\)-distributed WI proof system for \(R\) with soundness error \(\frac{1}{4} + \text{negl}(n)\), where \(n\) is the input length.

**Proof.** Let \(\tau = \tau(n)\) by a polynomial, and let \((P_{\text{PCP}}, V_{\text{PCP}})\) be the proof system described in Remark 7.20. We set the parameters \(l, k\) for \((P_{\text{stable}}, V_{\text{stable}})\) as follows: \(k = \log^{c_0} n\), \(l = 2^{\log^{c_0} n} + 1\) (we choose \(l\) to be odd so that the majority vote over the \(l\) proof copies generated by \(P_{\text{stable}}\) would be uniquely defined), and \(q^* = 2 c_0 (\tau + n)\), where \(c_0 \in \mathbb{N}\) is a constant such that for every natural \(c \geq c_0\), \(\log^{2c+2} (c^2) \leq \frac{c}{2} \cdot 8\) (Notice

\(^7\)Broadcast is used to ensure that the verifier does not contact more servers than allowed by the WI guarantee. Other alternatives are to extract the verification queries from an unpredictable source of public randomness; or have each server which is contacted by the verifier poll a small number of other servers in order to get an estimate of the number of servers that have been contacted by the verifier.

\(^8\)For the proof of this lemma, it would suffice that the power of the log would be \(2c_0\), but we will need the power to be \(c_0 + 2\) when we modify the system to have the properties guaranteed in Theorem 7.5.
that for this choice of \( q^* \), \( q^* \geq c_0 \) so \( \log^{2^{c_0}} (nq^*) + \tau \leq \log^{2^{c_0}} \left( (q^*)^2 \right) + \frac{q^*}{2} \leq q^* \). Therefore, the combined number of proof bits seen by the verifier \( V \), and \( \tau' \leq \tau \) servers, is at most \( q^* \). Notice that in this case the combined lengths of all copies is \( l \cdot \text{poly} (nq^*) = (\text{poly} \log n + O(\tau)) \cdot \text{poly} (nt) = \text{poly} (n) \), and we take the number of servers, \( m \), to be the corresponding polynomial. Nest, we analyze the properties of the protocol.

**Complexity.** The communication during the first round is \( m = \text{poly} (n) \), and during the second and third rounds the communication is proportional to the query complexity, namely \( \text{polylog} (n) \). (We note that the verifier needs to specify the indices, in a proof of length \( \text{poly} (n) \), it wishes to read, but this causes only a \( \text{polylog} (n) \) increase in the communication complexity.)

**Completeness.** Follows directly from the completeness of the underlying WIPCP, and the definition of the protocol.

**Soundness.** Let \( \pi^1, \ldots, \pi^l \) denote the proofs that \( P \) distributed between the servers, and let \( \pi^* \) be the (possibly ill-formed) proof that is consistent with \( \pi^1, \ldots, \pi^l \). Lemma 7.3.1 guarantees that the reconstructed answer to any single proof query is consistent with \( \pi^* \), except with at most \( 2^{1-\frac{\tau}{l}} \) probability. Using the union bound, the emulation of \( V_{\text{PCP}} \) is executed with \( \pi^* \), except with probability at most \( k \cdot q (n, q^*) \cdot 2^{1-\frac{\log^{\log q(n)} n}{q}} = \log^{\epsilon} (n) \cdot \log^{\epsilon} (nq^*) \cdot 2^{1-\frac{\log^{\log q(n)} n}{q}} = \text{negl} (n) \). Conditioned on the event that all reconstructed bits are consistent with \( \pi^* \), the soundness of the underlying WIPCP system guarantees that \( V_{\text{PCP}} \) (and consequently also \( V \)) accepts with probability at most \( \frac{1}{2} \), so the total soundness error is \( \frac{1}{2} + \text{negl} (n) \).

**Witness-indistinguishability.** Let \( A \) be a \( \tau \)-adversary corrupting \( V \) and a subset \( \mathcal{I} \) of \( \tau' \leq \tau \) servers. Let \( x \in L_R \) with witnesses \( w_1, w_2 \). Then \( V_{A,P; \{S_i: i \in [m]\setminus \mathcal{I}} (x, w_1) \) consists of the proof bits held by the servers \( \{S_i : i \in \mathcal{I}\} \), the proof bits sent to \( V \) in the third round, and the randomness \( r_{\text{stable}} \) for \( V_{\text{stable}} \). In total, the view of \( A \) consists of at most \( \tau' + \log^{2^{c_0}} (nq^*) \leq q^* \) bits (this is because for every random string \( r_{\text{stable}} \), \( V_{\text{stable}} \) makes at most \( k \cdot q (|x|, q^*) = \log^{\epsilon} (n) \cdot \log^{\epsilon} (nq^*) \leq \log^{2\epsilon} (nq^*) \) queries). Therefore, the WI property of the underlying WIPCP system against adaptive, possibly malicious verifiers guarantees that

\[
\text{SD} (V_{A,P; \{S_i: i \in [m]\setminus \mathcal{I}} (x, w_1), V_{A,P; \{S_i: i \in [m]\setminus \mathcal{I}} (x, w_2)) = \text{negl} (nq^*) = \text{negl} (n).
\]

(We note that WI against adaptive verifiers is required because \( A \) receives the proof bits held by the corrupted servers before determining the queries of \( V \). WI against malicious verifiers is required because the adversary sees the proof bits held by corrupted servers, and these correspond to (possibly malicious) proof queries. Notice that due to this reason, WI against malicious verifiers is needed even though the protocol enforces that the queries made by \( V \) correspond to the queries of an execution of \( V_{\text{PCP}} \).)

The proof system of Protocol 7.21 satisfies all the properties required in Theorem 7.5, except for the soundness error, which can be amplified in a standard way, as we now
show. We first restate Theorem 7.5.

**Theorem (Theorem 7.5, restated).** For every NP-relation $R$, and polynomial $\tau(n)$, there exists a polynomial $m(n) > \tau(n)$ such that $R$ has a 3-round sublinear $(\tau(n), m(n))$-distributed WI proof system, where $n$ is the input length.

**Proof of Theorem 7.5.** We amplify the soundness error of Protocol 7.21, and prove that the amplified system satisfies the requirements of Theorem 7.5. The amplification is performed as follows: the second and third rounds are repeated $\log^2 n$ independent times in parallel, and $V$ accepts if and only if $V_{PCP}$ accepted in all the iterations.

The amplification increases the communication complexity in the second and third rounds by a multiplicative factor of $\text{polylog}(n)$, so the protocol is still sublinear. Completeness holds just as in the proof of Lemma 7.6.3. As for soundness, $x \notin L_R$ is accepted if and only if $V_{\text{stable}}$ accepted $x$ in $\log^2 n$ random and independent executions (and for each execution this happens with probability $\frac{1}{4} + \text{negl}(n)$), which happens only with $\text{negl}(n)$ probability. Finally, notice that the view of a $\tau$ adversary $A$ now consists of at most $\log^2 n \cdot \log^{2\sigma} (nq^*) + \tau$ proof bits, which (by the choice of parameters in Lemma 7.6.3) is at most $q^*$, so the same argument used in the proof of Lemma 7.6.3 shows that the system is WI.

**Remark 7.22.** The distributed WI proof system crucially rely on the non-adaptivity of the honest WIPCP verifier. Indeed, if the honest verifier were adaptive then the protocol would have at least 4 rounds, since rounds cannot be compressed. Moreover, since the verifier may collude with a subset of servers, we needed a PCP system with WI against malicious verifiers.

### 7.6.2 Distributed Computational Zero-Knowledge Proof Systems

The techniques of Section 7.6.1 can be used to construct distributed proof systems for NP that guarantees zero-knowledge against computationally bounded $\tau$-adversaries, when all parties have access to a shared random string, as described in Theorem 7.6.

We first formally define the model.

**Definition 7.6.4 (m-distributed proof system with a CRS).** An $m$-distributed proof system with a common random string (CRS) is an $m$-distributed proof system as in Definition 7.6.1, where all parties have access to a shared random string $s$.

**Definition 7.6.5 ((\(\tau,m\))-distributed CZK proof system in the CRS model).** Let $R = R(x,w)$ be an NP-relation, $\sigma \in \mathbb{N}$ be a security parameter, $\tau = \tau(n)$ be a corruption threshold, and $m = m(n)$. We say that an $m$-distributed proof system as in Definition 7.6.4 is a $(\tau,m)$-distributed CZK proof system for $R$ in the CRS model if all parties have access to a shared random string $s \in \{0,1\}^\sigma$ for some $\sigma = \text{poly}(|x|)$, and the system has the following properties.
• Completeness. For every \((x, w) \in \mathcal{R}\), and every CRS \(s \in \{0, 1\}^\sigma\), \(V(x)\) outputs accept (with probability 1) in the execution \((P(x, w, s), S_1(s), \ldots, S_m(s), V(x, s))\) with the CRS \(s\).

• Soundness. For every \(x \notin L_{\mathcal{R}}\), and every (possibly malicious, possibly unbounded) \(\tau\)-adversary \(A\) corrupting the prover \(P\), and a subset of \(\tau' \leq \tau\) servers, \(V(x)\), in an execution \((P(x, s), S_1(s), \ldots, S_m(s), V(x, s))_A\) with a uniformly random CRS \(s \in_R \{0, 1\}^\sigma\) outputs reject except with \(\negl(n)\) probability.

• Computational zero-knowledge (CZK). Let \(A\) be a PPT \(\tau\)-adversary corrupting \(V\) and a subset \(I \subseteq [m]\) of \(\tau' \leq \tau\) servers. Then there exists a PPT simulator \(\text{Sim}\) such that for every \((x, w) \in \mathcal{R}\), \(\text{Sim}(x) \approx (s, V_{A,P_i\{S_i: i \in [m]\setminus I\}}(x, w, s))\), where \(\approx\) denotes computational indistinguishability; \(s \in_R \{0, 1\}^\sigma\); and \(V_{A,P_i\{S_i: i \in [m]\setminus I\}}(x, w, s)\) denotes the view of \(A\) (consisting of the views of the verifiers, and the servers \(\{S_i: i \in I\}\)), in the execution \((P(x, w, s), S_1(s), \ldots, S_m(s), V(x, s))_A\).

The CZK proof system of Theorem 7.6 is similar to Protocol 7.21, except that it uses a CZKPCP system instead of a WIPCP system. (Also, we include the soundness amplification in the protocol description, whereas Protocol 7.21 did not amplify soundness.)

Protocol 7.23. The distributed CZK system consists of the prover \(P\), \(m\) servers \(S_1, \ldots, S_m\), and the verifier \(V\), and uses a CZKPCP system \((P_{\text{PCP}}, V_{\text{PCP}})\) parameterized by the ZK parameter \(q^*\); and the length \(\sigma\) of the common reference string \(s\); where the honest verifier \(V_{\text{PCP}}\) is non-adaptive and makes \(q(n,q^*)\) queries to its proof oracle (\(n\) denotes the input length). The system also uses the system \((P_{\text{stable}}, V_{\text{stable}})\) obtained from \((P_{\text{PCP}}, V_{\text{PCP}})\) through the transformation of Construction 7.7, with parameters \(l, k\). (We use the variant of Remark 7.8, in which \(P_{\text{stable}}\) does not duplicate the input.) The protocol is executed as follows.

• Round 1. \(P\), on input \(x, w\), and given the CRS \(s\), generates a proof \(\pi \leftarrow P_{\text{stable}}(1^{q^*}, x, w, s)\) for \(q^* = O(t + |x|)\) (see the proof of Theorem 7.6 below for the exact constant). \(P\) distributes \(\pi\) between the servers (each server is given a single bit).

• Rounds 2 and 3. \(V\), and the servers, on input \(x\), and the CRS \(s\), perform the following \(\text{polylog}(n)\) independent times in parallel (see the proof of Theorem 7.6 below for the exact constant):
  
  - \(V\) picks a random string \(r_{\text{stable}}\) for \(V_{\text{stable}}\), and broadcasts it.
  - Each server uses \(x, r_{\text{stable}}\) to determine the \(qk, q \leq (|x|, q^*)\) queries that \(V_{\text{stable}}\) makes when using random string \(r_{\text{stable}}\), and the servers holding the corresponding bits broadcast their value. (For every proof query \(\pi_i\) that \(V_{\text{stable}}\) makes when using random string \(r_{\text{stable}}\), but \(\pi_i\) was not received during the third round, \(V\) sets \(\pi_i = 0\).)
We now use Protocol 7.23 to prove Theorem 7.6 (the proof is similar to the proof of Theorem 7.5, which uses Lemma 7.3.1). We first restate the theorem.

**Theorem (Theorem 7.6, restated).** Assume that OWFs exist. Then for every NP-relation $R$, and polynomial $\tau(n)$, there exists a polynomial $m(n) > \tau(n)$ such that $R$ has a 3-round sublinear $(\tau(n), m(n))$-distributed CZK proof system in the CRS model, where $n$ is the input length.

**Proof of Theorem 7.6.** Let $\tau = \tau(n)$ by a polynomial, and let $(P_{WIPCP}, V_{WIPCP})$ be the WIPCP system described in Remark 7.20. Let $(P_{PCP}, V_{PCP})$ be the CZKPCP of Proposition 4.3.5, obtained from $(P_{WIPCP}, V_{WIPCP})$ using a PRG $G : \{0,1\}^\sigma \rightarrow \{0,1\}^{2\sigma}$, whose outputs are $\epsilon_G(\sigma)$-pseudorandom against non-uniform distinguishers, where $\epsilon_G(\sigma) = \text{negl}(\sigma)$ (assuming OWFs exist, such a PRG exists by a standard reduction).

We take $\sigma = n$, then Proposition 4.3.5 guarantees that there exist constants $d_q, d_2$ such that the honest verifier $V_{PCP}$ makes $\log^{d_q}(nq^*)$ queries, and the proof has length $(nq^*)^{d_2}$. We set the parameters of Protocol 7.23 as follows: $k = \log^{d_q}n, l = 2^6(\tau + k) + 1$ (we choose $l$ to be odd so that the majority vote over the $l$ copies would be uniquely defined), $q^* = 2c_0(\tau + 2n)$ for a constant $c_0 \in \mathbb{N}$ such that for every natural $c \geq c_0$, $\log^{2d_q+2}(c^2) \leq \frac{c}{2}$, and the second and third round are repeated (in parallel) $\log^2 n$ times. (Notice that for this choice of $q^*$, $q^* \geq c_0$ so $\log^2(n) \cdot \log^{2d_q}(nq^*) + \tau \leq \log^{2d_q+2}(q^*3^2) + q^*_2 \leq q^*$. Therefore, the combined number of proof bits seen by the verifier $V$, and at most $\tau$ servers, is at most $q^*$.) Notice that in this case the proof has length $l \cdot \text{poly}(2nq^*) = O(\text{polylog}(n) + \tau) \cdot \text{poly}(n\tau) = \text{poly}(n)$, and we take the number of servers, $m$, to be the corresponding polynomial.

**Complexity.** In the first round, $P$ sends a single bit to each of the $m = \text{poly}(n)$ servers. In the second and third rounds, $V_{\text{stable}}$ is emulated $\log^2 n$ times, each emulation requires $V$ to broadcast, and receive, $\text{polylog}(n)$ bits (the random string consists of $k \cdot \text{polylog}(n) = \text{polylog}(n)$ bits, and for each such string there are $k \cdot \text{polylog}(n) = \text{polylog}(n)$ query answers).

**Completeness.** Follows directly from the protocol definition, and the completeness of the underlying WIPCP.

**Soundness.** Let $\pi^1, \ldots, \pi^l$ denote the proofs that $P$ distributed between the servers, and let $\pi^*$ be the (possibly ill-formed) proof that is consistent with $\pi^1, \ldots, \pi^l$. Lemma 7.3.1 guarantees that the reconstructed answer to any single proof query is consistent with $\pi^*$, except with at most $2^{1 - \frac{n}{s}}$ probability. Using the union bound, the emulation of $V_{\text{PCP}}$ is executed with $\pi^*$, except with probability at most $k \cdot q(n, q^*) \cdot 2^{1 - \frac{\log^{d_q}(n)}{s}} = \log^{d_q}(n) \cdot \log^{d_2}(nq^*) \cdot 2^{1 - \frac{\log^{d_q}(n)}{s}} = \text{negl}(n)$. Conditioned on the event that all reconstructed bits are consistent with $\pi^*$, the soundness of the underlying CZKPCP system guarantees that $V_{\text{stable}}$ accepts in a single iteration with probability at most $\frac{1}{4} + \text{negl}(n)$. Since $V$
accepts only if $V_{\text{stable}}$ accepted in all $\log^2(n)$ emulations, $V$ accepts $x$ only with $\text{negl}(n)$ probability.

**CZK.** By Proposition 4.3.5, there exists a simulator $\text{Sim}_{\text{PCP}}$ for $(P_{\text{PCP}}, V_{\text{PCP}})$, that can adaptively answer the queries of any (possibly malicious) $q^*$-bounded PPT verifier. Let $\mathcal{A}$ be a PPT $\tau'$-adversary corrupting $V$ and a subset $\mathcal{I}$ of $\tau' \leq \tau$ servers. We construct a simulator $\text{Sim}$ that simulates the view of $\mathcal{A}$. Let $x \in L_R$ with witness $w$, then $\text{Sim}$ on input $x$ runs $\text{Sim}_{\text{PCP}}$ to obtain the simulated CRS $s_{\text{Sim}}$, emulates $\mathcal{A}$ (with input $x, s_{\text{Sim}}$) to obtain the set $\mathcal{I}$ of $\tau' \leq \tau$ servers that $\mathcal{A}$ corrupts, and uses $\text{Sim}_{\text{PCP}}$ to simulate the proof bits held by these servers. $\text{Sim}$ provides $\mathcal{A}$ with these bits, and receives from $\mathcal{A}$ the random strings for $V_{\text{stable}}$ that $V$ would have broadcasted in the second round. $\text{Sim}$ uses these to determine the set $Q$ of queries that $V_{\text{stable}}$ would have made, and uses $\text{Sim}_{\text{PCP}}$ to simulate the oracle answers. When the emulation ends, $\text{Sim}$ outputs $s_{\text{Sim}}$, concatenated with the random strings of $V_{\text{stable}}$ chosen by $\mathcal{A}$, and the simulated proof bits seen by $\mathcal{A}$.

Let $V^*$ denote the PPT verifier (in the underlying CZKPCP system) that first queries the proof oracle about the bits corresponding to those held by the servers in $\mathcal{I}$, and then queries its oracle about the proof bits in $Q$. Then $(s, V, A, \{S_i : i \in [m] \setminus \mathcal{I}\} (x, w)) = (s, \text{Real}_{V^*, \text{PCP}} (x, w, s))$, where $\text{Real}_{V^*, \text{PCP}} (x, w, s)$ is obtained from $\text{Real}_{V^*, \text{PCP}} (x, w, s)$ by duplicating the bits of $\pi$ that appear several times in $\mathcal{I} \cup Q$. Notice that $|\mathcal{I} \cup Q| \leq \tau + \log^2 n \cdot k \cdot q (n, q) \leq \tau + \log^{2d+2} (nq^*)$ which by our choice of parameters is at most $q^*$. Therefore, the computational distance between $\text{Sim}_{\text{PCP}} (x)$ and $(x, s, \text{Real}_{V^*, \text{PCP}} (x, w, s))$ is at most $\text{negl}(n) + \epsilon_G (n) = \text{negl}(n)$. Since the output of $\text{Sim}$ is obtained from the output of $\text{Sim}_{\text{PCP}}$ by (possibly) duplicating the bits that appear several times in $\mathcal{I} \cup Q$, we conclude that $\text{Sim} (x)$ is computationally close to $(s, V, A, \{S_i : i \in [m] \setminus \mathcal{I}\} (x, w))$. □

### 7.7 Two-party Commit-and-Prove

In this section we construct a “2-party analog” of certifiable VSS, or alternatively, a “certifiable” generalization of a commitment scheme, which we call **Commit-and-Prove**. Roughly speaking, a commitment scheme is a two-phase protocol between a sender $S$, that has input $x$, and a receiver $R$, that has input $1^k$. In the first phase, called the **commit** phase, the sender and receiver interact freely, and the messages exchanged during this phase are called the **commitment**. In the second phase, called the **reveal** phase, $S$ sends $x$, together with a decommitment string $\text{dec}$ to $R$, and $R$ decides whether to accept of reject $x$, based on $\text{dec}$ and the commitment.

A commitment scheme should have the following properties. First, it should be (statistically) **hiding**, in the sense that for every pair $x, x'$ of sender inputs, the views of a (possibly malicious) receiver interacting with the honest sender in the commit phase of the protocol, when the sender has inputs $x, x'$, respectively, are statistically close (i.e., close up to a negligible statistical distance). Second, it should be (computationally) **binding**, namely except with negligible probability, no PPT (possibly malicious)
sender can find, after the interaction with \( R \) during the commit phase, distinct \( x, x' \) of the same length, and two decommitment strings \( \text{dec}, \text{dec}' \), such that \( R \) would accept \( x, x' \) with decommitment \( \text{dec}, \text{dec}' \), respectively. We will also consider the weaker notion of computational hiding, in which case hiding is only guarantees to hold against computationally-bounded receivers. (We note that one can also consider commitment schemes with statistical biding, but we will not do so in this section.)

A Commit-and-Prove protocol is certifiable in the sense that \( S \) not only commits to \( x \), but also proves that \( x \) satisfies some NP-predicate \( R \). (The relation between commitment schemes and Commit-and-Prove protocols is similar to the relation between VSS and certifiable VSS.) Specifically, it is similar to a commitment scheme, but at the end of the reveal phase \( R \) either outputs \( x, x' \in L_R \), or aborts. Since the honest \( R, S \) are both efficient algorithms, the sender cannot generally be expected to find on its own a “witness” for the claim “\( x \in L_R \)”. Therefore, \( S \) is given a witness \( w \) (in addition to the input \( x \)). The Commit-and-Prove functionality we consider has stronger guarantees than those discussed above, since we allow the sender to reveal some subset of the bits of \( x \), where the subset is determined by the receiver. More formally, the Commit-and-Prove functionality is defined as follows.

**Definition 7.7.1 (Commit-and-Prove).** Let \( \mathcal{R} \) be an NP-relation. A Commit-and-Prove (CP) protocol for \( \mathcal{R} \) is an interactive two-phase protocol between a Sender \( S \), and a Receiver \( R \), both PPT algorithms, such that the following holds.

- **Syntax.** \( S \) has input \((x, w)\) (where \( x \) is the secret and \( w \) is the witness), and \( R \) has input \( I \subseteq [\lvert x \rvert], 1^{\lvert x \rvert} \). The commit phase may consist of several rounds, and results in a joint output \( \text{com} = \text{com} \left( S \left( x, w \right), R \left( 1^{\lvert x \rvert}, I \right) \right) \) (the commitment to \( x \)) and a private output string \( \text{dec} = \text{dec} \left( S \left( x, w \right), R \left( 1^{\lvert x \rvert}, I \right) \right) \) of \( S \) (the decommitment). During the reveal phase, \( S \) is given \( I \), and sends \( \text{dec}|_{\mathcal{I}} = \{\text{dec}_i\}_{i \in \mathcal{I}} \), for some subset \( \mathcal{I}' \) determined by \( I \), to \( R \). \( R \) outputs \( x' \in \{0, 1\}^{\lvert I \rvert} \), or \( \bot \). Without loss of generality, the commitment to \( x \) consists of all messages exchanged during the commit phase.

- **Correctness.** If \( S, R \) are honest then for every input \((x, w) \in \mathcal{R} \) of \( S \), and input \( I \) of \( R \), \( R \) outputs \( x_I = \{x_i\}_{i \in I} \) at the end of the reveal phase (with probability 1).

- **Binding.**\(^9\) Every (possibly malicious) PPT sender algorithm \( S^* \) wins the following game with negligible probability.

  - \( S^* \) interacts with \( R \) in the commit phase of the protocol, with common input

\(^9\)This is a strengthening of the property defined in an earlier version of this work [IW14], which did not require partial opening. We thank Omer Paneth for pointing out to us that the previous definition was satisfiable using only standard public-coin interactive zero-knowledge protocols.
– At the end of the commit phase, $S^\ast$ outputs two pairs $(x, \text{dec})$, and $(x', \text{dec}')$, such that $|x| = |x'| = n$.

– $S^\ast$ wins if there exists $I \subseteq \{1, \ldots, n\}$, such that when given $\text{dec}|_I'$ for the subset $I'$ determined by $I$ (resp., $\text{dec}'|_I'$) during the reveal phase, $R$ outputs $\{x_i\}_{i \in I}$ (resp., $\{x'_i\}_{i \in I}$), and at least one of the following holds: either $(x_i)_{i \in I} \neq (x'_i)_{i \in I}$, or $x \notin \mathcal{L}_R$.

**Hiding.** There exists a PPT oracle machine $\text{Sim}$ such that for every (possibly malicious) PPT receiver algorithm $R^\ast$, every sender input $(x, w) \in \mathcal{R}$, and every $I \subseteq \{1, \ldots, |x|\}$, $\text{Sim}^R^\ast \left(1^{|x|}, I\right)$ is computationally indistinguishable from the view of $R^\ast$ during the commitment phase, when interacting with $S(x, w)$, with input $(1^{|x|}, I)$.

**Zero-knowledge after reveal.** There exists a PPT oracle machine $\text{Sim}$ such that for every (possibly malicious) PPT receiver algorithm $R^\ast$, every sender input $(x, w) \in \mathcal{R}$, and every $I \subseteq \{1, \ldots, |x|\}$, $\text{Sim}^R^\ast \left(1^{|x|}, \{x_i\}_{i \in I}, I\right)$ is computationally indistinguishable from the view of $R^\ast$ during the entire interaction with $S(x, w)$, where during the reveal phase $S$ is requested to reveal $\{x_i\}_{i \in I}$.

*Remark 7.24.* Similar to standard commitments, one can also consider stronger variants in which the binding property, or the hiding and zero-knowledge after reveal properties, are statistical (namely, they hold against computationally unbounded adversaries). If a protocol is computationally biding, statistically hiding, and statistically zero-knowledge after reveal, then we say it is a *statistical Commit-and-Prove* protocol.

We use HVZKPCPPs, and exponentially-hard collision-resistant hash functions (see Definition 7.7.2 below), to construct Commit-and-Prove protocols for NP which use the hash function in a *black-box* way, and use *polylogarithmic* communication during the commit phase. (By relaxing the communication requirements such that the communication during commit is sublinear, instead of polylogarithmic, the protocol can be based on a super-polynomially hard hash function.) More specifically, we describe two types of protocols. The first (Section 7.7.1) obtains statistical hiding, and statistical zero-knowledge after reveal, but requires super-constant many rounds.

**Theorem 7.25** (BB sublinear Commit-and-Prove, statistical hiding). Let $\mathcal{H}$ be any family of exponentially-hard collision-resistant hash functions. Then there exists a statistical Commit-and-Prove protocol with negligible soundness error, and polylogarithmic communication complexity during the Commit phase. Moreover, the protocol makes only black-box use of $\mathcal{H}$.

*Remark 7.26.* An additional feature of the Commit-and-Prove protocol of Theorem 7.25 is that it has *quasi-linear total communication*.

Our second protocol (Section 7.7.2) has a constant number of rounds in the coin-flipping hybrid model (intuitively, in this model the parties have access to an ideal coin-flipping functionality, see Section 7.7.2 for more details):
Theorem 7.27 (BB sublinear Commit-and-Prove, coin-flipping hybrid model). Let \( \mathcal{H} \) be any family of exponentially-hard collision-resistant hash functions. Then there exists a constant-round computationally-binding, and statistically-hiding, Commit-and-Prove protocol in the coin-flipping hybrid model, with negligible soundness error, polylogarithmic communication complexity during the Commit phase, and quasi-linear total communication. Moreover, the protocol makes only black-box use of \( \mathcal{H} \).

We also instantiate our constant-round Commit-and-Prove protocol in the plain model, where the hiding, and zero-knowledge after reveal properties, of the resultant protocol hold only against computationally-bounded receivers.

Corollary 7.28 (BB sublinear Commit-and-Prove, constant round). Let \( \mathcal{H} \) be any family of exponentially-hard collision-resistant hash functions. Then there exists a constant-round Commit-and-Prove protocol with negligible soundness error, polylogarithmic communication complexity during the Commit phase, and quasi-linear total communication. Moreover, the protocol makes only black-box use of \( \mathcal{H} \).

Remark 7.29. The Commit-and-Prove protocol of Corollary 7.28 is secure against expected polynomial-times senders, and is hiding, and zero-knowledge after reveal, against strictly polynomial-time receivers, but with expected polynomial-time simulators. This is because the security of constant-round coin-flipping protocols (which flip many random coins) are defined and proved secure in relation to expected polynomial-time adversaries.

Similar to the certifiable VSS protocols of Section 7.4.2, our Commit-and-Prove protocols are based on a robust secret sharing scheme, and an HVZKPCPP system \((P_{2k},V_{2k})\) (e.g., the HVZKPCPP system of Construction 5.10). A crucial difference from the construction of certifiable VSS is that for Commit-and-Prove, honest-verifier zero-knowledge suffices, because the sender can refuse to answer dishonest queries. (The difference from the distributed case is that in the distributed case the servers hold the proof, so the information revealed by corrupted servers corresponds to the information revealed through dishonest oracle queries, and one cannot prevent this “leakage” of information.) In addition, the protocol employs a family \( \mathcal{H} \) of collision-resistant hash functions, which we define next.

Definition 7.7.2 (CRHF). Let \( t \in \mathbb{N} \) be a security parameter, \( \epsilon : \mathbb{N} \rightarrow \mathbb{R}^+ \) be an error function, and \( T : \mathbb{N} \rightarrow \mathbb{N} \) be a size function. A family of functions

\[ \mathcal{H} = \{ h_\alpha : \{0,1\}^t \rightarrow \{0,1\}^t \}_{\alpha \in \{0,1\}^t, t \in \mathbb{N}} \]

is \((\epsilon(t), T(t))\)-hard collision-resistant if for every \( t \in \mathbb{N} \), and every family \( \{C_t\}_{t \in \mathbb{N}} \) of circuits of size at most \( T(t) \):

\[ \Pr_{\alpha \leftarrow \{0,1\}^t} \left[ (x, x') \leftarrow C_t(\alpha) : x \neq x' \land h_\alpha(x) = h_\alpha(x') \right] \leq \epsilon(t). \]
We say that $\mathcal{H}$ is \textit{exponentially-hard collision-resistant} if there exists a constant $c > 0$ such that $\mathcal{H}$ is $(2^{-c}, 2^c)$-hard. We say that $\mathcal{H}$ is \textit{super-polynomially hard collision-resistant} if there exists a negligible function $\epsilon : \mathbb{N} \to \mathbb{R}^+$ such that for every polynomial $p : \mathbb{N} \to \mathbb{N}$, $\mathcal{H}$ is $(\epsilon(n), p(n))$-hard collision-resistant.

Damgård et al. [DPP97] construct a computationally binding, and statistically hiding, commitment scheme based solely on CRHFs.

**Theorem 7.30** (Statistically hiding commitments from CRHF [DPP97]). Let

\[ \mathcal{H} = \{ h_\alpha : \{0, 1\}^{2t} \to \{0, 1\}^t \}_{\alpha \in \{0, 1\}^t, t \in \mathbb{N}} \]

be a family of super-polynomially hard CRHF. Then there exists a computationally-binding, and statistically-hiding, bit-commitment scheme with perfect completeness. The commitment scheme consists of a single message $\alpha \leftarrow \{0, 1\}^t$ from $R$ to $S$, followed by a single message of length $O(t)$ from $S$ to $R$. Moreover, the sender and receiver can engage in $\text{poly}(t)$-many parallel commitments, using the same first message $\alpha$, while maintaining the computational binding property.

**Remark 7.31.** The statistical hiding property of the bit-commitment scheme guarantees that if the sender only opens some of the commitments in the parallel version ($R$ can choose the identity of the commitments that will be opened after seeing the commitments), the unopened bits remain statistically hidden.

At a high level, our protocols operate as follows. During the commit phase, $R$ chooses a function $h \in \mathcal{H}$ and sends (the index of) $h$ to $S$. $S$ secret-shares every bit $x_i$ of $x \in L_R$ into shares $s_i^1, \ldots, s_i^m$ and uses $P_{VZK}$ to generate a proof $\pi$ for the claim “for every $1 \leq i \leq n$, $(s_1^i, \ldots, s_m^i)$ are close to a secret sharing of some $x_i^*$, and $(x_1^*, \ldots, x_n^*) \in L_R$”. Next, $S$ commits to $\pi$ using a commitment scheme $\text{Com}_h$, and uses a “Merkle Hash Tree” (MHT) [Mer87] to “compresses” the commitments. (That is, the commitments are compressed by repeatedly applying the hash function $h$ to pairs of adjacent strings, where every application of $h$ shrinks the input by half. Thus, a “tree” of hash values is generated, and its root is used as the compressed commitment.) $S$ commits to $s_1^1, \ldots, s_m^n$ in a similar manner. Then, $S$ sends the compressed commitments $C_\pi$ (of $\pi$), and $C_x$ (of $s_1^1, \ldots, s_m^n$), to $R$.

$R$ verifies the commitments as follows. Given randomness $r$ for $V_{VZK}$, $S$ determines the set $Q$ of oracle queries that $V_{VZK}$ makes when using randomness $r$. $S$ answers every query $q \in Q$ by decommitting the corresponding bit of $\pi, s_1^1, \ldots, s_m^n$ (using the reveal phase of $\text{Com}_h$), and sending the pre-images of all the hash values computed along the path in the MHT leading from that bit to the root. $R$ verifies that the values on the paths are consistent with $C_\pi, C_x$, and $h$, that $V_{VZK}$, with randomness $r$, makes the queries $Q$, and that $V_{VZK}$ accepts given these oracle answers.
During the reveal phase, $S$ is given $I$, and for every $i \in I$, sends to $R$ all the pre-images of all the hash values computed along the path in the MHT leading from $s_1^i, \ldots, s_m^i$ to the root of $C_x$, together with the random strings used to generates the commitments (through $\text{Com}_n$) to $s_1^i, \ldots, s_m^i$. For every $i \in I$, $R$ verifies that the commitments, and the MHT, are consistent with $s_1^i, \ldots, s_m^i$, and if so reconstructs $x_i$ from the shares. If all reconstructions succeeded, $R$ outputs $\{x_i\}_{i \in I}$. We note that the only difference between our protocols is in the method used to determine the randomness $r$ for $V_{ZK}$.

In our first protocol (Section 7.7.1) $R$ chooses $r$. This incurs a super-constant number of rounds since it requires soundness amplification. (More specifically, to achieve the parameters described in Theorem 7.25, we need to use an HVZKPCPP with a non-negligible soundness error, and amplify through sequential repetitions to obtain a negligible error.)

In our second protocol (Section 7.7.2), $r$ will be chosen jointly using a coin-flipping protocol. We note that when the parties jointly choose the randomness for the underlying HVZKPCPP verifier then the protocol can employ an HVZKPCPP with negligible soundness error, and therefore will not require amplification. (The difference is due to the different analysis of the hiding, and zero-knowledge after reveal properties.)

\begin{remark}
Let $(\text{Share}, \text{Rec})$ denote the secret sharing scheme of Definition 7.4.10, instantiated with $\hat{n}(n) = O(1 + t(n))$ such that $\frac{5t(n)+2}{4} \leq \hat{n}(n) \leq \frac{5t(n)+2}{2} \leq \text{poly}(n)$, where $t(n)$ is a parameter whose value will be set below. Then the scheme is $\frac{\hat{n}(n)}{10}$-robust. (Indeed, according to Observation 7.4.11, the scheme is $\tau(n)$ robust for $\tau(n) = \frac{\hat{n}(n)-t(n)-1}{2} \geq \frac{\hat{n}(n)-4t(n)}{2} = \frac{n(n)}{10}$.)

Let $R$ be an NP relation, and define

\[ R_{\text{Share}} = \left\{ (\{y^1, \ldots, y^n\}, w) : \forall 1 \leq i \leq n, y^i \in \text{supp}(\text{Share}) \land \text{Rec}(y^i) |_1 \in \{0, 1\} \right\} \]

\[ \land \left( (\text{Rec}(y^1) |_1, \ldots, \text{Rec}(y^n) |_1, w) \in R \right) \]

where $\text{Rec}(y) |_1$ denotes the first symbol in the output of $\text{Rec}$ on $y$.

$R_{\text{Share}} \in \text{DTIME}$ (since $R \in \text{DTIME}$), so by Theorem 5.19 there exists a PCPP system for $R_{\text{Share}}$ with proximity parameter $\frac{1}{10}$, soundness error $\frac{1}{2}$, randomness complexity $O(\log n)$ (where $n$ denotes the input length), and query complexity $O(1)$. Plugging this system into Theorem 5.16, we get for every constant $\epsilon \in (0, 1)$ an HVZKPCPP system $(P_{ZK}, V_{ZK})$ for the relation $R_{\text{Share}}$, with proximity parameter $\frac{1}{10}$, soundness error $\epsilon$, perfect completeness, and perfect honest-verifier zero-knowledge. Moreover, the proofs have size $\text{poly}(n)$, and the honest verifier tosses only $O(\log n)$ random coins, and non-adaptively reads only $q = O(1)$ bits from the oracles. Since $q$ is constant (in particular, independent of input length), we can take $t(n) = q$ in the secret sharing scheme, without changing the parameters of the system, in which case the proof length is $\text{poly}(n)$, and the randomness complexity is $O(\log n)$.

218
7.7.1 Statistical Commit-and-Prove with Super-Constant Many Rounds

In this section we describe a statistical Commit-and-Prove protocol with super-constant many rounds.

Protocol 7.33. Let $\mathcal{R} = \mathcal{R}(x, w)$ be an NP-relation, and $n \in \mathbb{N}$ denote the input length. The system is parameterized by a constant soundness parameter $\epsilon \in (0, 1)$, a security parameter $t_{\text{sec}} \in \mathbb{N}$, a family $\mathcal{H}$ of $T(t_{\text{sec}})$-hard CRHFs, the (parallelized version of the) commitment scheme $\text{Com}$ of Theorem 7.30 based on $\mathcal{H}$, and the HVZKPCPP of Remark 7.32.

The commit phase. $S$ has input $(x, w) \in \mathcal{R}$ such that $|x| = n$, and $R$ has input $1^n$ and $I \subseteq [n]$.

- $R$ samples $\alpha \leftarrow \{0, 1\}^{t_{\text{sec}}}$, and sends $\alpha$ to $S$.
- $S$ computes the commitment on $x$ in the following four steps.
  - Hiding $x$. Interprets every bit $x_i$ as an element in $\mathbb{F} := \mathbb{F}_n$, chooses $r^i \in R \mathbb{F}^{t(n)}$, and generates a secret sharing $c^i = \text{Share}(x_i, r^i)$. We denote $c^i = (a^i_1, \ldots, a^i_{2^l})$ (assuming for simplicity that all $c^i$ have length $2^l$ for some $l \in \mathbb{N}$).
  - Generating a ZKPCPP for the claim “$((c^1, \ldots, c^n), w) \in \mathcal{R}_{\text{Share}}$”. Uses $P_{\text{ZK}}$ to generate a proof $\pi$ for the claim “$((c^1, \ldots, c^n), w) \in \mathcal{R}_{\text{Share}}$”. We denote $\pi = (b_1, \ldots, b_{2^s})$ (assuming for simplicity that $\pi$ has length $2^s$ for some $s \in \mathbb{N}$).
  - Hiding the bits of $c^1, \ldots, c^n, \pi$. For every $a^i_k, b_j$ independently, $S$ computes the commitments $C^a_{i, k} = \text{Com}(a^i_k), C^b_j = \text{Com}(b_j)$ ($\alpha$ is used as the index of the underlying hash function).
  - Hashing-down the commitment. To hash the commitments to $a^1_1, \ldots, a^n_{2^l}$ into a single block $C^a$ of length $n$, $S$ generates a MHT [Mer87] of $(C^a_{i, 1}, \ldots, C^a_{n, 2^l})$, and a single block $C^b$ from $(b_1, \ldots, b_{2^s})$, and sends $(C^a, C^b)$ to $R$ as the commitment to $x$.
- $R$ verifies the commitment $(C^a, C^b)$ as follows.
  - Picks a random string $r$ for $V_{\text{ZK}}$, and sends $r$ to $S$ (if $R$ does not send a legal random string for $V_{\text{ZK}}$ then $S$ proceeds with $r = r_0$ for some fixed random string $r_0$).
  - Let $Q$ denote the set of queries that $V_{\text{ZK}}$ makes, when using randomness $r$. $S$ answers every input (proof) query $q \in Q$ by opening the corresponding commitment $C^a_q$ ($C^b_q$), and revealing the pre-image of all the hash values computed along the MHT path going from $C^a_q$ ($C^b_q$) to $C^a$ ($C^b$), which involves opening 2 values from every layer $i \in [l]$ ($i \in [s]$) of the tree.
- $R$ verifies the consistency of $(C^a, C^b)$ with the revealed commitments and hash values, and checks that $V_{ZK}$ accepts (given these answers).

**The reveal phase.** Given $I$, $S$ sends $R$ the secret shares $c^i$ for every $i \in I$, together with the pre-images of all the hash values on the MHT paths leading from $C^a_{i,1}, \ldots, C^a_{i,2^l}$ to the root of $C^a$, and the randomness used to generate the commitments $C^a_{i,1}, \ldots, C^a_{i,2^l}$. If $V_{ZK}$ rejected during the Commit phase, or the values received during the reveal phase are inconsistent with $C^a$ or with the commitments, then $R$ outputs ⊥. Otherwise, let $c^i \ast$ denote the secret sharing of $x^i$ that $R$ received during the phase, then $R$ computes $(x^i \ast, r^i \ast) = \text{Rec}(c^i \ast)$. If $x^i \ast \in \{0, 1\}$ for every $1 \leq i \leq n$ then $R$ outputs $\{x^i \ast\}_{i \in I}$, otherwise he outputs ⊥.

**Remark 7.34.** The ZK-arguments of Ishai et al. [IMS12] use a ZKPCP, where the proof is constructed by generating several independent ZKPCPs, hiding the ZKPCP bits using commitments, and hashing the commitments using the MHT. The verification sub-protocol run with $R$ is identical to the one in our construction. Therefore, our protocol is obtained from the ZK-arguments of [IMS12] by using a ZKPCPP instead of a ZKPCP, and supplying the receiver with the hashed-down value of (a randomized encoding of) the input. (In the case of ZK-arguments, there is no need to commit to the input or hide it, since it is given directly to the verifier.)

Next, we analyze the properties of Protocol 7.33. Let $t(n) = q$ be a privacy function (where $q$ is the query complexity of the honest verifier $V_{ZK}$ of the HVZKPCPP system), $\tau(n)$ be a robustness function, and $\hat{n}(n)$ be a secret sharing length function. We use MHT ($y$) to denote a MHT commitment to a string $y$ (constructed by first generating commitments to the bits of $y$, and then hashing the commitments to a single block, using the MHT).

**Lemma 7.7.3** (Binding). Let $\epsilon > 0$ be an error parameter, $(\text{Share}, \text{Rec})$ be the secret sharing scheme of Remark 7.32, and $(P, V)$ be an HVZKPCPP for the relation $\mathcal{R}_{\text{Share}}$ of Remark 7.32, with proximity parameter $\frac{1}{10}$, soundness error $\epsilon$, and a non-adaptive honest verifier. Let $\mathcal{H}$ be a super-polynomially hard family of CRHFs, and $\text{Com}$ be a computationally binding commitment scheme. Then Protocol 7.33, when instantiated with $(\text{Share}, \text{Rec}), (P, V)$, and $\text{Com}$, has $\epsilon + \text{negl}(n)$ binding.

**Proof.** Let $S^*$ be an efficient (possibly non-uniform) sender algorithm, and denote the commitment that $S^*$ sent to $R$ by $(C^a, C^b)$. We use $C^a, C^b$ to define effective (possibly illegal) secret-sharings $y^1, \ldots, y^n$ of the input bits, and an effective (possibly incorrect) “proof” $\pi^*$ as follows.

Consider the following mental experiment, in which $S^*$ is run sequentially with all possible random strings for $V_{ZK}$. That is, $S^*$ generates the commitments $C^a, C^b$ once, and then the verification procedure is repeated many times ($S^*$ is rewinded between each pair of consecutive executions), where each time $S^*$ is fed with a new random
string for $V_{ZK}$ as the randomness that $R$ chose for $V_{ZK}$. We arbitrarily fix to 0 the value of bits that $S^*$ fails to open, or that are never queried. (We can assume without loss of generality that all proof bits are queried, since proof bits that are never queried can be discarded from the proof. Moreover, the soundness of the ZKPCPP guarantees that at most an $\frac{1}{10}$-fraction of the bits of $y_1, \ldots, y^n$ are never queried by the verifier.)

The collision-resistance of the hash function, and the binding of the commitment scheme, guarantee that except with negligible probability, every bit of $y_1, \ldots, y^n, \pi^*$ is well defined (i.e., was either fixed to 0, or can only be opened by $S^*$ to a single, unique value). Therefore, conditioned on the event $E$ that $S^*$ did not break the collision-resistance of $\mathcal{H}$ (this event happens with overwhelming probability), or the binding of the commitment scheme, $y_1, \ldots, y^n, \pi^*$ are well defined.

Consider a second mental experiment, in which $V_{ZK}$ is run directly on $y_1, \ldots, y^n, \pi^*$, and notice that $\Pr \left[ V_{ZK}^{y_1,\ldots,y^n,\pi^*} = \text{acc} \right]$ is at least the probability that the commit phase passes (conditioned on $E$). We now consider 3 possible cases.

First, if $(y_1, \ldots, y^n)$ is $\frac{1}{10}$-far from all secret sharings of $(x_1, \ldots, x_n) \in F^n$ (in particular, $(y_1, \ldots, y^n)$ is $\frac{1}{10}$-far from $L_{R,\text{Share}}$), then the soundness of $(P_{ZK}, V_{ZK})$ guarantees that $V_{ZK}$ accepts with probability at most $\epsilon$, so in this case the commit phase fails (and $R$ outputs $\bot$) except with probability at most $\epsilon + \text{negl}(n)$ (the additive $\text{negl}(n)$ term bounds the probability that $V_{ZK}$ is run with oracles different than $y_1, \ldots, y^n, \pi^*$).

Second, if $(y_1, \ldots, y^n)$ is $\frac{1}{10}$-close to some sequence of secret sharings $(\text{Share}(x_1^*, r_1^*), \ldots, \text{Share}(x_n^*, r_n^*))$ such that $(x_1^*, \ldots, x_n^*) \notin L_R$, then the robustness of the secret sharing scheme of Remark 7.32 guarantees that $(y_1, \ldots, y^n)$ is at least $\frac{1}{10}$-far from $L_{R,\text{Share}}$. (Indeed, every $(z_1, \ldots, z^n) \in L_{R,\text{Share}}$ is a legal secret sharing, and the robustness of the secret sharing scheme guarantees that every pair of legal sharings differ in more than $2\epsilon = \frac{1}{5}$ coordinates, so $(y_1, \ldots, y^n)$ is $\frac{1}{5} - \frac{1}{10} = \frac{1}{10}$-far from every $(z_1, \ldots, z^n) \in L_{R,\text{Share}}$.) As in the first case, this implies that $R$ outputs $\bot$ at the end of the commit phase, except with probability at most $\epsilon + \text{negl}(n)$.

Finally, assume that $(y_1, \ldots, y^n)$ is $\frac{1}{10}$-close to some sequence $(\text{Share}(x_1^*, r_1^*), \ldots, \text{Share}(x_n^*, r_n^*))$ of secret sharings of some $(x_1^*, \ldots, x_n^*) \in L_R$. We claim that in this case $x^*$ is the unique input that can be decommitted, and for every $I \subseteq [n]$, the corresponding decommitment string consists of the pre-images on the paths leading from (the commitments to) $\{y_i\}_{i \in I}$ to the root in any honestly-generated MHT for $(y_1, \ldots, y^n)$. Indeed, if the commit phase passes successfully (i.e., $R$ detects no inconsistency) then for every $i \in I$, $\text{Rec}(y_i) = (x_i^*, r_i^*)$, due to the error correction property of the secret sharing scheme of Remark 7.32. By the collision-resistance of $\mathcal{H}$, and of the binding of the commitment scheme, $S^*$ can find a MHT for some $(y_1', \ldots, y^n') \neq (y_1, \ldots, y^n)$ (which is consistent with the interaction during the commit phase) only with $\text{negl}(n)$ probability. (The decommitment string must correspond to valid paths in a MHT, otherwise $R$ outputs $\bot$.) Therefore, if the commit phase succeeds then (except with negligible probability), $R$ outputs $\{x_i^*\}_{i \in I}$ or $\bot$. (If the commit phase fails then in any case $R$ outputs $\bot$.)
Hiding and zero-knowledge after reveal

We first sketch how to simulate the view of the honest receiver, then discuss the modifications required for the malicious case, and finally given a formal simulator construction.

The honest receiver case. Roughly speaking, the commit-phase view of $R$ can be simulated using the simulator $\text{Sim}_\text{ZK}$ of the underlying ZKPCPP system, where $\text{Sim}_\text{Com}$ answers the TTP-queries of $\text{Sim}_\text{ZK}$ with the corresponding bits in random secret sharings of $0^{1+\ell(n)}$. More specifically, $\text{Sim}$ chooses $\alpha \in_R \{0,1\}^\ell$, a random string $r$ for $V_{\text{ZK}}$ (which determines the set $Q$ of queries of $V_{\text{ZK}}$), and $r^1,\ldots,r^n \in_R \mathbb{F}_n^{\ell(n)}$. Then, for every $1 \leq i \leq n$, $\text{Sim}_\text{Com}$ generates a secret sharing $\hat{c}^i = \text{Share} (0, r^i)$, and uses $\text{Sim}_\text{ZK}$ to simulates the oracle answers $(a_1,\ldots,a_q)$, where the TTP-queries of $\text{Sim}_\text{ZK}$ are answered according to $\hat{c}^1,\ldots,\hat{c}^n$.

After the simulation terminates, $\text{Sim}$ constructs a “proof” $\hat{\pi}$ as follows. For every bit $i$ of $\hat{\pi}$, if $\hat{\pi}_i$ was queried in $Q$ then $\text{Sim}$ uses the corresponding simulated answer (obtained through the simulation of $\text{Sim}_\text{ZK}$), otherwise it sets $\pi_i = 0$. For the remainder of the simulation, $\text{Sim}$ acts as the honest sender would (i.e., commits to the values, hashes them down, etc.). Then the simulated transcript of the commit phase is statistically close to the real-world transcript with the honest receiver in Protocol 7.33. Indeed, $\alpha, r$ are equally distributed in both worlds, so we condition both transcripts on the value of $\alpha, r$. We first show that the transcripts are statistically close conditioned on the TTP answers given to $\text{Sim}_\text{ZK}$ being consistent with the input oracle of $V_{\text{ZK}}$, then show why the TTP answers are indeed consistent with the input oracle.

If the TTP answers are consistent, then the perfect honest-verifier zero-knowledge of $(P_{\text{ZK}}, V_{\text{ZK}})$ guarantees that the bits revealed from $c^1,\ldots,c^n, \pi$, are identically distributed to the corresponding bits of $\hat{c}^1,\ldots,\hat{c}^n, \hat{\pi}$. Therefore, we may condition both executions on the event that the same bit values are revealed. Let $W_R = \{(C_{n,1}^a,\ldots,C_{n,2}^a,C_{n,1}^b,\ldots,C_{n,2}^b, D_1,\ldots,D_q)\}$ denote the random variable consisting of the commitments to $(c^1,\ldots,c^n), \pi$, and decommitments to the $q$ queries of the real-world $R$. Similarly, we define $W_{\text{Sim}}$ based on the simulated proof $\hat{\pi}$, and secret sharings $\hat{c}^1,\ldots,\hat{c}^n$. Then by the statistical hiding of the commitment scheme of Theorem 7.30 (against poly-many selective decommitments, see Remark 7.31) $W_R, W_{\text{Sim}}$ are statistically close.

Therefore, it remains to show that the simulated TTP-answers provided to $\text{Sim}_\text{ZK}$ are identically distributed to the real-world. Indeed, $\text{Sim}_\text{ZK}$ makes at most $q = t(n)$ TTP queries (since $R$ makes $q$ queries). Moreover, the secret sharing scheme is $t(n)$-private, and for every $1 \leq i \leq n$, $c^i = \text{Share} (x_i, r^i)$ for $r^i \in_R \mathbb{F}_n^{\ell(n)}$, so every $q$ coordinates of $(c^1,\ldots,c^n)$ are identically distributed to the corresponding coordinates in $n$ random secret sharings of $0$. Therefore, the answers provided to $\text{Sim}_\text{ZK}$ are identically distributed to the answers of a real-world TTP for $c^1,\ldots,c^n$.

The view of $R$ at the end of the reveal phase can be simulated in a similar fashion.
Indeed, the simulator \( \text{Sim}_{\text{Rev}} \) is given \( \{x_i\}_{i \in \mathcal{I}} \), and can therefore honestly generate a random secret sharing \( \hat{c}^i \) of every \( x_i \), and (as in the hiding case) generate a random secret sharing of 0 for every \( i \notin \mathcal{I} \). \( \text{Sim}_{\text{Rev}} \) then commits to the bits of \( c_1, \ldots, c^n \). Then, \( \text{Sim}_{\text{Rev}} \) picks \( \alpha, r \) as above, and uses \( \text{Sim}_{\text{ZK}} \) to simulate the answers to the queries that \( R \) makes (i.e., the queries of \( V_{\text{ZK}} \) with randomness \( r \)), where the TTP queries of \( \text{Sim}_{\text{ZK}} \) are answered with the bits of \( c_1, \ldots, c^n \). The simulator then commits to the bits of \( \hat{\pi} \), defined as above (i.e., \( \hat{\pi}_i = 0 \) for every proof bit whose value was not determined during the simulation).

The malicious receiver case. Although \( S \) checks the validity of the receiver queries, this case differs from the honest-receiver case because \( R^* \) may choose the random string \( r \) for \( V_{\text{ZK}} \) based on the commitment \( C = (C^0, C^n) \) (while the honest receiver chooses a random \( r \), independent of \( C \)). Therefore, to extract \( r \), \( \text{Sim}_{\text{Com}} \) and \( \text{Sim}_{\text{Rev}} \) must first generate the commitment \( C \). We overcome this by having \( \text{Sim}_{\text{Com}}, \text{Sim}_{\text{Rev}} \) guess \( r \), and then generate \( C \) assuming \( R^* \) chooses the random string \( r \). If the guess is correct then the simulation can continue, and otherwise the simulator rewinds \( R^* \) and starts over.

To see why this strategy works, notice that the simulation can continue, and otherwise the simulator rewinds \( R^* \) guess \( r \), and then generate \( C \) assuming \( R^* \) chooses the random string \( r \). If the guess is correct then the simulation can continue, and otherwise the simulator rewinds \( R^* \) and starts over.

To see why this strategy works, notice that the simulation can continue, and otherwise the simulator rewinds \( R^* \) guess \( r \), and then generate \( C \) assuming \( R^* \) chooses the random string \( r \). If the guess is correct then the simulation can continue, and otherwise the simulator rewinds \( R^* \) and starts over. To see why this strategy works, notice that the simulation can continue, and otherwise the simulator rewinds \( R^* \) starts over.

We show how to simulate the view of \( R^* \) when given the commitments themselves (instead of the hashed-down values). Since the hashed-down commitments can be generated from the commitments themselves, this shows that the view of \( R^* \) can be simulated.

Construction 7.35. The commit-phase simulator \( \text{Sim}_{\text{Com}} \) operates as follows.

1. Initialization. Selects a fixed random string \( r \) for \( R^* \), and extracts the message \( \alpha \in \{0, 1\}^{\text{sec}} \) that \( R^* \) sends to \( S \) (which will also be fixed for the entire simulation).

2. Generating commitments. Guesses the random string \( r_{\text{ZK}} \) that \( R^* \) would choose, and extracts from it the \( q \) queries \( \hat{Q} = (\hat{q}_1, \ldots, \hat{q}_q) \) that \( V_{\text{ZK}} \) makes when using randomness \( r_{\text{ZK}} \). Then, \( \text{Sim}_{\text{Com}} \) generates \( n \) random secret sharing \( \hat{c}^1, \ldots, \hat{c}^n \) of 0, and uses \( \text{Sim}_{\text{ZK}} \) to simulate the oracle answers. (The TTP-queries of \( \text{Sim}_{\text{Com}} \) are answered with the corresponding bits in \( \hat{c}^1, \ldots, \hat{c}^n \).) A simulated proof \( \hat{\pi} \) is defined (based on these simulated answers) as follows. Every bit \( \hat{\pi}_i \) that was determined during the simulation is set to that value, and all other bits are set to 0. Then, \( \text{Sim}_{\text{Com}} \) generates random commitments \( \hat{C}_1, \hat{C}_2, \ldots \) for every bit of \( (\hat{c}^1, \ldots, \hat{c}^n) \), \( \hat{\pi} \) and sends them to \( R^* \).

3. Generating receiver queries and answers. Let \( r' \) denote the random string for \( V_{\text{ZK}} \) that \( R^* \) chooses when given the commitments \( \hat{C}_1, \hat{C}_2, \ldots \) (if \( R^* \) does not choose a valid random string, then \( \text{Sim}_{\text{Com}} \) sets \( r' = r_0 \), namely the same fixed random string for \( V_{\text{ZK}} \) that \( S \) would choose in this case). If \( r_{\text{ZK}} = r' \) then
Sim\textsubscript{Com} proceeds as the honest sender does (by opening the commitments to the corresponding bits). Otherwise, Sim\textsubscript{Com} returns to the second step.

**Construction 7.36.** The reveal-phase simulator Sim\textsubscript{Rev} operates similarly to the commit-phase simulator of Construction 7.35, except for the following modifications in the second step. First, for every \( i \in I \), Sim\textsubscript{Rev} honestly generates a secret-sharing \( c^i \) of \( x_i \) (by picking a random string \( r^i \) and computing \( c^i = \text{Share}(x_i, r^i) \) (for every \( i \notin I \), Sim\textsubscript{Rev} generates a random secret sharing \( c^i \) of 0, as in the second step of Construction 7.35); and the TTP-queries of Sim\textsubscript{ZK} are answered with the bits of \( c^1, \cdots, c^n \). Second, Sim\textsubscript{Rev} takes \( \hat{c} = c \).

Next, we show that the outputs of the commit-phase, and reveal-phase, simulators are statistically close to the real-world view of \( R^* \), thus proving hiding and zero-knowledge after reveal, respectively. The analysis follows the (somewhat simpler) analysis of the zero-knowledge interactive proof for graph 3-colorability of Goldreich et al. [GMW91]. The high level idea is as follows. First, we use the hiding of the commitment scheme to show that the random string \( r' \) that \( R^* \) chooses for \( V\text{ZK} \) is (almost) oblivious of the “secret sharing” and “proof” that the prover (or simulator) commits to. (This guarantees that both simulators guess correctly after “not too many” repetitions.) Then, we show that conditioned on the event that the simulators terminated, if the views are distinguishable then this contradicts either the zero-knowledge of the underlying HVZKPCPP system, the hiding of the commitment scheme, or the privacy of the secret sharing scheme.

Given a string \( s \), a random string \( r \) for \( R^* \), and a random string \( r\text{ZK} \) of \( V\text{ZK} \), let \( p_{s,r,r\text{ZK}} \) denote the probability that \( R^* \) chooses the random string \( r\text{ZK} \), conditioned on the event that \( R^* \) is run with random input \( r \), and is given random commitments to \( s \).

(The probability here is over the randomness used to generate the commitments to \( s \).)

The next lemma states that the random string that \( R^* \) chooses is essentially independent of the committed values he receives from \( S \) (or \( \text{Sim}_{\text{Com}}, \text{Sim}_{\text{Rev}} \)), and follows from the statistical hiding of the commitment scheme.

**Lemma 7.7.4.** If Com is statistically hiding, then for every pair of polynomials \( p, q : \mathbb{N} \to \mathbb{N} \), every pair of strings \( s, s' \in \{0,1\}^{q(n)} \), every random string \( r \) for \( R^* \), and every random string \( r\text{ZK} \) of \( V\text{ZK} \), there exists an \( n_0 \in \mathbb{N} \) such that for every \( n \geq n_0 \),

\[
|p_{s,r,r\text{ZK}} - p_{s',r,r\text{ZK}}| < \frac{1}{p(n)}.
\]

**Proof.** Assume towards contradiction that there exist polynomials \( p(n), q(n) \), and an infinite sequence \( \{s^n, s'^n, r^n, r^n\text{ZK}\}_n \) where \( s^n, s'^n \in \{0,1\}^{q(n)} \), and \( r^n, r^n\text{ZK} \) are as above, such that \( |p_{s^n,r^n,r^n\text{ZK}} - p_{s'^n,r^n,r^n\text{ZK}}| \geq \frac{1}{p(n)} \). We construct a distinguisher achieving \( \frac{1}{p(n)} \) advantage in distinguishing random commitments to \( s^n, s'^n \) for every such \( n \). (Using a standard hybrid argument, this contradicts the statistical hiding of the commitment scheme.) We incorporate \( R^* \) and \( r^n, r^n\text{ZK} \) (for every \( n \) in the sequence) into the distinguisher \( D \), who on input \( \bar{C} \) (a sequence of commitments to the bits of either \( s^n \) or \( s'^n \)
runs $R^*$ with random string $r^n$, $\vec{C}$ as the commitments of $S$. $D$ outputs 1 if and only if $R^*$ chooses the random string $r^n_{2K}$ for $V_{ZK}$. If $\vec{C}$ consists of random commitments to $s^n$ then $D$ outputs 1 with probability $p_{s^n,r^n,r_{2K}^n}$, and otherwise $D$ outputs 1 with probability $p_{s^n,r^n,r_{2K}^n}$.

The next lemma bounds the expected number of iterations in the simulations of $\text{Sim}_{\text{Com}}, \text{Sim}_{\text{Rev}}$, by showing that every iteration succeeds with a sufficiently high probability. For $(x, w) \in \mathcal{R}$, let $n := |x|$, and let $\ell = \ell(n)$ denote the combined lengths of the secret sharings $c^1, \ldots, c^n$ of $x_1, \ldots, x_n$ (respectively), and the ZKPCPP $\pi$ for $((c^1, \ldots, c^n), w)$.

**Lemma 7.7.5.** If $\text{Com}$ is statistically hiding, then every iteration of the simulators $\text{Sim}_{\text{Com}}, \text{Sim}_{\text{Rev}}$ (of Constructions 7.35, and 7.36, respectively) succeeds with probability at least $\frac{1}{2M}$, where $M$ denotes the number of random strings $r_{2K}$ of $V_{ZK}$. In particular, the expected number of iterations (until a correct guess) is $2M = 2 \cdot 2^{O(\log \ell(n))} = \text{poly} (\ell(n)) = \text{poly} (n)$.

**Proof.** We prove the lemma only for the simulator $\text{Sim}_{\text{Com}}$ of Construction 7.35 (the proof for the simulator $\text{Sim}_{\text{Rev}}$ of Construction 7.36 is similar). We prove the lemma conditioned on every possible random choice $r$ for $R^*$.

Let $p(n) = 2M$, which is polynomial in $n$ because $V_{ZK}$ uses $O(\log \ell(n)) = \log (\text{poly}(n))$ random coins. Then Lemma 7.7.4 guarantees that there exists an $n_0$ such that for every $n > n_0$, every $(x, w) \in \mathcal{R}, |x| = n$, every random string $r$ for $R^*$, and every random string $r_{2K}$ for $V_{ZK}$, the following holds. Let $(c^1, \ldots, c^n)$ be secret sharings for $(x_1, \ldots, x_n)$, and $\pi$ be an honestly-generated proof for $((c^1, \ldots, c^n), w)$. Similarly, let $(\hat{c}^1, \ldots, \hat{c}^n)$ be secret sharings of 0, and $\hat{\pi}$ be a simulated proof for $(\hat{c}^1, \ldots, \hat{c}^n)$ (i.e., the coordinates which $V_{ZK}$ queries when using randomness $r_{2K}$ were generated using $\text{Sim}_{ZK}$ with oracle $(\hat{c}^1, \ldots, \hat{c}^n)$, and the other coordinates are 0), then

$$|P((c^1, \ldots, c^n), \pi), r, r_{2K} - P((\hat{c}^1, \ldots, \hat{c}^n), \hat{\pi}), r, r_{2K}| < \frac{1}{2M}.$$ 

Consequently, for every large enough $n$, a single iteration of $\text{Sim}_{\text{Com}}$ succeeds with probability

$$\sum_{r_{2K}} \left( \Pr [\text{Sim}_{\text{Com}} \text{ guesses } r_{2K}] \cdot \sum_{(\hat{c}^1, \ldots, \hat{c}^n), \hat{\pi}} (p((\hat{c}^1, \ldots, \hat{c}^n), \hat{\pi}), r, r_{2K} \cdot \Pr [((\hat{c}^1, \ldots, \hat{c}^n), \hat{\pi}]) \right)$$

(the sum over $(\hat{c}^1, \ldots, \hat{c}^n), \hat{\pi}$ is over all secret sharings $(\hat{c}^1, \ldots, \hat{c}^n)$ of 0, and all simulated proofs $\hat{\pi}$, generated for the guess $r_{2K}$) which, by Lemma 7.7.4, is at least

$$\sum_{r_{2K}} \frac{1}{M} \cdot \sum_{(\hat{c}^1, \ldots, \hat{c}^n), \hat{\pi}} \left( p_{1 \ell(n), r, r_{2K}} - \frac{1}{2M} \right) \Pr [((\hat{c}^1, \ldots, \hat{c}^n), \hat{\pi})]$$

which is equal to

$$\sum_{r_{2K}} \frac{1}{M} \left( p_{1 \ell(n), r, r_{2K}} - \frac{1}{2M} \right) = \frac{1}{2M}.$$
We conclude that the expected number of iterations required to produce a correct guess is $2M$.  

**Remark 7.37.** Using Lemma 7.7.5, we can transform $\text{Sim}_{\text{Com}}$ (and $\text{Sim}_{\text{Rev}}$) into a strictly-polynomial time simulator $\text{Sim}'_{\text{Com}}$ as follows. $\text{Sim}'_{\text{Com}}$ emulates $n$ random and independent simulations of $\text{Sim}_{\text{Com}}$, where the second and third simulation steps of every emulation is repeated at most a bounded polynomial number (e.g. $6M$) of times, and output whatever $\text{Sim}_{\text{Com}}$ outputs in the first emulation that terminated (if none of the emulations terminated, $\text{Sim}'_{\text{Com}}$ outputs $\bot$). Using Markov’s inequality, every emulation terminates, except with probability at most $\frac{1}{3}$. As the emulations are independent, $\Pr[\text{Sim}'_{\text{Com}}(m) = \bot] \leq \left(\frac{1}{3}\right)^n \leq 2^{-n}$ (in particular, if $\text{Sim}_{\text{Com}}(m)$ is statistically close to the commit-phase view $V_{R^*}(x, w)$ of $R^*$, then $\text{Sim}'_{\text{Com}}(m), V_{R^*}(x, w)$ are also statistically close).

Next, we show that the simulated commit-phase view generated by $\text{Sim}_{\text{Com}}$ is statistically close to the real-world view of $R^*$. Intuitively, this follows from the statistical hiding of the commitment scheme, even after decommiting few commitments, which guarantees that the unopened commitments (to zeros in the simulation, and to “legal” bits of secret sharings and proof in the real-world) remain statistically close.

**Claim 7.7.6.** Let Protocol 7.33 be instantiated with:

- A statistically hiding commitment scheme $\text{Com}$.
- An HVZKPCPP system $(P_{ZK}, V_{ZK})$ with statistical HVZK, in which $V_{ZK}$ is non-adaptive and has query complexity $q$, and the simulator $\text{Sim}_{ZK}$ satisfies the following property. The set $Q$ of verifier queries that $\text{Sim}_{ZK}$ is asked to simulate completely determines the identity of the TTP queries that $\text{Sim}_{ZK}$ makes during the simulation.
- A $q$-private secret sharing scheme $(\text{Share}, \text{Rec})$.

Then Protocol 7.33 is statistically hiding.

**Proof.** Using Remark 7.37, it suffices to prove the claim conditioned on the event that $\text{Sim}_{\text{Com}}$ succeeded after at most $6M$ iterations (in particular, the simulation terminated after a finite number of steps). Moreover, as the random string $r$ for $R^*$ is equally distributed in both worlds, it suffices to prove the claim conditioned on every possible choice of $r$.

Given $(x, w) \in \mathcal{R}$, we define for every $r_{ZK}$ a pair of distributions $\mu_{r_{ZK}}(x), \eta_{r_{ZK}}(x, w)$ over the simulated and real-world views, which are the distributions over these views, conditioned on the event that $R^*$ chooses the random string $r_{ZK}$ (the simulated view is conditioned on $R^*$ choosing $r_{ZK}$ in one of the first $6M$ iterations). Let $(c^1, \cdots, c^n), (\hat{c}^1, \cdots, \hat{c}^n)$ be secret sharings of $(x_1, \cdots, x_n), 0^n$, respectively, $\pi$ be an honestly-generated proof for $((c^1, \cdots, c^n), w)$, and $\hat{\pi}$ be a simulated proof for
(\hat{c}^1, \ldots, \hat{c}^n) (generated for the simulator guess \(\hat{\tau} = r_{ZK}\)). Let \(p_{r_{ZK}}, q_{r_{ZK}}\) denote the probability that \(R^*\) chooses \(r_{ZK}\) in the ideal and real-world, respectively. By Lemma 7.7.4, \(p_{r_{ZK}}, q_{r_{ZK}}\) are statistically close.

Assume towards negation that the views are distinguishable, then there exists a polynomial \(p(n)\), an infinite sequence of \(n\)'s, and for every \(n\) a pair \((x^n, w^n) \in \mathcal{R}, |x^n| = n\), and a random string \(r^n_{ZK}\), such that \(|\mu_{r^n_{ZK}}(x^n) - \eta_{r^n_{ZK}}(x^n, w^n)| > \frac{1}{p(n^2)}\). Moreover, using Lemma 7.7.4, we can assume that for every such \(n\), \(R^*\) chooses \(r^n_{ZK}\) with non-negligible probability. Let \(D\) be a distinguisher achieving, for every such \(n\), advantage at least \(\frac{1}{p(n)}\) in distinguishing \(\mu_{r^n_{ZK}}(x^n), \eta_{r^n_{ZK}}(x^n, w^n)\).

For every \(n\) in the sequence let \(Q^n\) be the set of queries that \(V_{ZK}\) makes when using randomness \(r^n_{ZK}\). Let \(C^n, \hat{C}^n\) denote the distributions over random secret sharings of \((x^n_1, \ldots, x^n_4), 0^n\), respectively. Let \(\Pi^n\) denote the distribution over random proofs for the claims \((c_1^n, \ldots, c^n_4, w^n) \in \mathcal{L}_{R_{share}}\), where \((c_1^n, \ldots, c^n_4) \in C^n\) (i.e., the distribution is over the choice of \((c_1^n, \ldots, c^n_4)\), and then over the choice of the proof \(\pi^n\) for \((c_1^n, \ldots, c^n_4, w^n))\). Denote by \(\hat{\Pi}^n\) the distribution over simulated proofs for \((\hat{c}_1^n, \ldots, \hat{c}^n_4) \in \hat{C}^n\), generated for the guess \(\hat{\tau} = r^n_{ZK}\). (That is, the distribution is over the choice of \((\hat{c}_1^n, \ldots, \hat{c}^n_4)\), and then over the simulated proof. The indices of \(\hat{\pi}^n\) in \(Q^n\) were generated using \(\hat{Sim}_{ZK}\), and the others are 0.) Let \(\Pi^n\) denote the distribution over “proofs”, obtained from \(\Pi^n\) by replacing all the coordinates not in \(Q^n\) with zeros (i.e., we first pick a proof \(\pi^n\) according to \(\Pi^n\), and then replace all coordinates not in \(Q^n\) with zeros). Let \(\hat{\Pi}^n\) denote the distribution over simulated proofs, generated by \(\hat{Sim}_{ZK}\) for the input oracle \((c_1^n, \ldots, c^n_4) \in C^n\), and the queries \(Q^n\) (the distribution is over the choice of \((c_1^n, \ldots, c^n_4)\) and the randomness of \(\hat{Sim}_{ZK}\)).

For every \(n\) in the sequence we define hybrids as follows, where the views in all hybrids are conditioned on the event that \(R^*\) chooses the random string \(r^n_{ZK}\). \(H^n_0\) is the distribution over the view of \(R^*\) when given random commitments to \((c_1^n, \ldots, c^n_4)\), \(\pi^n\) drawn according to \(C^n, \Pi^n\) (in particular, \(H^n_0 = \eta_{r^n_{ZK}}(x^n, w^n)\)); \(H^n_1\) is the distribution of the view of \(R^*\) when given random commitments to \((\hat{c}_1^n, \ldots, \hat{c}^n_4)\), \(\hat{\pi}^n\) drawn according to \(\hat{C}^n, \hat{\Pi}^n\); \(H^n_2\) is the distribution over the view of \(R^*\) when \((\hat{c}_1^n, \ldots, \hat{c}^n_4)\), \(\hat{\pi}^n\) are drawn according to \(\hat{C}^n, \hat{\Pi}^n\), and then \(R\) is given random commitments to \(\hat{\pi}^n\) and \((\hat{c}_1^n, \ldots, \hat{c}^n_4)\) in the coordinates in \(Q^n\), and random commitments to zeros in the other coordinates; \(H^n_3\) is the distribution over the view of \(R^*\) when given random commitments to \(\hat{\pi}^n\), random commitments to \((\hat{c}_1^n, \ldots, \hat{c}^n_4)\) in the coordinates in \(Q^n\), and random commitments to zeros in the other coordinates; \(H^n_4\) is the distribution over the view of \(R^*\) when given random commitments to \(\hat{c}^n\), \(\hat{\pi}^n\) which were drawn according to \(\hat{C}^n, \hat{\Pi}^n\) (in particular, \(H^n_4 = \mu_{r^n_{ZK}}(x^n)\)).

Then for infinitely many \(n, r^n_{ZK}, x^n, w^n\) such that \(|x^n| = n\),

\[
\frac{1}{p(n)} \leq |\mu_{r^n_{ZK}}(x^n) - \eta_{r^n_{ZK}}(x^n, w^n)| = |H^n_4 - H^n_0| \leq \sum_{i=0}^{4} |H^n_{i+1} - H^n_i|
\]

227
In particular, for some $0 \leq i \leq 3$, $|H_{i+1}^n - H_i^n| > \frac{1}{4p(n)}$ for infinitely many $n$’s, i.e., there exists a distinguisher $D$ such that $|D(H_{i+1}^n) - D(H_i^n)| > \frac{1}{4p(n)}$.

We show that $|\Pr[D(H_i^n)] - \Pr[D(H_0^n)]| > \frac{1}{4p(n)}$, or $|\Pr[D(H_i^n)] - \Pr[D(H_0^n)]| > \frac{1}{4p(n)}$, contradicts the hiding of the commitment scheme. We show this for $i = 0$ (the case $i = 3$ can be proved similarly). Using an averaging argument, there exist $(c^{1,n}, \ldots, c^{n,n}), \pi^n$ such that the distinguishing gap does not decrease even when the hybrids are conditioned on these values. We construct a distinguisher $D'$ that can distinguish random commitments to zeros from random commitments to the coordinates of $\pi^n$ that are not in $Q^n$ (using a standard hybrid argument, this contradicts the hiding of the commitment scheme). We incorporate $(c^{1,n}, \ldots, c^{n,n}), \pi^n, r_2^n, Q^n$ into $D'$.

On input $Z^n$, $D'$ generates random commitments to $(c^{1,n}, \ldots, c^{n,n})$, random commitments to the bits of $\pi^n$ indexed by $Q^n$, and feeds $R^*$ with $Z^n$, the commitments it generated, and the openings of these commitments. If $R^*$ chooses $r' \neq r_2^n$, then $D'$ outputs 0 (in this case $D'$ “gives up” and does not try to distinguish between the views). Otherwise, $D'$ feeds the view of $R^*$ to $D$, and outputs whatever $D$ outputs.

Notice that if $Z^n$ consists of commitments to zeros then the input to $D$ is distributed according to $H^n_1$, and otherwise it is distributed according to $H^n_0$. Therefore, the advantage that $D'$ achieves is at least

$$|\Pr[D(H_{i+1}^n)] - \Pr[D(H_i^n)]| \cdot p_{r_2^n} - \text{negl}(n)$$

which is non-negligible since $p_{r_2^n}$ is non-negligible. (The loss of a negligible factor in the term above is due to the possible difference between the probability that $R^*$ chooses $r_2^n$ when given commitments to $(c^{1,n}, \ldots, c^{n,n}), \pi^n$ and $(c^{1,n}, \ldots, c^{n,n}), \tilde{\pi}$. By Lemma 7.7.4, this factor is negligible.)

We claim that $|\Pr[D(H_0^n)] - \Pr[D(H_i^n)]| > \frac{1}{4p(n)}$ contradicts the zero-knowledge of the ZKPCPP system. Indeed, using an averaging argument, for every $n$ there exist $(c^{1,n}, \ldots, c^{n,n})$ such that the distinguishing gap is preserved even when the views are conditioned on $(c^{1,n}, \ldots, c^{n,n})$. We construct a distinguisher $D'$ between the real-world and simulated view of $V_{2K}$, when verifying the claim “$((c^{1,n}, \ldots, c^{n,n}), w^n) \in \mathcal{R}_{\text{Share}}$”, by incorporating $(c^{1,n}, \ldots, c^{n,n}), r_2^n, Q^n$ into $D'$.

$D'$ on input $Z^n$ checks whether the random string used by $V_{2K}$ was $r_2^n$, and if not outputs 0 (in this case, $D'$ does not try to distinguish between the views). Otherwise, $D'$ generated random commitments to $(c^{1,n}, \ldots, c^{n,n})$, random commitments to zeros in the proof coordinates not in $Q^n$, and random commitments to the proof bits that appear in the view (namely, that appear in $Z^n$), and feeds $D$ with $Z^n$, the commitments, and the openings of the commitments to $Z^n$.

Notice that if $Z^n$ is a sampling from the real-world view of $V_{2K}$ then the input to $D$ is distributed according to $H^n_1$, and otherwise it is distributed according to $H^n_2$. 228
Therefore, the advantage of $\mathcal{D}'$ is at least

$$|\Pr[D(H_3^n) = 1] - \Pr[D(H_2^n) = 1]| \cdot \Pr[V_{\text{ZK}} \text{ chooses } r_{\text{ZK}}^n]$$

where $\Pr[V_{\text{ZK}} \text{ chooses } r_{\text{ZK}}^n]$ is non-negligible (since there is a polynomial number of random strings of $V_{\text{ZK}}$, and the choice of $V_{\text{ZK}}$ is independent of his oracles).

Finally, we show that $|\Pr[D(H_3^n) = 1] - \Pr[D(H_2^n)]| > \frac{1}{4p(n)}$ contradicts the $t(n)$-privacy of the secret sharing scheme (where $t(n) = q$). We incorporate $r_{\text{ZK}}^n$ into the distinguisher $\mathcal{D}'$, and show that there exists a subset $I \subseteq [n]$ such that $\mathcal{D}'$ achieves advantage at least $\frac{1}{4p(n)}$ in distinguishing $(c_1^n, \ldots, c_n^n) | I$, $(\hat{c}_1^n, \ldots, \hat{c}_n^n) | I$. $I$ consists of all the shares $s_i$ such that when asked to simulate $Q^n$, $V_{\text{ZK}}$ makes a TTP query to a bit of $s_i$.

$\mathcal{D}'$ on input $Z^n$ runs $V_{\text{ZK}}$, and queries it about $Q^n$. Whenever $V_{\text{ZK}}$ makes a TTP query, $\mathcal{D}'$ extracts the answer to the query from $Z^n$. Let $\pi''$ denote the simulated bits of the proof oracle which $V_{\text{ZK}}$ generates during the simulation, then $\mathcal{D}'$ generates random commitments, and openings of these commitments, to $\pi''$, $Z^n$, and generates random commitments to zeros in all other coordinates. Then, $\mathcal{D}$ runs $R^*$ with the commitments, and if $R^*$ chooses $r' \neq r_{\text{ZK}}^n$ then $\mathcal{D}'$ outputs $0$. Otherwise, $\mathcal{D}'$ runs $\mathcal{D}$ with the view of $R^*$ as input, and outputs whatever $\mathcal{D}$ outputs.

Notice that if $Z^n$ is distributed according to $(c_1^n, \ldots, c_n^n) | I$ then the input of $\mathcal{D}$ is distributed according to $H_2^n$, and otherwise it is distributed according to $H_3^n$, so using Lemma 7.7.4, the advantage of $\mathcal{D}'$ is at least $\frac{1}{4p(n)} \cdot p_{\text{ZK}} - \text{negl}$, which is non-negligible.$\square$

**Claim 7.7.7.** Let $(P_{\text{ZK}}, V_{\text{ZK}})$ be an HVZKPCPP system with statistical HVZK, where $V_{\text{ZK}}$ is non-adaptive, and $\text{Com}$ be a statistically hiding commitment scheme. Then Protocol 7.33 has statistical zero-knowledge after reveal.

**Proof.** Similar to the proof of Claim 7.7.6, it suffices to prove that for every input $(x, w)$ of $S$, every $\mathcal{I} \subseteq [n]$, and every random string $r_{\text{ZK}}$ for $V_{\text{ZK}}$, the simulated and real-world views are statistically close, conditioned on the event that $R^*$ chooses $r_{\text{ZK}}$ in both worlds with non-negligible probability. Otherwise, there exist a distinguisher $\mathcal{D}$, a polynomial $p(n)$, infinitely many $n$'s, infinitely many input pairs $(x^n, w^n) \in R$, $|x| = n$ of $S$, infinitely many subsets $\mathcal{I}_n \subseteq [n]$, and infinitely many random strings $r_{\text{ZK}}^n$ for $V_{\text{ZK}}$ which $R^*$ chooses with non-negligible probability, such that $|\text{Sim}_{\text{Rev}}\left(\left\{x_i^n\right\}_{i \in \mathcal{I}_n}\right) - \left(\forall R^* \left( x^n, w^n, \mathcal{I}_n \right), \text{dec} \right)| > \frac{1}{p(n)}$, where both views are conditioned on the event that $R^*$ chooses the random string $r_{\text{ZK}}^n$ for $V_{\text{ZK}}$. (Here, $\forall R^* \left( x^n, w^n, \mathcal{I}_n \right)$ denotes the view of $R^*$ after the reveal phase, in an execution of the protocol in which $S$ has input $(x^n, w^n)$, and is given the subset $\mathcal{I}_n$ during the reveal phase.)

Let $Q^n$ denote the coordinates that $V_{\text{ZK}}$ queries when using randomness $r_{\text{ZK}}^n$. Remember that for every $i \in \mathcal{I}_n$, the secret sharing $c_i^n$ that $\text{Sim}_{\text{Rev}}$ generates is identically distributed to the real world secret sharing (since it is a random secret sharing of $x_i^n$), so we can condition both $\text{Sim}_{\text{Rev}}(x^n)$, and $\left(\forall R^* (x^n, w^n), \text{dec} \right)$, on $\left\{c_i^n\right\}_{i \in \mathcal{I}_n}$.
Let $\Pi^n$ denote the distribution over honestly-generated proofs for $((c_1^n, \ldots, c_n^n), w^n)$. Let $\Pi^\rho$ denote the distributions over simulated proofs for $((c_1^n, \ldots, c_n^n), w^n)$, generated for the guess $\hat{r} = r^\rho_{ZK}$. (That is, for every $\hat{\pi}^n \in \Pi^\rho$, the indices of $\hat{\pi}^n$ in $Q^n$ were generated using $\text{Sim}_{ZK}$, and the others are 0.) Let $\hat{\Pi}^\rho$ denote the distributions over simulated proofs for $((\hat{c}_1^n, \ldots, \hat{c}_n^n), \hat{w}^n)$, generated for the guess $\hat{r} = r^\rho_{ZK}$, where for every $i \in I_n$, $\hat{c}_i^n = c_i^n$, otherwise $\hat{c}_i^n$ is a random secret sharing of 0. (That is, for every $\hat{\pi}^n \in \Pi^\rho$, the indices of $\hat{\pi}^n$ in $Q^n$ were generated using $\text{Sim}_{ZK}$, and the others are 0.) Let $\Pi_0^n$ denote the distribution over “proofs”, obtained from $\Pi^n$ by replacing all the coordinates not in $Q^n$ with zeros (i.e., we first pick a proof $\pi^n$ according to $\Pi^n$, and then replace all coordinates not in $Q^n$ with zeros).

We define hybrids as follows, where the views in all hybrids are conditioned on the event that $R^*$ chooses the random string $r^n_{ZK}$. $H_0^n$ is the distribution over the view of $R^*$ when given random commitments to $(c_1^n, \ldots, c_n^n), \pi^n$, where $\pi^n$ was drawn according to $\Pi^n$; $H_1^n$ is the distribution over the view of $R^*$ when given random commitments to $(c_1^n, \ldots, c_n^n), \pi_0^n$, where $\pi_0^n$ was drawn according to $\Pi_0^n$; $H_2^n$ is the distribution over the view of $R^*$ when given random commitments to $(\hat{c}_1^n, \ldots, \hat{c}_n^n), \hat{\pi}^n$, where $\hat{\pi}^n$ was drawn according to $\hat{\Pi}^\rho$; $H_3^n$ is the distribution over the view of $R^*$ when given random commitments to $(\hat{c}_1^n, \ldots, \hat{c}_n^n), \hat{\pi}^n$, where $\hat{\pi}^n$ was drawn according to $\hat{\Pi}^\rho$.

Similar to the proof of Claim 7.7.6, there exists an $0 \leq i \leq 2$ such that $\mathcal{D}$ distinguishes $H_{i+1}^n$ from $H_i^n$, for infinitely many $n$’s, with advantage at least $\frac{1}{\text{poly}(n)}$. For $i = 0$, or $i = 2$, this contradicts the hiding of the commitment scheme, and for $i = 1$ this contradicts the zero-knowledge of the ZKPCPP system (the proof is similar to the proof of Claim 7.7.6).

**Reducing the Soundness Error**

Protocol 7.33 achieves a constant soundness error (i.e., some malicious sender $S^*$ can possibly win the binding game with constant probability $\epsilon$). To amplify soundness, $S$ and $R$ sequentially repeat the commitment phase of Protocol 7.33 many independent times, where $R$ is convinced that $S$ is committed to some value $x \in L_R$ only if all the sequential commitment phases succeeded (i.e., no inconsistency was detected). Consequently, if the commitment phase is executed $k$ sequential times, then a malicious sender $S^*$ can win the binding game in the sequential repetition of the protocol, only if he was able to do so in each sequential commitment phase, so the error reduces to $\epsilon^k$.

This sequential repetition preserves input-hiding during commit. (Indeed, zero-knowledge with respect to auxiliary inputs is preserved under sequential repetition [GO94], and every zero-knowledge interactive proof system with a black-box simulator can be extended to a zero-knowledge proof with respect to auxiliary inputs.) Indeed, if the protocol is repeated a polynomial number of times, then by Lemma 7.7.5 after a total of $\text{poly}(k, n)$ iterations, the simulator successfully guesses the queries of $R^*$ in all iterations. (This holds since the simulator only needs to sequentially guess the
queries of $R^*$, i.e., in every simulation of the commit phase, he can rewind $R^*$ to the last state in which his guess regarding the queries of $R^*$ was correct.)

Zero-knowledge after reveal is also preserved, since the simulator can emulates several random copies of $\text{Sim}_{\text{rev}}$ (with the same subset $I$). Consequently, we can achieve a negligible soundness error with polylogarithmic communication complexity during the commitment phase. (For example, setting $k = \log^2 n$ yields a protocol with soundness error $2^{-O(\log^2 n)} = n^{-O(\log n)} = \text{negl}(n)$, and communication complexity $k \cdot \text{polylog} n = \text{polylog} n$.) This sequential repetition proves Theorem 7.25, as we now show.

**Proof of Theorem 7.25.** We instantiate Protocol 7.33 with: the HVZKPCPP system, and the secret sharing scheme of Remark 7.32; an exponentially-hard collision-resistant family $\mathcal{H}$; and the commitment scheme obtained from $\mathcal{H}$ through Theorem 7.30. Let $c$ denote the constant such that $\mathcal{H}$ is $(2^{-c}, 2^{c})$-hard, then we take $t_{\text{sec}} = (\log n)^{\frac{2}{c}}$. We modify Protocol 7.33 such that the commitment phase is repeated sequentially $k = \log^2 n$ times. We show that the protocol obtained in this manner has the required properties.

**Communication complexity during the commit phase.** We first analyze the communication complexity of every iteration of the commitment phase. The first two messages during the commit phase are of length $t_{\text{sec}} = \text{polylog} n$. The third message is of size $O(t \cdot \log(n)) = O(1) \cdot \log(n) = O(\log n)$ (since the queries are to indices in a $\text{polylog}(n)$-length proof). The forth message has size at most $O(q \cdot (l + s) \cdot t_{\text{sec}}) = \text{polylog} n$ since $2^s, 2^l = \text{poly}(n)$, so the total communication complexity during each iteration of the commitment phase is $\text{polylog} n$. Since the commit phase is repeated $k = \log^2 n$ times, the total communication during the commit phase is $\text{polylog} n$.

We note that the communication complexity during the reveal phase is $\tilde{O}(n \cdot \tilde{n}(n))$. Indeed, the binary representation of $c$ consists of $\tilde{O}(\tilde{n}(n))$ bits, so the total length of the commitments to all its bits is $\tilde{O}(\tilde{n}(n)) \cdot O(t_{\text{sec}}) = \tilde{O}(\tilde{n}(n))$. The MHT that $S$ send to $R$ during the reveal phase contains $\tilde{O}(\tilde{n}(n))$ “nodes”, namely $S$ sends at most $\tilde{O}(\tilde{n}(n))$ messages, each of size $\text{polylog} \tilde{n}(n)$. Since $\tilde{n}(n) = O(n)$, the total communication is $\tilde{O}(n)$.

**Completeness.** Follows directly from the perfect completeness of the underlying HVZKPCPP system $(P_{ZK}, V_{ZK})$, and the commitment scheme of Theorem 7.30.

**Binding.** Lemma 7.7.3 guarantees that every iteration of the commit phase has $\epsilon + \text{negl}(n)$ binding, where $\epsilon$ is constant. As the commitment phase is repeated $k = \log^2 n$ times, the protocol has $\text{negl}(n)$-binding.

**Hiding.** Follows directly from Claim 7.7.6, because the commitment scheme is statistically hiding, $(P_{ZK}, V_{ZK})$ has statistical HVZK, and the identity of the TTP queries of $\text{Sim}_{ZK}$ are completely determined by the identity of the queries that it is asked to simulate (see Remark 5.17), and the secret sharing scheme is $q$-private.

**Zero-knowledge after reveal.** Follows directly from Claim 7.7.7, because the com-
mitment scheme is statistically hiding, and $(P_{ZK}, V_{ZK})$ has statistical HVZK.

We note that if $H$ only satisfies the usual notion of super-polynomial hardness, the communication complexity (during commit) of the resulting Commit-and-Prove protocol can be $O(n^\gamma)$, for an arbitrarily small $\gamma > 0$.

### 7.7.2 Computational Commit-and-Prove with a Constant Number of Rounds

In this section we present a Commit-and-Prove protocol with a constant number of rounds in the coin-flipping hybrid model, in which the parties can use an ideal coin-flipping functionality $F_{CF}$, that on input $1^t$ outputs a random string $r \in_R \{0,1\}^t$ to both parties. Then, we instantiate $F_{CF}$ using a computationally-secure coin-flipping protocol, obtaining a constant-round Commit-and-Prove protocol in which the binding, hiding, and zero-knowledge after reveal properties hold against computationally-bounded adversaries. (This should be contrasted with the Commit-and-Prove protocol of Section 7.7.1, which guarantees hiding, and zero-knowledge after reveal, even against computationally-unbounded receivers, but requires super-constant many rounds to achieve a negligible error.)

The protocol in the coin-flipping hybrid model differs from Protocol 7.33 only in the manner in which the parties choose the random string for the verifier of the underlying HVZKPCPP system.

**Protocol 7.38.** The Commit-and-Prove protocol in the coin-flipping hybrid model is identical to Protocol 7.33, except that $S, R$ use the coin-flipping functionality $F_{CF}$ to determine the random string $r_{ZK}$ for the HVZKPCPP verifier.

Next, we analyze the properties of Protocol 7.38, using the same notations used in Section 7.7.1. That is, $t(n) = q$ denotes a privacy function (where $q$ is the query complexity of the honest verifier $V_{ZK}$ of the HVZKPCPP system), $\tau(n)$ is a robustness function, and $\hat{n}(n)$ denotes a secret sharing length function. We use MHT $(y)$ to denote a MHT commitment to a string $y$ (constructed by first generating commitments to the bits of $y$, and then hashing the commitments to a single block, using the MHT).

The following lemma states that the protocol is binding, and can be proven similarly to Lemma 7.7.3.

**Lemma 7.7.8.** Let $\epsilon > 0$ be an error parameter, $(\text{Share}, \text{Rec})$ be the secret sharing scheme of Remark 7.32, and $(P, V)$ be an HVZKPCPP for the relation $R_{Share}$ of Remark 7.32, with proximity parameter $\frac{1}{\Omega}$, soundness error $\epsilon$, and a non-adaptive honest verifier. Let $H$ be a super-polynomially hard family of CRHFs, and $\text{Com}$ be a computationally binding commitment scheme. Then Protocol 7.38, when instantiated with $(\text{Share}, \text{Rec}), (P, V)$, and $\text{Com}$, has $\epsilon + \text{negl}(n)$ binding.

The next lemma states that the protocol is hiding.
Lemma 7.7.9. Let Protocol 7.38 be instantiated with:

- A statistically hiding commitment scheme $\text{Com}$.
- An HVZKPCPP system $(P_{ZK}, V_{ZK})$ with perfect HVZK, in which $V_{ZK}$ is non-adaptive and has query complexity $q$.
- A $q$-private secret sharing scheme $(\text{Share}, \text{Rec})$.

Then Protocol 7.38 is statistically hiding in the coin-flipping hybrid model.

Proof. We describe a commit-phase simulator $\text{Sim}_{\text{Com}}$. Let $R^*$ be a (possibly malicious) receiver, and let $(x, w) \in \mathcal{R}, |x| = n$, then $\text{Sim}_{\text{Com}}$ on input $1^n$ chooses a random string $r^*$ for $R^*$, and extracts from it the message $\alpha \in \{0,1\}^* \text{uc}$ that $R^*$ sends to $S$. Then, $\text{Sim}_{\text{Com}}$ chooses a random string $r_{ZK}$ for $V_{ZK}$ (as the output of the coin-flipping functionality $\mathcal{F}_{\text{CF}}$), and extracts from it the set $Q = (q_1, \ldots, q_q)$ of queries of $V_{ZK}$. Next, $\text{Sim}_{\text{Com}}$ computes $n$ secret sharings $\hat{c}^i = \text{Share}(0, r^i)$ for $1 \leq i \leq n$, and random $r^i$'s, and uses $\text{Sim}_{ZK}$ to simulate the oracle answers $(a_1, \ldots, a_q)$, where the TTP-queries of $\text{Sim}_{ZK}$ are answered according to $\hat{c}^i, \ldots, \hat{c}^n$. When the simulation of $\text{Sim}_{ZK}$ terminate, $\text{Sim}_{\text{Com}}$ constructs a “proof” $\hat{\pi}$, where bits that $V_{ZK}$ queries are set to the corresponding simulated values, and all other bits are set to zero. For the remainder of the simulation, $\text{Sim}_{\text{Com}}$ follows the protocol as the honest sender would (i.e., constructs the MHTs for $(\hat{c}^1, \ldots, \hat{c}^n), \hat{\pi}$, etc.).

We claim that the simulated transcript is statistically close to the transcript of the commit phase in Protocol 7.38 in the coin-flipping hybrid model. We prove a stronger claim, that the transcripts are statistically close even if the commitments to the secret-sharing and proof are not hashed-down. As $r^*, r_{ZK}$ are distributed identically in both worlds, we prove this conditioned on every possible choice of values for these queried bits. This conditioning implies that the TTP answers given to $\text{Sim}_{ZK}$ are distributed identically to the input oracle of $V_{ZK}$. Therefore, the perfect honest-verifier zero-knowledge of $(P_{ZK}, V_{ZK})$ guarantees that the simulated bits of $\pi, \hat{\pi}$ are identically distributed, so we can further condition both distributions on the event that the same bit values were decommitted.

The statistical indistinguishability now follows from the statistical hiding of the commitment scheme. Otherwise, there exist a distinguisher $\mathcal{D}$, a polynomial
is run with random commitments to

Similar to the proof of Lemma 7.7.9, we define hybrid distributions over the views of

are conditioned on the random string

(x, w) \in \mathcal{R}

such that

Pr[D(W_{R^*}) = 1] - Pr[D(W_{Sim}) = 1] > \frac{1}{p(n)} \quad (\text{where } W_{R^*}, W_{Sim} \text{ denote the restriction of } W_{R^*}, W_{Sim} \text{ to the distributions in which the inputs are } (x, w), 1^n, \text{ respectively}).

Using an averaging argument, for every n in the sequence, there exist

(c_1, \ldots, c_n), (\tilde{c}_1, \ldots, \tilde{c}_n), \tilde{\pi}

(consistent with the values we have already fixed) such that the distinguishing advantage is preserved, even when

W_{R^*}, W_{Sim}

are conditioned on

(c_1, \ldots, c_n), \pi, (\tilde{c}_1, \ldots, \tilde{c}_n), \tilde{\pi}

(Recall that in

(c_1, \ldots, c_n), (\tilde{c}_1, \ldots, \tilde{c}_n),

the decommitted coordinates are identical, and the same holds for \pi, \tilde{\pi}).

We construct a distinguisher \mathcal{D}' that can distinguish, for every such n, random commitments to

((c_1, \ldots, c_n), \pi) \mid_{Q_n}

(Recall that

((c_1, \ldots, c_n), \pi)

not indexed by \mathcal{Q}^n, and \mathcal{Q}^n is the set of bits that V_{ZK} queries when using randomness r_{ZK}^n), from random commitments to

((c_1, \ldots, c_n), \tilde{\pi}) \mid_{Q_n}

\mathcal{D}' on input 1^n, \tilde{Z} generates random commitments to

((c_1, \ldots, c_n), \pi) \mid_{Q^n}

decommissions of these values, concatenates the commitments and decommitments to \tilde{Z}, gives this as input to \mathcal{D}, and outputs whatever \mathcal{D} outputs. Notice that if \tilde{Z} is distributed as random commitments to

((c_1, \ldots, c_n), \pi) \mid_{Q_n}

then the input to \mathcal{D} is distributed according to \mathcal{W}_{\mathcal{D}}^\pi, and otherwise it is distributed according to \mathcal{W}_{Sim} (since we have conditioned on the event that

((c_1, \ldots, c_n), \pi) \mid_{Q^n} = ((\tilde{c}_1, \ldots, \tilde{c}_n), \tilde{\pi}) \mid_{Q^n}.

Therefore, \mathcal{D}' achieves a non-neglible advantage in distinguishing sequences of poly(n) commitments which (by a standard hybrid argument) contradicts the statistical hiding of the commitment scheme.

Lemma 7.7.10. Let (P_{ZK}, V_{ZK}) be an HVZKPCPP system with perfect HVZK, where V_{ZK} is non-adaptive, and Com be a statistically hiding commitment scheme. Then Protocol 7.38 has statistical zero-knowledge after reveal in the coin-flipping hybrid model.

Proof. The proof is similar to the proof of Lemma 7.7.9, but simpler since the reveal-phase simulator \text{Sim}_{Rev} is given the input \{x_i\}_{i \in I}, and can therefore generate secret sharings that are consistent with x in the bits indexed by I. We present only the points in which the proof deviates from the proof of Lemma 7.7.9. The simulation is the same, except that \text{Sim}_{Rev} on input \{x_i\}_{i \in I} generates the secret sharings \(\tilde{c}_i\) as follows: for every \(i \in I\), \(x_i\) is a random secret sharing of \(x_i\), and for \(i \notin I\), \(\tilde{c}_i\) is a random secret sharing of 0.

Define \mathcal{W}_{R}, \mathcal{W}_{Sim} as in the proof of Lemma 7.7.9 (in particular, both distributions are conditioned on the random string \(r^*\) of \(R^*\), and the random string \(r_{ZK}\) used for the emulation of \(V_{ZK}\)). Notice that for every \(i \in I\), \(c_i, \tilde{c}_i\) are identically distributed, so it suffices to prove indistinguishability conditioned on every possible value of \(\{c_i\}_{i \in I}\).

Similar to the proof of Lemma 7.7.9, we define hybrid distributions over the views of \(R^*\) as follows. \(H_0^n\) is the distribution over the view of \(R^*\) when he is run with random commitments to \((c_1, \ldots, c_n), \pi, H_1^n\) is the distribution over the view of \(R^*\) when he is run with random commitments to \((c_1, \ldots, c_n), \pi_0, \text{ where } \pi_0\) is created by honestly

234
generating a proof $\pi$ for $(c^1, \ldots, c^n)$, then replacing every bit of $\pi$ not queried by $R^*$ (when using $r_{ZK}$) with 0; $H^n_2$ is the distribution over the view of $R^*$ when he is run with random commitments to $(c^1, \ldots, c^n)$, $\hat{\pi}$, where $\hat{\pi}$ is a simulated proof for $(c^1, \ldots, c^n)$ (in particular, bits of $\hat{\pi}$ not queried by $R^*$, when using $r_{ZK}$, are 0); and $H^n_3$ is the distribution over the view of $R^*$ when he is run with random commitments to $(\hat{c}^1, \ldots, \hat{c}^n)$, $\hat{\pi}$, where $\hat{\pi}$ is a simulated proof for $(\hat{c}^1, \ldots, \hat{c}^n)$. Then the statistical distance of the simulated and actual views follows from the fact that every pair of adjacent hybrids are statistically close: for $H^n_0$, $H^n_1$ this follows from the statistical hiding of the underlying commitment scheme; for $H^n_1$, $H^n_2$ this follows from the perfect honest-verifier zero-knowledge of $(P_{ZK}, V_{ZK})$; and for $H^n_2$, $H^n_3$ this follows from the privacy of the secret sharing scheme.

The proof of Theorem 7.27 will use the following HVZKPCPP system.

**Remark 7.39.** Let $\epsilon > 0$ be a soundness parameter, and $R_{Share}$ be the NP relation of Remark 7.32, then according to the remark it has a PCPP system with proximity parameter $\frac{1}{10}$, soundness error $\frac{1}{2}$, randomness complexity $O(\log n)$ (where $n$ denotes the input length), and query complexity $O(1)$. Plugging this system into Theorem 5.16 with soundness parameter $\epsilon$, we get an HVZKPCPP system $(P_{ZK}, V_{ZK})$ for $R_{Share}$, with proximity parameter $\frac{1}{10}$, soundness error $\epsilon$, perfect completeness, and perfect honest-verifier zero-knowledge. Moreover, the proofs have size $\log \frac{1}{\epsilon} \cdot \text{poly}(n)$, and the honest verifier tosses only $\log n \cdot \text{polylog}\frac{1}{\epsilon}$ random coins, and non-adaptively reads only $q = O(\log \frac{1}{\epsilon})$ bits from the oracles.

Since $q$ is independent of the input length, we can take the privacy parameter of the secret sharing scheme $(\text{Share}, \text{Rec})$ of Remark 7.32 to be $t(n) = q$, which would only incur an increase of $\text{polylog}\frac{1}{\epsilon}$ in the proof length, and query complexity, so we obtain an HVZKPCPP system with proofs of length $\text{poly}(n, \log \frac{1}{\epsilon})$, and randomness complexity $\log n \cdot \text{polylog}\frac{1}{\epsilon}$.

**Proof of Theorem 7.27.** Let $\epsilon = n^{-\log n}$. We instantiate Protocol 7.38 with: the HVZKPCPP system, and the secret sharing scheme of Remark 7.39 (where $\epsilon$ is the soundness parameter of the HVZKPCPP system); an exponentially-hard collision-resistant family $H$; and the commitment scheme obtained from $H$ through Theorem 7.30. Let $c$ denote the constant such that $H$ is $(2^{-c}, 2^{-c})$-hard, then we take $t_{sec} = (\log n)^2$.

**Communication complexity during commit.** The first two message have length $t_{sec} = \text{polylog} n$. The ideal coin-flipping functionality $\mathcal{F}_{CF}$ provides each of $S, R$ with $\text{rand}$ bits, where $\text{rand} = \log n \cdot \text{polylog}\frac{1}{\epsilon}$ denotes the number of random bits that $V_{ZK}$ uses. The next message (in which $S$ answers the queries of $V_{ZK}$) has size at most $O(q \cdot (l + s) \cdot t_{sec})$, where $2^l, 2^s = \text{poly}(n)$, so the total communication complexity during the commit phase is $\text{polylog}(n, \frac{1}{\epsilon}) = \text{polylog} n$.

We note that the communication complexity during the reveal phase is $\log \frac{1}{\epsilon} \cdot \tilde{O}(n)$ (i.e., for $\epsilon = n^{-\log n}$ as chosen above, the total communication complexity is $\tilde{O}(n)$).
Indeed, the binary representation of \( c \) consists of \( n = O(n + \log \frac{1}{\varepsilon}) \) bits, so the total length of the commitments to all its bits is \( O(n + \log \frac{1}{\varepsilon}) \cdot O(t_{\text{sec}}) \leq \log \frac{1}{\varepsilon} \cdot \tilde{O}(n) \). The MHT that \( S \) send to \( R \) during the reveal phase contains \( O(\hat{n}(n)) \leq \log \frac{1}{\varepsilon} \cdot \tilde{O}(n) \) “nodes”, namely \( S \) sends at most \( \log \frac{1}{\varepsilon} \cdot \tilde{O}(n) \) messages, each of size \( \text{polylog} n \), for a total of \( \tilde{O}(n) \) communication.

**Completeness.** Follows directly from the perfect completeness of the underlying HVZKPCPP system \((P_{2K}, V_{2K})\), and the correctness of the commitment scheme of Theorem 7.30.

**Binding.** Lemma 7.7.8 guarantees that Protocol 7.38 has \( \varepsilon + \text{negl}(n) \) binding, which (for our choice of \( \varepsilon = \text{negl}(n) \)) gives \( \text{negl}(n) \)-binding.

**Hiding and zero-knowledge after reveal.** Lemma 7.7.9, and Lemma 7.7.10, guarantee that Protocol 7.38 is statistically hiding, and has zero-knowledge after reveal, respectively, in the coin-flipping hybrid model.

As was the case in Section 7.7.1, if \( \mathcal{H} \) only satisfies the usual notion of superpolynomial hardness, the communication complexity (during commit) of the resulting Commit-and-Prove protocol can be \( O(n^\gamma) \), for an arbitrarily small \( \gamma > 0 \).

Though we consider our Commit-and-Prove protocol in the hybrid model to be our main contribution in this section, we also describe a Commit-and-Prove protocol in the plain model. We replace the ideal coin-flipping functionality \( F_{\text{CF}} \) with the computationally-secure 2-party coin-flipping protocol of [PW09]. (The protocol allows a pair of parties to generate poly-many random coins, is secure against expected polynomial time adversaries, and makes a black-box use of a one-way function.) Using a composition theorem for malicious parties [Can00] we can now prove Corollary 7.28.

**Proof of Corollary 7.28.** We repeat the proof of Theorem 7.27, for the protocol obtained from Protocol 7.38 when the call to \( F_{\text{CF}} \) is replaced with the coin-flipping protocol of [PW09].

**Communication complexity during commit.** Similar to the proof of Theorem 7.27, the communication complexity during commit is \( \text{polylog} n \), and the communication complexity during reveal is \( \tilde{O}(n) \). (The only difference is in the commit phase, in which \( S, R \) run a coin-flipping protocol. However, the protocol uses \( \text{poly}(\text{rand}) \) bits, where \( \text{rand} = \text{polylog}(n, \frac{1}{\varepsilon}) = \text{polylog}(n) \) denotes the number of random bits \( V_{2K} \) uses, so the communication complexity remain polylogarithmic in \( n \).)

**Completeness.** Follows directly from the perfect completeness of the underlying HVZKPCPP system \((P_{2K}, V_{2K})\), the correctness of the coin-flipping protocol, and the correctness of the commitment scheme of Theorem 7.30.

**Binding.** Lemma 7.7.8 guarantees that every efficient sender algorithm \( S_{\text{CF}}^* \) wins the binding game in the coin-flipping hybrid model only with \( \text{negl}(n) \) probability. The same proof can be used to show that the advantage of every expected polynomial-time sender is negligible.
Assume towards negation that there exists an expected polynomial time sender strategy $S^*$ that wins the binding game with non-negligible probability in the real world. Define $\text{Real}_{S^*, R}(1^n)$ to be the random variable describing the execution in the real world with sender $S^*$. Given a sender strategy $S^*_\text{CF}$ in the hybrid model, let $\text{Hybrid}_{S^*_\text{CF}, R}(1^n)$ be the random variable describing the execution in the hybrid model with sender $S^*_\text{CF}$.

The composition theorem of [Can00] guarantees that there exists an expected polynomial-time sender strategy $S^*_\text{CF}$ in the hybrid model, such that the executions $\text{Real}_{S^*, R}(1^n), \text{Hybrid}_{S^*_\text{CF}, R}(1^n)$ are computationally indistinguishable. (Technically, the security definition for which the composition theorem holds is of simulation of polynomial time adversaries in strict polynomial time, while the coin-flipping protocol is secure with expected polynomial time simulators. However, as Protocol 7.38 makes a single oracle call, then the composition theorem can be used.) In particular, $S^*_\text{CF}$ wins the binding game in the hybrid model with non-negligible probability, which contradicts Lemma 7.7.8 (since $\epsilon = \text{negl}(n)$).

**Hiding and zero-knowledge after reveal.** Lemma 7.7.9 and Lemma 7.7.10 guarantee that Protocol 7.38 is statistically hiding and has zero-knowledge after reveal, respectively, in the coin-flipping hybrid model. In particular, these properties hold for expected polynomial time adversaries.

Using the composition theorem of [Can00], the protocol remains computationally hiding, and computationally zero-knowledge after reveal, even when $F_{\text{CF}}$ is replaced with a secure coin-flipping protocol. Indeed, as the sender has no output then the real-world and ideal-world emulations of the protocol with a (possibly malicious) $R^*$ consist only of the view of $R^*$. The real-world and hybrid-world views are computationally indistinguishable (by the composition theorem of [Can00]), and the hybrid-world and ideal-world views are statistically close (by Lemmas 7.7.9 and 7.7.10). We note that as the coin-flipping protocol is secure with expected polynomial-time simulators, the hiding and zero-knowledge after reveal properties of the real-world Commit-and-Prove protocol hold with expected polynomial-time simulators.

We note that similar to Theorem 7.27, if $H$ only satisfies the usual notion of super-polynomial hardness, the communication complexity (during commit) of the protocol of Corollary 7.28 is $O(n^\gamma)$ for an arbitrarily small $\gamma > 0$.

A non-black-box alternative

We have shown how to apply HVZKPCPPs towards obtaining sublinear-communication Commit-and-Prove protocols that make a black-box use of any (super-polynomially hard, or exponentially-hard) collision-resistant hash function. Settling for a non-black-box use of the hash function, one could avoid the use of HVZKPCPPs by combining a sublinear commitment $\text{Com}$ with sublinear zero-knowledge arguments of knowledge [Bar01, BG08]. Concretely, during the commit phase $S$ first commits to $x$ using $\text{Com}$, and then proves to $R$ that he knows a witness $w$, and randomness $r$, such that $(x, r)$ are consistent with
the transcript of \( \text{Com} \), and \((x,w) \in \mathcal{R} \). For the reveal phase, given a subset \( I \), \( S \) sends \( \{x_i\}_{i \in I}, r \) to \( R \). (This is possible if the commitment has the additional property that the sender can reveal only part of the committed value, which is the case, e.g., when using a MHT.) Both of the above primitives can be based on a collision-resistant hash function. However, the commit phase of this protocol is inherently non-black-box because the sublinear argument applies to an NP-relation which depends on the hash function (since \( \text{Com} \) depends on the hash function).

### 7.7.3 Generalized (Multiparty) Commit-and-Prove

In this section we construct reactive one sided functionalities (ROSFs), which are reactive \((m + 1)\)-party functionalities taking their inputs from a single party. ROSFs generalize the notion of Commit-and-Prove to the reactive, and the multiparty, settings.

**Definition 7.7.11 (Reactive one-sided functionality).** An \((m + 1)\)-party reactive one-sided functionality \( F_{f,\text{ROS}} \) runs with a sender \( S \) and \( m \) receivers \( R_1, \ldots, R_m \) (all parties are PPT algorithms), and is parameterized by a function \( f \) taking 2 inputs and returning \( m + 1 \) outputs. \( F_{f,\text{ROS}} \) is executed in phases and maintains an internal state which is updated at the end of each phase (initially, the internal phase \( S_0 \) is empty). Phase \( i \) is executed as follows: upon receiving a message \( x_i \) from \( S \), the functionality computes \( f(x_i, S_{i-1}) = (y_{i,1}, \ldots, y_{i,m}, S_i) \), where \( S_{i-1} \) denotes the internal state after round \( i - 1 \); sends \( S_i \) to \( S \); and sends \( y_{i,j} \) to \( R_j, j \in [m] \). Additionally, the internal state is updated to the value \( S_i \).

A generalized Commit-and-Prove protocol implements a reactive one-sided functionality \( F_{f,\text{ROS}} \), is executed in phases, and has the following properties. First, the protocol is sound, that is, the outputs of the receivers are guaranteed to be consistent with some vector of sender inputs, even if the computationally-bounded sender is malicious. Second, the protocol is private, namely any coalition \( B \) of malicious receivers learns only the outputs of \( f \) restricted to \( B \).

**Definition 7.7.12 (Generalized Commit-and-Prove protocol).** An \( m \)-receiver generalized Commit-and-Prove protocol \( \Pi \) for \( F_{f,\text{ROS}} \) is executed in phases between a PPT sender \( S \) and \( m \) PPT receivers \( R_1, \ldots, R_m \). At the end of every phase \( i \), each receiver \( R_j \) outputs a value \( y_{i,j} \). The protocol has the following properties:

- **Binding.** For every efficient non-uniform sender algorithm \( S^* \) running in \( \Pi \) there exists an ideal (not necessarily efficient) sender strategy \( S \) participating in the execution of \( F_{f,\text{ROS}} \) such that the following holds for every phase \( i \) of \( \Pi \). Let \( \text{Real}_{S^*,i,\Pi,i} \) denote the random variable describing the outputs of \( R_1, \ldots, R_m \) in the first \( i \) phases of the execution of \( \Pi \) with \( S^* \), and let \( \text{Ideal}_{f,i} \) denote the random variable describing the outputs of \( R_1, \ldots, R_m \) in the first \( i \) phases of the execution of \( F_{f,\text{ROS}} \) with sender \( S \), where in any phase \( S \) can send an abort message to \( F_{f,\text{ROS}} \).
which causes the execution to terminate. Then for every phase $i$, $\text{Real}_{S^*,\Pi,i}$ and $\text{Ideal}_{f,i}$ are computationally close, denoted $\text{Real}_{S^*,\Pi,i} \approx \text{Ideal}_{f,i}$.

- **Privacy.** For every $B \subseteq [m]$, and every collection $\{R_j^i\}_{j \in B}$ of efficient (possibly non-uniform) receiver algorithms, there exists a simulator $\text{Sim}$ interacting in phases with an external “environment” $Z$, where in phase $i$, $Z$ gives $\{y_j\}_{j \in B}$ to $\text{Sim}$ as input, and $\text{Sim}$ outputs a view $\{V_{i,j}(y_j)\}_{j \in B}$ such that the following holds for every phase $i$, and every sender inputs $(x_1, \ldots, x_i)$. Let $V_B(x_1, \ldots, x_i)$ denote the view of $\{R_j^i\}_{j \in B}$ after the first $i$ phases of $\Pi$ when $S$ has inputs $(x_1, \ldots, x_i)$, then $V_{B}(f_j(x_i,f_{m+1}(x_{i-1})))_{j \in B} \approx V_{B}(x_1, \ldots, x_i)$, where $f_j$ denotes the $j$'th output of $f$.

**ZKPCPP-based Generalized Commit-and-Prove Protocols**

We use ZKPCPPs to construct generalized Commit-and-Prove protocols. More specifically, we prove the following:

**Theorem 7.40.** Let $\mathcal{H}$ be a family of exponentially-hard collision resistant hash functions, and $\mathcal{F}_{\text{ROS}}$ be an $(m+1)$-party reactive one-sided functionality. Then there exists an $m$-receiver generalized Commit-and-Prove protocol for $\mathcal{F}_{\text{ROS}}$, which makes only black-box use of $\mathcal{H}$, and every function evaluation phase uses only $m \cdot (\text{polylog}(nm) + \tilde{O}(k))$ communication bits, where $n, k$ denote the input and output lengths, respectively.

**Warm-up: generalized Commit-and-Prove with a single receiver.** The single-receiver case resembles the setting of 2-party Commit-and-Prove protocols (as in Section 7.7). However, these protocols are “one-shot”, and in particular reveal the sender input to the receiver. Therefore, the high-level idea is to use Protocol 7.33 with a modified relation. Given a relation $\mathcal{R} = \mathcal{R}(x, w)$, let

$$\mathcal{R}^f = \{(y,(x,w)) : x \in \{0,1\}^n \land y = f(x,w) \land (x,w) \in \mathcal{R}\}.$$

When Protocol 7.33 is executed with relation $\mathcal{R}^f$, then it follows from the soundness property that at the end of the commit phase, $R$ holds a commitment to $f(x^*,w^*)$ for some pair $(x^*,w^*) \in \mathcal{R}$. However, reactive one-sided functionalities also guarantee computation continuity in the following sense: in phase $i$, the function was evaluated on the internal state generated by phase $i-1$. Computation continuity is not guaranteed when every round of $\mathcal{F}_{\text{ROS}}^f$ is implemented using Protocol 7.33 with relation $\mathcal{R}^f$. To overcome this, we replace $\mathcal{R}^f$ with a relation that is connected to the randomized functionality $F$ that returns, in addition to the receiver output (of $f$), also a short commitment to the current internal state. Formally:

**Definition 7.7.13.** For a string $y$, let $\text{MHT}(y)$ denote the set of Merkle Hash Tree commitments to $y$ (where these commitments are generated as in steps 3-4 of the commit phase of Protocol 7.33), and let $\text{MHT}_{r}(y)$ denote the Merkle Hash Tree commitment
to \( y \), generated using randomness \( r \). Define \( F(x_i, S_{i-1}) = (f_1(x_i, S_{i-1}), c_i) \), where \( c_i \in R \text{MHT}(f_2(x_i, S_{i-1})) \). Given a random string \( r \) for \( F \), let \( F_r(\cdot, \cdot) \) denote the output of \( F \) when it uses randomness \( r \). Let \( R_{\text{ROS}} \) be the relation defined as follows

\[
\{(y_i, c_i, c_{i-1}), (x_i, S_{i-1}, r_i, r_{i-1}) \mid (y_i, c_i) = F_{r_i}(x_i, S_{i-1}) \wedge c_{i-1} = \text{MHT}_{r_{i-1}}(S_{i-1})\}.
\]

\( R_{\text{ROS}} \) guarantees computation continuity because it enforces that (with high probability) in each phase \( i \), the second input \( S_{i-1} \) of \( F \) is the second output of phase \( i - 1 \). Thus, in every phase, \( S \) commits to the output of \( f \), and to the new internal state \( S_i \) (i.e., the state at the end of phase \( i \)). Moreover, \( S \) “proves” to \( R \) that \( f \) was applied to the “correct” internal state \( S_{i-1} \).

**Generalized Commit-and-Prove with multiple receivers.** We first describe the issues arising when the network includes several receivers. Intuitively, every function evaluation phase between \( S, R_j \) could use Protocol 7.33, run with the relation \( R_{\text{ROS}}^{j} \) consisting of all pairs of the form

\[
((f_j(x_i, S_{i-1}), c_i, c_{i-1}), (x_i, S_{i-1}, r_i, r_{i-1}))
\]

where \( c_i = \text{MHT}_{r_i}(f_{m+1}(x_i, S_{i-1})) \), and \( c_{i-1} = \text{MHT}_{r_{i-1}}(S_{i-1}) \). However, such executions do not guarantee **consistency between different receivers**. Indeed, the soundness of Protocol 7.33 only guarantees that for every phase-\( i \) output \( y_{i,j} \) of \( R_j \), there exists an input pair \( (x_{i,j}, S_{i-1,j}) \) such that \( y_{i,j} = f_j(x_{i,j}, S_{i-1,j}) \), but does not guarantee the existence of a **single** input that simultaneously satisfies all these equations. Therefore, \( S \) should “prove” he used the same input in all executions. Roughly speaking, this is done by incorporating commitments to all outputs into every relation \( R_{\text{ROS}}^{j} \), where \( S \) broadcasts these commitments during the reveal phase. Thus, every receiver can verify that the commitments are consistent with his phase-\( i \) output.

Concretely, we use the following set of relations:

**Definition 7.7.14.** For a function \( f \) taking \( 2 \) inputs and returning \( m + 1 \) outputs, let \( f_j(\cdot, \cdot), 1 \leq j \leq m \) denote the \( j \)’th output of \( f \). For every \( j \in [m] \), \( R_{\text{ROS}}^{j} \) consists of all pairs of the form

\[
((y_{i,j}, z_{i,1}, \ldots, z_{i,m+1}, c_{i-1}), (x_i, S_{i-1}, r_{i-1}, r_{i,1}, \ldots, r_{i,m+1}))
\]

such that \( y_{i,j} = f_j(x_i, S_{i-1}), z_{i,k} = \text{MHT}_{r_{i,k}}(f_k(x_i, S_{i-1})) \) for every \( 1 \leq k \leq m + 1 \), and \( c_{i-1} = \text{MHT}_{r_{i-1}}(S_{i-1}) \).

We now use the relations of Definition 7.7.14 to construct a generalized Commit-and-Prove protocol for \( m \geq 1 \) receivers.

**Protocol 7.41.** The protocol is executed between a sender \( S \) with input \((x_1, \ldots, x_n)\), and \( m \) receivers \( R_1, \ldots, R_m \). Let \( \Pi_1, \ldots, \Pi_m \) denote Protocol 7.33, when run with relations
\( R_{\text{ROS}}^1, \ldots, R_{\text{ROS}}^m \), respectively. The protocol starts with an initialization phase, followed by function evaluation phases, as described next.

- **Initialization.** Initially, \( S_0 = \bot \) (the initial state). \( S \) generates a Merkle Hash Tree commitment \( c^0 \) to \( S^0 \) using randomness \( r \), and broadcasts \( c_0, r \). Every \( R_j \) verifies that \( c^0 \) is the commitment to \( S^0 \), obtained with randomness \( r \), and otherwise aborts with output \( \bot \).

- **Function evaluation.** Phase \( i \) is executed as follows. Let \( S_{i-1} \) denote the state at the end of phase \( i - 1 \), and \( c_{i-1} = \text{MHT}_{r_{i-1}}(S_{i-1}) \) denote the commitment which the receivers hold (obtained during phase \( i - 1 \)).\(^{10}\) Let \( f(x_i, S_{i-1}) = (y_{i,1}, \ldots, y_{i,m}, S_i) \). For every \( 1 \leq j \leq m \) let \( z_{i,j} = \text{MHT}_{r_{i,j}}(y_{i,j}) \) for a uniformly random \( r_{i,j} \), and \( z_{i,m+1} = \text{MHT}_{r_{i,m+1}}(S_i) \) for a uniformly random \( r_{i,m+1} \).
  
  - For every \( 1 \leq j \leq m \), \( S \) and \( R_j \) run the commit phase of \( \Pi_j \), where \( S \) uses input \( ((y_{i,j}, z_{i,1}, \ldots, z_{i,m+1}, c_{i-1}), (x_i, S_{i-1}, r_{i-1}, r_{i,1}, \ldots, r_{i,m+1})) \). If the commitment phase fails then \( R_j \) broadcasts a complaint, outputs \( \bot \), and halts.
  
  - If no \( R_k \) broadcasted a complaint in previous rounds, then every \( R_j, 1 \leq j \leq m \) executes the reveal phase of \( \Pi_j \) with \( S \), and let \( (y_{i,j}, z_{i,1}^j, \ldots, z_{i,m+1}^j, c_{i-1}^j) \) denote the value revealed during the phase. In addition, \( S \) broadcasts the values \( z_{i,1}, \ldots, z_{i,m+1} \). For every \( 1 \leq j \leq m \), if \( c_{i-1}^j \neq c_{i-1} \), \( c_{i-1} \neq z_{i-1,m+1}^j \), or \( z_{i,k}^j \neq z_{i,k} \) for some \( 1 \leq k \leq m + 1 \) then \( R_j \) broadcasts a complaint and aborts with outputs \( \bot \). (In the first phase \( i = 1 \), instead of checking that \( c_{i-1} \neq z_{i-1,m+1} \), \( R_j \) checks that the value \( c_0 \) broadcasted in the first phase is consistent with the value broadcasted during initialization.) If at least one receiver broadcasted a complaint during the phase, then \( R_j \) aborts with output \( \bot \). Otherwise, \( R_j \) output \( y_{i,j} \), and continues to the next phase.

The next sequence of lemmas analyzes the binding and privacy properties of Protocol 7.41.

**Lemma 7.7.15.** Let \( n \in \mathbb{N} \) be an input length parameter. For \( i \in \mathbb{N} \), let \( y_{1,1}, \ldots, y_{1,m}, \ldots, y_{i,1}, \ldots, y_{i,m} \) denote the outputs of (the honest) \( R_1, \ldots, R_m \) in an \( i \)-phase execution of Protocol 7.41. We say that \( x_k \) is consistent with the phase-\( k \) receiver outputs \( y_{k,1}, \ldots, y_{k,m} \) if \( f_j(x_k, S_{k-1}) = y_{k,j} \) for every \( 1 \leq j \leq m \), where \( S_{k-1} \) denotes the internal state at the end of phase \( k - 1 \), and \( S_0 = \bot \). Then for every \( i = \text{poly}(n) \), and for any sequence \( x_1, \ldots, x_{i-1} \) of inputs that are consistent with \( y_{1,1}, \ldots, y_{m,1}, \ldots, y_{i-1,1}, \ldots, y_{i-1,m} \) (and \( S_0, S_1, \ldots, S_{i-1} \)), except with negligible probability there exists an \( x_i \) such that \( f_j(x_i, S_{i-1}) = y_{i,j} \) for every \( 1 \leq j \leq m \).

\(^{10}\)We implicitly assume all receivers hold the same commitment. This will be enforced by \( S \) broadcasting the commitment in the last round of each phase. We refer to \( S_{i-1} \) as the “state” since this is the case in an honest execution of \( F_{\text{ROS}}^i \).
Proof. Notice that it suffices to consider executions in which no receiver aborted. Otherwise, all receivers abort in the same phase, and we can take $x_i$ to be an abort message (and all outputs from that phase onwards will be identically distributed in both the real and the ideal executions).

We prove the claim by induction on $i$. For the base case $i = 1$, notice that by the initialization phase of Protocol 7.41, $R_1, \ldots, R_m$ all hold commitments $c_{i-1}$ to $S_0 = \perp$. For every $1 \leq j \leq m$, the binding of $\Pi_j$ guarantees that (since $R_j$ did not abort in the phase) except with negligible probability there exists a single set $x_{1,i}, S_{0,i}, r_{0,i}, r_{1,1,i}, \ldots, r_{1,m+1,i}$ such that $y_{1,j} = f_j(x_{1,i}, S_{0,i})$: for every $k \in [m]$, $z_{1,k}^i = \text{MHT}_{r_{1,k}}(f_k(x_{1,i}, S_{0,i})); c_0^i = \text{MHT}_{r_0^i}(f_{m+1}(x_{1,i}, S_{1,i}^i)) = \text{MHT}_{r_0^i}(S_{1,i}^i)$; and $S$ can successfully decommit to $x_{1,i}, S_{0,i}, r_{0,i}, r_{1,1,i}, \ldots, r_{1,m+1,i}$.

Using the union bound, it suffices to prove the basis conditioned on the event that such inputs exist for every $j \in [m]$. Since $R_j$ compares $c_0^i$ to $c_0$, we know that with negligible probability (namely, the probability that the computationally-bounded $S$ was able to break the binding of the underlying commitment scheme, or the collision-resistance of the underlying CRHF), $S_0 = S_0^i$ for every $1 \leq j \leq m$.

Using the union bound again, we can further condition on the event that $S_0 = S_0^i$ for every $1 \leq j \leq m$. We set $x_{1,i} := x_{1,i}^i$; and for every $2 \leq k \leq m$, let $y_{1,k}^i = f_k(x_{1,i}, S_{0,i})$, then $z_{1,k}^i = \text{MHT}_{r_{1,k}}(f_k(x_{1,i}, S_{0,i})) = \text{MHT}_{r_{1,k}}(y_{1,k}^i)$. Moreover, since every $R_j$ compares the commitments $z_{1,1,i}, \ldots, z_{1,m,i}$ to the broadcasted values $z_{1,1}, \ldots, z_{1,m}$, we know that for every $2 \leq k \leq m$, $z_{1,k}^i = z_{1,k}^i$. Remember that by our conditioning, $z_{1,k}^i = \text{MHT}_{r_{1,k}}(f_k(x_{1,i}, S_{0,i})) = \text{MHT}_{r_{1,k}}(y_{1,k}).$ Then by the computational binding of the underlying commitment scheme, and the collision-resistance of the CRHF, except with negligible probability, $y_{1,k} = y_{1,k}^i = f_k(x_{1,i}, S_{0,i})$ for every $1 \leq k \leq m$.

For the step, let $y_{1,1}, \ldots, y_{1,m}, y_{2,1}, \ldots, y_{2,m}$ denote the outputs of $R_1, \ldots, R_m$ in an $i$-phase execution of Protocol 7.41, and let $x_{1,1}, \ldots, x_{i-1}$ be inputs that are consistent with $y_{1,1}, \ldots, y_{1,m}, y_{2,1}, \ldots, y_{2,m}$ (whose existence follows from the induction hypothesis). That is, there exist $S_0 = \perp, S_{1,1}, \ldots, S_{1,i-1}$ such that $y_{i,j} = f_j(x_t, S_{t-1})$ for every $1 \leq t \leq i - 1, 1 \leq j \leq m$, and $S_t = f_{m+1}(x_t, S_{t-1})$ for every $1 \leq t \leq i - 1$.

For every $1 \leq j \leq m$, as $R_j$, $j \in [m]$ did not abort in phase $i$ then the binding of $\Pi_j$ guarantees that except with negligible probability, there exist $x_{1,i}, S_{1,i-1}$ such that $y_{i,j} = f_j(x_{1,i}, S_{1,i-1})$, and (since $R_j$ compares $z_{1,k}^i, z_{1,k}^i, k \in [m + 1]$) the values $z_{1,1}, \ldots, z_{1,m+1}$ that $S^*$ broadcasted at the end of phase $i$ are MHT commitments to $f_1(x_{1,i}, S_{1,i-1}), \ldots, f_{m+1}(x_{1,i}, S_{1,i-1})$, respectively, and $c_{i-1}^i$ is a MHT commitment to $S_{1,i-1}$ (namely, to the internal state at the end of phase $i$, as it was used in the execution of $\Pi_1$). Therefore, we can prove the claim conditioned on the existence of such inputs. Moreover, since $R_1$ did not abort in phase $i$ then the binding of $\Pi_1$ (together with the fact that $R_1$ verifies that $c_{i-1}^i = c_{i-1} = z_{1,i-1,m+1}^i$, which, since $R_1$ did not abort in phase $i = 1$, is equal to $z_{1,i-1,m+1}$, which is a commitment to the internal state at the end of phase $i - 1$) guarantees that except with negligible probability, the commitment $c_{i-1}$
that $S^*$ broadcasted in phase $i - 1$, is a MHT commitment to $S_{i-1}$, so we can further condition on this event. In this case, except with negligible probability (which is the probability that $S^*$ found a collision in the MHT), $S_{i-1} = S^1_{i-1}$.

We take $x_i := x_{i,1}$, and claim that except with negligible probability, $y_{i,j} = f_j (x_i, S_{i-1})$ for every $1 \leq j \leq m$. By the union bound, it suffices to show that for every $1 \leq j \leq m$, $y_{i,j} = f_j (x_i, S_{i-1})$ except with negligible probability. Fix an index $j$. As discussed above, except with negligible probability, $z_{i,k}$ is a MHT commitment to $f_k (x_i, S_{i-1})$ for every $1 \leq k \leq m$. Moreover, the binding of $\Pi_j$ guarantees that except with negligible probability, $z_{i,j}$ is a MHT commitment to the value $y_{i,j}$ that $R_j$ outputs. Therefore, $y_{i,j} = f_j (x_i, S_{i-1})$ except with negligible probability (i.e., the probability that $S^*$ breaks the computational binding of the underlying commitment scheme, or the collision-resistance of the CRHF).

$$\square$$

The next lemma states that the joint real-world views of (any) subset of receivers in a single phase of Protocol 7.41 are statistically close to the simulated views. Intuitively, this follows from the “zero-knowledge after reveal guarantee” of Protocol 7.33, since an online simulator can use the simulator $Sim_{in}$ of Protocol 7.33 to independently generate the receiver views in every phase.

More specifically, the zero-knowledge after reveal of Protocol 7.33 guarantees that the real-world and simulated views of the receiver of Protocol 7.33 remain statistically close, even after the reveal phase of the protocol. In particular, when the underlying relation is $R^{jt}_{ROS}, j \in [m]$, then whatever can be deduced from the additional information that $R_j$ learns in the second step of the function evaluation phase (i.e., from the broadcasted values), follows from the phase-$i$ output or $R_j$. (Indeed, an honest $S$ broadcasts the values that would have been revealed to $R_j$ during the reveal phase of $\Pi_j$.) Moreover, since $R^{jt}_{ROS}$ is defined over pairs in which the internal state can take an arbitrary value, indistinguishability holds even if the computation begins at an arbitrary internal state of $F^{jt}_{ROS}$, which captures the execution of a single arbitrary phase $i$ (not necessarily the first) of Protocol 7.41.

Lemma 7.7.16. Let $B \subseteq [m]$, and let $\mathcal{B} := \{R^*_j : j \in B\}$ be a collection of (possibly malicious) receivers in Protocol 7.41. Then there exists a simulator $Sim_{\text{single}}$, whose running time is polynomial in the joint running time of the receivers in $\mathcal{B}$, such that for every sender input $x$, and every internal state $S$ of $F^{jt}_{ROS}$, $Sim_{\text{single}} ([x], \{f_j (x, S)\}_{j \in \mathcal{B}})$ is statistically close to the joint view $(V_j)_{j \in \mathcal{B}}$ of $\{R^*_j : j \in \mathcal{B}\}$ at the end of a single function evaluation phase, in which the sender $S$ uses $(x, S)$ as the input to $f$.

Proof. Denote $B := \{j_1, \ldots, j_k\}$, and $Sim_{in}$ denote the simulator for Protocol 7.33, whose existence follows from the zero-knowledge after reveal property. For every receiver algorithm $R^*_j, j \in B$ let $R_{in,j}$ denote the receiver algorithm in protocol $\Pi_j$, induced by the operations $R^*_j$ performs during the execution of $\Pi_j$. Denote the view of $R^*_j (R_{in,j})$, when $S$ uses $(x, S)$ as input to $f$, by $V_j (x, S) (V_{in,j} (x, S))$. For every $1 \leq j \leq m$, let
Let $y_j := f_j(x, S)$ be the output of $R_j$ at the end of the phase. Let $\Sim_{in}(|x|, y_j)$ denote the simulated view of $R_{in,j}$ (generated using random, independent, instantiations of $\Sim_{in}$). Let $\Real_B(x, S) := (V_j(x, S))_{j \in B}$, and $\Ideal_B(|x|, \{y_j\}_{j \in B}) := (\Sim_{in}(|x|, y_j))_{j \in B}$. We show that for every $(x, S)$, $\Real_B(x, S)$ and $\Ideal_B(|x|, y_1, \ldots, y_{j,k})$ are statistically close.

For every $(x, S)$, we define a sequence of hybrids $H_0(x, S), \ldots, H_k(x, S)$, where for every $0 \leq i \leq k$,

$$H_i(x, S) = (V_{j_1}(x, S), \ldots, V_{j_i}(x, S), \Sim_{in}(|x|, y_{j_{i+1}}), \ldots, \Sim_{in}(|x|, y_{j_k})).$$

Notice that $H_0(x, S) = \Ideal_B(|x|, y_1, \ldots, y_{j_k})$, and $H_k(x, S) = \Real_B(x, S)$.

If the claim does not hold, then there exist a distinguisher $D$, a polynomial $p(n)$, an infinite sequence of $n$'s, and for every $n$ in the sequence an input pair $(x_n, S_n)$ such that $D$ achieves advantage at least $\frac{1}{np(n)}$ in distinguishing $H_k(x_n, S_n)$ from $H_0(x_n, S_n)$. This implies that for some $0 \leq i \leq k-1$, $D$ achieves advantage at least $\frac{1}{np(n)}$ in distinguishing $H_{i+1}(x_n, S_n)$ from $H_i(x_n, S_n)$, for an infinite sequence of input pairs $(x_n, S_n)$. We use $D$ to achieve the same advantage in distinguishing $V_{j_{i+1}}(x_n, S_n)$ from $\Sim_{in}(|x|, y_{j_{i+1}})$, thus contradicting the zero-knowledge after reveal of Protocol 7.33 (and the choice of $\Sim_{in}$).

Notice that there exist $V_{j_1}(x_n, S_n), \ldots, V_{j_i}(x_n, S_n)$, and $\Sim_{in}(|x_n|, y_{j_{i+2}}, \ldots, y_{j_k})$ such that $D$ achieves advantage at least $\frac{1}{np(n)}$ in distinguishing between $H_{i+1}(x_n, S_n), H_i(x_n, S_n)$, even when the hybrids are conditioned on $V_{j_1}(x_n, S_n), \ldots, V_{j_i}(x_n, S_n), \Sim_{in}(|x_n|, y_{j_{i+2}}, \ldots, y_{j_k})$. For every such $n$, we incorporate $x_n, S_n$ and $V_{j_1}(x_n, S_n), \ldots, V_{j_i}(x_n, S_n), \Sim_{in}(|x_n|, y_{j_{i+2}}, \ldots, y_{j_k})$ into our distinguisher $D'$. On input $y$, which is distributed according to either $V_{j_{i+1}}(x_n, S_n)$ or $\Sim_{in}(|x_n|, y_{j_{i+1}})$, $D'$ concatenates its input with $V_{j_1}(x_n, S_n), \ldots, V_{j_i}(x_n, S_n), \Sim_{in}(|x_n|, y_{j_{i+2}}, \ldots, y_{j_k})$, runs $D$, and answers as $D$ does. Then the advantage of $D'$ is at least $\frac{1}{np(n)}$. 

We can now prove Theorem 7.40.

**Proof of Theorem 7.40.** We show that Protocol 7.41 has the required properties.

**Communication complexity.** Using Theorem 7.25, the communication complexity during every function evaluation round is $m \cdot \left(\polylog (mn) + \tilde{O}(k)\right)$, where $n$ denotes the input length, and $k$ is the maximal length of the output (i.e., $k = \max \{|S_i|, \max_{j \in [m]} |\{y_j\}|\}$). Indeed, the broadcasted commitments each have length $\polylog (k) = \polylog (\poly (n)) = \polylog (n)$, and there are $O(m)$ commitments in total. Moreover, for every $1 \leq j \leq m$, the execution of $\Pi_j$ requires $\polylog (mn) + \tilde{O}(k)$ communication ($\polylog (mn)$ communication bits for the commit phase since it is run on an input of length $m \cdot (\polylog (k) + k)$ where $k = \poly (n)$; $m \cdot \polylog (k)$ broadcasted bits; and at most $\tilde{O}(k)$ bits for the reveal phase, by Remark 7.26).

**Black-box use of $\mathcal{H}$.** The protocol makes a black-box use of $\mathcal{H}$ since $\mathcal{H}$ is only used
in the underlying executions of Protocol 7.33, which makes black-box use of $H$.

**Binding.** Let $y_{1,1}, \ldots, y_{1,m}, \ldots, y_{i,1}, \ldots, y_{i,m}$ denote the sequence of receiver outputs in an $i$-phase execution of protocol 7.41 with the (possibly malicious) sender $S^*$. The by Lemma 7.7.15, for every $1 \leq j < i$, and for every sequence $x_1, \ldots, x_j$ of inputs that are consistent with $y_{1,1}, \ldots, y_{1,m}, \ldots, y_{j,1}, \ldots, y_{j,m}$, expect with negligible probability there exists an $x_{j+1}$ such that $x_1, \ldots, x_{j+1}$ are consistent with $y_{1,1}, \ldots, y_{1,m}, \ldots, y_{j+1,1}, \ldots, y_{j+1,m}$.

Therefore, in every phase $S$ fails to find an input for the phase (that is consistent with the receiver outputs so far and the inputs $S$ chose in the previous phases) only with negligible probability. As there are at most $\text{poly}(n)$ phases, $S$ can simulate all phases, except with negligible probability.

**Privacy.** Let $q = q(n)$ be an upper-bound on the running time of $S$. In particular, Protocol 7.41 has at most $q$ phases. Let $B \subseteq [m]$, and for every $j \in B$ and every phase $i$, let $V_j(x_1, \ldots, x_i)$ denote the view of $R^*_j$ in the first $i$ phases of the execution of Protocol 7.41 in which $S$ has inputs $x_1, \ldots, x_i$. Let $y_{1,j}, \ldots, y_{i,j}$ denote the outputs of $R^*_j$ in this execution. (That is, for every $1 \leq l \leq i$, $y_{l,j} = f_j(x_l, S_{l-1})$, where $S_0 = \bot$, and $S_l = f_{m+1} (x_l, S_{l-1})$ for every $1 \leq l \leq i$.) Let $\text{Sim}$ be the simulator that in phase $i$, given input $\langle x_i, \{y_{i,j}\}_{j \in B}\rangle$, runs $\text{Sim}_{\text{single}}$ (the simulator of Lemma 7.7.16) on his input, and outputs whatever $\text{Sim}_{\text{single}}$ outputs.

We claim that for every sequence $x_1, \ldots, x_t, t \leq q$ of inputs, $(V_j(x_1, \ldots, x_t))_{j \in B}$ is statistically close to $(\text{Sim}(y_{1,j}))_{j \in B}, \ldots, \text{Sim}(y_{t,j}))_{j \in B})$. Otherwise, there exist a distinguisher $D$, a polynomial $p(n)$, an infinite sequence of $n$'s, and for every $n$ a sequence of inputs $x^n_1, \ldots, x^n_t$, such that $D$ achieves advantage at least $\frac{1}{p(n)}$ in distinguishing $(V_j(x^n_1, \ldots, x^n_t))_{j \in B}$ from $(\text{Sim}(y^n_{1,j}))_{j \in B}, \ldots, \text{Sim}(y^n_{t,j}))_{j \in B})$. We define a sequence $H^n_1, \ldots, H^n_t$ of hybrids, where the views in phases $1, 2, \ldots, i$ in $H^n_i$ are the real-world views of $R^*_j, j \in B$, and the views in phases $i + 1, \ldots, t$ are the simulated views. Notice that $H^n_i = (V_j(x^n_1, \ldots, x^n_t))_{j \in B}$ and $H^n_i = (\text{Sim}(y^n_{1,j}))_{j \in B}, \ldots, \text{Sim}(y^n_{t,j}))_{j \in B})$. As $D$ can distinguish $H^n_1$ from $H^n_t$ with advantage at least $\frac{1}{p(n)}$, there exists an $0 \leq i_n < t$ such that $D$ can distinguish $H^n_{i_n}$ from $H^n_{i_n+1}$ with advantage at least $\frac{1}{p(n)}$.

This contradicts Lemma 7.7.16, since it implies the existence of a distinguisher $D'$ achieving advantage at least $\frac{1}{p(n)}$ in distinguishing between the real-world and simulated views of $(R^*_j)_{j \in B}$ in a single phase, on an infinite sequence of input pairs $(x^n_{i_n+1}, S^n_{i_n})$ ($S^n_{i_n}$ is the internal state after phase $i_n$, in an execution on inputs $x^n_1, \ldots, x^n_t$). Indeed, using an averaging argument, for every $n$ there exists a fixed sequence $\alpha_n$ of views of $R^*_j, j \in B$ up to phase $i_n$ (including), and a fixed sequence $\beta_n$ of simulated views from phase $i_n + 2$, such that $D$ achieves advantage at least $\frac{1}{p(n)}$, even when $H^n_{i_n}, H^n_{i_n+1}$ are conditioned on $\alpha_n, \beta_n$. We incorporate $i_n, \alpha_n, \beta_n$ into $D'$, and on input $y$, distributed according to either the real-world of simulated views of $R^*_j, j \in B$ in phase
$i_n + 1$, $D'$ concatenates its input with $\alpha_n$, $\beta_n$, runs $D$, and outputs whatever $D$ outputs. Then the advantage of $D'$ is at least $\frac{1}{n \cdot p(n)}$.

7.7.4 Updatable Databases

In this section we describe an application of reactive one-sided functionalities for updatable database schemes. In a nutshell, an updatable database scheme consists of a server $S$, and $m$ clients $C_1, \ldots, C_m$. The server stores a database $D$, on which the clients can perform read and write operations. Every client $C_j$ is associated with sets $\text{read}_j, \text{write}_j$ defining the database locations she can read from and write to, respectively.

We say a read (write) operation of $C_j$ is valid, if $C_j$ reads from (writes to) a location in $\text{read}_j$ ($\text{write}_j$). The scheme is executed in phases, where in every phase a single client performs a single (read or write) operation. The scheme is $\tau$-private if for every coalition $B$ of at most $\tau$ (possibly malicious) efficient client algorithms, the only information $B$ learns during the execution are the outcomes of valid read operations of the clients in $B$. The scheme is secure, if for every (possibly malicious) efficient server $S^*$, if $S^*$ tries to cheat (i.e., return an incorrect value following a read operation, or update the database to an incorrect value following a write operation) then $S^*$ is caught except with negligible probability.

Updatable databases can be represented as a private case of reactive one-sided functionalities, captured by the following function $f_{r-w}$.

**Definition 7.7.17.** $f_{r-w}$ takes 2 inputs, returns $m + 1$ outputs, and is parameterized by $2m$ sets $\text{read}_1, \text{write}_1, \ldots, \text{read}_m, \text{write}_m$. On input $((\text{op}, j, l, v, i), S)$, $f_{r-w}$ returns the outputs $(y_1, \ldots, y_{m+1})$ defined as follows. (Intuitively, this input represents that client $C_j$ requested the operation $\text{op}$ to be performed in phase $i$ on location $l$ with value $v$, where the current internal state of the server is $S$.)

- **If** $i = 1$, and $S = \perp$, then $y_k = \text{init}$ for every $1 \leq k \leq m$, and $y_{m+1} = (v, 1)$ (init is an initialization message). Otherwise $y_k = \perp$ for every $1 \leq k \leq m$, and $y_{m+1} = S$.\(^{12}\)

- **If** $i > 1$, then the following is executed, where if $S = \perp$ then $S$ is taken to be the all-zeros string.\(^{13}\)

  - $\text{op} = \text{read}$: if $l \in \text{read}_j$ then $y_j = (S_l, \text{op}, j, l)$, $y_{m+1} = S$, and $y_k = \perp$ for every $j \neq k \in [m]$. Otherwise, $y_k = \perp$ for $1 \leq k \leq m$, and $y_{m+1} = S$.

\(^{11}\)We assume all inputs to the function have this format. This is without loss of generality, since inputs with invalid formats can be ignored.

\(^{12}\)The restriction that $S$ is empty will guarantee that when an updatable database scheme is based on the reactive one-sided functionality corresponding to $f_{r-w}$, then the database initialization is executed only once - in the first phase of the protocol.

\(^{13}\)We do not execute operations on non-initialized databases. If the database has not been initialized, it is set to some arbitrary value (in this case: the all-zeros string).
\[ \text{\texttt{op}} = \text{write}: \text{if } l \in \text{write}_j \text{ then } y_k = \bot \text{ for every } j \neq k \in [m], \ y_j = (\text{op}, j, l, v), \] and \( y_{m+1} = S' \), where \((S')_t = v\), and \( S'_t = S_t \) for every \( t \neq l \). Otherwise, \( y_k = \bot, 1 \leq k \leq m \), and \( y_{m+1} = S \).

Intuitively, \( f_{r-w} \) captures read and write operations since its input \(((\text{op}, j, l, v), S_t)\) denotes that in phase \( i \), client \( C_j \) performs operation \( \text{op} \) on location \( l \) of the database, whose value at the beginning of the phase is \( S \). (Moreover, if \( \text{op} \) is a write, then \( v \) is the value to be written.) More specifically, when \( i = 1 \) the server \( S \) can set the “internal state” of \( F_{ROS}^{f_{r-w}} \) to any value (namely, the functionality will store any initial database value). Valid writes to the database update the internal state of the functionality (i.e., the database), and valid read operations do not change the database, and return an output only to the client requesting the read operation.

Next, we use generalized Commit-and-Prove protocols to design an updatable database scheme. Specifically, our scheme relies on Protocol 7.41 for the functionality \( F_{ROS}^{f_{r-w}} \). Intuitively, the security of Protocol 7.41 guarantees that the requested database operations are indeed performed. However, since \( S \) is the only party passing an input to the functionality, clients need to send all read or write requests to \( S \), which then passes them on to the functionality. Consequently, a malicious server \( S \) can perform any set of operations it chooses, regardless of the client requests. Therefore, we have every client also verify that the operation that was actually performed on the database is indeed the requested operation.

\textit{Protocol 7.42.} The scheme is run between a server \( S \), and \( m \) clients \( C_1, \ldots, C_m \), where \( S \) has as input a database \( \text{Db} \), and every client \( C_j, j \in [m] \) is associate with the sets \( \text{read}_j, \text{write}_j \) (indicating the locations from which \( C_j \) can read, and to which she can write, respectively). The scheme uses as a building block Protocol 7.41 for \( F_{ROS}^{f_{r-w}} \), denoted \( \Pi_{in} \). The \textit{updatable database scheme} \( \Pi_{UD} \) consists of an initialization phase, followed by computation phases, each consisting of a single read or write.

- \textbf{Initialization.} The parties execute the initialization phase of \( \Pi_{in} \), and then run the first function evaluation phase, where \( S \) uses input \((\bot, \bot, \bot, \text{Db}, 1)\).

- \textbf{Read-Write.} Following the initialization phase, the parties perform read and write operations by repeatedly executing the function evaluation phase of \( \Pi_{in} \), where phase \( i \) is executed as follows.

  - \texttt{read} \((j, l)\):
    * \( C_j \) sends \( l \) to \( S \).
    * \( S, C_1, \ldots, C_m \) run a function evaluation phase of \( \Pi_{in} \), where \( S \) uses input \((\text{read}, j, l, \bot, i)\). Let \((v, c)\) denote the output of \( C_j \) in that phase of \( \Pi_{in} \).
    * If \( l \in \text{read}_j \), and \( c \neq (\text{read}, j, l) \), then \( C_k \) broadcasts a complaint, and all clients abort. Otherwise, \( C_j \) outputs \( v \), and every \( C_k, k \neq i \) outputs \( \bot \).
- write \((j, l, v)\):
  * \(C_j\) sends \(l, v\) to \(S\).
  * \(S, C_1, \ldots, C_m\) run a function evaluation phase of \(\Pi_{in}\) in which \(S\) uses input \((\text{write}, j, l, v, i)\). Let \(c\) denote the output of \(C_j\) in the current phase of \(\Pi_{in}\).
  * If \(l \in \text{write}_j\), and \(c \neq (\text{write}, j, l, v)\) then \(C_j\) broadcasts a complaint, and all clients abort. Otherwise, all clients output \(\bot\), and the parties continue to the next phase.
Chapter 8

Conclusions

In this thesis we have proposed several “zero-knowledge” variants of probabilistic verification and testing techniques, which extended the existing concept of ZKPCPs. We have shown connections between probabilistic proofs and codes to secure multiparty computation, using LRCCs and MPC protocols to construct ZKPCPs and ZKPCPPs, respectively. We have improved known ZKPCP constructions in terms of the adaptivity of the honest verifier, and suggested several applications of zero-knowledge probabilistic proofs and codes. Additionally, we have generalized the notion of LRCCs, constructing LRCCs that guarantee correctness of the computation even in the presence of corrupted parties. Using these generalized LRCCs we designed leakage-secure protocols based on a single leaky trusted hardware device.

Several interesting and important questions still remain open, as we outline next.

Constructions of ZK probabilistic proofs and codes. A main open question is that of obtaining zero-knowledge PCPs and PCPPs with comparable parameters to non-ZK PCPs and PCPPs, and specifically in terms of proof length, zero-knowledge type, and the adaptivity of the honest verifier. Concretely, though there exist PCPs and PCPPs whose length is quasi-linear in the witness length [BS05, Din06], the shortest currently known ZKPCPs and ZKPCPPs have length at least quadratic. This raises the question of characterizing the exact “cost” of zero-knowledge, and determining whether quasi-linear length ZKPCPs and ZKPCPPs exist.

In terms of adaptivity of the honest verifier, the transformation of Chapter 4 only applies to PCPs (and not to PCPPs), so current ZKPCPPs (as constructed in Chapter 5) require adaptive verification. Constructing non-adaptively verifiable ZKPCPPs remains an interesting open problem. Such a construction would complement non-adaptively verifiable ZKPCPs, by improving the round complexity of the certifiable VSS protocols of Chapter 7, and several other cryptographic tasks.

Another open question is that of improving the ZK guarantee of non-adaptively verifiable ZKPCPs. (Recall that our constructions in Chapter 4 achieve somewhat weaker notions of zero-knowledge: either zero-knowledge with an inefficient simulator;
or zero-knowledge against computationally-bounded receivers, in the CRS model.)

Finally, ZKPCPs, ZKPCPPs, and ZKLTCs, have already proven (Chapter 7) most useful in improving the complexity, and underlying assumptions, of various cryptographic tasks. Thus, investigating additional cryptographic settings in which they could be used is a very appealing research direction.

**Leakage-resilient computation.** Perhaps the most important open question left by this work, is that of extending the leakage-resilience of multiparty LRCCs (and SAT-respecting LRCCs) to broader leakage classes, which would better capture the types of leakage exploited by known “side-channel” attacks. Another interesting open question is that of constructing SAT-respecting LRCCs with efficient simulation. Finally, our constructions cause a polynomial blowup in the size of the compiled circuit. The question of reducing this blowup remains open.
Appendix A

The Complexity of the Arora-Safra PCP [AS92]

In this Chapter we analyze the PCP system of Arora and Safra [AS92], and show that it has the properties needed to construct the NA-WIPCP system of Chapter 4 (Section 4.3). Specifically, the NA-WIPCP used the PCP system of Arora and Safra [AS92, Theorem 1] as a building block (concretely, the system used in [AS92] to prove that $3\text{SAT} \in \text{PCP}[\log n, \log^2 n, \frac{1}{2}, \text{poly}(n)]$), and relied on the following property: every proof bit in that system is computable by a low-depth, polynomial-sized circuit with “few” $\oplus$ gates (specifically, in $L_{O(1), \text{poly}(n), \oplus O(1)}$, see Definition 3.6.2).

The PCP system of [AS92] is constructed for the NP-complete language $3\text{SAT}$, where the prover, on input a 3CNF $\varphi$ with $n$ clauses and $n$ variables, and a satisfying assignment $A$ for $\varphi$, transforms $A$ into a PCP $\pi$ for the claim “$\varphi \in 3\text{SAT}$”. They show a construction over large fields, which we described in Section A.1. We then show (Section A.2) that this construction applies also to extension fields of $\text{GF}(2)$. The analysis in this section is very succinct, and is focused on the properties needed for the proof of Theorem 4.1. We refer the reader to [AS92] for a more detailed description of the PCP generation and verification.

A.1 The Construction over Large Fields

At a high level, the PCP system of [AS92] operates as follows. The prover $P$, given a 3CNF $\varphi$, and a satisfying assignment $A$ for $\varphi$, generates a low-degree polynomial $\hat{A} : F^m \rightarrow F$, where $m \in \mathbb{N}$, and $F$ is a large finite field, such that $\hat{A}$ “represents” $A$ (specifically, $\hat{A}$ is the Reed-Muller encoding of $A$). $P$ then uses $\hat{A}$, and the structure of $\varphi$, to construct another low-degree polynomial $g_A$ over $F$.

This representation is useful because [AS92] show that there exist an $H \subseteq F$, and a (not too large) set $P$ of low-degree polynomials $P_1, P_2, \ldots$, such that if $P_1$ is chosen

---

1The assumption that there are $n$ clauses and variables is without loss of generality, since the number of clauses and variables can be made equal by introducing new clauses or variables, if necessary.
uniformly at random from $\mathcal{P}$, then: if $A$ satisfies $\varphi$ then $P_i \cdot g_A$ sums to 0 on $H^m$ with probability 1; and otherwise this happens with probability at most $\frac{1}{2}$. Moreover, the set $\mathcal{P}$ is independent of $A$ and can be efficiently constructed in polynomial time. Consequently, to verify that $\varphi \in \text{3SAT}$, the verifier $V$ can pick a random $P_i \in \mathcal{P}$ and check that $g_A \cdot P_i$ vanishes on $H^m$.

Arora and Safra show that if $g_A$ is guaranteed to be a low-degree polynomial, then given access to the truth-table of $g_A$, and to an auxiliary proof, one can verify that $g_A \cdot P_i$ sums to 0 on $H^m$ (this is called the sum-check test). (Notice that $V$ can compute $P_i$ on its own, so $V$ does not need access to the truth-table of $P_i$, or $g_A \cdot P_i$.) Moreover, they show that the value of $g_A$ at any random point can be computed given only $\varphi$ and the value of $\hat{A}$ at 3 (related) random points, so it suffices to give $V$ the truth-table of $\hat{A}$. Furthermore, they show how to efficiently verify, given the truth-table of any function $f$, and an auxiliary proof, that $f$ is a low-degree polynomial (this is called the low-degree test).

Thus, the PCP prover $P$ outputs a proof consisting of the truth-table of $\hat{A}$, a proof for the low-degree test for $A$, and for every $P_i \in \mathcal{P}$, a proof for the sum-check test of $g_A \cdot P_i$. The verifier $V$ first verifies that $\hat{A}$ has low degree, then picks a random $P_i \in \mathcal{P}$ and performs the sum-check test on $g_A \cdot P_i$ (using the truth table of $\hat{A}$ to generate values of $g_A$, when needed). We now describe the ingredients of the proof, and how they are constructed, in more detail.

Let $\varphi$ be a 3CNF, and assume without loss of generality that $\varphi$ has $n$ variables and $n$ clauses. Let $h = O(\log n)$ and $m = O\left(\frac{\log n}{\log \log n}\right)$ such that $n = (h + 1)^m$. Denote $H = \{0, 1, \ldots, h\}$, and let $\mathbb{F} = \mathbb{GF}(2^t)$ for $t = O(\log h)$ (the constant in the definition of $O(\cdot)$ is taken to be large enough, such that $H \subseteq \mathbb{F}$). Identify every variable $x_i$ and every clause $c_i$ of $\varphi$ with a vector $v \in H^m$ (e.g., the representation of $i$ in basis $h$). Interpret the assignment $A$ to $\varphi$ as a function $A : H^m \rightarrow \mathbb{F}$ where $A(x_i) \in \{0, 1\}$ is the value that $A$ assigns to $x_i$.

**Constructing an extension-polynomial for the assignment $A$.** Let $\mathbb{F}_h[x_1, \ldots, x_m]$ denote the class of $m$-variate polynomials of individual degree at most $h$ over variables $x_1, \ldots, x_m$ with coefficients in $\mathbb{F}$. (A polynomial $p$ has individual degree at most $h$ if for every $x_i$, and every monomial $M$ of $p$, the degree of $x_i$ in $M$ is at most $h$.)

Let $\hat{A} \in \mathbb{F}_h[x_1, \ldots, x_m]$ be a polynomial such that $A(y) = \hat{A}(y)$ for every $y \in H^m$. $\hat{A}$ is called the polynomial extension of $A$ over $\mathbb{F}$ (because the degree is $h$, it is also unique), and can be constructed using the Lagrange polynomials $L_y$ as follows:

$$\hat{A}(x_1, \ldots, x_m) = \Sigma_{y=(y_1, \ldots, y_m) \in H^m} A(y) \cdot \Pi_{i=1}^{m} L_{y_i}(x_i)$$

where for every $y_i \in H$, $L_{y_i}(x_i) := \frac{\Pi_{y_i \neq z \in H(x_i \neq z)}}{\Pi_{y_i \neq z \in H(y_i \neq z)}}$. Notice that for every $y_i \in H$, $L_{y_i}$ depends only on $H$, and is independent of both $A$ and $\varphi$.

**Representing the formula $\varphi$ as a polynomial.** Arora and Safra show that there
exists a family $P$ of $l := O(mh)$ polynomials $P_1, \ldots, P_l \in \mathbb{F}_7^{4m}$ (that depend both on $\hat{A}$ and on $\varphi$), constructible in polynomial time, possessing the following 3 properties.

- If $A$ satisfies $\varphi$ then $\sum_{x_1, \ldots, x_{4m} \in H} P_i(x_1, \ldots, x_{4m}) = 0$ for every $1 \leq i \leq l$.
- If $A$ does not satisfy $\varphi$ then $\Pr_{P \in \mathcal{P}_{l, \ldots, P_l}} [\sum_{x_1, \ldots, x_{4m} \in H} P(x_1, \ldots, x_{4m}) = 0] \leq \frac{1}{8}$.
- For every $1 \leq i \leq l$ and every point $\alpha \in \mathbb{F}^{4m}$, the value of $P_i$ at $\alpha$ is determined by the value of $\hat{A}$ at 3 points $a_1, a_2, a_3 \in \mathbb{F}^m$. Moreover, if $\alpha$ is uniformly distributed in $\mathbb{F}^{4m}$ then each of $a_1, a_2, a_3$ is uniformly distributed in $\mathbb{F}^m$.

Concretely, $P_1, P_2, \ldots$ are obtained through the following procedure:

- (Arithmetization of $\varphi$.) For $j = 1, 2, 3$, define $\chi_j : H^{2m} \to \{0, 1\}$ such that $\chi_j(c, v) = 1$ if and only if $v$ is the $j$'th variable in clause $c$, and define $s_j : H^m \to \{0, 1\}$ such that $s_j(c) = 1$ if and only if the $j$'th variable in clause $c$ is not negated (both polynomials can be defined using the Lagrange polynomials as explained above). Let $\hat{\chi}_j, \hat{s}_j$ be the polynomial extensions of $\chi_j, s_j$ over $\mathbb{F}$, respectively. Notice that $\hat{\chi}_j \hat{s}_j$ can be computed locally given $\varphi$, i.e., they are independent of $A$.

- (Arithmetization of $A$.) Define $g_A : \mathbb{F}^{4m} \to \mathbb{F}$ as follows: $g_A(z, w_1, w_2, w_3) = \prod_{j=1}^3 \hat{\chi}_j(z, w_j) \left( \hat{s}_j(z) - \hat{A}(w_j) \right)$. Then $g_A$ has individual degree at most $6h$, and its evaluation at any point in $\mathbb{F}^{4m}$ can be locally computed given $\varphi$ and the value of $\hat{A}$ at 3 corresponding points in $\mathbb{F}^m$. Notice that $A$ satisfies $\varphi$ if and only if $g_A$ vanishes on $H^{4m}$.

- Let $R_1, \ldots, R_l \in \mathbb{F}_7^{4m}$ be such that for every $f : H^{4m} \to \mathbb{F}$ that is not identically 0,

$$\Pr_{R \in \mathcal{R}_{l, \ldots, R_l}} [\sum_{x_1, \ldots, x_{4m} \in H} R(x_1, \ldots, x_{4m}) \cdot f(x_1, \ldots, x_{4m}) = 0] \leq \frac{1}{8}.$$

Such a family exists and can be constructed in polynomial time (see [AS92, Lemma 10] for details). As will become evident later, the manner in which this family is constructed is of no interest to us, because every $R_i$ has poly($n$) monomials, and is independent of $\varphi, A$ (and can therefore be locally constructed by the verifier).

For every $1 \leq i \leq l$, set $P_i(x_1, \ldots, x_{4m}) := R_i(x_1, \ldots, x_{4m}) g_A(x_1, \ldots, x_{4m})$. Notice that given the evaluations of $\hat{A}$ on $\mathbb{F}^m$, the verifier can locally compute the values of $P_i(a_1, \ldots, a_{4m})$, for every $(a_1, \ldots, a_{4m}) \in \mathbb{F}^{4m}$, which requires the value of $\hat{A}$ at 3 points.

The proof. The proof consists of a low-degree proof for the low-degree test, and a sum-check proof for the sum-check test, and is generated as follows:

- The proof contains the evaluation of $\hat{A}$ on all points in $\mathbb{F}^m$. (This is used for both tests.)
• For every $j \in [m]$, and every $a_{-j} = (a_1, \ldots, a_{j-1}, a_{j+1}, \ldots, a_m) \in \mathbb{F}^{m-1}$, let
  \[ \hat{A}_{j,a_{-j}}(x) := \hat{A}(a_1, \ldots, a_{j-1}, x, a_{j+1}, \ldots, a_m), \]
  then the proof contains the evaluation of $\hat{A}_{j,a_{-j}}$ on every point in $\mathbb{F}$ (i.e., the proof contains the evaluations of restrictions of $\hat{A}$ to all lines in $\mathbb{F}^{m-1}$). (This is used for the low-degree test.)

• For every polynomial $P_i$, every $j \in [4m]$, and every $a_1, \ldots, a_{j-1} \in \mathbb{F}$, let
  \[ P_{i,a_1,\ldots,a_{j-1}}(x) := \sum_{z_{j+1},\ldots,z_{4m} \in \mathcal{H}} P_i(a_1, \ldots, a_{j-1}, x, z_{j+1}, \ldots, z_{4m}), \]
  then the proof contains evaluations of $P_{i,a_1,\ldots,a_{j-1}}$ on all points in $\mathbb{F}$. (This is used for the sum-check test.)

**Verification.** To verify that $\varphi \in \text{3SAT}$, the verifier performs the following checks.

• Checks that $\hat{A}$ has individual degree at most $h$, by running the low degree test on $\hat{A}$.

• Picks a random $i \in [l]$, and checks that $P_i$ sums to 0 on $\mathcal{H}^m$ by running the sum-check test. Whenever the sum-check test requires the value of $P_i$ at some (random) point, the verifier reads the value of $\hat{A}$ at (the) 3 (corresponding) points, and locally computes the value of $P_i$ at that point.

### A.2 The Construction over $\mathbb{GF}(2)$

In this section we show how to transform the PCP of [AS92] (which consists of field elements) into a proof over bits. (We are interested in representing the PCP over bits, because our NA-WIPCP construction needs the function generating the PCP from the NP witness to be in the leakage class that the SAT-respecting relaxed LRCC withstands. In our case, this class consists of functions computable by low-depth boolean circuits.)

We note first that since $[\mathbb{F} : \mathbb{GF}(2)] = t$, then it is isomorphic to the field $\mathbb{GF}(2)^t / \mathcal{P}_t(\mathcal{X})$, where $\mathcal{P}_t(\mathcal{X})$ is an irreducible polynomial of degree $t$, so we can assume that $\mathbb{F}$ is the field $\mathbb{GF}(2)^t / \mathcal{P}_t(\mathcal{X})$. Therefore, there exists a bijection $\mathcal{B}$ between $\mathbb{F}$ and $(\mathbb{GF}(2))^t$, which maps a polynomial $\sum_{i=0}^{t-1} a_i \mathcal{X}^i$ to the vector $(a_0, \ldots, a_{t-1})$ of its coefficients. ($\mathcal{B}, \mathcal{B}^{-1}$ are computable in time $O(t)$.) We will need the following fact connecting polynomials over extension fields to polynomials over the base field.

**Fact A.2.1.** Let $\mathbb{F} = \mathbb{GF}(2^t)$, and $d, m \in \mathbb{N}$. Then every polynomial $p \in \mathbb{F}_d[x_1, \ldots, x_m]$ corresponds to a collection of $t$ multi-linear polynomials $p_0, \ldots, p_{t-1} \in \mathbb{GF}(2)^t[y_1, \ldots, y_m]$, of total degree at most $\min\{md, mt\}$, each with at most $2^{mt}$ monomials, such that for every $\alpha_1, \ldots, \alpha_m \in \mathbb{F}$,

\[ \mathcal{B}(p(\alpha_1, \ldots, \alpha_m)) = (p_0(\mathcal{B}(\alpha_1), \ldots, \mathcal{B}(\alpha_m)), \ldots, p_{t-1}(\mathcal{B}(\alpha_1), \ldots, \mathcal{B}(\alpha_m))). \]
Moreover, given the truth-table of \( p \), the value of every \( p_i \) at any point in \( \beta \in \text{GF}(2)^{mt} \) is computable in time \( O(tm) \), and requires the value of \( p \) at a single point \( \alpha \in \text{F}^m \); and given the truth-tables of \( p_0, \ldots, p_{t-1} \), the value of \( p \) at any point in \( \alpha \in \text{F}^m \) is computable in time \( O(tm) \), and requires the values of \( p_0, \ldots, p_{t-1} \) at a single point \( \beta \in \text{GF}(2)^{mt} \).

The PCP described in the previous section consists of the evaluations of a collection of polynomials over \( \text{F} \). The general idea of the bit-PCP is to replace the evaluations of these polynomials with the evaluations of the corresponding \( t \) polynomials over \( \text{GF}(2) \) whose existence is guaranteed by Fact A.2.1. Then the proof consists of the evaluations of all these polynomials.

The verification procedure. The verification procedure is carried out essentially in the same way, except that the values of every polynomial \( \hat{A} \) and \( \hat{A}_{j,a-j}, P_i,a_1,\ldots,a_{j-1} \) at points \((\alpha_1, \ldots, \alpha_m) \in \text{F}^m \) and \( \alpha \in \text{F} \), respectively, are reconstructed using the values of the corresponding multivariate polynomials over \( \text{GF}(2) \) at \((B(\alpha_1), \ldots, B(\alpha_m)) \) and \( B(\alpha) \), respectively.

### A.3 Setting the Parameters

We set the parameters of the construction to be \( h = O(\log n) \), \( m = \frac{\log n}{\log \log n} = O\left(\frac{\log n}{\log \log n}\right) \) and \( \log |\text{F}| = t := c_h \log h = c \log \log n \) (for large enough constants \( c, c_h \)). For this choice of parameters, the construction (used in [AS92, Theorem 1] to prove that 3SAT \( \in \text{PCP} \{\log n, \log^2 n, \frac{1}{2}, \text{poly}(n)\} \) yields a PCP system with the following properties:

- **Query complexity.** The verifier makes \( O(m) \) queries to symbols of size \( h \log h \), reading a total of \( q := m \cdot h \cdot \log h = O\left(\frac{\log n}{\log \log n} \log n \cdot \log \log n\right) = O(\log^2 n) \) bits of the proof.

- **Soundness.** The system has constant soundness error \( \frac{1}{2} \).

- **Proof length.** The proof has length \( \text{poly}(n) \). Indeed, it consists of the evaluations of the following polynomials.

  - \( \hat{A} \in \text{F}_h[x_1, \ldots, x_m] \), which requires the evaluations on \( \text{F}^m \) (the evaluation at every point is a field element). This requires \( |\text{F}|^m \) field elements, i.e.,
    \[
    (2^c \log n)^O\left(\frac{\log n}{\log \log n}\right) = 2^{O\left(\log \log n \cdot \frac{\log n}{\log \log n}\right)} = \text{poly}(n) \text{ bits}.
    \]
  - \( \hat{A}_{j,a-j} \in \text{F}_h[x] \), for \( j \in [m] \) and \( a-j \in \text{F}^{m-1} \). There are \( m \cdot |\text{F}|^{m-1} \) such polynomials, and the evaluation of each requires the value (in \( \text{F} \)) at \( \text{F} \) points.
Therefore, these evaluations require \( m |F|^{m-1} \cdot |F| = m |F|^m \) field elements, i.e., \( |F|^m \log |F| = \text{poly} (n) \) bits.

- \( P_{i,j,a_1,\ldots,a_{j-1}} \in F_h [x] \), for every \( i \in [l], j \in [m], \) and \( (a_1, \ldots, a_{j-1}) \in \mathbb{F}^{j-1} \).
  There are at most \( l \cdot m \cdot |F|^{m-1} \) such polynomials, where their value at each point of \( F \) is a field element in \( F \). Therefore, these evaluations require \( l \cdot m \cdot |F|^{m-1} \cdot |F| = O (hm) \cdot m \cdot |F|^m \) field elements, and \( \frac{\log^4 n}{\log \log n} \cdot \text{poly} (n) = \text{poly} (n) \) bits.

- **Proof structure.** Every proof symbol (in the construction over large fields, Section A.1) is computable by a multivariate polynomial over \( F \) with at most \( 4m \) variables, of individual degree at most \( O (h) = O (\log n) \), with at most \( \text{poly} (n) \) monomials.

Therefore, by Fact A.2.1 every proof bit in the bit-implementation of the PCP of [AS92] is computable by a multilinear polynomial over \( \text{GF} (2) \) with at most \( 4mt = O \left( \frac{\log n}{\log \log n} \cdot \log \log n \right) = O (\log n) \) variables, of total degree at most \( \min \{ 4m \cdot O (h), 4m \cdot c \log \log n \} = O \left( \frac{\log^2 n}{\log \log n} \right), \) with at most \( 2^{4m \cdot t} = 2^{O \left( \frac{\log n}{\log \log n} \right) \cdot \log \log n} = 2^{O (\log n)} = \text{poly} (n) \) monomials. In particular, every proof bit can be computed by an \( \text{AC}^0 \) circuit of depth 2 with a single \( \oplus \) gate of unbounded fan-in. (The first layer will compute all the monomials, using (for every monomial) a single AND gate of unbounded fan-in; and the second would sum all monomials, using a single \( \oplus \) gate of unbounded fan-in.)

- **Computation time.** The proof over \( F \) can be constructed and verified in polynomial time. Generating the proof over \( \text{GF} (2) \) requires the prover to generate the truth tables of the polynomials \( \{ p_{A,0}, \ldots, p_{A,t-1} \}, \quad \{ p_{A_j,0,\ldots,t-1}, \ldots, p_{A_j,a_j,0,\ldots,t-1} \}, \) and \( \{ p_{P_{a_1,\ldots,a_j,0,\ldots,a_{j-1},t-1}}, \ldots, p_{P_{a_1,\ldots,a_{j-1},a_j,0,\ldots,a_{j-1},t-1}} \} \). The generation of every such point from the truth-tables of \( A, A_j, a_j \) and \( P_{a_1,\ldots,a_{j-1}} \) requires time \( O (4mt) = O (\log n) \), and so the proof generation time increases by a multiplicative \( O (\log n) \) factor.

The verification of the proof requires the verifier to generate, given the queries to the proof over \( F \), the corresponding queries over \( \text{GF} (2) \) (for every query, this requires at most \( O (mt) = O (\log n) \) time), and then map the query answers in \( \text{GF} (2)^l \) back to field elements in \( F \), which takes time \( O (mt) \) for every query. Therefore, the verification time also increases by a multiplicative factor of \( O (\log n) \).
Appendix B

Amplifying Transformations for Locking Schemes

The construction of ZKPCPPs with zero-knowledge against malicious verifiers (Chapter 5) employs a locking scheme (See Definition 5.2.7, Chapter 5) to “lock” the answers to “honest” queries. To achieve polynomial-length proofs that guarantee negligible soundness error, and zero-knowledge with a negligible statistical distance, the underlying locking scheme should have similar properties.

In spirit of simplifying PCP and PCPP constructions, and making them combinatorial, we prefer to use the locking scheme of [IMS12] over the locking scheme of [KPT97]. However, due to a non-optimal tradeoff between the hiding property and the lock length, the locking scheme of [IMS12] does not achieve these desired properties. Consequently, we describe general binding and equivocation enhancing transformations, and use them to transform the locking scheme of [IMS12] into a scheme with poly-length locks, $1 - 2^{-\sigma}$-binding, and equivocation with $2^{-\sigma}$ statistical distance, where $\sigma$ is a security parameter.

B.1 Equivocation Amplification

In this section we describe an equivocation amplifying transformation. The natural solution is to secret share the locked message, locking each secret share in a lock of its own. The intuition behind this solution is that to gather information about the locked message, the receiver must gather information about “enough” secret shares, i.e., it should break the equivocation of several locks. This should be harder than breaking the equivocation of a single lock. (The analysis is in fact more involved, as we explain below.) This intuition is formalized in the next construction.

Construction B.1. Let $\sigma$ be a security parameter, $k \in \mathbb{N}$ be an amplification parameter, and $(S, R)$ be a locking scheme for the message space $W$. The enhanced locking scheme $(S_{eq}, R_{eq})$ operates as follows.
• **Commitment.** The sender $S_{eq}$ has input $1^\sigma, 1^t, w \in \mathcal{W}$, where $|w| = n$. $S_{eq}$ additively secret-shares $w$ into $k$ shares $w_1, \ldots, w_k$, and uses $S(1^\sigma, w_1), \ldots, S(1^\sigma, w_k)$ to generate $k$ lock-key pairs $(L_1, K_1), \ldots, (L_t, K_k)$. $S_{eq}$ outputs $(L = (L_1, \ldots, L_k), K = (K_1, \ldots, K_k))$. The receiver $R_{eq}$ has input $1^\sigma, 1^t, 1^k$, and oracle access to $L$.

• **Decommitment.** $S_{eq}$ sends $K = (K_1, \ldots, K_k)$ to $R_{eq}$. For every $1 \leq i \leq k$, $R_{eq}$ uses $R_{eq}(1^\sigma, 1^t, K_i)$ to decommit some value $w'_i$. If for every $1 \leq i \leq k$, $w'_i \neq \perp$ then $R_{eq}$ outputs $w' = w'_1 \oplus \cdots \oplus w'_k$. Otherwise, he outputs $\perp$.

To guarantee that Construction B.1 enhances the equivocation of $(S, R)$, the underlying locking scheme should have a stronger equivocation guarantee. Before giving the formal definition, we first explain what goes wrong when only (standard) equivocation is assumed.

Assume that $(S, R)$ is $(u, \epsilon)$-equivocal with simulator $\text{Sim}$, namely $\text{Sim}$ can simulate up to $u$ queries to any single lock, and the simulated answers are $\epsilon$-statistically close to the answers of a real-world lock. To demonstrate the problem, consider the simpler case in which $\text{Sim}$ has the following guarantee: $\text{Sim}$ can abort (with probability at most $\epsilon$) when the simulation enters the second phase, otherwise the simulation is perfect. Since Construction B.1 is a general transformation that could be applied to any locking scheme (in particular, we know nothing regarding how $\text{Sim}$ operates), then to simulate the answers of a lock $L = (L_1, \ldots, L_k)$ generated by $S_{eq}$, the simulator $\text{Sim}_{eq}$ for $(S_{eq}, R_{eq})$ should use $\text{Sim}$ to simulate the answers of the underlying locks $L_1, \ldots, L_k$.

Intuitively, an equivocation guarantee with better statistical distance, say $(u, \epsilon^k)$-equivocation, means that $\text{Sim}_{eq}$ aborts only if $\text{Sim}$ aborts in all underlying simulations. Therefore, it should be the case that as long as there is one “good” simulation in which $\text{Sim}$ does not abort, then $\text{Sim}_{eq}$ can prevent $\text{Sim}$ from aborting in all other simulations.

Consider the following simpler case that in every simulation of $\text{Sim}$ there exists a message $w$ such that if $w$ is the message locked in the lock, then $\text{Sim}$ does not abort. When the simulation enters the second phase, $\text{Sim}_{eq}$ is given a secret $W$, from which he should generate random shares for the underlying simulations. ($\text{Sim}_{eq}$ must randomly secret share $W$, since an emulation of $\text{Sim}$ is guaranteed to be close to the real-world interaction, only if the locked message that $\text{Sim}$ receives in the second phase of the simulation is distributed similarly to the real world.) Notice that if $\text{Sim}_{eq}$ knows, for every simulation $i$ of $\text{Sim}$, the value of the message $w_i$ on which $\text{Sim}$ is guaranteed to not abort, then $\text{Sim}_{eq}$ can prevent $\text{Sim}$ from aborting in the first $k - 1$ emulations, by simply fixing the first $k - 1$ shares to $w_1, \ldots, w_{k-1}$. Notice that the value of the message $W$ locked in $L$, and the first $k - 1$ shares $w_1, \ldots, w_{k-1}$, completely determines the value $w_k$ of the last secret share. This introduces two further complications.

First, we can no longer claim that the shares $\text{Sim}_{eq}$ uses are distributed as in the real-world (and so we cannot use the equivocation of the underlying system). Second, it is possible that setting the last share to the value $w_k$ causes $\text{Sim}$ to abort in the $k$’th
emulation (since we have changed the values provided to the simulations of Sim, his
decision whether to abort or not may now change). The following strong equivocation
property, guarantees that the above situation does not happen. (Notice that strong
equivocation implies “standard” equivocation.)

**Definition B.1.1 (Strong equivocation).** Let σ be a security parameter, and (S, R)
be a locking scheme for the message space W with simulator Sim. We say that (S, R) has
(u, ε)-strong equivocation if the following holds for every (possibly malicious) receiver
R∗.

- At the onset of the simulation, Sim chooses a w0 ∈ R W,1 and a simulated key K.
- If the simulation terminates before entering the second phase then
  (Sim(1σ, 1n), K) ≡ (ViewR∗Lw0(1σ, 1n, Kw0), Kw0), where (Lw0, Kw0) is a ran-
dom lock-key pair for w0.
- When the simulation enters the second phase there exists a “bad event” B such
  that the following holds.
  - Whether or not B occurs depends only on the value of the simulated key,
    and the queries of R∗ in the first phase. (In particular, Sim can determine
    whether B occurred.) More specifically, B occurs only if the queries of R∗
during the first phase intersect a “hard” set H.
  - Pr[B] ≤ ε.
  - Pr[Sim aborts|B] = 0.
  - If Sim did not abort then for every message w given to Sim in the second
    phase, (Sim(1σ, 1n), K) ≡ (ViewR∗Lw(1σ, 1n, Kw), Kw).
  - Pr[Sim abort|L = Lw0] = 0 (even if B occurred).

We claim that construction B.1 exponentially improves equivocation, while incurring
only a linear (in k) increase in the size of the locks and keys. This is formalized in the
following claim.

**Claim B.1.2 (Equivocation enhancement for LSs).** Let σ be a security parameter, k ∈
N be an amplification parameter, and (S, R) be a locking scheme for the message space
W with (1 − δ)-binding, (u, ε)-hiding, and (u, ε)-strong equivocation. Then (Seq, Req)
is a locking scheme for W with (1 − kδ)-binding, (u, 2εk)-hiding, and (u, εk)-strong
equivocation.

**Proof.** Completeness follows from the completeness of the underlying locking scheme
(S, R) (and the completeness of the secret sharing scheme). Hiding follows from the
equivocation property and the triangle inequality.

---

3We note that this property is not needed to prove strong equivocation of Construction B.1, but
is needed to prove that the binding-enhancing transformation (Construction B.2) preserves the next
properties. See Section B.2 for additional details.
Binding. Fix some lock oracle \( L = (L_1, \ldots, L_k) \). As the underlying locking scheme is \((1 - \delta)\)-binding, then for every \( 1 \leq i \leq k \), there exists a \( w_i \) such that for any key \( K_i \), 
\[
\Pr \left[ R_{eq}^L (1^\sigma, 1^n, K_i) \notin \{ w_i, \perp \} \right] < \delta.
\]
Denote \( w = w_1 \oplus \ldots \oplus w_k \), we claim that for any key \( K, Pr \left[ R_{eq}^L (1^\sigma, 1^n, K) \notin \{ w, \perp \} \right] < k\delta \). Indeed, if \( L_1, \ldots, L_k \) decommit to \( w_1, \ldots, w_k \) (respectively), then \( L \) decommits to \( w \). Therefore, if \( L \) successfully decommits to \( w' \neq w \), there exists a lock \( L_i, 1 \leq i \leq k \), which successfully decommitted to \( w'_i \neq w_i \), which (by the \((1 - \delta)\)-binding of \((S, R)\)) happens with probability at most \( \delta \). Using the union bound, the probability that at least one of the locks \( L_i \) successfully decommitted to some value \( w'_i \neq w_i \) is at most \( k\delta \).

\((u, \epsilon^k)\)-strong equivocation.\(^2\) Let \( R^* \) be a (possibly malicious) receiver, then the simulator \( \text{Sim} \) for \( R^* \) uses (in a non-black-box manner) \( k \) independents copies \( \text{Sim}_{in,1}, \ldots, \text{Sim}_{in,k} \) of the simulator \( \text{Sim}_{in} \) for \((S, R)\), and operates in the following way.

- At the onset of the simulation, every \( \text{Sim}_{in,i} \) chooses a \( w'_i \in R \mathcal{W} \), and a simulated key \( K_i \). Then \( \text{Sim} \) sets \( w = w'_1 \oplus \ldots \oplus w'_k \), and \( K = (K_1, \ldots, K_i) \).

- During the first phase of the simulation, \( \text{Sim} \) forwards every query of \( R^* \) to some lock \( L_i \) to \( \text{Sim}_{in,i} \).

- When the simulation enters the second phase, \( \text{Sim} \) is given a message \( w \). For every \( 1 \leq i \leq k \), let \( \mathcal{B}_i \) be the “bad event” for simulator \( \text{Sim}_{in,i} \), and let \( \mathcal{B} = \{ i \in [k] : \mathcal{B}_i \text{ occurred} \} \). We define the “bad event” \( \mathcal{B} \) for \( \text{Sim} \) to be the event that \( \mathcal{B} = [k] \). If \( w = w' \) then \( \text{Sim} \) sends \( w'_1, \ldots, w'_k \) to \( \text{Sim}_{in,1}, \ldots, \text{Sim}_{in,k} \) as the locked messages. Otherwise, if \( \mathcal{B} = [k] \) then \( \text{Sim} \) aborts. If \( w \neq w' \), and \( \mathcal{B} \neq [k] \), then \( \text{Sim} \) generates the secret shares as follows. Set some \( j \notin \mathcal{B} \), and for every \( j \neq l \in \mathcal{B}, \text{Sim} \) chooses \( w_i \) at random, for every \( i \in \mathcal{B}, \text{Sim} \) sets \( w_i = w'_i \), and finally, \( w_j := w \oplus \left( \bigoplus_{i \in \mathcal{B}} w'_i \right) \oplus \left( \bigoplus_{j \notin \mathcal{B}} w_i \right) \). The secret shares \( w_1, \ldots, w_k \) of \( w \) are given to \( \text{Sim}_{in,1}, \ldots, \text{Sim}_{in,k} \) as the messages locked in the locks.

- During the second phase of the simulation, \( \text{Sim} \) continues to forward the queries of \( R^* \) to \( \text{Sim}_{in,1}, \ldots, \text{Sim}_{in,k} \).

We claim that \((S_{eq}, R_{eq})\) achieves strong equivocation with simulator \( \text{Sim} \). Notice that since \( w'_i \in R \mathcal{W} \) for every \( 1 \leq i \leq k \), then \( w'_1 \oplus \ldots \oplus w'_k = w' \in R \mathcal{W} \). Assume first that the simulation did not enter the second phase. Then by the strong equivocation of \((S, R), (\text{Sim}_{in,i} (1^\sigma, 1^n), K_i) \equiv \left( \text{View}_{R} (L_{w'_i} (1^\sigma, 1^n, K_{w'_i})), K_{w'_i} \right) \) for every \( 1 \leq i \leq k \), where \( R^*_i \) is the receiver induced by the queries that \( R^* \) sent to the \( i \)th sub-lock, and \( (L_{w'_i}, K_{w'_i}) \) is a random lock-key pair for \( w'_i \). Therefore, \((\text{Sim} (1^\sigma, 1^n), K) \equiv (\text{View}_{R^*_i} (L_{w'_i} (1^\sigma, 1^n, K_{w'_i}), K_{w'_i}) \) where \( (L_{w'}, K_{w'}) \) is a random lock-key pair for \( w' \), and \( w' \in R \mathcal{W} \).

\(^2\)Since strong equivocation implies equivocation, this will also show that the system is \((u, \epsilon^k)\)-equivocal.
Next, we analyze the “bad event” $\mathcal{B}$. Notice first that $\mathcal{B}$ occurs if and only if $B_i$ occurred for every $1 \leq i \leq k$. Moreover, in the first phase the underlying simulations are independent, so

$$\Pr[\mathcal{B}] = \Pr[\bigwedge_{i \in [k]} B_i] = \prod_{i \in [k]} \Pr[B_i] \leq \epsilon^k.$$ 

Furthermore, whether or not $B_i$, $i \in [k]$ occurred depends solely on the simulated key $K_i$, and the queries of $R^*$ to the $i$’th lock in the first phase, so whether or not $\mathcal{B}$ occurred depends only on the simulated key $K$, and the queries of $R^*$ during the first phase. (In particular, every simulator $\text{Sim}_{in}$ can determine whether $B_i$ occurred, so $\text{Sim}$ can determine whether $\mathcal{B}$ occurred.) Notice also that if $\mathcal{B}$ does not occur, then $\text{Sim}$ does not abort.

We claim that conditioned on the event that $\text{Sim}$ did not abort the simulation is perfect. If $w = w'$ then the values that $\text{Sim}$ sends to the underlying simulators in the second phase are $w_1', \ldots, w_k'$, so by the strong equivocation of $(S, R)$, then

$$(\text{Sim}_{in,i}(1^n, 1^n), K_i) \equiv \left( \bigvee_{R^*_i \in L_i} (1^n, 1^n, K_{w_i}), K_{w_i}' \right)$$

for every $1 \leq i \leq k$, where $R^*_i \in L_i$ is the receiver induced by the queries of $R^*$ to the $i$’th lock. Since in this case $w_1', \ldots, w_k'$ is a random secret-sharing of $w$, $(\text{Sim}(1^n, 1^n), K) \equiv (\bigvee_{R^*_i \in L_i} (1^n, 1^n, K), K)$. Second, assume that $w \neq w'$, and notice that in this case $\mathcal{B} \neq [k]$ (because we have conditioned on the event that $\text{Sim}$ did not abort). We claim that by the choice of $w_1, \ldots, w_k$, these are a random secret sharing of $w$, and therefore identically distributed to the values locked in a random real-world lock for $w$. Indeed, for every $i \in \mathcal{B}$, $w_i = w_i'$ is random, and so is $w_j$ for every $j \neq l \in \mathcal{B}$.

Therefore, we can condition the sharing on the shares $w_i, i \in \mathcal{B}$ and $w_j, j \neq l \in \mathcal{B}$, in which case $w_j$ (such a $j$ exists since $\mathcal{B} \neq [k]$) has the right distribution (the only possible value that yields a sharing of $w$). Moreover, non of the underlying simulators abort. Indeed, for $i \in \mathcal{B}$ this holds by the strong equivocation of $(S, R)$, because $w_i = w_i'$; and for $i \notin \mathcal{B}$ this holds because $\mathcal{B}_i$ did not occur. (Recall that whether or not $\mathcal{B}_i$ occurs depends only on the receiver queries, and the key, so changing the value of the message locked in the lock does not cause $\mathcal{B}_i$ to occur.)

Consequently, the strong equivocation of $(S, R)$ guarantees that for every $1 \leq i \leq k$,

$$(\text{Sim}_{in,i}(1^n, 1^n), K_i) \equiv \left( \bigvee_{R^*_i \in L_i} (1^n, 1^n, K_{w_i}), K_{w_i}' \right),$$

where $R^*_i \in L_i$ is the receiver induced by the queries of $R^*$ to the $i$’th lock. Since $w_1, \ldots, w_k$ is a random secret sharing of $w$ (and therefore identically distributed to the values locked in a random lock for $w$), we conclude that the simulation in this case is perfect.

Finally, we show that $\text{Sim}$ does not abort when given the message $w'$ as the message locked in the lock (even if $\mathcal{B}$ occurs). By the strong equivocation of $(S, R)$, for every $1 \leq i \leq k$ $\text{Sim}_{in,i}$ does not abort when given the message $w_i'$ as the message locked in $L_i$. Since $w = w_1' + \ldots + w_k'$, if $\text{Sim}$ is given $w'$ as the locked message then non
of the underlying simulators abort (since the secret sharing $\text{Sim}$ uses in this case is $w_1', \ldots, w_k'$).

\section*{B.2 Amplifying Binding}

In this section we describe a binding-amplifying transformation for locking schemes. The natural method for binding amplification is to use several locks that lock the same value, as described in the next construction.

Construction B.2. Let $\sigma$ be a security parameter, $k \in \mathbb{N}$ be an amplification parameter, and $(S, R)$ be a locking scheme for the message space $W$. The enhanced locking scheme $(S_{\text{bind}}, R_{\text{bind}})$ is defined as follows.

- **Commitment.** The sender $S_{\text{bind}}$ has input $1^\sigma, 1^k, w \in W$, where $|w| = n$. $S_{\text{bind}}$ uses $S(1^\sigma, w)$ to generate $k$ random and independent lock-key pairs $(L_1, K_1), \ldots, (L_k, K_k)$, and outputs $(L = (L_1, \ldots, L_k), K = (K_1, \ldots, K_k))$. The receiver $R_{\text{bind}}$ has input $1^\sigma, 1^n, 1^k$, and oracle access to $L$.

- **Decommitment.** $S_{\text{bind}}$ sends $K = (K_1, \ldots, K_k)$ to $R_{\text{bind}}$. For every $1 \leq i \leq k$, $R_{\text{bind}}$ uses $R^{L_i}(1^\sigma, 1^n, K_i)$ to decommit some value $w_i'$. If $w_1' = \ldots = w_k'$ then $R_{\text{bind}}$ outputs $w_1'$. Otherwise, he outputs $\perp$.

We claim that Construction B.2 exponentially improves the binding, while incurring a multiplicative $O(k)$ increase in the size of locks and keys. Roughly speaking, the binding amplification holds because to decommit the lock to a value different than the one it was locked to, most of the underlying locks must decommit to a value which is different than the one they was locked with. This is formalized in the next claim.

Claim B.2.1 (Binding enhancement for LSs). Let $\sigma$ be a security parameter, $k \in \mathbb{N}$ be an amplification parameter, and $(S, R)$ be a locking scheme with $(1-\delta)$-binding, $(u, \varepsilon)$-hiding, and $(u, \varepsilon)$-strong equivocation. Then Construction B.2, when applied to $(S, R)$, is $\left(1 - (2e\delta)^{\frac{1}{2}}\right)$-binding, $(u, 2ke)$-hiding, and has $(u, ke)$-strong equivocation.

\textbf{Proof.} Completeness follows from the completeness of the underlying locking scheme $(S, R)$. Hiding follows from the equivocation property and the triangle inequality.

\textbf{Binding.} Fix some lock oracle $L = (L_1, \ldots, L_k)$. We claim that there exists a $w \in W$ such that for every key $K'$, $\Pr [R_{\text{bind}}^{L_i}(1^\sigma, 1^n, 1^k, K') \notin \{w, \perp\}] < (2e\delta)^{\frac{1}{2}}$. By the $(1-\delta)$-binding of $(S, R)$, for every $1 \leq i \leq k$ there exists a $w_i \in W$ such that for every key $K_i$, $\Pr [R^{L_i}(1^\sigma, 1^n, K_i) \notin \{w_i, \perp\}] < \delta$. We say that $L_i$ \textit{agrees to} $w$ if $w_i = w$.

Let $w$ such that the largest number of locks agree to $w$, then we show that binding holds for $w$. Indeed, $R_{\text{bind}}$ outputs $w' \notin \{w, \perp\}$ only if all the locks successfully decommit to $w'$. As $w' \neq w$, at most $\frac{k}{2}$ of the locks agree to $w'$. In particular, if $R_{\text{bind}}$ outputs $w'$ then at least $\frac{k}{2}$ of the locks that do not agree to $w'$ must decommit to $w'$ which,
by the binding of \((S, R)\), happens with probability at most \(\delta\). As this holds for every \(w' \notin \{w, \bot\}\) then for any key \(K' = (K'_1, \ldots, K'_k)\),

\[
\Pr \left[ R^L_{\text{bind}} \left( 1^n, 1^n, 1^k, K' \right) \notin \{w, \bot\} \right]
\]

is at most

\[
\Pr \left[ \text{all the locks } L_1, \ldots, L_k \text{ successfully decommit to some value } \notin \{w, \bot\} \right]
\]

which is upper-bounded by

\[
\Pr \left[ s \geq \frac{k}{2} \right]
\]

of the locks that agree to \(w\) successfully decommit to some value \(\notin \{w, \bot\}\)

which is at most

\[
\left( \frac{k}{2} \right) \cdot \frac{k}{2} \cdot \frac{k}{2} \leq \left( \frac{ke}{2^n} \right) \cdot \frac{k}{2} \cdot \frac{k}{2} \leq (2e\delta)^{\frac{k}{2}}
\]

\((u, ke)\)-strong equivocation. Let \(R^*\) be a (possibly malicious) receiver, then the simulator \(\text{Sim}\) for \(R^*\) uses (in a non-black-box manner) \(k\) independent copies \(\text{Sim}_{\text{in},1}, \ldots, \text{Sim}_{\text{in},k}\) of the simulator \(\text{Sim}_{\text{in}}\) for \((S, R)\) (whose existence follows from the strong equivocation of the system), and operates as follows.

- At the onset of the simulation, \(\text{Sim}\) randomly selects a secret \(w' \in_R W\), and instantiates \(\text{Sim}_{\text{in},1}, \ldots, \text{Sim}_{\text{in},k}\) where every \(\text{Sim}_{\text{in},i}\) uses \(w'\) as the simulated message.\(^3\)
  (This can be guaranteed by an appropriate choice of the randomness of \(\text{Sim}_{\text{in}}\), and since the simulations are not clack-box, then \(\text{Sim}\) can determine how to choose the randomness appropriately) For every \(1 \leq i \leq k\), \(\text{Sim}_{\text{in},i}\) chooses a simulated key \(K_i\), and \(\text{Sim}\) sets \(K = (K_1, \ldots, K_k)\).
- During the first phase of the simulation, \(\text{Sim}\) forwards every query of \(R^*\) to some lock \(i\) to \(\text{Sim}_{\text{in},i}\).
- When the simulation enters the second phase, i.e. \(\text{Sim}\) receives an arbitrary \(w \in W\), it sends \(w\) to \(\text{Sim}_{\text{in},1}, \ldots, \text{Sim}_{\text{in},k}\). If one of the underlying simulators aborts, then so does \(\text{Sim}\).
- During the second phase of the simulation, \(\text{Sim}\) continues to forward the queries of \(R^*\) to \(\text{Sim}_{\text{in},1}, \ldots, \text{Sim}_{\text{in},k}\).

We claim that \((S_{\text{bind}}, R_{\text{bind}})\) achieves strong equivocation with simulator \(\text{Sim}\). First, notice that \(w' \in_R W\) by definition. Next, assume that the simulation did not enter

\(^3\)We note that this step is the reason that the property of strong equivocation allows the simulator to choose the simulated message. Otherwise, if we are only guaranteed that there exists some message with which the simulation is consistent, then these messages may differ between different invocations of \(\text{Sim}_{\text{in}}\), and so the simulation would significantly differ from the real-world transcript of the interaction with a valid lock.
the second phase. Then by the strong equivocation of \((S,R)\), \((\text{Sim}_{\text{in},i}(1^\sigma,1^n),K_i)\) is equivalent (w.r.t. the receiver induced by the queries of \(R^*\)) to \(\left(\mathcal{V}_{R^*_i \cdot L_{w'}} (1^\sigma,1^n,K_{w'}),K_{w'}\right)\) for every \(1 \leq i \leq k\), where \(R^*_i\) is the receiver induced by the queries of \(R^*\) to the \(i\)’th sub-lock, and \((L_{w'},K_{w'})\) is a random lock-key pair for \(w'\). Therefore, \((\text{Sim}(1^\sigma,1^n),K)\) is equivalent (w.r.t. the receiver induced by the queries of \(R^*\)) to \(\left(\mathcal{V}_{R^* \cdot L_{w'}} (1^\sigma,1^n,K_{w'}),K_{w'}\right)\), where \((L_{w'},K_{w'})\) is a random lock-key pair for \(w'\).

Next, we define a “bad event” \(\mathcal{B}\) for \(\text{Sim}\), and analyze its properties. For every \(1 \leq i \leq k\), let \(\mathcal{B}_i\) denote the “bad event” for simulator \(\text{Sim}_{\text{in},i}\) (whose existence follows from the strong equivocation of \((S,R)\)). Let \(\mathcal{B} = \bigvee_{1 \leq i \leq k} \mathcal{B}_i\) (i.e., \(\mathcal{B}\) is the event that at least one of the bad events \(\mathcal{B}_i\) occurred). By the strong equivocation of \((S,R)\), whether or not \(\mathcal{B}_i, i \in [k]\) occurred depends only on the simulated key \(K_i\), and the queries of \(R^*\) to the \(L_i\) in the first phase, so whether or not \(\mathcal{B}\) occurred depends only on the simulated key \(K\) and the queries of \(R^*\) during the first phase. (In particular, \(\text{Sim}\) can determine whether \(\mathcal{B}\) occurred, since every \(\text{Sim}_{\text{in},i}\) can determine whether \(\mathcal{B}_i\) occurred.) Moreover, for every \(1 \leq i \leq k\) \(\Pr[\mathcal{B}_i] \leq \epsilon\), so \(\Pr[\mathcal{B}] = \sum_{i \in [k]} \Pr[\mathcal{B}_i] \leq k\epsilon\) by the union bound. We claim also that \(\Pr[\text{Sim aborts}|\mathcal{B}] = 0\). Indeed, \(\text{Sim}\) aborts only if one of the underlying simulator aborts. Conditioned on \(\mathcal{B}\), for every \(1 \leq i \leq k\) the event \(\mathcal{B}_i\) does not occur, so the strong equivocation of \((S,R)\) guarantees that \(\text{Sim}_{\text{in},i}\) does not abort, and so \(\text{Sim}\) does not abort.

We claim that conditioned on the event that \(\text{Sim}\) did not abort then the simulation is perfect. If \(\text{Sim}\) does not abort, then for every \(1 \leq i \leq k\), \(\text{Sim}_{\text{in},i}\) does not abort, in which case the strong equivocation of \((S,R)\) guarantees that \((\text{Sim}_{\text{in},i}(1^\sigma,1^n),K_i)\) is equivalent (w.r.t. the receiver induced by the queries of \(R^*\)) to \(\left(\mathcal{V}_{R^*_i \cdot L_{w'}} (1^\sigma,1^n,K_{w'}),K_{w'}\right)\), where \(R^*_i\) is the receiver induced by the queries of \(R^*\) to the \(i\)’th lock, and \((L_{w'},K_{w'})\) is a random lock-key pair for \(w'\). Consequently, \((\text{Sim}(1^\sigma,1^n),K)\) is equivalent (w.r.t. the receiver induced by the queries of \(R^*\)) to \(\left(\mathcal{V}_{R^* \cdot L_{w'}} (1^\sigma,1^n,K_{w'}),K_{w'}\right)\). (Here we also use the fact that in this case, \(\text{Sim}\) sends \(w\) as the locked message to all simulators.)

Finally, by the strong equivocation of \((S,R)\), when \(\text{Sim}_{\text{in},i}, i \in [k]\) is \(w'\) as the message locked in the lock, then \(\text{Sim}_{\text{in},i}\) does not abort. Therefore, \(\text{Sim}\) does not abort when given the message \(w'\) as the message locked in the lock. \(\square\)

Remark B.3. Claim B.2.1 shows that the binding-amplifying transformation of Construction B.2 preserves strong equivocation (with a certain loss in the parameters). However, we note that this stronger property is not needed for the binding-amplifying transformation. That is, if the original locking scheme has (standard) equivocation, then (using a standard hybrid argument) one can show that the amplified lock is also equivocal. This should be contrasted with the equivocation-amplifying transformation of Construction B.1, for which strong equivocation seems necessary.

Complexity of the enhancing transformations

Construction B.1, and Construction B.2, incur a linear increase in the length of locks and keys. That is, if \(\ell_{\text{lock}}, \ell_{\text{key}}\) are (upper bounds on) the size of locks and keys generated
by $S(1^\sigma, w)$ (respectively), then for every $(L_{eq}, K_{eq}) \in S_{eq}(1^\sigma, 1^k, w)$, $(L_{bind}, K_{bind}) \in S_{bind}(1^\sigma, 1^k, w)$, $|L_{eq}|, |L_{bind}| \leq k\ell_{lock}$ and $|K_{eq}|, |K_{bind}| \leq k\ell_{key}$. Moreover, if the honest receiver $R$ of the original scheme tosses $r$ coins, and makes $q$ queries to the lock during the decommitment phase, then $R_{eq}, R_{bind}$ toss $kr$ coins, and make $kq$ queries during the decommitment phase.

B.3 Strong Equivocation of the Locking Scheme of [IMS12]

In the next section, we construct a locking scheme with “good” hiding and binding guarantees, and “short” locks, by applying our amplifying transformations (Sections B.1 and B.2) to the locking scheme of Ishai et al. [IMS12]. These transformations require that the locking scheme be strongly equivocal. Therefore, in this section we amplify the binding of the locking scheme of [IMS12], and show that it is strongly equivocal. The locking scheme of [IMS12] is described in the next construction.

Construction B.4. The Locking scheme $(S^{IMS}, R^{IMS})$ of [IMS12] for message space $\{0, 1\}$ is parameterized by a security parameter $\sigma$.

- **Commitment.** The sender $S^{IMS}$ on input $1^\sigma$, and $b \in \{0, 1\}$, generates a key $K \in_R \{0, 1\}^\sigma$, and a locking oracle $L = (h, f, g)$, which can be represented as a function $L : \{0, 1\}^{3\sigma + 2} \rightarrow \{0, 1\}^{3\sigma}$ consisting of the following three functions:
  - A random function $h : \{0, 1\}^{3\sigma} \rightarrow \{0, 1\}^{\sigma}$.
  - A random function $f : \{0, 1\}^{\sigma} \rightarrow \{0, 1\}^{3\sigma}$.
  - A function $g : \{0, 1\}^{3\sigma} \rightarrow \{0, 1\}^{3\sigma}$, defined as $g(r) = b \cdot r + f(K + h(r))$ (additions and multiplications are component-wise in $GF(2)$).

  The receiver $R^{IMS}$ has input $1^\sigma$, and oracle access to the lock $L$.

- **Decommitment.** To decommit $b$, $S^{IMS}$ sends the key $K$ to $R^{IMS}$. $R^{IMS}$ chooses $r \in_R \{0, 1\}^{3\sigma}$, and queries $h(r), f(K + h(r))$ and $g(r)$ to determine the value of $b$.

  Let $(L, K) \in S^{IMS}(1^\sigma, b)$ for some $b \in \{0, 1\}$. We define the $k$-enhanced decommitment step, for an amplification parameter $k \geq 1$.

**Definition B.3.1** ($k$-enhanced decommitment phase, Construction B.4). Given the key $K$, $R^{IMS}$ repeats the decommitment phase $k$ independent times. If in all decommitments the same bit $b$ was decommitted, then the receiver outputs $b$ as the secret locked in $L$. Otherwise, he outputs $\perp$.

  We show that if the decommitment phase of Construction B.4 is replaced with the $2\sigma$-enhanced decommitment phase, then the resultant locking scheme is at least $1 - 2^{-\sigma}$-binding.
Lemma B.3.2 (Binding, Construction B.4). The locking scheme of Construction B.4, with the $k$-enhanced decommitment phase, is $1 - 2^{-\sigma}$-binding for every $k \geq 2\sigma \geq 4$.

Remark. Whenever we use the locking scheme of [IMS12] (e.g. in the proof of Corollary B.6), we actually use Construction B.4 above with the $2\sigma$-enhanced decommitment phase.

The proof is similar to the proof of the binding property in [IMS12].

**Proof of Lemma B.3.2.** For a given locking oracle $L = (h, f, g)$, let

$$B_L = \left\{ r \in \{0, 1\}^{3\sigma} : \exists y_1, y_2 \in f_L (\{0, 1\}^{\sigma}) \text{ such that } y_1 + r = y_2 \right\}$$

then $|B_L| \leq 2^{2\sigma}$. Indeed, $|f_L (\{0, 1\}^{\sigma})| \leq 2^\sigma$, therefore the number of $r \in \{0, 1\}^{3\sigma}$ for which there exists a pair $\{y_1, y_2\} \subseteq f_L (\{0, 1\}^{\sigma})$ such that $y_1 + r = y_2$ is at most $2^\sigma \cdot 2^\sigma = 2^{2\sigma}$ (setting $y_1, y_2$ determines $r$, and in particular for every such pair exactly one $r$ is possible).

If the locking scheme is not $1 - 2^{-\sigma}$-binding, then there exist a lock $L = (h, f, g)$ and a pair of keys $K_0, K_1 \in \{0, 1\}^{\sigma}$ (not necessarily different) such that

$$\Pr \left[ \left( R^{\text{IMS}} \right)^L (1^{\sigma}, 1, K_b) = b \right] > 2^{-\sigma} \text{ for } b = 0, 1.$$

Let

$$\text{Rand}_{0,K_0} := \left\{ r \in \{0, 1\}^{3\sigma} : g (r) = f (K_0 + h (r)) \right\}$$

and

$$\text{Rand}_{1,K_1} := \left\{ r \in \{0, 1\}^{3\sigma} : g (r) = r + f (K_1 + h (r)) \right\}.$$

During the decommitment phase, $R^{\text{IMS}}$ performs the $k$-enhanced decommitment phase, which consists of choosing $k$ random and independent values $r \in \{0, 1\}^{3\sigma}$, and unlocking $L$ with each of them. $R^{\text{IMS}}$ with key $K_b$ outputs $b$ after this phase if and only if for all $k$ random strings $r$ he used, $L$ with key $K_b$ and randomness $r$ decommitted the value $b$. In particular, this means that every randomness $r$ chosen during the $k$-enhanced decommitment phase was in $\text{Rand}_{b,K_b}$. As the random strings $r$ chosen during decommitment are random and independent, then for every $b \in \{0, 1\}$,

$$\Pr \left[ \left( R^{\text{IMS}} \right)^L (1^{\sigma}, 1, K_b) = b \right] = \left( \frac{|\text{Rand}_{b,K_b}|}{2^{3\sigma}} \right)^k.$$

In particular,

$$2^{-\sigma} < \left( \frac{|\text{Rand}_{b,K_b}|}{2^{3\sigma}} \right)^k \iff |\text{Rand}_{b,K_b}| > \left( 2^{-\sigma} \cdot 2^{3\sigma} k \right)^{\frac{1}{k}} = 2^{3\sigma - \frac{2}{k}} \geq 2^{3\sigma - \frac{1}{2}}$$

where the right inequality holds since $k \geq 2\sigma$.

On the one hand, both $\text{Rand}_{0,K_0}$ and $\text{Rand}_{1,K_1}$ are subsets of $\{0, 1\}^{3\sigma}$, so

$$|\text{Rand}_{0,K_0} \cap \text{Rand}_{1,K_1}| \geq 2 \cdot 2^{3\sigma - \frac{1}{2}} - 2^{3\sigma}.$$
On the other hand, \( \text{Rand}_{0,K_0} \cap \text{Rand}_{1,K} \subseteq B_L \), so \(|\text{Rand}_{0,K_0} \cap \text{Rand}_{1,K_1}| \leq 2^{2\sigma} \). To recap,

\[
2^{3\sigma + \frac{1}{2}} - 2^{3\sigma} = 2 \cdot 2^{3\sigma - \frac{1}{2}} - 2^{3\sigma} \leq |\text{Rand}_{0,K_0} \cap \text{Rand}_{1,K_1}| \leq 2^{2\sigma}
\]

which holds if and only if

\[
2^{2\sigma} < 2^{3\sigma - 2} < 2^{3\sigma} \left(\sqrt{2} - 1\right) = 2^{3\sigma + \frac{1}{2}} - 2^{3\sigma} \leq 2^{2\sigma}
\]

(where the left-most inequality holds when \( \sigma > 2 \), a contradiction. \( \square \)

Next, we prove that Construction B.4 has strong equivocation.

**Lemma B.3.3** (Strong equivocation, Construction B.4). Let \( \sigma \in \mathbb{N} \) be a security parameter, then the locking scheme of Construction B.4 has \( \left(2^\frac{7}{2}, 2^{-\frac{7}{2}}\right) \)-strong equivocation.

Let \( R^* \) be a (possibly malicious) receiver, \( L = (h, f, g) \) be a locking oracle locking the secret \( b \), and \( K \) be the corresponding key. Fix some transcript of the interaction of \( R^* \) with \( L \), and let \( H, F, G \) denote the sets of queries that \( R^* \) makes to \( h, f, g \), respectively, where every \( g \)-query is also added to \( H \). We use the following claim (stated as \([\text{IMS12}, \text{Claim 4.4}]\)):

**Claim B.3.4** ([\text{IMS12}]). All the answers to the query sets \( H, F, G \) will look random to \( R^* \), as long as the following two conditions hold:

- The queries to \( h \) do not collide (i.e., \(|H| = |h(H)|\)).
- \((K + h(H)) \cap F = \emptyset\) (namely, there exists no \( r \in h(H) \) such that \( K + r \in F \)).

**Proof of Lemma B.3.3.** Let \( R^* \) be a (possibly malicious) receiver, then the simulator \( \text{Sim} \) for \( R^* \) operates as follows. At the beginning of the simulation, \( \text{Sim} \) randomly selects a key \( K \in \{0, 1\}^\sigma \), and a random message \( b^* \in_R \{0, 1\} \). During the simulation, let \( H, F, G \) be the sets of queries that \( R^* \) sent to \( h, f, g \), respectively (where \( g \)-queries are also added to \( H \)). Then \( \text{Sim} \) answers every query \( r \) of \( R^* \) as follows:

- **r is a query to h.** \( \text{Sim} \) answers with a random value, unless \( r \in H \) in which case \( \text{Sim} \) answers according to the value \( h(r) \) already determined during the simulation.

- **r is a query to f.** \( \text{Sim} \) answers with a random value, unless \( r = h(r') + K \) for some \( r' \in G \), in which case \( \text{Sim} \) answers with the value \( f(K + h(r')) = g(r') + r \cdot b^* \).

- **r is a query to g.** \( \text{Sim} \) chooses \( h(r), f(K + h(r)) \) (as defined above) and adds \( r \) to \( H \). The answer to the query is \( g(r) = r \cdot b^* + f(K + h(r)) \).

After having answered at most \( 2^{\frac{7}{2}} \) queries of \( R^* \), the simulation enters the second phase, and \( \text{Sim} \) is given the secret \( b \) that is locked in the real-world lock. We define a “bad event” \( \mathcal{B} \) as follows: \( \mathcal{B} \) occurs if there are \( h \)-collisions (i.e., \(|h(H)| \neq |H|\)), or

267
When the simulation enters the second phase, \( b^* \) is chosen at random for this phase. The key (both in the real-world and the simulation) is chosen uniformly at random in \( \{0,1\}^\sigma \), and the answer to a query \( g_{b,K} \) is chosen at random to be consistent with the key \( K \). During the first simulation phase, the answers of the simulator are consistent with the key \( K \), if the simulation terminated before entering the second phase, then the simulated view is identically distributed to a real-world random lock for key \( K \). During the second phase of the simulation, \( b^* \) is chosen at random, and the answer to a query \( g \) is random such that they are consistent with the key \( K \), and since \( \text{Sim} \) did not abort, then at any point during the second phase, then the simulated view is identical to a real-world random lock for \( b^* \) with key \( K \).

Next, we analyze the properties of \( B \). Notice first that whether or not \( B \) occurred depends only on the simulated key \( b^* \) and the queries of \( R^* \) during the first simulation phase. Moreover, the simulator can determine whether or not \( B \) occurred. We claim that \( \Pr[B] \leq 2^{-2\sigma}. \) Indeed, \( B \) occurs only if there is an \( h \) collision, or if \( (K + h(H)) \cap F \neq \emptyset \). \( h \)-queries are answered with random values in \( \{0,1\}^\sigma \), so for any pair \( x, x' \) of \( h \)-queries, \( \Pr[h(x) = h(x')] \leq 2^{-\sigma}. \)

Using the union bound, the probability that there is an \( h \) collision is at most \( 2^{-\sigma} \cdot |H|^2 \). Moreover, for \( x \in H, x' \in F \) the probability that \( K + h(x) = x' \) is at most \( 2^{-\sigma} \) (because answers to \( h \)-queries are random). Therefore, \( \Pr[(K + h(H)) \cap F \neq \emptyset] \leq 2^{-\sigma} \cdot |H| \cdot |F|. \)

Using the union bound, the probability that either \( (K + h(H)) \cap F \neq \emptyset \), or there is an \( H \) collision is at most

\[
2^{-\sigma} \cdot |H|^2 + 2^{-\sigma} \cdot |H| \cdot |F| = 2^{-\sigma} |H| \cdot (|H| + |F|) \leq 2^{-\sigma} \cdot 2^\sigma \cdot 2^\sigma = 2^{\sigma}. 
\]

Moreover, notice that if \( b^* = b \) then \( \text{Sim} \) does not abort, and also that \( \Pr[\text{Sim aborts}|B] = 0. \)

Finally, we show that conditioned on the event that \( \text{Sim} \) did not abort, then \((\text{Sim}(1\sigma), 1, K) \equiv (\text{View}_{R^*}(1\sigma, b, L_b), K_b)\), where \((L_b, K_b) \in R^{\text{IM}}(1\sigma, b)\). If \( b = b^* \) then the distributions are trivially identical, so we assume that \( b \neq b^* \). Using Claim B.3.4, and since \( \text{Sim} \) did not abort, then at any point during the first phase of the simulation (in particular, when the simulation enters the second phase) the answers in both worlds are randomly distributed. Therefore, we can prove indistinguishability, conditioned on every possible set of answers (that do not cause \( \text{Sim} \) to abort).

When the simulation entered the second phase, \( \text{Sim} \) added \( K + h(G) \) to \( F \), and chose the answers such that they will be consistent with \( b \) (which is possible since \( G \subseteq H \) and \( K + h(H) \cap F = \emptyset \), so for every \( x \in K + h(G) \), \( f(x) \) has not been determined before). In particular, the answers to these queries are random such that they are consistent with \( g, b, K \) (since the answers to \( g \) queries until that point are randomly distributed). The answers to \( h, f \) in the remainder of the simulation were chosen at random, and the answer to a query \( r \) to \( g \) is random such that it is consistent with
exists a locking scheme complete, during the decommitment phase he tosses lock. (locking scheme size $\text{poly}$ he tosses hiding, and keys have length $\text{poly}$ $(\text{poly}$ size $\text{O}$). Corollary B.6. Let $\sigma$ be a security parameter. Then there exists a locking scheme $(S'^{\text{IMS}}, R'^{\text{IMS}})$ for the message space $\{0, 1\}$ that is perfectly-complete, $(1 - 2^{-\sigma})$-binding, $(2^\sigma, 2^{-\sigma})$-hiding, and $(2^\sigma, 2^{-\sigma})$-strongly equivocal. The keys have length $O(\sigma)$, and the locks have size $\text{poly}(2^\sigma)$. Moreover, $R$ has adaptivity 2, and during the decommitment phase he tosses $O(\sigma^2)$ coins, and reads $O(\sigma^2)$ bits from the locking oracle.

Furthermore, for every input length parameter $n \in \mathbb{N}$ there exists a locking scheme $(S'_n^{\text{IMS}}, R'_n^{\text{IMS}})$ for the message space $\{0, 1\}^n$ with $(1 - n \cdot 2^{-\sigma})$-binding, $(2^\sigma, n \cdot 2^{-\sigma})$-hiding, and $(2^\sigma, n \cdot 2^{-\sigma})$-strongly equivocation. The keys have length $O(n\sigma)$, and the locks have size $\text{poly}(n \cdot 2^\sigma)$. Moreover, $R$ has adaptivity 2, and during the decommitment phase he tosses $O(n\sigma^2)$ coins, and reads $O(n\sigma^2)$ bits from the lock.

Let $\sigma$ be a security parameter, $q^*$ be a zero-knowledge parameter, and $(S'^{\text{IMS}}, R'^{\text{IMS}})$ be the locking scheme of [IMS12] with security parameter $\sigma' = 8 \log (q^* \sigma)$. Applying the binding and hiding amplifications (of Sections B.1 and B.2) with parameter $k = \sigma$ gives the following result.

Corollary B.6. Let $\sigma$ be a security parameter, and $q^* \in \mathbb{N}$, then there exists a locking scheme $(S, R)$ for $\{0, 1\}$ with $1 - 2^{-\sigma}$-binding, $(q^*, \frac{1}{2^\sigma} \cdot 2^{-\sigma-1})$-hiding, and $(q^*, \frac{1}{2^\sigma} \cdot 2^{-\sigma-1})$-strongly equivocation, the locks have length $\text{poly}(\sigma, q^*)$, and the keys have size $\text{poly}(\sigma, \log q^*)$. Moreover, $R$ has adaptivity 2, and during the decommitment phase he tosses $O\left(\sigma^2 (\log q^* + \log \sigma)^3\right)$ coins and reads $O\left(\sigma^2 (\log q^* + \log \sigma)^4\right)$ bits from the lock.

Remark B.7. For every input length parameter $n \in \mathbb{N}$, the scheme $(S, R)$ can be extended to a locking scheme $(S_n, R_n)$ for the message space $\{0, 1\}^n$. (This can be achieved from the locking scheme for bit strings by setting the security parameter of the initial scheme to $\sigma' = 8 \log (\sigma n q^*)$). $(S_n, R_n)$ is $1 - 2^{-\sigma}$-binding, $(nq^* \cdot \frac{1}{nq^*}, 2^{-\sigma-1})$-hiding, and $(nq^* \cdot \frac{1}{nq^*}, 2^{-\sigma-1})$-strongly equivocal, the locks have size $\text{poly}(\sigma, n, q^*)$, and the keys have size $\text{poly}(\sigma, n, \log q^*)$. During the decommitment phase, the honest receiver tosses $O\left(n\sigma^2 (\log \sigma + \log n + \log q^*)^3\right)$ random coins, and reads $O\left(n\sigma^2 (\log \sigma + \log n + \log q^*)^4\right)$ bits from the lock.
Bibliography


[Cle86] Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In STOC, pages 364–369, 1986.


275


כפיים נדד דלף שמיפה והיה בשתיות של הtyardו שבניו אופי, ובלוח ייחודי כיהא לתליעתי לא דוד
כתייה, בתﲔה, במדבב שמתאמה למפקד המסקנה את אופי פועלית המועג המтокיר על אוספ כלשהו של
קלטים (שערכם ניבע על פי קלטי המועג המקופל, שهجوم עריסים על חיות מתקדימים ברווח). בכלי
מקופלים זרים בערבית האיבר סוכלים לתוך这个地方 נקנס ראש מוראי מדשה למות בכנה בתיו
העידי באביר קלף שאין.Sdkודד היה. ואת, בעדו בודק והיוו עור למיניהם במקופלים בצורת התוך
מצבות את בכנהי התוינז עשה ומקופל כיול החיה והיה ורודו. בנוסוף, כאית ומשתמשים
בקופים המבוקשים נדד דלף על מנטת לטכניק ריבר הזרה חיד שעושי לפליקר (אך מלבט את מובטח
כפי אשה פועלו בכנה) עשהועבד ומנהיג בתו יקר הבנים מספר שהבכין. ריבר זורה היא יコレ לتضום
מקופלים שלם, ומקופלים בכנה הדרים, אפורי בכנהו קבוצות מקופלים עם מייל שמקופלים דלף
לע הtyardו המתחעם בתוכ ריבר הזרה, ושימושי לטונה ואפ糧ת בכנהי התוינז על ידי שימוש
בקלטים שלד קודדים רואים.
In the last decades, methods of testing the robustness of systems have dramatically increased in complexity. In order to verify evidence, the proximity of codes to their correction has become more significant. In parallel, attacks on “side channel” channels led to significant security breaches, resulting in increasing interest among the cryptographic community in developing secure methods of running protocols. The research dealt with various versions of methods of testing the robustness with the property of “zero knowledge”, and their inclusion in secure cryptographic protocols, and their use for improvement of protocols. Results of the research are relevant to two central topics.

Firstly, we examined various versions of systems of testing the robustness with the property of “zero knowledge”. First, systems of evidence testing, which are systems of evidence that enable ‘‘knowledge-based” verification, i.e., the verification of evidence that is based on a small number of bits from the evidence. Secondly, systems of evidence testing for proximity, which are systems of evidence testing that enable the verification of proximity of any code to its correction with a higher probability. Lastly, codes are tested for proximity within a code “correctable” code, which is a code for the correction of local errors, within the code, which is not known, the code is not known, and the code is not known. However, the code is not known, and the code is not known. The codes and codes within the “zero knowledge” property of systems of evidence testing, systems of evidence testing for proximity, and codes are used for building of protocols, in particular, as compared to the complexity of their implementation for cryptographic purposes, such as checking the distribution of secrets between several players, the distribution of secrets, and the evidence zero-knowledge in the model of the three players in the sense of the complexity of the protocol (as compared to the complexity of the protocol).


תודה

אני מצה ד lagi על התמחות הכפיפות הגניזה בשש להפקודות.
חישוב בטוח בדיקה הסתברותית

 Blessor Ul Machker

לשמם מילוי חלקי של הדרישות לקבלת התואר

דוקטור לפילוסופיה

מור ייס

הווש לוסטט טסנגי - מלון טקנולגיה לישראלי
נוכי המושיע - חפף
2016
希וש בטווח ובذلك הסתברות

מור וייס