An Efficient Non-Interactive Zero-Knowledge Proof System for NP with General Assumptions

Joe Kilian * Erez Petrank †

Abstract

We consider noninteractive zero-knowledge proofs in the shared random string model proposed by Blum, Feldman and Micali [BFM88]. Until recently there was a sizable polynomial gap between the most efficient noninteractive proofs for NP based on general complexity assumptions [FLS90] versus those based on specific algebraic assumptions [Dam92]. Recently, this gap was reduced to a polylogarithmic factor [Kil94]; we further reduce the gap to a constant factor. Our proof system relies on the existence of one-way permutations (or trapdoor permutations for bounded provers).

Our protocol is based on the random committed bit model introduced by Feige, Lapidot and Shamir. We show how to prove that an n-gate circuit is satisfiable, with error probability $1/n^{O(1)}$, using only $O(n \lg n)$ random committed bits. For this error probability, this result matches to within a constant factor the number of committed bits required by the most efficient known *interactive* proof systems.

1 Introduction

A basic issue in cryptography is the tradeoff between resources and security properties. Ordinary zero-knowledge proofs obtain greater security at the price of requiring greater interaction between the prover and the verifier. Fiat and Shamir [FS86] give a heuristic means for eliminating the need for interaction, at a cost of slightly weaker security properties and more importantly the ability (for now) of rigorously analyzing the resulting protocols. Blum, Feldman and Micali [BFM88] propose a more theoretically appealing compromise, known as the *shared random string model* for noninteractive zero-knowledge proofs. In this model, the prover and verifier are given a shared string of bits that are guaranteed to be completely random. In this model, the prover can then send a zero-knowledge proof of a theorem as a single message to the verifier who can then check the proof without further interaction.

Such non-interactive zero-knowledge proofs turned out to be an important cryptographic primitive. For example, it is used in the signature scheme framework of [BG89] and to obtain secure public-key encryption schemes that are robust against chosen message attacks [NY90]. This motivates the questions of what assumptions are necessary for noninteractive zero-knowledge proofs and how efficient can these proofs be made.

1.1 Previous results

Blum, Feldman, and Micali [BFM88] show how to construct a non-interactive zero-knowledge proof system for any language in NP given a very specific number theoretic assumption. The assumption

*NEC Research Institute. E-mail: joe@research.nj.nec.com

†Technion - Israel Institute of Technology, Haifa, Israel. E-mail: erez@cs.technion.ac.il. Most of this work was done while the author was visiting the NEC Research Institute.

was relaxed to quadratic residuosity by De-Santis, Micali, and Persiano [DSMP88]. The original solution required n^4 shared random bits to prove that a n-node graph is 4-colorable, but the efficiency of protocols based on quadratic residuosity has since been recently greatly improved [Dam92, BP94]. Given a cryptographic security parameter k, which must be large, the most efficient of these families of solutions requires $O(k^2 n)$ shared random bits.

Feige, Lapidot, and Shamir [FLS90] gave the first non-interactive zero-knowledge for NP under a general complexity assumption. Their construction requires one way permutations or, for polynomially bounded provers, certified trapdoor permutations (the technical "certification" requirement was eliminated in [BY92]). They also obtained much greater generality in how their proofs could be used (e.g., many provers could prove many theorems using the same random string). To achieve this they introduced the *random committed bit* model; all subsequent progress using general complexity assumptions has used this model. In the random committed bit model, the prover and verifier are dealt a sequence of random committed bits. The prover is allowed to see the contents of these bits and may, as his only action, reveal a subset of these bits to the verifier. The verifier can only look at those bits that have been revealed and decide whether to accept or reject. A protocol for the random committed bit model may then be compiled into a protocol for the shared random string model.

The proof system proposed by [FLS90] was much less efficient than those based on quadratic residuosity: It required $O(k \cdot n^{11/2})$ shared random bits to prove that an n-node graph contains a Hamiltonian cycle. Subsequently, Feige [Fei] has shown how to prove that n by n by n matrix has a 3-dimensional matching, using only $O(k \cdot n^{7/2})$ shared random bits.

Recently, a more efficient proof system for NP was given by Kilian [Kil94] who presented a proof system for circuit satisfiability using $O(k^2 \cdot n \cdot \log^c n)$ shared random bits for some constant c. In addition to the efficiency of this proof, circuit satisfiability seems to be the preferred NP-problem for reducing a general predicate. Kilian's construction is based on random committed bit model, and hence is based on the same complexity assumptions as those of [FLS90].

1.2 Our results

We improve the protocol of [Kil94] and get a non-interactive zero-knowledge proof system for circuit satisfiability which uses only $O(k^2 n)$ shared random bits. Our proof system is significantly simpler than the one in [Kil94]. As in [FLS90]+[BY92],[Kil94] we also rely on the general cryptographic assumption that there exists a one way permutation, or trapdoor permutations if the prover only runs in polynomial time.

In the random committed bit model, we show how to prove circuit satisfiability using $O(n \lg n)$ random committed bits, and achieving an error of $1/n^{O(1)}$. We do not know how to achieve such an error probability in the interactive model without using $\Omega(n \lg n)$ bit commitments. Note however that the compilation technique of [FLS90] requires a significant overhead and hence noninteractive zero-knowledge proofs remain asymptotically less efficient than their interactive counterparts.

1.3 Outline of the paper

In Section 2 we introduce some definitions and technical details that we use in the paper. In Section 3 we give a short introduction to our approach and a top-down preview of the proof system. In Section 4 we give a bottom-up full description of the proof system. In Section 5 we prove that the proof system is valid, and in Section 6 we explain what modifications are needed in the proof system so that the prover can be implemented efficiently.

2 Preliminaries

Let us set the notations and definitions we use. A parameterized ensemble of distributions over a language L is a set $E = \{E(x,k) : x \in L, k \geq 1\}$ of distributions, one for each $x \in L$ and security parameter k. We say that two ensembles of distributions E_1 and E_2 over L are indistinguishable on input x if for any polynomial time machine (tester) T, all constants c and all sufficiently large k it holds that:

$$\left|\text{Prob}_{s \leftarrow E_1(x,k)}[T(s) = 1] - \text{Prob}_{s \leftarrow E_2(x,k)}[T(s) = 1]\right| \leq \frac{1}{k^c}$$

where $s \leftarrow E(x,k)$ stands for s being sampled according to the distribution $E(x,k)$.

Let us now define non-interactive zero-knowledge proof systems.

Definition 2.1 *A* non-interactive zero-knowledge proof system *for a language L with security parameter k consists of a probabilistic proving algorithm P, a probabilistic polynomial time verification algorithm V and a probabilistic polynomial time machine (simulator) M with the following property. Let σ be a random string shared by the prover and verifier, and let $k(|x|)$ be a security parameter. Then:*

1. **Completeness:** *For any $x \in L$ and security parameter k,*

$$\text{Prob}[V(x, \sigma, P(x, \sigma, k)) = \text{accept}] \geq 1 - 2^{-k(|x|)}$$

where the probability space is taken over the random coin tosses of P, V, and over the uniform and independent the choice of the random bits in the shared random string σ. We can in fact obtain perfect completeness in all of our protocols (i.e., V accepts with probability 1).

2. **Soundness:** *For any $x \notin L$, and any (possibly cheating) prover P', and security parameter k,*

$$\text{Prob}[V(x, \sigma, P'(x, \sigma, k), k) = \text{accept}] \leq 2^{-k(|x|)}$$

where the probability space is the same as in the completeness condition.

3. **Zero-knowledge:** *There exists a simulator M that, on input x and security parameter k, produces a distribution space, and the ensemble $\{M(x,k)\}_{x \in L}$ is indistinguishable from the ensemble $\{(\sigma, P(x, \sigma, k))\}_{x \in L}$ which is defined over the distribution space of a random uniformly chosen σ over $\{0,1\}$ and the random choices of the prover P.*

In a reasonable settings, k is greater than $\lg n$ since we want the properties of the protocol to hold also when we run the protocol polynomially many times. We assume that $k \geq \log n$ in the analysis of our protocol. [1]

In this paper, we address the problem of circuit satisfiability. However, in order to simplify the presentation, we are going to reduce this problem to a more restricted satisfiability problem: 3-SAT-5. The problem 3-SAT-5 is defined as follows:

Definition 2.2 The problem 3-SAT-5:

Input: *A 3-CNF formula φ with each variable appearing in at most 5 clauses.*

Question: *Is φ satisfiable?*

[1] However, our protocol may be modified to deal with other settings yielding a protocol which requires $O(n \log n \cdot k^2)$ shared random bits.

It is important that there exists a reduction from circuit satisfiability to 3-SAT-5 which is linear since we present a proof system for 3-SAT-5 with complexity $O(n \cdot k^2)$, where n is the size of the 3-SAT-5 formula, and we claim that this proof can serve for proving circuit satisfiability with complexity $O(n \cdot k^2)$ where n is the size of the circuit. Indeed, one can easily reduce circuit satisfiability to 3-SAT using a linear reduction. Furthermore, the reduction from 3-SAT to 3-SAT-5 can also be done linearly, by creating a new variable for each copy of the original variable and adding clauses which force all copies of the same variable to have equal assignment. Thus, it is sufficient to present a non-interactive zero-knowledge proof system for 3-SAT-5.

3 An overview of the protocol

In Section 4, we describe the protocol in full detail. The description there is a bottom up description of how the proof system is built on top of the random string. In this section, we first discuss at a high level the basic approach of [Kil94], the sources of its inefficiency and our approach to eliminating these inefficiencies. We then give a shorter top-down description of the proof system, which may help going through the details of Section 4.

3.1 Influence Games

The protocol of [Kil94] is based on implementing an *influence game*, a further abstraction from the random committed bits model. In an influence game, the dealer deals the prover and the verifier a sequence of committed pairs of random strings $\{(s_i^1, s_i^2)\}$. However, for each pair (s_i^1, s_i^2) the dealer uniformly chooses a bit $b \in \{0, 1\}$, and makes s_i^b a "wild card" string. The prover can reveal any bit of the "committed" s_i^b as any value he desires.[2] However, the verifier is not told which string is fixed to a random value and which string is wild, and cannot distinguish whether a revealed bit was fixed at a random value or chosen by the prover.

In [Kil94] it is shown how to in prove that circuit satisfiability (via 3-CNF-O(1)) in zero-knowledge using an influence game where each committed string has size $O(\lg n)$, where n is the circuit size. The error probability for this proof is $1/n^c$ where c may be set to any constant (affecting the constant in the size of the committed strings). Thus, only $O(n \lg n)$ committed bits are required.

3.2 Sources of Inefficiency in [Kil94]

The inefficiency in [Kil94] comes from implementing influence games within the random committed bit model. Let ℓ be the length of the strings in the influence game; for the application in [Kil94], $\ell = \Theta(\lg n)$. In [Kil94] string pairs are represented by a committed bit sequence of size $\ell^{\Theta(1)}$ which occurs at random with probability $1/\ell^{\Theta(1)}$. The procedure for "distilling" these representations will occasionally let through invalid representations that could be exploited by a cheating prover. To eliminate this problem, the [Kil94] construction uses $\ell^{\Theta(1)}$ representations, so that most representations will be correct with high probability. Each $\ell^{\Theta(1)}$ factor results in an additional polylogarithmic overhead over the $O(n \lg n)$ basic requirement.

3.3 Our Approach

We obtain our greater efficiency making effective use of constant-size string pairs. We use a much simpler encoding that represents a string pair of size ℓ with $O(\ell)$ committed bits. The probability that a legal encoding occurs in a sequence of random committed bits is only $1/2^{-\Theta(\ell}$, but this is $\Omega(1)$ when $\ell = O(1)$.

[2] The name "influence game" comes from the prover's ability to influence the values of these strings.

Unfortunately, the soundness of the zero-knowledge proof scheme requires that we use an influence game with size-$c \lg n$ for some specific constant c. To obtain such large pairs we use the distillation method of [Kil94] (with different parameter settings) to generate $O(\lg n)$ string pairs each of size $O(1)$. This requires only $O(\lg n)$ random committed bits. We then adapt an expander-based method of [Kil94] to combine these pairs into a single pair whose strings are of size $c' \lg n$, where c' is a slightly larger constant. We use a different and more powerful method of analyzing the result of this combining procedure.

As before, not all of the string pairs we distill are well formed; indeed, under our parameter settings a constant fraction might not be. This prevents us from generating an ideal pair (s_i^1, s_i^2) such that one is completely under the prover's influence and the other is completely outside the prover's influence. Instead, we show that the prover has only a limited amount of influence on one of the two strings. We then argue that this extra influence does not eliminate the soundness of the reduction to influence games.

To analyse what the prover can do to cheat on $x \notin L$, we conceptually imagine that first the prover sends his message M and then the shared random string σ is chosen. We show that for any M the probability that the verifier accepts (M, σ) is extremely small when σ is chosen uniformly. Indeed, this probability remains small even when multiplied by the total number of well-formed messages the prover can send. This implies that with high probability there will be no message that the prover can send to make the verifier falsely accept.

3.4 A top-down description of our protocol

The input to the protocol is a 3-CNF formula φ with variables $x_1, x_2, \ldots, x_n$ and each variable appears in at most 5 clauses. Suppose that so far the prover and verifier have established the following (complex) object out of the shared random string. For each variable x_i, the prover and verifier have a pair of "character" strings (s_i^1, s_i^2). A character is an entity with the following 3 properties:

1. A character can have three possible values: 0, 1, and WC (wild character).
2. A character is given to the verifier in a hidden (committed) form. It is known to the prover and is unknown to the verifier.
3. The prover can reveal the content of the character to the verifier in the following way. If the character has value 0 or 1, then the prover must reveal the correct value of the character. However for a WC character, the prover can reveal either the value 0 or the value 1 (he can choose whichever he wants).

Returning to the strings (s_i^1, s_i^2), we assume that the strings of characters which were established at previous stages for x_i have the following property. One of the strings s_i^1 and s_i^2 is a random string of characters with values 0 or 1, and the other string contains only WC characters. We still have to specify the order in which the strings appear in each pair. Namely, which one of the strings s_i^1 and s_i^2 contains the WC characters and which one does not contain any WC character.

Let τ be a satisfying assignment to φ, then the order of the strings in the pair has been determined according to $\tau(x_i)$. If $\tau(x_i) =$**false** then the string s_i^1 is the WC string, and if $\tau(x_i) =$**true** then the string s_i^2 is the WC string. This summarizes the description of what we assume was already built in previous stages of the protocol. Let us now explain how the prover uses this construction to show that indeed τ satisfies φ.

For each of the five clauses in which x_i is involved we select the following sub-string. if x_i appears negated in a clause C_j, then we select a fifth of the string s_i^1 to represent the variable x_i in the clause C_j, whereas if x_i is positive (not negated) in the clause then we select a quarter of

the string s_i^2 to represent x_i in the clause C_j. The substrings are selected with no overlap, i.e., a character in a string is selected only for one clause. (Here we use the property that the variable appears in at most five clauses, and we selected only a fifth of the given string). Note that the selection is made exactly so that if τ on x_i satisfies the clause then the selected string is a WC string and the prover can open it any way he wants, whereas if the clause is not satisfied by the assignment τ to the variable x_i then x_i is represented by a completely random string over $\{0,1\}$.

After all strings for all clauses have been selected, the prover opens them and the verifier checks that for each clause the three bit strings revealed by the prover sum up to 0. Namely, the bit-wise exclusive-or of the revealed strings is a string of zeros. If τ is indeed a satisfying assignment to φ then the prover can easily pass this test. For each clause, one of the selected strings contain only WC characters and he can open this string to whatever bit string he needs. However, if τ does not satisfy φ then there exists a clause for which all the selected strings are completely random and the prover will be caught with high probability. (The strings are of length $O(k)$.)

So given a pair of strings for each variable with the above characteristics, it is possible to check whether τ satisfies φ. However, we still have to explain how to build a pair of strings for each character. Suppose again that we have already passed a preliminary stage in the protocol and we have already established a long stream of "characters", i.e., hidden characters as described above. These characters are uniformly and independently selected in the set $\{0,1,\mathrm{WC}\}$ and our goal is to gather the (completely random) characters such that for each variable we get a pair of strings: One of them contains no WC characters and the other contains only WC characters.

Loosely speaking, this part of the proof-system goes as follows. First, the stream of characters is gathered into blocks of characters. These blocks contain only a constant number of characters. (We shall have to use $O(k)$ such blocks to get the strings for the variables which are composed of $O(k)$ characters.) Next, the prover selects "good" blocks: Those that either contain only WC characters or those that contain no WC characters. In fact, he selects only pairs of good-blocks that have one WC block and one block that contains no WC characters. The prover then gets rid of the pairs of blocks that are not good by stating which they are and revealing their content. The verifier checks that the prover indeed got rid only of bad pairs of blocks and that the number of good pairs of blocks that the prover found is close to its expected value. This should happen with high probability by the Chernoff Inequality. All the above is done for each variable separately, and for each variable the prover collects $O(k)$ blocks.

Next, the prover selects the order of the blocks in each pair according to the assignment to the corresponding variable. Namely, he can reverse the order of the blocks in each pair by announcing the need to reverse the pair in the proof. Then the verifier makes a test (which we do not describe here) for each variable x_i that all the pairs which correspond to the variable x_i are consistently ordered (polarized). The efficient construction of this test relies on the properties (and existence) of an expander graph.

After the verifier is convinced that the pairs of blocks are consistently ordered, all the pairs are concatenated to get a pair of long strings. One of the strings in the pair contains only WC characters, and the other is random over $\{0,1\}$ as needed.

Of-course, we still have to explain how hidden characters can be constructed from the bits in the shared random tape. We explain this in the full description of the protocol in the following Section.

4 The protocol in detail

We present a non-interactive zero-knowledge protocol for 3-SAT-5. As mentioned in Section 2, this implies the following theorem:

Theorem 1 *Circuit Satisfiability can be proven by a non-interactive zero knowledge proof system whose length is $O(n \cdot k^2)$, where k denotes the security parameter and n is the size of the circuit.*

The compilation procedure of [FLS90] requires that the security parameter k be large, as it must be as large as the security parameter for the one-way permutations or trap-door permutations. In the actual construction for the random committed bit model we can set it to be $O(\lg n)$. We give a sketch of this compilation procedure in Section A.

The Protocol: Denote the input formula by φ and denote the number of variables in φ by n. Recall that in our model, the prover and verifier share a random tape and of-course the input formula. The proof consists of the prover sending one message. The verifier, based on this message (the proof) and the random tape, decides whether to accept the claim that the input formula is satisfiable.

In our proof system (as in other non-interactive zero-knowledge proof systems) the prover's message includes a statement or an interpretation of the structure of the shared random string in a way that several properties are satisfied if and only if the input (φ) is in the language (3-SAT-5). If there were no zero knowledge requirement, the proof would have been simple. The prover would have given the indices in the random string in which the verifier can find a satisfying assignment for the formula φ. Note that if there exists a satisfying assignment, then with very high probability it appears as a substring of a (polynomial-length) random tape. However, our protocol is more involved since we would like the proof to be zero knowledge.

Our proof system is a direct improvement over the protocol in [Kil94] which in turn is inspired by the protocol of [FLS90].

4.1 Interpreting pairs of bits as characters

The prover and verifier first establish *characters* in the random string. A character can have 3 possible values: 0, 1, and a wild-character (WC). Characters are encoded by pairs of committed bits in the following way:
10 - the character 0
01 - the character 1
11 - the character WC.
Of-course the random committed bit stream of the random tape also contains the pair 00, but the pair 00 will be irrelevant to the proof. On one hand, the pairs 00 will not foil the ability of the prover to convince the verifier that φ is satisfiable when indeed it is the case, and on the other hand, these pairs will not help a cheating prover to prove that φ is satisfiable when this is not the case.

This new interpretation of the random tape considers the committed bits of the previous stage in pairs, and gets a string of characters by the above encoding. In order to get rid of the non relevant pairs '00', the prover opens all the commitment of '00' pairs and these pairs are not considered any more during the rest of the proof. This opening of the irrelevant pairs is the first step of the proof.

Now that we have defined what a character is, let us also set a procedure by which the prover can "open" a character. Namely, a prover can show the verifier that a character (which consists of a pair of committed bits) contains a certain value. Throughout the proof, the prover will open a character either to the value 1 or to the value 0. To prove that a character has value 0, the prover will open the commitment of the first bit and show that the (committed) bit is 1. To show that a character is 1, the prover will open the commitment of the second bit in the pair and show that it is 1. Clearly, a wild-character can be open both ways, since both the first and the second bit in

the wild character are 1. We shall make an explicit use of the fact that the character WC can be opened by the prover both to the value 0 and to the value 1.

To summarize our interpretation of the random tape so far, we have a stream of uniformly independently chosen characters of value 0,1, and WC. The prover can open the character 0 and 1 to their correct value, and he can open the character WC to both values 0 or 1 as he wishes.

4.2 Sieving good blocks

In the next step of the interpretation of the random string we consider blocks of α consecutive characters (where α is a constant to be determined later). In fact, we consider the given stream of characters as a sequence of pairs of blocks. We call a pair of blocks *good* if one of the blocks in the pair contains only WC characters and the other block does not contain any WC character. For each variable x_i, $1 \leq i \leq n$, the prover initially considers βk pairs of blocks (for a constant β to be determined later, and the security parameter k) and lets only the good blocks prevail for the rest of the proof.

In the second part of the proof the prover completely opens each character in every block that is not good. These blocks are not used again in the proof. The prover not only opens these characters (as we defined an opening of a character by opening one of its committed bits) but also opens the two committed bits in each of the characters, so that the verifier can check that the discarded blocks are indeed not good.

First part of the proof: *The prover opens all committed bits in all pairs of blocks which are not good.*

The verifier verifies that all blocks that were opened were indeed not good. Otherwise he stops immediately and rejects. Also, if any of the characters opened is irrelevant (contains the pair 00) the verifier stops and rejects. Last, the verifier checks for each variable x_i that the number of pairs of blocks, ℓ_i, that remained, i.e., the number of pairs of blocks that were claimed to be good for x_i (these are still hidden) is close to its expected value. If ℓ_i is not in the range $[t_1, t_2]$ to be specified below, then the verifier stops immediately and rejects.

Let us compute the expected value of ℓ_i and determine the bounds t_1 and t_2. The probability that a random pair of blocks is good is $2 \cdot (1/3)^\alpha \cdot (2/3)^\alpha$ (choose which member of the pair is the WC block, then it has probability $(1/3)^\alpha$ to consist only of WC characters, and the other block has probability $(2/3)^\alpha$ to contain no WC character inside). Therefore, the number of good blocks between the βk random blocks initially considered for the variable x_i is a random variable ℓ_i that has expected value $2^{\alpha+1} \cdot 3^{-2\alpha} \cdot \beta \cdot k$.

By the Chernoff bound, the deviation of ℓ_i from its expected value is big with low probability. Specifically,

$$\text{Prob}\left[\left| \ell_i \; - \; 2^{\alpha+1} \cdot 3^{-2\alpha} \cdot \beta k \right| \; \geq \; \gamma \beta k \right] \; \leq \; 2 \cdot 2^{-k\beta \cdot \gamma^2 \cdot 2^{-\alpha-2} \cdot 3^{2\alpha}}. \tag{1}$$

The parameter γ is a constant which determines the tightness of the bound. We shall set its value later. Now set the deviation bounds that the verifier insists on as:

$$t_1 \stackrel{\text{def}}{=} \left(2^{\alpha+1} \cdot 3^{-2\alpha} - \gamma\right) \beta k$$

$$t_2 \stackrel{\text{def}}{=} \left(2^{\alpha+1} \cdot 3^{-2\alpha} + \gamma\right) \beta k$$

By the Chernoff bound, if we set the constant γ to:

$$\gamma \; = \; \sqrt{\frac{2 \cdot 11 \cdot 2^{\alpha+1} \cdot 3^{-2\alpha}}{\beta}} \tag{2}$$

then the random variable ℓ_i is between t_1 and t_2 with probability at least $1 - 2^{-10k}$ for a random stream of characters.

To summarize this step, for each variable x_i, the (honest) prover and verifier are left with ℓ_i good pairs of blocks, each consisting of one block of WC characters and the other block contains no WC character in the block. The number ℓ_i is at least t_1 and at most t_2. A non-honest prover may claim that some non-good pair of blocks are good, but with probability at least $1 - 2^{-10k}$ the number of pairs of blocks he can cheat on is at most $t_2 - t_1$ (assuming the worst case in which there are only t_1 good pair of blocks but the prover claims that there are t_2).

Thus we end this step having a set of good pairs of blocks associated with each variable. Next, we begin to associate the shared random string (which is now a random stream of good pair of blocks) with a satisfying assignment of the formula φ.

4.3 Setting the polarity of the pairs of blocks according to the satisfying assignment to φ

Each pair of blocks contains a block of WC characters and a (random) block without WC characters. The order of the blocks in each pair is random. In the third part of the proof, the prover is going to set order in all these pairs according to some satisfying assignment to the variables in the formula φ. Let τ denote such an assignment. The prover is going to set the order of all the pair of blocks that correspond to the variable x_i such that if $\tau(x_i) =$**false**, then the first block in each pair of blocks that correspond to x_i is set to be the WC block and if $\tau(x_i) =$**true**, then the second block in each pair of blocks that correspond to x_i is set to be the WC block. The setting of order in the pairs of blocks is the third part of the proof.

Second part of the proof: *For each pair of blocks (got from the shared random tape), which was not dropped in the previous step of the proof, the prover specifies a bit. If a pair is assigned the bit 0 then its order is kept, whereas, if a pair of blocks is assigned 1, then the order of the blocks in the pair is reversed for the rest of the proof.*

The verifier does not check anything, but only reverses the order of the blocks where necessary.

To summarize this step, the honest prover has now pairs of blocks such that for each variable, all pairs are ordered (polarized) in the same manner, and this manner corresponds to the assignment of the variable associated with the pair.

In the following step, we are going to check that the pairs of blocks are indeed polarized consistently. This test is not intended to check whether this polarization matches a satisfying assignment to the formula φ.

4.4 Checking the consistency of the polarization

We now describe a consistency test for the pairs of blocks associated with a variable x_i. This test is repeated for each of the variables. Recall that we have ℓ_i pairs of blocks which are supposed to be polarized. The test consists of many basic tests, each basic consistency test is performed on a couple of pairs. We have to specify what the basic test is and also which pairs are going to be tested against each other. Let us begin with the second.

We consider an expander graph with ℓ_i vertices. Each vertex correspond to a pair of blocks, and the edges, as appearing in the expander, determine which pairs are going to be checked one against the other (through the basic test). In the sequel, we denote the degree of the expander by d and the expansion rate by $1 + \delta$. Namely, each subset A of the vertices of cardinality at most $|A| \leq \frac{\ell_i}{4}$ has at least $\delta \cdot |A|$ neighbors which are not in A.

Next we describe the basic consistency test as applied on two pairs of blocks. Let these two pairs be number j and k, and denote by (B_j^1, B_j^2) and (B_k^1, B_k^2) the blocks in the pair. Recall that

the requirement to make this basic test originates from the existence of an edge $e = (j,k)$ in the expander. We are going to use $\frac{\alpha}{2d}$ unique characters of each of the four blocks involved in this test. Denote the character-strings which are used by the basic test as $(s_j^1(e), s_j^2(e))$ and $(s_k^1(e), s_k^2(e))$, where $s_q^p(e)$, $(1 \leq p \leq 2$ and $q \in \{j,k\})$ is the substring of B_q^p which is devoted for the basic test specified by the edge e (we never use the same character for two different tests). Note that if the prover is honest and the blocks are good and well polarized, then either $s_k^1(e)$ and $s_j^1(e)$ contain only wild characters and $s_k^2(e)$ and $s_j^2(e)$ are random over $\{0,1\}$, or $s_k^1(e)$ and $s_j^1(e)$ are are random over $\{0,1\}$ and $s_k^2(e)$ and $s_j^2(e)$ contain only wild characters. In this case, the prover can open the values of these characters such that

$$s_k^1(e) \oplus s_j^2(e) = s_j^1(e) \oplus s_k^2(e) = 0^{\alpha/2d}$$

since the prover can open the WC characters to whatever value he wants (the operation "$\oplus$" denotes bit-wise exclusive-or). We define this as the basic consistency test.

Third part of the proof: *The prover opens half of the characters in each of the blocks such that all the basic consistency tests associated with all the expander edges hold.*

The verifier verifies that all the characters which should have been opened were indeed opened, and that for each edge of each expander, the consistency test is satisfied.

To summarize, in this step we "lost" half of the characters which we cannot use in the sequel, but we gained some assurance that there is some consistency in the polarity of the pairs. We shall analyze the degree of this assurance in the analysis of the protocol (See Section 5).

For the next stage, we do the following concatenation process. For each of the variable, we concatenate all the first blocks in all its corresponding pairs to a single string, and concatenate all the second blocks in all its pairs to another string. So for each variable we have two strings, and if the prover is honest then one of these strings is a string of WC characters only and the other string is a random string over $\{0,1\}$. The length of each string is $\ell_i \cdot \alpha/2$. To make all strings (of all variables) have the same length, we truncate them to $t_1 \cdot \alpha/2$ (recall that $\ell_i \geq t_1$ or the verifier has already rejected).

4.5 Last step: showing that τ satisfies φ

In this last step, the prover uses the polarization of the strings, which should represent τ, to show that the formula φ is satisfiable. If the prover behaves according to the protocol it should hold that for each variable x_i, $1 \leq i \leq n$, the prover and verifier share a pair of strings of (hidden) characters. One of the strings consists of WC characters only, and the other string is a random string over $\{0,1\}$ which contains no wild characters. Also, the WC string is the first in the pair if $\tau(x_i) =$**false** and the second in the pair if $\tau(x_i) =$**true**. Recall that the prover has the ability to open all the characters in the WC string either to 0 or to 1 as he wishes, but the other string is fixed.

Now, for each clause, we make a test. Loosely speaking, for each variable in the clause we take part of its string (of characters) so that if the variable satisfies the clause (i.e., it is assigned **true** and appears positively in the clause or it is assigned **false** and appears with a negation in the clause) then the prover can completely control the opening of the string that correspond to the variable. If τ assigns x_i a value that does not help to satisfy the clause, then the prover can open the corresponding string only in one way which corresponds to a random string over $\{0,1\}$. After selecting the 3 strings that correspond to the clause, the prover has to open the strings such that their bit-wise exclusive-or equals the zero constant. So if the prover can control one of the strings (or equivalently, τ makes one of the literals in the clause getting the value **true**) then he can easily pass this test. Otherwise, the prover has to open 3 random strings, and their exclusive-or is zero with small probability.

More formally, recall that each variable appears in at most 5 clauses. So we partition the pair of strings of each variable to 5 pairs of strings (each string in each of the 5 pairs is now of size $\alpha \cdot t_1/10$) and we use one of the pairs for each appearance of the variable in a clause. We stress that the polarity of the strings remains. Namely, the pair is divided into 5 sub-pairs where the first string in each sub-pair is a substring of the first string in the original pair, and the second string in each sub-pair is taken from the second string in the original pair. Also, no overlapping is allowed. So throughout the proof the prover never uses the same character twice.

Let a clause contain the variables x_i, x_j, and x_k. ¿From the pair of sub-strings that each of the variables contribute to this clause, we select the first sub-string if the variable appears negated in the clause and the second sub-string if the variable appears positive in the clause. Thus, we select for each clause 3 strings of characters of length $\alpha \cdot t_1/10$. Intuitively, if the assignment τ satisfies one of the literals in the clause than the string that correspond to this literal contains only WC characters.

After selecting the 3 strings of the clause, the prover opens all the characters in all of these strings so that the bit-wise exclusive-or of the strings equals $0^{\alpha \cdot t_1/10}$.

In case the prover has more control over the opening of these characters (e.g., if more then one variable is assigned **true** in the clause) then the prover uses his degrees of freedom randomly. Namely, he opens all WC characters at random except the ones which have to be fixed so that the test is satisfied. This random selection is crucial for the zero knowledge property.

Fourth part of the proof: *The prover opens all the remaining characters so that for each clause, its three corresponding strings have a bit-wise exclusive-or equal* $0^{\alpha \cdot t_1/10}$. *The proof ends.*

The verifier checks that the test for each clause indeed passes, and that all characters were indeed opened legitimately. In this case, he accepts. Otherwise, he rejects. The verification process ends.

5 Analysis of the protocol

5.1 Completeness

This is the easy part of the proof. It should be clear that the prover can perform all his tasks when the input formula φ is indeed satisfiable. He will fail only when the Chernoff inequality does not hold (see Equation (1)), namely, the number of good blocks ℓ_i, for some variable x_i, is not in the range between t_1 and t_2. This happens with probability at most $n \cdot 2^{-10k}$, and since $n \leq 2^k$ (for reasonable settings, see Section 2) we get that this happens with probability at most 2^{-9k}.

5.2 Soundness

Let us analyze what is the probability that the prover can produce a convincing proof (on a random string) for a non-satisfiable formula φ. The prover convinces the verifier if:

1. For each variable x_i, the number of good blocks ℓ_i satisfies $t_1 \leq \ell_i \leq t_2$.
2. All polarity tests hold.
3. All clause tests hold.

The first condition does not depend on the input formula. The probability that the first condition hold is at least $1 - 2^{-9k}$. We assume (in a worst case manner) that if the first condition doesn't hold, than strange things happen and the prover can convince the verifier always. This happens with probability at most 2^{-9k}. Conditioning on the event that the first condition hold, We compute the probability that Conditions (2) and (3) hold. We show that this happens with (conditional)

probability at most 2^{-9k}. Thus, the probability that the prover can convince the verifier is at most $2 \cdot 2^{-9k}$. Of-course, we can reduce the upper bound on this probability to 2^{-ck} for any constant $c > 0$ increasing the number of bit commitment used (The parameters α, β, and γ used in the protocol).

So assume now that Condition (1) holds. We first note that this conditioning does not foil one important property which holds in the unconditional space. The non-WC characters in the good block pairs are uniform and independent over $\{0,1\}$. Namely, we note that the number of good pairs of blocks is independent of the specific content of the non-WC characters in the good pairs of blocks. This property is essential for the rest of the proof.

Let us fix the following worst case scenario for the rest of this proof. Suppose there are t_1 good pair of blocks for each of the variables, but the prover claims that there are t_2 such pairs. Also, in all the non-good pairs of blocks which the prover announces as good (false pairs of blocks), all the characters are WC. Namely, the prover has complete control over these pairs and he can open each of the characters there to whatever he wants. Moreover, we assume that the prover can decide the places of the $t_2 - t_1$ false pairs of blocks in between the t_2 overall pairs of blocks. However, note that in each of the t_1 good pairs, even in this worst case analysis, we still assume that the block that does not contain WC characters is a uniformly and independently chosen string in $\{0,1\}^{\alpha}$.

Next, we define an assignment τ to the variables of the formula φ. We shall show that if this assignment does not satisfy φ (as must be the case here since we assume that φ is not satisfiable) then Conditions (2) and (3) hold with small probability. Consider the good pairs of blocks after being polarized polarized in the third part of the (non-interactive) proof. For any variable x_i consider the majority of the polarization in the t_1 good block-pairs which correspond to x_i. Since the prover is not necessarily honest, some of these pairs may have their WC-block as the first block and some of the pairs may have their WC block as the second block. We define $\tau(x_i)$ to be **false** if in the majority of the pairs the first block in the pair is the WC block, and we define $\tau(x_i)$ to be **true** if in the majority of the pairs the second block is the WC block.

Each variable has a degree of consistency with this assignment. Define the consistency of the variable x_i to be the fraction of the good pairs that have polarization equal to the majority polarization of the variable (by which $\tau(x_i)$ is determined). Clearly, each variable has consistency at least $1/2$. We are going to partition the analysis into two cases. One possible case is that one of the variables has low consistency, and then we will show that the consistency test of the polarization (Part (4) of the proof) passes with low probability. The second possibility is that all variables are close to being consistent. In this case, we recall that τ cannot satisfy all clauses (since φ is not satisfiable) and we show that the prover can pass the test of a clause which is not satisfied by τ with low probability.

Formally, we partition the analysis into two cases:

1. There exists a variable whose consistency is less then $9/10$.
2. For each variable in the formula, its consistency is at least $9/10$.

We are going to show that whichever of the above is valid, the prover succeeds in convincing the verifier with probability at most 2^{-10k}.

5.2.1 A probabilistic argument

For each of the above possibilities, we use the following method by which we prove that the prover fails with high probability. We define a randomized prover which chooses its cheating strategy at random, and compute the probability that the proof which the randomized prover produces passes the relevant tests. Next, we show that even if the prover chooses its best strategy (rather then

selecting his strategy at random), the probability that he manages to convince the verifier still remains low.

5.2.2 Case I: There exists an inconsistent variable

Assume that there exists a variable x_i $(1 \leq i \leq n)$ whose consistency is lower then 9/10 and let us calculate the probability that the prover has a winning strategy for Part (4) of the proof (the consistency test of the polarization). As mentioned, we begin by calculating the probability that the prover can pass the consistency test of Step (4) when he chooses his strategy at random. Recall that we are assuming a worst case scenario in which there are t_1 good pairs of blocks for x_i and there are also $t_2 - t_1$ pairs of blocks that were declared good by the prover but contain, in fact, only WC characters.

In the polarization consistency test the prover has to open half of the characters in each block. The randomized prover opens all the non-WC characters to 0 or 1 as he must, and he opens all the WC characters in the best way, so that the test that they are engaged in is satisfied. (Recall that each character is engaged in at most one test.) However, the prover still has two things to choose. First, he can determine where to put the $t_2 - t_1$ false pairs of blocks amongst the t_2 pairs. Second, he can decide which pairs are inconsistently polarized. We let the randomized prover pick these at random. Namely, he picks at random $t_2 - t_1$ places for the false pairs between the t_2 pairs, and he picks $t_1/10$ (or more) pairs that will be polarized inconsistently (with the polarity of the majority of the pairs).

What is the probability that the prover passes the consistency test? Recall that each basic consistency test is performed on two pairs. If the pairs are polarized in the same manner then the test is easily passed by the prover opening the wild characters properly. Also, if anything is tested against a false pair then the prover easily passes the test since it has one of the pairs consisting of wild characters only. However, when two good pairs of opposite polarization are tested against each other, then each of the non-WC blocks in both pairs contributes a completely random string of $\alpha/2d$ bits and the two strings must be equal. This happens with probability exactly $2^{-\alpha/2d}$.

Note a delicate point in our claim that the strings compared are random. Indeed the contents of the strings were chosen uniformly at random, but it is known in advance which pairs are tested against each other and the prover may use his advantage in setting the *places* of the false pairs or deciding *which pairs* are inconsistently polarized in order to set specific pairs against each other and enhance the probability of the good pairs to pass the test. Nevertheless, this is not the case here. We are considering now the randomized prover who chooses the places of the false pairs and the good pairs that are inconsistently polarized at random, and thus the good pairs relate to one another in a completely independent and random manner.

Now let's compute how many oppositely polarized good pairs are tested against each other. By our assumption, there are at least $t_1/10$ good pairs which are not polarized consistently with the majority of the pairs. By the expansion property of the expander graph we use, these pairs have at least $t_1 \cdot \delta/10$ neighbors that are either consistently polarized with the majority or they are false pairs. Since the number of false pairs is at most $t_2 - t_1$, we get that at least $m \stackrel{\text{def}}{=} t_1 \cdot \delta/10 - (t_2 - t_1)$ tests are made between pairs that are oppositely polarized. Since we do not recycle characters, all these tests are completely independent, and the test is passed with probability $2^{-\alpha m/2d}$.

Recall that

$$t_1 = \left(2^{\alpha+1} \cdot 3^{-2\alpha} - \gamma\right) \beta k$$

and

$$t_2 = \left(2^{\alpha+1} \cdot 3^{-2\alpha} + \gamma\right) \beta k.$$

Thus,

$$m = k\beta\left(\frac{\delta}{10}\cdot 2^{\alpha+1}\cdot 3^{-2\alpha} - \gamma(2+\frac{\delta}{10})\right)$$

Setting $\beta = \frac{2\cdot 11\cdot 60^2}{2^{\alpha+1}\cdot 3^{-2\alpha}\cdot\delta^2}$, we get (see Equation (2)) $\gamma = \frac{2^{\alpha+1}\cdot 3^{-2\alpha}\cdot\delta}{60}$, and since γ is that small, we get:

$$m \geq k\beta\frac{\delta}{20}\cdot 2^{\alpha+1}\cdot 3^{-2\alpha}.$$

To summarize, the probability that the randomized prover passes this test is at most

$$2^{-\frac{\alpha m}{2d}} \leq 2^{-k\cdot\frac{\beta\alpha\delta}{40d}\cdot 2^{\alpha+1}\cdot 3^{-2\alpha}}.$$

Returning to the real prover. The randomized prover chooses between its strategies at random. The choice is made between the $\binom{t_2}{t_1}$ possibilities to fix the places of the false pairs, and between choosing the inconsistently polarized pairs. The number of options for this second choice can be bounded by 2^{t_1}. Also, the randomized prover passes the test with probability at most $2^{-k\cdot\frac{\beta\alpha\delta}{40d}\cdot 2^{\alpha+1}\cdot 3^{-2\alpha}}$. By simple counting arguments, the best strategy of the prover passes this test with probability at most

$$\begin{aligned}\binom{t_2}{t_1}\cdot 2^{t_1}\cdot 2^{-k\cdot\frac{\beta\alpha\delta}{40d}\cdot 2^{\alpha+1}\cdot 3^{-2\alpha}} &\leq 2^{t_1+t_2}\cdot 2^{-k\cdot\frac{\beta\alpha\delta}{40d}\cdot 2^{\alpha+1}\cdot 3^{-2\alpha}} \\ &= 2^{k\beta 2^{\alpha+1}\cdot 3^{-2\alpha}\cdot(2-\frac{\delta\alpha}{40d})}\end{aligned}$$

Setting $\alpha = \frac{120d}{\delta}$ and since $\beta \geq \frac{10}{2^{\alpha+1}\cdot 3^{-2\alpha}}$, we get that the prover passes the test with probability at most 2^{-10k}.

5.2.3 Case II: All variables are almost consistent

Assume that the consistency of all variables is at least 9/10. Again, we consider the randomized prover that acts as before. I.e., selects at random the places of the $t_2 - t_1$ false pairs of blocks in between the t_2 blocks, and selects at random which good pairs will have an inconsistent polarization. In all other senses (i.e., opening WC characters) the prover always makes his best choice in order to pass all tests.

Recall that we have defined an assignment τ that corresponds to the majority of the polarities in each variable. Since the input formula is not satisfiable (we are analyzing the soundness of the protocol) then there exists a clause that τ does not satisfy. Let the literals in this clause be y, z, and w. (Each of them is either a variable of a negation of a variable). Each variable contributes a string of characters s_x, s_y, and s_z of length $\frac{\alpha}{10}$. Since τ assigns each of the literals the value **false** and the literals are 9/10 consistent, then we can deduce the following on each one of the strings.

Each of the strings s_x, s_y, and s_z is composed of at least $9t_1/10$ random blocks (i.e., $\frac{9}{10}\cdot t_1\cdot\frac{\alpha}{10}$ random characters), at most $t_1/10$ WC blocks that originate from inconsistently polarized good blocks and from $t_2 - t_1$ WC blocks that originate from false pairs of blocks. Also, since we have truncated the strings from length t_2 blocks to t_1 blocks (in order to have the same length for all strings) then we assume that all the $t_2 - t_1$ blocks that were truncated were good. To summarize, we have in each string $(9t_1/10 - (t_2 - t_1))\frac{\alpha}{10}$ random characters, and $(t_1/10 + (t_2 - t_1))\frac{\alpha}{10}$ WC characters.

Note again that the claim on randomness is based on the prover not selecting the places of the false pairs and the inconsistently polarized good-pairs at random.

Recall that for the test of the clause the prover has to open all characters such that the bitwise exclusive or of the strings s_x, s_y, and s_z equals $0^{t_1 \cdot \frac{\alpha}{10}}$. What is the probability that the randomized prover passes this test? The best possible arrangement of the WC characters in the three strings s_x, s_y, and s_z is when their indices don't overlap, and then the prover has control over $3(t_1/10+(t_2-t_1))\frac{\alpha}{10}$ characters. Still, all the rest of the characters are completely random and sum up to 0 with probability 1/2. Since all these characters are completely random and independent, the probability that the randomized prover passes this test is at most

$$2^{-(7t_1/10-3(t_2-t_1))\frac{\alpha}{10}}$$

Again, we return to the real prover. The number of possible strategies from which the randomized prover has chosen at random is bounded by $\left(2^{t_1+t_2}\right)^3$ (as in Case (I) but for three variables). So if the prover chooses the best strategy instead of picking a strategy at random, the probability that he passes the test, p, satisfies:

$$\begin{aligned} p &\leq 2^{3(t_1+t_2)-\frac{7}{10}\cdot t_1\cdot\frac{\alpha}{10}+3(t_2-t_1)\cdot\frac{\alpha}{10}} \\ &= 2^{k\beta\cdot\left(6\cdot 2^{\alpha+1}\cdot 3^{-2\alpha}-\frac{7}{10}\cdot(2^{\alpha+1}\cdot 3^{-2\alpha}-\gamma)\cdot\frac{\alpha}{10}+6\cdot\gamma\cdot\frac{\alpha}{10}\right)} \\ &= 2^{k\beta\cdot\left(6\cdot 2^{\alpha+1}\cdot 3^{-2\alpha}-\frac{\alpha}{10}\cdot(\frac{7}{10}2^{\alpha+1}\cdot 3^{-2\alpha}-6.7\gamma)\right)} \end{aligned}$$

Since $6.7\gamma \leq \frac{2}{10}2^{\alpha+1}\cdot 3^{-2\alpha}$ we get that

$$p \leq 2^{k\beta\cdot 2^{\alpha+1}\cdot 3^{-2\alpha}\cdot\left(6-\frac{\alpha}{10}\cdot\frac{1}{2}\right)}$$

Since $\alpha \geq 140$ and since $\beta \geq \frac{10}{2^{\alpha+1}\cdot 3^{-2\alpha}}$ we get that the probability that the prover passes this test even when he chooses his best strategy is at most 2^{-10k}.

5.2.4 Combining both cases

In both cases the prover is caught with high probability. If the variables are not polarized consistently then he is caught at the consistency test with probability $1-2^{-10k}$ and if he sets the polarization of the variables almost consistent, then he is caught in at least one of the clause tests with probability at least $1-2^{-10k}$. Combining both, we conclude that the prover is caught with probability at least $1-2^{-10k}$. Also, our analysis was conditioned on the event that the conditions given in the Chernoff inequality hold (see Equation (1)) and by the setting of the parameters this happens with probability 2^{-9k}. Therefore, the overall probability that the prover manages to convince the verifier on an input φ which is not satisfiable is at most $2\cdot 2^{-9k}$.

One may note that setting the parameters α, β, and γ appropriately enables us to lower the upper bound on this probability to 2^{-ck} for any constant $c>0$.

5.3 Zero knowledge

In order to prove the zero knowledge property of the protocol, we have to show that there exists a probabilistic polynomial time simulator M whose output distribution on $\varphi \in$3-SAT-5 is indistinguishable from the distribution $(\sigma, P(\sigma,\varphi))$ where the shared random tape σ is uniformly chosen in $\{0,1\}^{\alpha\beta k^2 n}$. Note that for the honest prover when φ is indeed satisfiable, the shared random tape σ completely determines the proof.

The way to produce this pair, i.e., shared random tape with the corresponding proof can be done in the following way. First, we note that we can efficiently produce a random commitment for any given bit. For example, in the specific manner we use bit commitment, we have to produce a random string x such that $B(x)=b$, and then operate the one way permutation P on x. For any given $b\in\{0,1\}$ this can be done efficiently.

In this construction, we assume that it is impossible to distinguish efficiently between a commitment on 0 and a commitment on 1. (This is the general requirement of a commitment scheme, and in our particular scheme this is guaranteed by the hard-coredness of B.) In fact, the shared random string output by the simulator will be different from the original shared random string in the sense that the original random string will contain commitments on random bits indeed, but the shared random string output by the simulation will contain specific places where the commitment will be only on the bit 1.

The simulator begins by producing a real random string of committed bits which are known to the simulator. Next, the simulator follows the prover strategy as far as it can. Namely, it produces characters, throws away the irrelevant '00' pairs, gather the characters into blocks and opens all the bad blocks. At this stage (before the third part of the proof), the simulator replaces all the characters in the good blocks by WC characters. The simulator does that by producing random commitments on the bit '1' and replacing the original commitments in these places by the new commitments. We stress that the shared random tape is indistinguishable from a real random tape since the bit commitments are hidden for an efficient machine.

After substituting the commitments, the simulator polarizes the blocks randomly. Namely, for each remaining block, a bit is chosen uniformly and independently and these bits are sent in the third part of the proof. The remaining pairs of blocks are subsequently used in the order specified by this random polarization.

Next, the simulator passes all the required tests by opening the relevant characters appropriately. Note that each test is made of 2 or 3 characters of which at least one must be a WC character and the rest can be completely random. The opening of the WC character in the original proof is done so that it matches the other predetermined random characters (over $\{0,1\}$). In the simulation the simulator produces the same distribution on the opened characters. Namely, all the opened characters are uniformly chosen in $\{0,1\}$ except for the characters that must be fixed to pass the test. Note that it wouldn't be good to always open a pair to 0-1 when the exclusive or of these bits has to equal 1, since in the original proof the pair 0-1 has the same probability as 1-0. Since no character is used twice, the distribution of the proof output by the simulation (on satisfiable formulas) is identical to the distribution of the real proof.

To summarize, two points are important in the simulation. First the shared random string output by the simulation is polynomially indistinguishable from a truly random string, and second, all the characters that are opened during the proof are uniformly chosen in $\{0,1\}$ except for the restrictions imposed by the tests. Therefore, the output of the simulation is indistinguishable from the original proof.

5.4 A remark on perfect completeness

It is desired that a proof system will have a perfect completeness property. Namely, that if the input formula is indeed satisfiable then the prover will always be able to prove that this is the case. However, in our protocol even if the input formula φ is satisfiable, the prover may still not be able to prove it since the conditions stated in the Chernoff bound do not always hold for the shared random string of the proof.

To fix the protocol to have perfect completeness, we allow the prover to show that this is the case (by opening all the characters of the shared random string and showing that the number of good blocks is not within the desired bound). In this case the verifier accepts (although he gets no statement about the input formula). Using this augmented protocol, we gain perfect completeness and the soundness remains unchanged since we have assumed that in this rare case, the prover always succeeds in convincing the verifier.

5.5 Complexity

Our proof uses $O(k)$ committed bits for each variable, i.e., $O(k^2)$ random bits for each variable. Thus the overall number of committed bits we use is $O(nk)$ and the length of the shared random tape we use is $O(k^2 \cdot n)$.

In this extended abstract, we have not tried to push the constants down, and so our constants are in fact huge. In the final version of this paper, we intend to keep the constants as low as we can. We believe that the problem of designing a protocol with more efficient constants is an interesting open question.

6 Efficient provers

Our proof system can also be implemented by an efficient prover (who has an auxiliary input containing a witness for the NP problem). However, a few changes have to be made which affect the cryptographic assumption we use.

We follow [FLS90, Kil94] by assuming the existence (and specifically by using) a *family of certified trap-door permutations*. Loosely speaking, a family $\mathcal{F}$ of trap-door permutations is a set of permutations $\left\{f_i \ : \ \{0,1\}^k \to \{0,1\}^k\right\}_{i \in I}$ such that: $f_i(x)$ can be efficiently computed given (i,x), it is hard to compute $f_i^{-1}(y)$ given (i,y), for each i there exists a *trapdoor* $t(i)$ such that it is possible to compute efficiently $f_i^{-1}(y)$ given $(i,y,t(i))$, and there exists an efficient algorithm that given the security parameter k, samples $i \in I \cap \{0,1\}^k$ uniformly together with $t(i)$, i.e., there is an efficient sampler that samples $(i,t(i))$ such that i is uniformly chosen in $I \cap \{0,1\}^k$.

The additional "certification" property is that given i, it is possible to verify efficiently that $i \in I$, i.e., $f_i \in \mathcal{F}$. Specifically, if f_i is in $\mathcal{F}$ then we know that f_i is a permutation.

Our proof system for an efficient prover involves an additional preliminary step. Note that the only use we made of the power of the prover is in inverting the one way permutation while interpreting the shared random string in the beginning of the proof.

In order to remove this need of powerful computation, we let the prover choose at random $(i,t(i))$ such that $i \in I \cap \{0,1\}^k$ and the whole proof is made on the one way permutation f_i. So at the beginning of the proof, the prover states the index i he is going to use and the verifier verifies that indeed $i \in I$. Note that it is important to verify this since f_i must be a permutation for the soundness of the proof system. After that, the proof system is the same and the efficient prover can invert f_i (using the trapdoor $t(i)$) whenever needed.

The completeness, and the zero knowledge analysis remains the same, but the soundness property has to be computed again, since the prover gains a new power: He can interpret the shared random string in $\left|I \cap \{0,1\}^k\right|$ different manners (according to the choice of $f_i \in \mathcal{F}$) and perhaps he can choose f_i that most helps him to cheat on a non-satisfiable input formula φ.

We treat the soundness analysis in the same manner as we did in the original soundness analysis. Namely, first we analyze what happens when the prover really chooses uniformly $f_i \in \mathcal{F}$ and then we compute the advantage he may get when picking the best possible $f_i \in \mathcal{F}$. So when f_i is picked at random, the probability that the prover succeeds in cheating the verifier is the same as computed in Section 5.2. Therefore, the probability that the prover succeeds in cheating when he can pick $f_i \in \mathcal{F}$ that best serves his purpose, is at most

$$\begin{aligned} 2 \cdot 2^{-9k} \cdot \left|I \cap \{0,1\}^k\right| &\leq 2 \cdot 2^{-9k} \cdot 2^k \\ &\leq 2^{-k} \end{aligned}$$

as needed.

To summarize, in order to run the protocol with an efficient prover, we use a family $\mathcal{F}$ of certified trap door permutations, we add a preliminary stage in which the prover chooses and states a member $f_i \in \mathcal{F}$, and then the prover and verifier interpret the shared random string according to the permutation f_i in the rest of the proof.

6.1 Getting rid of the certification requirement

The need for certification in this setting was first noted by [BY92]. They also suggested a way to get rid of this requirement. Loosely speaking, they replace the requirement for efficient verification that f_i is a permutation by a proof (of the prover) that f_i is "close" to a permutation. The idea of the proof is that the prover has to invert random strings in $\{0,1\}^k$. In our setting, after the prover selects $i \in I \cap \{0,1\}^k$, he uses a predetermined part of the shared random string to demonstrate the permutation property of f_i. This part of the random string is partitioned into blocks of k bits and for each block $u \in \{0,1\}^k$, the prover specifies $f_i^{-1}(u)$, i.e. the inverse of the block u under the mapping f_i.

If an ϵ fraction of the strings in $\{0,1\}^k$ do not have preimages and the prover tries to invert $10k \cdot \epsilon^{-1}$ strings in $\{0,1\}^k$, then he is caught with probability at least 2^{-10k}. Denote by l the number of committed bits used in the proof, i.e., $l = O(n \cdot k)$. We select $\epsilon \stackrel{\text{def}}{=} \frac{k}{l}$ to be the fraction of bad strings in $\{0,1\}^k$ which we allow. Therefore, we have to use additional $10 \cdot l$ tests to make sure that with probability at most 2^{-10k} f_i violates this property and the prover is not caught. Note that we only multiply the length of the random string by a constant (11), so the complexity is not changed.

We first note that since there are at most 2^k possible permutations that the prover can choose, the test fails all of them with probability at least $1 - 2^{-9k}$. So now assume that we are dealing with a function l_i for which at most ϵ fraction of the points in $\{0,1\}^k$ have two inverses. So in the proof the prover gains an additional power: The ability to open the committed bits that have two inverses in both possible way. We call these bits *bad*. As before, we let the prover choose the opening of the bad committed bits at random and we then conclude that the prover can use this to raise his probability to cheat the verifier by a factor of at most 2^b where b is a random variable that represents the number of bad committed bits between the l bits used for the original proof. (Note that the prover has also the advantage of choosing which $i \in \{0,1\}^k$ to choose so that the bad bits appear in good places for him, but this advantage of selecting f_i was already taken in his favor previously in this section.

If the number of bad committed bits $b \leq 5k$, then we are done, since our soundness analysis is robust to the prover increasing his probabilities by a factor of 2^{5k}. So let us compute the probability that the number of bad bits b does not exceed $5k$. The expected number of bad bits is $l \cdot \epsilon = k$. By the Chernoff Inequality, we get:

$$\begin{aligned} \text{Prob}\left[b \geq 5k\right] &\leq \text{Prob}\left[|b - E[b]| \geq 4k\right] \\ &\leq 2^{-3k} \end{aligned}$$

It was shown in [BY92] that this proof is zero knowledge. It should also be clear that if $\varphi \in$3-SAT-5 and the (honest) prover indeed selects a permutation then he cannot fail.

Summing up, we can use the technique of [BY92] in order to relax the cryptographic assumption from the existence of a family of *certified* trapdoor permutations to the existence of a general family of trapdoor permutations. The cost is at most a constant multiplicative factor in the length of the shared random tape.

7 Open questions

The result in this paper follows earlier attempts to make the non-interactive zero-knowledge protocols more efficient. However, there is no known lower bound on the complexity of a non-interactive proof for NP-Hard languages. It is a most intriguing open question whether non-trivial lower bounds can be proven in this case.

References

[BFM88] M. Blum, P. Feldman, and S. Micali. Non-Interactive Zero Knowledge and Its Applications. In *Proc. Twentieth ACM Symp. Theor. Comput.*, pages 103–112, 1988.

[BG89] M. Bellare and S. Goldwasser. New Paradigms for Digital Signatures and Message Authentication Based on Non-Interactive Zero-Knowledge Proofs. In J. Feigenbaum, editor, *Advances in Cryptology - CRYPTO '89 proceeding*, volume 435 of *Lecture notes in computer Science*, pages 194–211. Springer-Verlag, 1989.

[BP94] J. Boyar and R. Peralta. Efficient Zero-Knowledge Proofs of Circuit Satisfiability. Technical Report 1, ISSN No. 09033920, Institut for Matematik og Datalogi, Odense Universitet, 1994.

[BY92] M. Bellare and M. Yung. Certifying Cryptographic Tools: The Case of Trapdoor Permutations. In *Advances in Cryptology - CRYPTO '92 proceeding*, Lecture notes in computer Science. Springer-Verlag, 1992.

[Dam92] I. Damgård. Non-Interactive Circuit-Based Proofs and Non-Interactive Perfect Zero Knowledge with Preprocessing. In *Advances in Cryptology - Eurocrypt '92 proceeding*, Lecture notes in computer Science. Springer-Verlag, 1992.

[DSMP88] A. De Santis, S. Micali, and G. Persiano. Non-Interactive Zero-Knowledge with Preprocessing. In S. Goldwasser, editor, *Advances in Cryptology - CRYPTO '88 proceeding*, volume 403 of *Lecture notes in computer Science*, pages 27–35. Springer-Verlag, 1988.

[Fei] U. Feige. Personal Communication.

[FLS90] U. Feige, D. Lapidot, and A. Shamir. Multiple Non-Interactive Zero-Knowledge Proofs Based on a Single Random String. In *Proc. 31th Ann. Symp. Found. Comput. Sci.*, pages 308–317, 1990.

[FS86] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology - CRYPTO '86 proceeding*, *Lecture notes in computer Science*, pages 186–189. Springer-Verlag, 1986.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM J. Comput.*, 18 (1):186–208, 1989.

[Kil94] J. Kilian. On the Complexity of Bounded-Interaction and Non-Interactive Zero-Knowledge Proofs. In *Proc. 35th Ann. Symp. Found. Comput. Sci.*, 1994.

[NY90] M. Naor and M. Yung. Public Key Cryptosystems Provably Secure Against Chosen Ciphertext Attacks. In *Proc. Twenty-second Ann. ACM Symp. Theor. Comput.*, pages 427–437, 1990.

A Interpreting the random tape as bit commitments

For completeness, we include the technique of [FLS90] for implementing random committed bits in the shared random string model.

We interpret the random string as a stream of committed random bits. For this we assume the existence of a one way permutation P. This approach was introduced in [FLS90]. We interpret a string r of k consecutive bits in the random string as encoding the bit $B(P^{-1}(r))$, where B is a hard-core bit of the one way permutation P.

The properties we need from this encoding are that the prover can "open" a committed bit to the verifier (simply by stating $P^{-1}(t)$ in this case), the prover cannot open a committed bit both to appear as a 0 and as a 1 (in our case this follows from P being a permutation), the computationally bounded verifier cannot guess the values of the bits from their commitment (this is guaranteed by the hard core property of B), and last, a uniformly chosen string of length t should be a commitment of a uniformly chosen bit in $\{0,1\}$ (this follows from P being a permutation and by the fact that B is unbiased).

So before the prover and verifier begin the protocol, they partition the random string to non-overlapping strings of length k and treat each such string as a commitment on a random bit. In the sequel, we refer to the shared random string as being composed of random committed bits.