ON ACHIEVING THE “BEST OF BOTH WORLDS” IN SECURE
MULTIPARTY COMPUTATION*

YUVAL ISHAI†, JONATHAN KATZ‡, EYAL KUSHILEVITZ†, YEHUDA LINDELL§, AND
EREZ PETRANK†

Abstract. Two settings are traditionally considered for secure multiparty computation, de-
pending on whether or not a majority of the parties are assumed to be honest. Existing protocols
that assume an honest majority provide “full security” (and, in particular, guarantee output delivery
and fairness) when this assumption holds, but are completely insecure if this assumption is violated.
On the other hand, known protocols tolerating an arbitrary number of corruptions do not guarantee
fairness or output delivery even if only a single party is dishonest. It is natural to wonder whether
it is possible to achieve the “best of both worlds”: namely, a single protocol that simultaneously
achieves the best possible security in both the above settings. Here, we rule out this possibility (at
least for general functionalities) and show some positive results regarding what can be achieved.

Key words. theory of cryptography, secure computation

AMS subject classifications. 94A60, 97P20

DOI. 10.1137/100783224

1. Introduction. Protocols for secure multiparty computation [4, 8, 12, 22] allow a set of mutually distrustful parties to compute a function in a distributed fashion while guaranteeing (to the extent possible) the privacy of the parties’ inputs and the correctness of their outputs. Security is typically formulated by requiring that a real execution of a protocol be indistinguishable from an ideal execution in which the parties hand their inputs to a trusted party who computes the function and returns the outputs to the appropriate parties. Thus, whatever security is implied by the ideal model is also guaranteed in a real-world execution of the protocol (see [5]).

The vast body of research in this area can be divided into two, almost disjoint, lines of work: one dealing with the case when a majority of the parties are assumed to be honest, and the other dealing with the case when an arbitrary number of parties may be corrupted. These settings differ not only in the approaches that are used to construct secure protocols but also in the results that can be achieved (and hence in the ideal models thus defined). Further details are as follows:

• Secure computation with an honest majority. When a majority of the participants are honest, it is possible to obtain the strongest level of security one could hope for (i.e., “full security”) [12]. Fully secure protocols ensure not only privacy and correctness but also fairness (namely, if one

*Received by the editors January 19, 2010; accepted for publication (in revised form) October 21, 2010; published electronically February 10, 2011. This is the full version of [18, 19]. This research was supported by grant 36/03 from the Israel Science Foundation, US-Israel Binational Science Foundation grant 2004240, and NSF CAREER award 0447075.
http://www.siam.org/journals/sicomp/40-1/78322.html
†Department of Computer Science, Technion, Haifa 32000, Israel (yuvali@cs.technion.ac.il, eyalk@cs.technion.ac.il, erez@cs.technion.ac.il).
‡Department of Computer Science, University of Maryland, College Park, MD 20742 (jkatz@cs.umd.edu). Portions of this work were done while the author was visiting the Institute for Pure and Applied Mathematics (IPAM), UCLA.
§Department of Computer Science, Bar-Ilan University, Ramat Gan 52900, Israel (lindell@cs.biu.ac.il). Portions of this work were done while the author was visiting the Technion.
1Here and in the rest of the paper, we assume a synchronous network of secure point-to-point channels and a broadcast channel. Formal definitions of security are given in section 2.
party receives its output, then all parties do) and guaranteed output delivery (i.e., honest parties are guaranteed to successfully complete the computation). In the presence of an honest majority, full security can even be obtained unconditionally [2, 4, 8, 10, 20].

- **Secure computation with no honest majority.** Results of Cleve [9] imply that full security (for general functionalities) is only possible when an honest majority is present. Specifically, while privacy and correctness are still attainable without an honest majority, it is impossible (in general) to guarantee fairness or output delivery.\(^2\) Thus, when no honest majority is assumed, a relaxed notion of security (“security with abort”) is used in which privacy and correctness still hold but the adversary is allowed to abort the computation after obtaining its own outputs. Unconditional security is no longer possible in this setting (for general functionalities), but protocols realizing this notion of security for any number of corrupted parties can be constructed based on the existence of enhanced trapdoor permutations or oblivious transfer [3, 12, 13, 14, 22].

An unfortunate drawback of existing protocols for each of the above settings is that they do not provide any security beyond what is implied by the definitions. Specifically, existing protocols designed for the case of honest majority are completely insecure once half (or more) of the parties are corrupted; e.g., honest parties’ inputs are entirely revealed, and even correctness may be violated. On the other hand, known protocols that achieve security with abort against an arbitrary number of corruptions do not guarantee fairness or output delivery even if only a single party is corrupted.

To get a sense for the magnitude and importance of this problem, consider trying to decide which type of protocol is more appropriate to implement secure voting. Since we would like privacy of individuals’ votes to hold (to the extent possible) regardless of the number of corruptions, we are forced to use a protocol of the second type that provides only security with abort. But then a single corrupted machine (in fact, even one which simply fails in the middle of the election) may be able to perform a denial-of-service attack that prevents all honest parties from learning the outcome. Neither option is very appealing. The above state of affairs raises the following natural question:

*To what extent can we design a single protocol achieving the “best of both worlds” regardless of the number of corruptions, i.e., a protocol that simultaneously guarantees full security in the case when a majority of the parties are honest, and security with abort otherwise?*

### 1.1. Our results

In this paper, we initiate the study of the above question. Our main results settle the question in the negative, showing that (in general) it is impossible to construct a single protocol that achieves the best possible security regardless of the number of corruptions. We consider both standard functionalities, which receive inputs and deliver outputs in “one shot,” and reactive functionalities, which receive inputs and deliver outputs in multiple phases and maintain (distributed) state information between phases. (See section 2 for definitions of these and other terms we use.) We show the following.

**Theorem 1.1.** Let \( t + s \geq n \) with \( 1 \leq t \leq s \). There exists a standard (non-reactive) functionality \( f \), for which there is no \( n \)-party protocol computing \( f \) that is simultaneously (1) fully secure when \( t \) parties are corrupted, and (2) secure with abort when \( s \) parties are corrupted.

\(^2\)For certain exceptions, see [15, 16, 17] and the discussion in section 1.2.
In fact, our negative results are even stronger than indicated above. Fix $s, t, n$ as above. For nonreactive functionalities, it is impossible (in general) to simultaneously achieve full security against $t$ fail-stop corruptions and privacy against $s$ fail-stop corruptions (recall that fail-stop corruption only allows the adversary to deviate from the protocol by aborting). This holds even if we allow protocols with expected polynomial round complexity. For reactive functionalities we show an even stronger result: one cannot (in general) simultaneously obtain full security against $t$ fail-stop corruptions and privacy against $s$ semihonest corruptions.

In light of the above, we are led to explore what security guarantees can be achieved with regard to different corruption thresholds. Considering the natural relaxations, we show two incomparable positive results in this regard. First, we show that when $t + s < n$ (and $t < n/2$), then the “best of both worlds” is, indeed, possible.

**Theorem 1.2.** Assume the existence of enhanced trapdoor permutations, and let $t + s < n$ and $t < n/2$. For any (reactive or nonreactive) functionality $f$, there exists a protocol computing $f$ that is simultaneously (1) fully secure when $t$ parties are corrupted, and (2) secure with abort when $s$ parties are corrupted.

We also show that in the case of nonreactive functionalities, the optimal security thresholds (i.e., $t < n/2$ and $s < n$) are obtainable if we restrict to semihonest adversaries when there is no honest majority.

**Theorem 1.3.** Assume the existence of enhanced trapdoor permutations, and let $t < n/2$ and $s < n$. For any nonreactive functionality $f$, there exists a protocol computing $f$ that is simultaneously (1) fully secure against $t$ malicious corruptions, and (2) fully secure against $s$ semihonest corruptions.

Table 1 summarizes our results along with pointers to the corresponding theorems from the technical sections.

<table>
<thead>
<tr>
<th></th>
<th>Standard functionalities</th>
<th>Reactive functionalities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full security against $t &lt; n/2$ malicious parties and privacy against $s &lt; n$ semihonest parties</td>
<td>Yes</td>
<td>Theorem 4.2</td>
</tr>
<tr>
<td>Full security against $t &lt; n/2$ malicious parties and privacy against $s &lt; n$ malicious parties</td>
<td>No</td>
<td>Theorem 3.5</td>
</tr>
</tbody>
</table>

**1.2. Future directions.** The stark negative results in this paper, coupled with the fact that it can be difficult to determine what security bounds are appropriate to assume in practice, suggest looking for other ways to obtain “best of both worlds” type results besides those already discussed above. We briefly mention two possibilities that have been explored previously and then highlight some promising directions for future work.

In previous versions of this work [18, 19], two feasibility results have been investigated (which are not included in the current version). Specifically, Ishai et al. [18] show a protocol for any nonreactive functionality $f$ that provides full security against $t < n/2$ malicious parties and also ensures the following guarantee against $s < n$ malicious parties (informally): the malicious parties achieve (and learn) no more than they could achieve in $s$ invocations of an ideal party evaluating $f$, where the malicious parties may use different inputs in different invocations of $f$. For certain functional-
ities (with voting serving as a prime example), this provides a meaningful notion of security. In another direction, Katz [19] (following [18]) explored what is possible in the case of a nonrushing adversary or, equivalently, under the assumption of simultaneous message transmission. He shows, for any nonreactive functionality \( f \) and any polynomial \( p \), a protocol that is fully secure against \( t < n/2 \) malicious parties, as well as \( \frac{1}{p} \)-secure with abort” for any number of malicious corruptions. Roughly speaking, this latter notion means that the actions of any real-world adversary can be simulated by an ideal-world adversary (who has the ability to abort the protocol) such that the resulting outcomes cannot be distinguished with advantage better than \( O(1/p) \). (The protocol provides additional security guarantees as well; see [19] for details.)

With regard to future work in this area, several directions seem promising:

- **Nonrushing adversaries.** We currently have only a partial answer to what can be achieved if a nonrushing adversary is assumed. The results of Katz [19] leave open the possibility of protocols in this model that achieve the true “best of both worlds”: simultaneous full security against \( t < n/2 \) corruptions, and security with abort against \( s < n \) corruptions. (However, it is known that there are no constant-round [18] or even logarithmic-round [19] protocols of this sort.) Alternatively, it might be possible in this model to obtain full security against \( t < n/2 \) corruptions and \( \frac{1}{p} \)-security without abort against \( s < n \) corruptions. (In fact, this may be possible even for rushing adversaries by building on the ideas of [17].)

- **Specific functionalities.** The impossibility results presented here rule out protocols for general functionalities, but leave open the question of what might be obtained for specific functionalities of interest. Recent work [15, 16] has shown that protocols with full security against an arbitrary number of corruptions can be constructed for certain (nontrivial) functionalities. (This is even better than what a “best of both worlds”-type result would imply.) For what other functionalities can positive results be obtained?

- **Definitional relaxations.** In the current work we have focused on the standard notions of full security and security with abort. Given the impossibility results we have obtained, it may be worthwhile to explore relaxations of these definitions such as those considered in [1, 17].

2. **Preliminaries and definitions.** In this work, \( k \) denotes the security parameter and PPT stands for “probabilistic polynomial time.”

2.1. **Overview.** Our default network model consists of \( n \) parties, \( P_1, \ldots, P_n \), who interact in synchronous rounds via private and authenticated point-to-point channels. We also assume that the parties have access to a broadcast channel. We consider both rushing and nonrushing adversaries. A rushing adversary may delay sending the messages of the corrupted parties in any given round until after the honest parties send their messages in that round; thus, the round-\( i \) messages of the corrupted parties may depend on the round-\( i \) messages of the honest parties. In contrast, a nonrushing adversary must decide on what messages the corrupted parties should send in any given round before seeing the honest parties’ messages in that round. Assuming a nonrushing adversary is essentially equivalent to assuming that there exists a mechanism for simultaneous message exchange. The standard definition of secure computation assumes a rushing adversary. Nevertheless, in one of our lower bounds we consider nonrushing adversaries since this only strengthens the result.

We consider both malicious adversaries, who have total control over the behavior
of corrupted parties and may instruct them to deviate arbitrarily from the protocol specification, and semihonest adversaries, who record all information viewed by corrupted parties as they run the protocol but do not otherwise modify their behavior. We also consider fail-stop adversaries, who follow the protocol honestly (as semihonest adversaries do) except that they may abort the protocol early.

Throughout the paper, we consider security against computationally bounded adversaries and assume for simplicity that the adversary is static, i.e., that the set of corrupted parties is chosen at the onset of the protocol in a nonadaptive manner. This strengthens our negative results and is not essential for our positive results; see Remark 4.5.

The security of a multiparty protocol is defined with respect to a functionality $f$. A nonreactive $n$-party functionality is a (possibly randomized) mapping of $n$ inputs to $n$ outputs. A multiparty protocol for computing a nonreactive functionality $f$ is a protocol running in polynomial time and satisfying the following correctness requirement: if parties $P_1, \ldots, P_n$ holding inputs $(1^k, x_i)$, respectively, all run an honest execution of the protocol, then the joint distribution of the outputs $y_1, \ldots, y_n$ of the parties is statistically close to $f(x_1, \ldots, x_n)$.

A reactive functionality $f$ is a sequence of nonreactive functionalities $f = (f_1, \ldots, f_\ell)$ computed in a series of phases, where the functionality maintains a secret state which is updated in each phase. Let $x_i^j$ denote the input of $P_i$ in phase $j$, and let $s^j$ denote the state of the computation after phase $j$. Computation of $f$ proceeds by setting $s^0$ equal to the empty string and then computing $(y_1^j, \ldots, y_n^j, s^j) \leftarrow f_j(s^{j-1}, x_1^j, \ldots, x_n^j)$ for $j = 1$ to $\ell$, where $y_i^j$ denotes the output of $P_i$ at the end of phase $j$. A multiparty protocol computing $f$ also runs in $\ell$ phases, at the beginning of which each party holds an input and at the end of which each party obtains an output. (Note that parties may wait to decide on their phase-$j$ input until the beginning of that phase.) Parties maintain state throughout the entire execution. The correctness requirement is that, in an honest execution of the protocol, the joint distribution of all the outputs $\{y_1^j, \ldots, y_n^j\}_{j=1}^\ell$ of all the phases is statistically close to the joint distribution of all the outputs of all the phases in a computation of $f$ on the same inputs used by the parties.

2.2. Defining security. In this section, we present the (standard) security definitions used in this paper. We assume the reader is familiar with the simulation-based approach for defining secure computation, as described in detail in [5, 7, 11]. This definitional approach compares the real-world execution of a protocol for computing some function with an ideal-world evaluation of the function by a trusted party. Security is then defined by requiring that whatever can be achieved in the real world could have also been achieved (or simulated) in the ideal world. More formally, it is required that for every adversary $\mathcal{A}$ attacking the real execution of the protocol there exists an adversary $\mathcal{A'}$, sometimes referred to as a simulator, which "achieves the same effect" in the ideal world. This is made more precise in what follows.

The real model. Let $\pi$ be a multiparty protocol computing a nonreactive functionality $f$. It is convenient [7] to view an execution of $\pi$ in the presence of an adversary $\mathcal{A}$ as being coordinated by a nonuniform environment $\mathcal{Z} = \{Z_k\}$. At the outset, $\mathcal{Z}$ gives input $(1^k, x_i)$ to each party $P_i$, and gives $I$, $\{x_i\}_{i \in I}$, and $z$ to $\mathcal{A}$, where $I \subset [n]$ represents the set of corrupted parties and $z$ denotes an auxiliary input. The parties then interact, with each honest (i.e., uncorrupted) party $P_i$ behaving as instructed by the protocol (using input $x_i$) and corrupted parties behaving as directed by $\mathcal{A}$. In the case of a semihonest adversary, $\mathcal{A}$ directs the parties to follow the protocol.
on their given inputs. At the conclusion of the protocol, \( A \) gives to \( Z \) an output which is an arbitrary function of \( A \)'s view throughout the protocol, and \( Z \) is additionally given the outputs of the honest parties. Finally, \( Z \) outputs a bit. We let \( \text{real}_{\pi,A,Z}(k) \) be a random variable denoting the value of this bit.

For reactive functionalities, the environment \( Z \) operates in a series of phases. At the outset of the execution, \( Z \) gives \( I \) and \( z \) to \( A \). Then, at the beginning of each phase \( j \), the environment gives input \( x'_i \) to each party \( P_i \) and gives \( \{x'_i\}_{i \in I} \) to \( A \). The parties then run the \( j \)th phase of protocol \( \pi \). At the end of each phase, \( A \) gives to \( Z \) an output which is an arbitrary function of \( A \)'s view thus far, and \( Z \) is additionally given the outputs of the honest parties in this phase. If the adversary aborts the protocol in some phase (formally, if the output of some honest party at the end of the phase is \( \bot \)), execution is halted; otherwise, execution continues until all phases are completed (i.e., the protocol is finished). Once the execution terminates, \( Z \) outputs a bit; we let \( \text{ideal}_{\pi,A,Z}(k) \) be a random variable denoting the value of this bit.

**The ideal model—full security.** In the ideal model, there is a trusted party who computes \( f \) on behalf of the parties. The first variant of the ideal model, discussed now, corresponds to a notion of security where fairness and output delivery are guaranteed.

Once again, we have an environment \( Z \) that gives inputs \( x_1, \ldots, x_n \) to the parties and provides \( I, \{x_i\}_{i \in I}, \) and \( z \) to \( A' \). Execution then proceeds as follows:

- Each honest party \( P_i \) sends its input \( x_i \) to the trusted party. Corrupted parties may send the trusted party arbitrary inputs as instructed by \( A' \). (Any missing or “invalid” value is substituted by a default value.) Denote by \( x'_i \) the value sent by party \( P_i \). In the case of a semihonest adversary, we require that \( x'_i = x_i \) for all \( i \in I \).

- The trusted party computes \( f(x'_1, \ldots, x'_n) = (y_1, \ldots, y_n) \) and sends \( y_i \) to party \( P_i \). (If \( f \) is randomized, this computation involves random coins that are generated by the trusted party.)

After the above, \( A' \) gives to \( Z \) an output which is an arbitrary function of the view of \( A' \), and \( Z \) is also given the outputs of the honest parties. Finally, \( Z \) outputs a bit. We let \( \text{ideal}_{f,A',Z}(k) \) be the random variable denoting the value of this bit.

For the case of reactive functionalities, execution once again proceeds in a series of phases. At the outset, \( Z \) gives \( I \) and \( z \) to \( A' \). At the beginning of each phase \( j \), the environment provides input \( x'_i \) to party \( P_i \) and gives \( \{x'_i\}_{i \in I} \) to \( A' \). Inputs/outputs are then sent to/from the trusted party as above. At the end of each phase, \( A' \) gives to \( Z \) an output which is an arbitrary function of its view thus far, and \( Z \) is additionally given the outputs of the honest parties in this phase. After all phases have been completed, \( Z \) outputs a bit. Once again, we let \( \text{ideal}_{\pi,A',Z}(k) \) be a random variable denoting the value of this bit.

**The ideal model—security with abort.** In this second variant of the ideal model, fairness and output delivery are no longer guaranteed. This is the standard relaxation used when a strict majority of honest parties is not assumed. (Other variants are also possible [11, 14].)

As in the first ideal model, we have an environment \( Z \) that provides inputs \( x_1, \ldots, x_n \) to the parties and provides \( I, \{x_i\}_{i \in I}, \) and \( z \) to the adversary \( A' \). Execution then proceeds as follows:

- As before, the parties send their inputs to the trusted party, and we let \( x'_i \) denote the value sent by \( P_i \). Once again, for a semihonest adversary we require \( x'_i = x_i \) for all \( i \in I \).
• The trusted party computes \( f(x'_1, \ldots, x'_n) = (y_1, \ldots, y_n) \) and sends \( \{y_i\}_{i \in I} \) to the adversary.

• The adversary chooses whether to continue or abort; this is formalized by having the adversary send either a \textit{continue} or \textit{abort} message to the trusted party. (A semihonest adversary never aborts.) In the former case, the trusted party sends to each uncorrupted party \( P_i \) its output value \( y_i \). In the latter case, the trusted party sends the special symbol \( \perp \) to each uncorrupted party.

After the above, \( A' \) gives to \( Z \) an output which is an arbitrary function of the view of \( A' \), and \( Z \) is also given the outputs of the honest parties. Finally, \( Z \) outputs a bit. We let \( \text{ideal}_{f, A', Z}(k) \) be the random variable denoting the value of this bit; the subscript "⊥" indicates that the adversary now has the ability to abort the trusted party in the ideal model.

The extension to the case of reactive functionalities is the same as before. As in the real-world model, execution is halted immediately after any phase in which an honest party outputs \( \perp \).

\textbf{Defining security.} With the above in place, we can now define our notions of security.

\textbf{Definition 2.1 (security; security with abort).} Let \( \pi \) be a multiparty protocol for computing a functionality \( f \), and fix \( s \in \{1, \ldots, n\} \).

1. We say that \( \pi \) securely computes \( f \) in the presence of malicious (resp., semihonest) adversaries corrupting \( s \) parties if for any \( \text{PPT adversary (resp., semihonest adversary)} A \) there exists a \( \text{PPT adversary (resp., semihonest adversary)} A' \) such that for every polynomial-size circuit family \( Z = \{Z_k\} \) corrupting at most \( s \) parties the following is negligible:

\[ |\Pr[\text{real}_{f, A, Z}(k) = 1] - \Pr[\text{ideal}_{f, A', Z}(k) = 1]|. \]

2. We say that \( \pi \) securely computes \( f \) with abort in the presence of malicious adversaries corrupting \( s \) parties if for any \( \text{PPT adversary} A \) there exists a \( \text{PPT adversary} A' \) such that for every polynomial-size circuit family \( Z = \{Z_k\} \) corrupting at most \( s \) parties the following is negligible:

\[ |\Pr[\text{real}_{f, A, Z}(k) = 1] - \Pr[\text{ideal}_{f, A', Z}(k) = 1]|. \]

We also consider the weaker notion of \textit{privacy} which, roughly speaking, ensures only that the adversary cannot learn anything about honest parties' inputs other than what is implied by its own inputs and outputs. Accordingly, the definition of privacy only requires that the adversary’s view in the real model can be simulated in the ideal model.

Formally, define \( \text{real}'_{f, A, Z}(k) \) analogously to \( \text{real}_{f, A, Z}(k) \) except that \( Z \) is not given the outputs of the honest parties (that is, \( Z \) is only given the output of \( A \), which is an arbitrary function of \( A \’ \)'s view). We define \( \text{ideal}'_{f, A', Z}(k) \) similarly.

\textbf{Definition 2.2 (privacy).} Let \( \pi \) be a multiparty protocol for computing a functionality \( f \), and fix \( s \in \{1, \ldots, n\} \). We say that \( \pi \) privately computes \( f \) in the presence of malicious (resp., semihonest) adversaries corrupting \( s \) parties if for any \( \text{PPT adversary (resp., semihonest adversary)} A \) there exists a \( \text{PPT adversary (resp., semihonest adversary)} A' \) such that for every polynomial-size circuit family \( Z = \{Z_k\} \) corrupting at most \( s \) parties the following is negligible:

\[ |\Pr[\text{real}'_{f, A, Z}(k) = 1] - \Pr[\text{ideal}'_{f, A', Z}(k) = 1]|. \]

Note that privacy is weaker than security with abort.
3. Impossibility results.

3.1. Reactive functionalities. In this section we present a strong impossibility result for the case of reactive functionalities. The threshold in the theorem that follows is tight; see section 4.1. The restriction of the theorem to reactive functionalities is also essential, as we show in section 4.2 a positive result for the case of nonreactive functionalities.

**Theorem 3.1.** Let \( n, t, s \) be such that \( t + s = n \) and \( t \geq 1 \). There exists a reactive \( n \)-party functionality \( f \) for which there is no protocol that simultaneously

- securely computes \( f \) in the presence of malicious adversaries corrupting \( t \) parties, and
- privately computes \( f \) in the presence of semihonest adversaries corrupting \( s \) parties.

This holds even if the adversary in the first case is restricted to be a nonrushing, fail-stop adversary.

**Proof.** If \( t \geq n/2 \), then the theorem follows from the fact that there exist nonreactive functionalities that cannot be computed securely without an honest majority [9]. Thus, we assume \( t < n/2 \) (implying \( n \geq 3 \) and \( s > 0 \)) in what follows.

We prove Theorem 3.1 using a two-phase functionality \( f \) that corresponds (roughly) to commitment; see Figure 1. Take any protocol \( \pi \) computing \( f \), and let \( S_1 = \{P_1, \ldots, P_t\} \) and \( S_2 = \{P_{t+1}, \ldots, P_n\} \); note that \(|S_2| = s\). The intuition for the theorem is as follows:

1. Say \( \pi \) privately computes \( f \) in the presence of semihonest adversaries corrupting \( s \) parties. Then the parties in \( S_2 \) should (jointly) know nothing about \( b \) after phase 1.
2. Say \( \pi \) securely computes \( f \) in the presence of malicious adversaries corrupting \( t \) parties. Then the parties in \( S_1 \) should not be able to prevent \( P_n \) from learning \( b \) in the second phase. In particular, this should hold even if all parties in \( S_1 \) abort before the second phase.

Intuitively this gives a contradiction since, following the first phase of the protocol, the parties in \( S_2 \) can jointly simulate an execution of the second phase of the protocol when all parties in \( S_1 \) abort. The formal proof of the theorem is slightly more involved since item 2, above, is not quite true in the presence of a malicious \( P_1 \) who might change his input before running the protocol.

---

**Fig. 1.** The reactive functionality for the proof of Theorem 3.1.
We now prove this formally. Let \( \pi \) be a protocol computing \( f \) that is secure in the presence of \( t \) nonrushing, fail-stop adversaries; we will show that \( \pi \) cannot also be private in the presence of \( s \) semihonest adversaries. Since \( \pi \) is secure in the presence of \( t \) fail-stop adversaries, we may assume without loss of generality that the output of \( P_n \) in \( \pi \) is always a bit (and never \( \perp \)) as long as parties in \( S_1 \) behave honestly, except that they may abort the protocol early. We also assume for simplicity that, in an honest execution of \( \pi \), the output of \( P_n \) is exactly \( P_1 \)'s input value \( b \) with probability 1.

We consider two real-world executions of \( \pi \). In both cases, \( Z \) chooses \( b \) and \( b' \) uniformly and independently.

**First execution.** Here we consider a nonrushing, fail-stop adversary \( A_1 \) who corrupts the parties in \( S_1 \) and instructs them to behave as follows. Run the first phase of \( \pi \) honestly. In the second phase, if \( b' = b \), then run the second phase honestly; otherwise, abort immediately. We let \( \Pr_1[\cdot] \) denote the probability of events in this execution.

**Second execution.** Here we consider a semihonest adversary \( A_2 \) who corrupts the parties in \( S_2 \). At the conclusion of phase 1 of \( \pi \), this adversary simulates an execution of the second phase of \( \pi \), assuming all parties in \( S_1 \) abort, and outputs the resulting output of \( P_n \). We let \( \Pr_2[\cdot] \) denote the probability of events in this execution.

**Claim 3.2.** \( \Pr_1[P_n \text{ outputs } b'] = \frac{1}{2} + \frac{1}{2} \cdot (1 - \Pr_2[A_2 \text{ outputs } b]) \).

**Proof.** In the first execution, we can consider a mental experiment in which the parties in \( S_2 \) simulate the second phase of \( \pi \) assuming all parties in \( S_1 \) abort. By definition of \( A_2 \), the probability that the output of \( P_n \) in this mental experiment is not equal to \( b \) is exactly \( 1 - \Pr_2[A_2 \text{ outputs } b] \). Furthermore, this probability is independent of the value \( b' \) used in the second phase.

In the first execution, the parties in \( S_1 \) abort with probability exactly \( 1/2 \); moreover, when they do not abort, the output of \( P_n \) is \( b = b' \). When the parties in \( S_1 \) do abort, then \( b' \neq b \), and so \( P_n \) outputs \( b' \) iff the output of \( P_n \) is not equal to \( b \). Thus,

\[
\Pr_1[P_n \text{ outputs } b'] = \frac{1}{2} + \frac{1}{2} \cdot \Pr_1[P_n \text{ outputs } b \mid \text{parties in } S_1 \text{ abort}]
\]

\[
= \frac{1}{2} + \frac{1}{2} \cdot (1 - \Pr_2[A_2 \text{ outputs } b]),
\]

as desired.

**Claim 3.3.** \( |\Pr_1[P_n \text{ outputs } b'] - \frac{1}{2}| \) is negligible.

**Proof.** Here we rely on the assumption that \( \pi \) securely computes \( f \) in the presence of a nonrushing, fail-stop adversary corrupting \( t \) parties. Consider an ideal-world execution of \( f \) in the presence of an adversary corrupting the parties in \( S_1 \). In the ideal world, the final output of \( P_n \) is equal to whatever value \( P_1 \) sends to the trusted party in the first phase. Since the adversary has no information about \( b' \) in the first phase, the probability that \( P_1 \) sends \( b' \) to the trusted party in the first phase (and hence the probability that \( P_n \) outputs \( b' \) in the second phase) is exactly \( 1/2 \). The claim follows.

**Claim 3.4.** If \( \pi \) privately computes \( f \) in the presence of a semihonest adversary corrupting \( s \) parties, then \( |\Pr_2[A_2 \text{ outputs } b] - \frac{1}{2}| \) is negligible.

**Proof.** This follows from the fact that, in an ideal-world execution of \( f \), an adversary corrupting the parties in \( S_2 \) cannot guess the value of \( b \) after phase 1 with probability different from \( 1/2 \).

---

3Allowing \( Z \) to be randomized does not affect our definitions of security.
The preceding three claims imply that $\pi$ cannot privately compute $f$ in the presence of a semihonest adversary corrupting $s$ parties. This concludes the proof of Theorem 3.1.

3.2. Nonreactive functionalities. We now show an impossibility result for the case of standard (i.e., nonreactive) functionalities. This result is incomparable to the result proved in the previous section, since here we rule out privacy only against malicious adversaries and explicitly make use of the fact that the adversary can be rushing.\footnote{For nonrushing adversaries, similar negative results for protocols with $O(\log k)$ rounds appear in \cite{18, 19}.} The thresholds in the theorem are tight; see section 4.2.

**Theorem 3.5.** Let $n, t, s$ be such that $t + s = n$ and $t \geq 1$. There exists a nonreactive functionality $\tilde{f}$ for which there is no protocol that simultaneously

- securely computes $\tilde{f}$ in the presence of malicious adversaries corrupting $t$ parties, and
- privately computes $\tilde{f}$ in the presence of malicious adversaries corrupting $s$ parties.

This holds even if we consider only fail-stop adversaries in each case and even if we allow protocols with expected polynomial round complexity.

When $t \geq n/2$ the theorem follows from the existence of functionalities that cannot be computed securely without an honest majority \cite{9}. Thus, we assume $t < n/2$ (and hence $n \geq 3$ and $s > 0$) in what follows. We prove the theorem in two stages. In section 3.2.1 we present a functionality $f$ for which there is no protocol that simultaneously

- securely computes $f$ in the presence of malicious adversaries corrupting $t$ parties, and
- securely computes $f$ with abort in the presence of malicious adversaries corrupting $s$ parties.

Extending this, we then show in section 3.2.2 a (slightly different) functionality $\tilde{f}$ that suffices to prove the theorem.

3.2.1. Ruling out security with abort. Fix $n, s,$ and $1 \leq t < n/2$ with $t + s = n$. Define $f$ as follows: parties $P_1$ and $P_n$ have as input $b_1, b_n \in \{0, 1\}$, respectively, and each receives as output $b_1 \oplus b_n$ (no other parties receive output). Let $\pi$ be a protocol that securely computes $f$ in the presence of a fail-stop adversary corrupting $t$ parties. We assume $\pi$ operates in segments, each exactly $n$ rounds long, where only party $P_i$ sends a message in the $i$th round of a segment (i.e., in any given segment first $P_1$ speaks, then $P_2$, etc., until $P_n$ speaks and then the next segment begins). If $\pi$ is secure against a rushing adversary, then it can always be transformed into a protocol of this form without affecting its security.

Let $r = r(k)$ be a polynomial with the following property:

- If $P_1$ (resp., $P_n$) is honest and at most $t$ parties are corrupted, then with probability at least $15/16$, party $P_1$ (resp., $P_n$) has generated its output by the end of segment $r$. Moreover, if $P_1$ and $P_n$ are both honest and at most $t$ parties are corrupted, then with probability at least $15/16$, parties $P_1$ and $P_n$ have both generated their outputs by the end of segment $r$.

(Note that if $P_1$ and $P_n$ are both honest and at most $t$ parties are corrupted, then security of $\pi$ implies that—for large enough $k$—with probability at least $7/8$, parties $P_1$ and $P_n$ both generate identical outputs by the end of segment $r$.)

For protocols
with bounded (polynomial) round complexity, \( r \) can just be taken as the upper bound on the number of rounds; for protocols with expected polynomial round complexity \( r'(k) \), we may take \( r(k) = 16r'(k) \). We assume for simplicity that honest parties always run \( \pi \) for at least \( r \) segments (e.g., by sending dummy messages until segment \( r \) if the protocol otherwise would have terminated before then).

Define \( A \overset{\text{def}}{=} \{ P_1, \ldots, P_t \} \), \( B \overset{\text{def}}{=} \{ P_{t+1}, \ldots, P_{n-t} \} \), and \( C \overset{\text{def}}{=} \{ P_{n-t+1}, \ldots, P_n \} \). Consider the real-world execution in which \( Z \) chooses inputs for \( P_1 \) and \( P_n \) uniformly and independently (see footnote 3) and then all parties run the protocol honestly, except that parties in \( A \) or \( C \) may (possibly) abort at some round. (Parties in \( B \) run the protocol honestly and never abort.) Let \( v^t_i \), with \( 0 \leq i \leq r \), denote the final output of \( P_i \) when parties in \( C \) all abort in segment \( i+1 \) or, in other words, when segment \( i \) is the last segment in which parties in \( C \) send any messages. (For \( i = 0 \) this means that parties in \( C \) abort the protocol immediately without sending any messages.) Define \( v^t_n \) similarly to be the output of \( P_n \) when all parties in \( A \) abort in segment \( i+1 \) (i.e., send messages for the final time in segment \( i \)). Note that \( v^t_i \) can be computed from the joint view of the parties in \( A \cup B \) as soon as all parties in \( C \) have sent their segment-\( i \) messages, and similarly \( v^t_n \) can be computed from the joint view of the parties in \( B \cup C \) once all parties in \( A \) have sent their segment-\( i \) messages.

Security of \( \pi \) implies that, for all \( i \), we have \( v^t_i, v^t_n \in \{ 0, 1 \} \) (and, in particular, \( v^t_i \neq \perp \)) with all but negligible probability. This is true since \( \pi \) provides full security against t fail-stop adversaries, and at most \( t \) parties abort in the experiment defining \( v^t_i, v^t_n \). In what follows, we will assume for simplicity that \( v^t_i, v^t_n \in \{ 0, 1 \} \) with probability 1.

Consider the following summation, where all probabilities are with respect to the real-world execution described earlier:

\[
\Pr[v^0_1 = 1 \land v^0_n = 1] + \Pr[v^0_1 = 0 \land v^r_n = 1] - \frac{1}{2}
\]

\[
\Pr[v^0_1 = 0 \land v^0_n = 0] + \Pr[v^0_1 = 1 \land v^r_n = 0] - \frac{1}{2}
\]

\[
\sum_{i=0}^{r-1} \left[ \Pr[v^i_1 = 0 \land v^{i+1}_n = 0] + \Pr[v^i_1 = 1 \land v^i_n = 0] - \frac{1}{2} \right]
\]

\[
\Pr[v^i_1 = 1 \land v^{i+1}_n = 1] + \Pr[v^i_1 = 0 \land v^{i+1}_n = 1] - \frac{1}{2}
\]

\[
\Pr[v^{i+1}_n = 0 \land v^{i+1}_n = 0] + \Pr[v^i_1 = 0 \land v^{i+1}_n = 1] - \frac{1}{2}
\]

\[
\Pr[v^{i+1}_n = 1 \land v^{i+1}_n = 1] + \Pr[v^i_1 = 1 \land v^{i+1}_n = 0] - \frac{1}{2}
\]

which evaluates to

\[
\Pr[v^0_1 = 1 \land v^0_n = 1] + \Pr[v^0_1 = 0 \land v^r_n = 1] \\
+ \Pr[v^0_1 = 0 \land v^0_n = 0] + \Pr[v^0_1 = 1 \land v^r_n = 0] - 1 \\
+ \sum_{i=0}^{r-1} \left[ \Pr[v^i_1 = 1 \land v^i_n = 0] + \Pr[v^i_1 = 0 \land v^i_n = 1] \\
+ \Pr[v^{i+1}_n = 0 \land v^{i+1}_n = 0] + \Pr[v^{i+1}_n = 1 \land v^{i+1}_n = 1] - 1 \right]
\]

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
Consider the adversary given auxiliary input $z = i$ who corrupts the parties in $A \cup B$ and acts as follows: it runs the protocol honestly up to the end of segment $i$ (if $i = 0$, this is just the beginning of the protocol). At this point, as noted earlier, the parties in $A \cup B$ jointly have enough information to compute $v^i_1$. If $v^i_1 = 1$, then the adversary immediately aborts all parties in $A$. If $v^i_1 = 0$, then the parties in $A$ send...
their (honestly computed) messages for segment $i+1$ but send no more messages after that (i.e., they abort in segment $i + 2$). In either case, parties in $B$ continue to run the remainder of the protocol honestly.

The probability that $P_n$ outputs 0 in a real execution of the protocol is exactly

$$\Pr [v_1^i = 0 \land v_{n}^{i+1} = 0] + \Pr [v_1^i = 1 \land v_n^i = 0] \geq \frac{1}{2} + p(k).$$

However, as before, in an ideal execution with any adversary corrupting parties in $A \cup B$, the honest party $P_n$ will not output 0 with probability greater than $\frac{1}{2}$. Thus, in this case $\pi$ does not securely compute $f$ with abort in the presence of a fail-stop adversary corrupting $s$ parties.

**Case 2(b).** If there exists an $i$ such that $\Pr [v_1^i = 1 \land v_n^{i+1} = 1] + \Pr [v_1^i = 0 \land v_n^i = 1] - \frac{1}{2} \geq p(k)$, an argument as above gives an adversary corrupting parties in $A \cup B$ who forces $P_n$ to output 1 more often than can be achieved by any adversary in the ideal world.

**Case 3(a).** Say there exists an index $i \in \{1, \ldots, r(k)\}$ such that

$$\Pr [v_1^i = 0 \land v_n^i = 0] + \Pr [v_1^{i-1} = 0 \land v_n^i = 1] - \frac{1}{2} \geq p(k).$$

(Note that all indices have been shifted by 1 for convenience.) Consider the adversary given auxiliary input $z = i$ who corrupts parties in $B \cup C$ and acts as follows: it runs the protocol honestly up to the point when it is $P_{n-t+1}$’s turn to send a message in segment $i$. (Recall that $P_{n-t+1}$ is the party with lowest index who is in $C$.) At this point, the parties in $B \cup C$ can jointly compute $v_n^i$. If $v_n^i = 1$, then all parties in $C$ abort in this segment and do not send any more messages (so the last messages sent by any parties in $C$ were sent in segment $i-1$). If $v_n^i = 0$, then all parties in $C$ send their (honestly generated) messages in segment $i$ but abort in segment $i+1$. In either case, parties in $B$ continue to run the entire rest of the protocol honestly.

The probability that $P_1$ outputs 0 in a real execution of the protocol is exactly

$$\Pr [v_1^i = 0 \land v_n^i = 0] + \Pr [v_1^{i-1} = 0 \land v_n^i = 1] \geq \frac{1}{2} + p(k).$$

However, in an ideal execution with any adversary corrupting parties in $B \cup C$, the honest party $P_1$ will not output 0 with probability greater than $\frac{1}{2}$ (given that its input is chosen uniformly at random). We conclude that in this case $\pi$ does not securely compute $f$ with abort in the presence of a fail-stop adversary corrupting $s$ parties.

**Case 3(b).** If there exists an $i$ such that $\Pr [v_1^i = 1 \land v_n^i = 1] + \Pr [v_1^{i-1} = 1 \land v_n^i = 0] - \frac{1}{2} \geq p(k)$, an argument as above gives an adversary corrupting parties in $B \cup C$ who forces $P_1$ to output 1 more often than can be achieved by any adversary in the ideal world.

### 3.2.2. Ruling out privacy.

The argument in the previous section shows that we cannot hope to achieve the “best of both worlds.” However, we might hope that for every functionality there is a protocol $\pi$ that is secure with an honest majority and is at least private when there is no honest majority. Building on the result of the previous section, we rule out this possibility as well.

Given $n, t, s$ as before, we define a function $f$ that takes inputs from $P_1$ and $P_n$, and returns output to $P_1, P_n$, and also $P_{t+1}$. On input $(b_1, a_0, a_1)$ from $P_1$ and
(b_n, \beta_0, \beta_1) from P_n, where b_1, b_n, \alpha_0, \alpha_1, \beta_0, \beta_1 \in \{0, 1\}, functionality \( \hat{f} \) computes \( v = b_1 \oplus b_n \), gives \( v \) to \( P_1 \) and \( P_n \), and gives \( (v, \alpha_v, \beta_v) \) to \( P_{t+1} \). That is,
\[
\hat{f}((b_1, \alpha_0, \alpha_1), \lambda, \ldots, \lambda, (b_n, \beta_0, \beta_1)) \\
\quad \overset{\text{def}}{=} (b_1 \oplus b_n, \lambda, \ldots, \lambda, (b_1 \oplus b_n, \alpha_{b_1 \oplus b_n}, \beta_{b_1 \oplus b_n}), \lambda, \ldots, \lambda, b_1 \oplus b_n),
\]
output of \( P_{t+1} \)

where we let \( \lambda \) denote an empty input/output.

Let \( \pi \) be a protocol that securely computes \( \hat{f} \) in the presence of a malicious adversary corrupting \( t \) parties. Let \( A, B, C \) be a partition of the parties as in the previous section, and recall that \( P_{t+1} \in B \). Consider an experiment in which \( Z \) chooses inputs for \( P_1 \) and \( P_n \) uniformly and independently, and all parties run protocol \( \pi \) honestly except that parties in \( A \) or \( C \) (but never \( B \)) may possibly abort. An argument exactly as in the previous section shows that there exists a real-world adversary \( A \) who either

- corrupts the parties in \( A \cup B \) and causes \( P_n \) to output some bit \( v \) with probability noticeably greater than \( 1/2 \), or
- corrupts the parties in \( B \cup C \) and causes \( P_t \) to output some bit \( v \) with probability noticeably greater than \( 1/2 \).

Assume without loss of generality that the first case holds with \( v = 0 \), and so there is a fail-stop adversary \( A \) who corrupts the \( s \) parties in \( A \cup B \) and causes \( P_n \) to output 0 with probability at least \( 1/2 + p(k) \) for some noticeable function \( p \). The key observation is that \( A \) causes only the \( t \) parties in \( A \) to abort (in one of two possible segments), and the remaining corrupted parties in \( B \) execute the entire protocol honestly. Since \( \pi \) is secure in the presence of a malicious adversary corrupting \( t \) parties, all parties in \( B \cup C \) will receive their outputs (except possibly with negligible probability) even if all parties in \( A \) abort. Moreover, security of \( \pi \) implies that the output of the honest-looking \( P_{t+1} \) will be consistent with the input and output of the honest \( P_n \) (except with negligible probability). Taken together, this means that the view of \( A \)—which includes the output generated by \( P_{t+1} \)—includes \( \beta_0 \) with probability at least \( 1/2 + p'(k) \) for some noticeable function \( p' \), and furthermore \( A \) knows when this occurs (since the output of \( P_{t+1} \) includes \( v \overset{\text{def}}{=} b_1 \oplus b_n \) in addition to \( \beta_v \)). Thus, \( A \) can output a guess for \( \beta_0 \) that is correct with probability
\[
\frac{1}{2} + p'(k) + \frac{1}{2} \cdot \left( \frac{1}{2} - p'(k) \right) = \frac{3}{4} + \frac{p'(k)}{2}.
\]

In contrast, no ideal-world adversary \( A' \) corrupting \( A \cup B \) can output a guess for \( \beta_0 \) which is correct with probability better than \( 3/4 \) when \( Z \) chooses \( P_n \)'s inputs uniformly at random. This shows that \( \pi \) does not privately compute \( \hat{f} \) in the presence of malicious adversaries corrupting \( s \) parties.

4. Positive results. In this section we show two positive results regarding when a “best of both worlds”-type guarantee is possible. First, we briefly describe a “folklore” protocol for any (reactive or nonreactive) functionality \( f \) that is simultaneously secure against malicious adversaries corrupting any \( t < n/2 \) parties and secure-with-abort against malicious adversaries corrupting \( n - t - 1 \) parties. In light of Cleve’s result [9] and Theorems 3.1 and 3.5, these thresholds are the best possible (for general functionalities).

We next show a protocol whose security is incomparable to the above. Our second protocol is simultaneously secure against malicious adversaries corrupting \( t < n/2 \)
PARTIES AS WELL AS SEMIHONEST ADVERSA RIES CORRUPTING ANY $s < n$ PARTIES. (WE STRESS THAT, IN CONTRAST, TYPICAL PROTOCOLS OFFERING FULL SECURITY AGAINST $t$ MALICIOUS PARTIES ARE COMPLETELY INSECURE AGAINST EVEN $t + 1$ SEMIHONEST PARTIES.) THIS RESULT APPLIES ONLY TO NONREACTIVE FUNCTIONALITIES; AS SHOWN BY THEOREM 3.1, THIS IS INHERENT.

4.1. Achieving the “best of both worlds” for suitable thresholds.

**Theorem 4.1.** Let $n, s, t$ be such that $t + s < n$ and $t < n/2$, and assume the existence of enhanced trapdoor permutations. For any probabilistic polynomial-time (reactive or nonreactive) functionality $f$, there exists a protocol that simultaneously

- securely computes $f$ in the presence of malicious adversaries corrupting $t$ parties, and
- securely computes $f$ with abort in the presence of malicious adversaries corrupting $s$ parties.

**Proof.** If $s \leq t$, the theorem follows from the known results of [11], so assume $s > t$. We begin with the case of nonreactive functionalities. In this case, a protocol $\pi$ with the claimed properties can be derived easily by suitably modifying known protocols that achieve security for an honest majority. In particular, such a protocol $\pi$ can be obtained by following the general approach of [11, Construction 7.5.39] with the following changes:

- The verifiable secret sharing scheme used should have threshold $s + 1$, so that any $s + 1$ shares suffice to recover the secret but any $s$ shares give no information (in at least a computational sense) about the secret.
- The “handling abort” procedure is modified as follows. If a party $P_i$ aborts (or is detected cheating) at some point during the protocol, all remaining parties broadcast their shares of $P_i$’s input and random tape. If at least $s + 1$ valid shares are revealed, the protocol continues with parties carrying out the execution on $P_i$’s behalf. If fewer than $s + 1$ valid shares are revealed, then all parties abort the protocol with output $\perp$.

Note that when $t$ parties are corrupted, at least $n - t \geq s + 1$ valid shares are always revealed during the “handling abort” procedure. Thus, security of $\pi$ in the presence of a malicious adversary corrupting $t$ parties follows using the same analysis as in [11]. When $s$ parties are corrupted, they may cause the protocol to abort but cannot otherwise affect the computation since the sharing threshold is set to $s + 1$.

A standard argument can be used to show that $\pi$ securely computes $f$ with abort in this case.

For reactive functionalities, we proceed as sketched in [11, section 7.7.1.3] with natural modifications. As there, the system state $s^j$ at the end of the $j$th phase of the protocol is shared among the parties; here, however, this sharing is done using threshold $s + 1$ as above. □

4.2. Security against a malicious minority or a semihonest majority.

**Theorem 4.2.** Let $n, s, t$ be such that $t < n/2$ and $s < n$, and assume the existence of enhanced trapdoor permutations. For any probabilistic polynomial-time nonreactive functionality $f$, there exists a protocol that simultaneously

- securely computes $f$ in the presence of malicious adversaries corrupting $t$ parties, and
- securely computes $f$ in the presence of semihonest adversaries corrupting $s$ parties.

**Proof.** Assume $t = \lfloor(n - 1)/2\rfloor$ and $s = n - 1$. We also assume that $f$ is a single-output function, i.e., a function where all parties receive the same output. This is without loss of generality since secure computation of a general functionality
(y_1, \ldots, y_n) = \hat{f}(x_1, \ldots, x_n) can be reduced to secure computation of the single-output functionality \( (y_1 \oplus r_1) \cdots (y_n \oplus r_n) \leftarrow f((r_1, x_1), \ldots, (r_n, x_n)) \), where \( r_i \) is a random pad chosen by \( P_i \) at the outset of the protocol. This reduction is secure for any number of corruptions.

Before describing our protocol \( \pi \) computing \( f \), we first define a related functionality \( \mathcal{SS}_f \) that corresponds to computing an authenticated secret sharing of \( f \) (with threshold \( t + 1 \)). That is, \( \mathcal{SS}_f \) denotes the (randomized) functionality that

1. evaluates \( y = f(x_1, \ldots, x_n) \);
2. computes a \((t+1)\)-out-of-\( n\) Shamir secret sharing \[11\] of \( y \), sending to each party its share \( y_i \); and
3. authenticates each share \( y_i \) for every other party \( P_j \) using an information-theoretic MAC (denoted Mac). The resulting tag is given to \( P_i \), and the key is given to \( P_j \).

See Figure 2 for details.

---

### Functionality \( \mathcal{SS}_f \)

**Parameters:** \( \mathbb{F} \) is a finite field that includes both \([n] \) and the range of \( f \).

**Inputs:** Each party \( P_i \) provides input \( x_i \).

**Computation:**

1. Compute \( y = f(x_1, \ldots, x_n) \) (using random coins if \( f \) is randomized).
2. Choose random \( p_1, \ldots, p_t \in \mathbb{F} \) and set \( p(x) = y + \sum_{i=1}^t p_i x^i \).
3. Set \( y_i = p(i) \) for \( i \in [n] \).
4. Choose random \( k_{i,j} \in \{0,1\}^k \) for all \( i, j \in [n] \).
5. Compute \( \text{tag}_{i,j} = \text{Mac}_{k_{i,j}}(y_i) \) for all \( i, j \in [n] \).
6. Each party \( P_i \) is given as output \( y_i, \{k_{i,j}\}_{j \in [n]}, \) and \( \{\text{tag}_{i,j}\}_{j \in [n]} \).

---

**Fig. 2.** Functionality \( \mathcal{SS}_f \) for authenticated secret sharing of the output of \( f \).

Our protocol \( \pi \) computing \( f \) relies on two subprotocols: a subprotocol \( \pi_n \) that computes \( \mathcal{SS}_f \), and a subprotocol \( \pi_{n/2} \) that computes \( f \). We require the following security guarantees from these protocols:

- \( \pi_n \) securely computes \( \mathcal{SS}_f \) with abort in the presence of malicious adversaries corrupting \( t \) parties and also securely computes \( \mathcal{SS}_f \) in the presence of semihonest adversaries corrupting \( s \) parties.
- \( \pi_{n/2} \) securely computes \( f \) in the presence of malicious adversaries corrupting \( t \) parties.

Protocols with the above properties can be constructed assuming the existence of enhanced trapdoor permutations \[11\]. We describe protocol \( \pi \) in Figure 3.

Intuition for the claimed security properties of \( \pi \) is as follows. Consider first the case of a semihonest adversary corrupting \( s \) parties in some set \( I \). In this case, Phase I always completes successfully (and so Phase II is never executed) and all honest parties learn the correct output \( y \). Moreover, since \( \pi_n \) is secure in the presence of semihonest adversaries corrupting \( s \) parties, the adversary learns nothing from the execution of \( \pi \) other than the secret shares \( \{y_i\}_{i \in I} \), which is equivalent to learning \( y \).

Next, consider the case of a malicious adversary corrupting \( t \) parties in some set \( I \). Here, there are two possibilities:
Protocol π

Inputs: Each party $P_i$ has input $(1^k, x_i)$.

Output: Each party $P_i$ gets output $y = f(x_1, \ldots, x_n)$.

Phase I:
Each party $P_i$ does as follows:

1. Run a protocol $\pi_n$ computing the functionality $SS_f$, using input $x_i$. Let $y_i, \{k_{j,i}\}_{j \in [n]}$, and $\{\text{tag}_{j,i}\}_{j \in [n]}$ denote the output of $P_i$ following execution of this protocol.

2. If $y_i = \bot$, then go to Phase II. Otherwise, do:
   (a) For every other party $P_j$, send $(y_i, \text{tag}_{i,j})$ to $P_j$. Receive in return $(y_j, \text{tag}_{j,i})$ from each other party $P_j$.
   (b) Set $y'_i = y_i$. For $j \neq i$, if $\text{Mac}_{k_{j,i}}(y_j) = \text{tag}_{j,i}$ set $y'_j = y_j$; otherwise, set $y'_j = \bot$.
   (c) Reconstruct the output $y$ using the shares $y'_1, \ldots, y'_n$.

Phase II:
Each party $P_i$ runs protocol $\pi_{n/2}$ computing the functionality $f$, using its original input $x_i$. Each $P_i$ outputs the output value it obtained in $\pi_{n/2}$.

**Claim 4.3.** If $\pi_n$ securely computes $SS_f$ in the presence of semihonest adversaries corrupting $s$ parties, then $\pi$ securely computes $f$ in the presence of semihonest adversaries corrupting $s$ parties.

**Proof.** We analyze $\pi$ in a hybrid model, where the parties have access to a trusted party computing $SS_f$ (this trusted party computes $SS_f$ according to the first ideal model, where the adversary cannot abort the computation) and show that in
this hybrid model \( \pi \) securely computes \( f \) in the presence of semihonest adversaries corrupting \( s \) parties. Standard composition theorems \([5, 11]\) imply the claim.

Let \( \mathcal{A} \) be a semihonest adversary in a hybrid-model execution of \( \pi \) (as described above). We describe the natural semihonest adversary \( \mathcal{A}' \), running an ideal-world evaluation of \( f \), whose behavior provides a perfect simulation of the behavior of \( \mathcal{A} \). Adversary \( \mathcal{A}' \) receives the set of corrupted parties \( I \subset [n] \), their inputs \( \{x_i\}_{i \in I} \), and auxiliary input \( z \). It sends the inputs of the corrupted parties to the trusted party evaluating \( f \) and receives in return an output \( y \). Next, \( \mathcal{A}' \) simply runs steps 2–6 of functionality \( SS_f \) (cf. Figure 2) using the value of \( y \) it obtained. The resulting values \( \{(y_i, \{k_{j,i}\}_{j \in [n]}, \{\text{tag}_{i,j}\}_{j \in [n]})\}_{i \in I} \) are given to \( \mathcal{A} \) as the outputs of the corrupted parties from functionality \( SS_f \). The values \( \{(y_i, \{\text{tag}_{i,j}\}_{j \in [n]})\}_{i \in I} \) are then given to \( \mathcal{A} \) as the messages sent by the honest parties in the final round of Phase I. Finally, \( \mathcal{A}' \) outputs whatever \( \mathcal{A} \) does. It is straightforward to see that the joint distribution of the honest parties’ outputs and the view of \( \mathcal{A} \) (run as a subroutine of \( \mathcal{A}' \)) in the ideal world is identical to the joint distribution of the honest parties’ outputs and the view of \( \mathcal{A} \) in the hybrid world. The claim follows.

**Claim 4.4.** If \( \pi_n \) securely computes \( SS_f \) with abort in the presence of malicious adversaries corrupting \( t \) parties, and \( \pi_{n/2} \) securely computes \( f \) in the presence of malicious adversaries corrupting \( t \) parties, then \( \pi \) securely computes \( f \) in the presence of malicious adversaries corrupting \( t \) parties.

**Proof.** Once again, we analyze \( \pi \) in a hybrid model. Now, the parties have access to two trusted parties:

- a trusted party computing \( SS_f \) according to the second ideal model, where the adversary may prematurely abort the computation; and
- a trusted party computing \( f \) according to the first ideal model, where the adversary cannot abort the computation.

We show that in this hybrid model, \( \pi \) securely computes \( f \) in the presence of a malicious adversary corrupting \( t \) parties. Standard composition theorems \([5, 11]\) imply the claim.

Let \( \mathcal{A} \) be a malicious adversary in a hybrid-model execution of \( \pi \) (as described above). We describe a malicious adversary \( \mathcal{A}' \), running an ideal-world evaluation of \( f \), whose behavior provides a perfect simulation of the behavior of \( \mathcal{A} \). Adversary \( \mathcal{A}' \) receives the set of corrupted parties \( I \subset [n] \), their inputs \( \{x_i\}_{i \in I} \), and auxiliary input \( z \). It passes these values to \( \mathcal{A} \) and then receives inputs \( \{x'_i\}_{i \in I} \) sent by \( \mathcal{A} \) to the trusted party computing \( SS_f \) (recall that \( \mathcal{A} \) operates in the hybrid world where there is a trusted party computing this functionality). \( \mathcal{A}' \) chooses random \( y_i \in \mathbb{F} \) for \( i \in I \) and then runs steps 4–6 of \( SS_f \) (cf. Figure 2). The resulting values \( \{(y_i, \{k_{j,i}\}_{j \in [n]}, \{\text{tag}_{i,j}\}_{j \in [n]})\}_{i \in I} \) are given to \( \mathcal{A} \) as the outputs of the corrupted parties from functionality \( SS_f \). We stress that \( \mathcal{A}' \) has not yet sent anything to its own trusted party computing \( f \).

There are now two subcases, depending on whether or not \( \mathcal{A} \) aborts the computation of \( SS_f \):

- If \( \mathcal{A} \) aborts computation of \( SS_f \), adversary \( \mathcal{A}' \) continues with simulation of Phase II as described below.
- If \( \mathcal{A} \) allows computation of \( SS_f \) to continue, then \( \mathcal{A}' \) sends the inputs \( \{x'_i\}_{i \in I} \) to the trusted party computing \( f \) and receives in return a value \( y \). It interpolates a degree-\( t \) polynomial \( p \) satisfying \( p(0) = y \) and \( p(i) = y_i \) for \( i \in I \), and sets \( y_i = p(i) \) for \( i \notin I \). Finally, it gives \( \{(y_i, \{\text{tag}_{i,j}\}_{j \in I})\}_{i \notin I} \) to \( \mathcal{A} \) as the messages sent by the honest parties in the final round of Phase I and outputs whatever \( \mathcal{A} \) outputs.
If $A$ had aborted computation of $A_f$, then $A'$ now continues with simulation of Phase II. $A$ receives (possibly different) inputs $\{x''_i\}_{i \in I}$ from $A$, sends these to its trusted party computing $f$, and receives in return an output value $y$. It gives $y$ to $A$ and outputs whatever $A$ outputs.

It is not hard to verify that the joint distribution of the honest parties’ outputs and the view of $A$ (run as a subroutine of $A'$) in the ideal world is statistically close to the joint distribution of the honest parties’ outputs and the view of $A$ in the hybrid world (where the only difference arises from the possibility that $A$ manages to forge a tag on an invalid share). The claim follows.

Since protocols $\pi_n$ and $\pi_{n/2}$ with the desired security properties can be constructed from enhanced trapdoor permutations, the preceding claims complete the proof of the theorem.

Remark 4.5 (realizing adaptive security and/or universal composability.). Theorem 4.2 refers to our default model of stand-alone security against static adversaries. It is not hard to see, however, that protocol $\pi$ described in the proof of that theorem can be proved secure against adaptive adversaries and/or universally composable [6] if the underlying protocols $\pi_n, \pi_{n/2}$ satisfy these stronger security properties.

REFERENCES


