On Combining Privacy with Guaranteed Output Delivery in Secure Multiparty Computation*

Yuval Ishai[1], Eyal Kushilevitz[1], Yehuda Lindell[2]**, and Erez Petrank[1]

[1] Technion ({yuvali,eyalk,erez}@cs.technion.ac.il)
[2] Bar-Ilan University (lindell@cs.biu.ac.il)

Abstract. In the setting of multiparty computation, a set of parties wish to jointly compute a function of their inputs, while preserving security in the case that some subset of them are corrupted. The typical security properties considered are privacy, correctness, independence of inputs, guaranteed output delivery and fairness. Until now, all works in this area either considered the case that the corrupted subset of parties constitutes a strict minority, or the case that a half or more of the parties are corrupted. Secure protocols for the case of an honest majority achieve full security and thus output delivery and fairness are guaranteed. However, the security of these protocols is *completely compromised* if there is no honest majority. In contrast, protocols for the case of no honest majority do not guarantee output delivery, but do provide privacy, correctness and independence of inputs for *any number* of corrupted parties. Unfortunately, an adversary controlling only *a single party* can disrupt the computation of these protocols and prevent output delivery.
In this paper, we study the possibility of obtaining general protocols for multiparty computation that *simultaneously* guarantee security (allowing abort) in the case that an arbitrary number of parties are corrupted *and* full security (including guaranteed output delivery) in the case that only a minority of the parties are corrupted. That is, we wish to obtain the best of both worlds in a single protocol, depending on the corruption case. We obtain both positive and negative results on this question, depending on the type of the functionality to be computed (standard or reactive) and the type of dishonest majority (semi-honest or malicious).

1 Introduction

1.1 Background

Secure multiparty computation (MPC) [43, 28, 6, 13] allows a set of mutually distrusting parties to compute a function in a distributed way while guaranteeing (to the extent possible) the privacy of their local inputs and the correctness of the outputs. To be more exact, security is typically formulated by comparing a real execution of a protocol to an ideal execution where the parties just send

* Research supported by grant 36/03 from the Israel Science Foundation.
** Much of this work was carried out while the author was visiting the Technion.

their inputs to a trusted party and receive back their outputs. A real protocol is said to be secure if an adversary can do no more harm in a real execution than in an ideal execution (which is secure by definition). The main security properties that have been identified, and are implied by this formulation, are: privacy (parties learn nothing more than their output), correctness (the outputs are correctly computed), independence of inputs (parties must choose their inputs independently of the other parties), fairness (if one party receives its output, all parties receive their output), and guaranteed output delivery (the honest parties are guaranteed to successfully complete the computation).

The vast body of research in this area can be divided into two, almost disjoint, lines of work: one dealing with the case of an *honest majority* and another dealing with the case of *no honest majority*. There are inherent differences between the two settings both in terms of the type of security guarantees one can hope to obtain and in terms of the techniques used for implementing protocols. We elaborate on this below.

Secure computation with honest majority. If a majority of the participants are honest, then it is possible to obtain the strongest type of security one could hope for (including all of the properties described above). In fact, it is even possible to obtain this level of security unconditionally, without relying on unproven complexity assumptions (as long as we do assume secure point-to-point channels and either the existence of a broadcast channel [41] or of a trusted preprocessing phase [40][1]). Protocols for the honest-majority case include unconditionally secure protocols from [6, 13, 41, 3, 17, 23, 15, 31] and computationally secure protocols from [28, 5, 18, 19, 20, 25]. In this setting, n parties can compute an arbitrary polynomial-time computable function of their inputs, while providing unconditional security against $s < n/2$ malicious parties [41, 15]. Moreover, settling for computational security, the same can be done in a constant number of rounds (assuming that one-way functions exist) [5, 20]. It is known that the bound $s < n/2$ is tight even for simple functionalities such as coin-flipping [14] and even if one settles for computational security. We refer to protocols with security against $s < n/2$ malicious parties as being of type $\mathcal{P}_{n/2}$.

Secure computation with no honest majority. If a half or more of the participants *may* be corrupted, then it is impossible to construct protocols that guarantee (full) fairness, let alone output delivery. Rather, all protocols must allow the adversary to *abort* the protocol (possibly, even after first learning its output, but before the honest parties receive their output) [14].[2] We will refer to this type of security as "security with abort" (and will sometimes call security for the case of an honest majority full security in order to clarify which level of security is discussed). Unlike the honest majority case, protocols in the case of no honest majority can only guarantee *computational* security and need to rely on crypto-

[1] The assumptions of broadcast or trusted preprocessing are not needed in the case where more than two thirds of the parties are honest [6, 13].

[2] In fact, the protocols can guarantee that the latter fairness problem occurs only if some specified distinguished party, say the first, is corrupted (cf. [27, 30]).

graphic assumptions. Accordingly, the techniques used for obtaining such protocols involve cryptographic tools such as zero-knowledge proofs, commitment, and oblivious transfer. Protocols in this setting include the two-party protocols of [43, 28, 36, 39, 33] and the multi-party protocols of [28, 4, 29, 34, 30, 12, 38, 2]. In this setting, n parties can compute any polynomial-time computable function of their inputs, while providing security with abort against $t < n$ malicious parties [43, 28] (assuming enhanced trapdoor permutations exist).[3] This can also be done in a constant number of rounds (assuming that trapdoor permutations and collision-resistant hash functions exist) [36, 34, 38]. We refer to protocols that are secure with abort against $t < n$ malicious parties as being of type $\mathcal{P}_n$. We stress that, although output delivery and fairness are not guaranteed, there is no compromise on privacy, correctness and independence of inputs.

A major *qualitative advantage* of type $\mathcal{P}_n$ protocols is that they provide security (albeit with abort) against an *arbitrary* number of corrupted parties. In particular, each party is assured that the *privacy* of its input is maintained even if *all the rest of the world* conspires against it (the same holds for correctness and independence of inputs as well). Thus, the privacy of each party is entirely in its own hands. On the other hand, a major *qualitative disadvantage* of type $\mathcal{P}_n$ protocols is that they fail to be resilient even against a single malicious party. In particular, a single malicious party can abort the protocol and prevent honest parties from receiving any outputs. This disadvantage becomes more severe as the number of parties grows, to the point of rendering the guarantee of security with abort essentially useless in a system with many parties. (Consider, for instance, a voting system in which each voter can disrupt the elections.)

The above state of affairs raises the following natural question:

> *Is it possible to construct a single protocol for general MPC that will provide security with abort (or even just privacy) in the case that an arbitrary number of parties are corrupted and,* simultaneously, *will provide full security (including guaranteed output delivery) in the case that only a minority (or even just one) of the parties are corrupted?*

In other words, is it possible to combine the main qualitative advantages of the two types of protocols? To the best of our knowledge, the above fundamental question has not been studied before. This is a major gap in our current understanding of the feasibility of secure computation.

1.2 Our Results

We initiate the study of the above question, obtaining both negative and positive results. We distinguish between standard and reactive secure computation

[3] Here too, one either needs to rely on the availability of a broadcast channel, or alternatively rely on some setup assumption such as PKI. Another alternative is to relax the requirement of a *unanimous* abort, allowing a situation where only some of the honest parties abort [30, 22].

tasks. A standard (or non-reactive) functionality receives a single input from every party and provides a single output to every party. This corresponds to the usual notion of secure function evaluation. In contrast, a reactive functionality can receive inputs and provide outputs in multiple rounds, maintaining a state information between subsequent invocations. (For example, a commitment functionality is reactive because it has distinct "commit" and "reveal" phases.) We also distinguish between two main variants of the above question, depending on the type of adversary that is present. We will consider semi-honest and malicious adversaries. A semi-honest adversary follows the protocol specification but may try to infer additional information about the honest parties' inputs from what it sees. In contrast, a malicious adversary may instruct the corrupted parties under its control to arbitrarily deviate from the protocol specification. In order to strengthen our negative results, we will relax the requirements and allow the protocol to achieve only privacy in the case of no honest majority (without requiring the full definition of security-with-abort).

A negative result for reactive functionalities. We observe that the answer to the above question is negative for the case of reactive functionalities. This is proved by considering a type of *commitment* or *secret-sharing* functionality and showing that it is impossible to achieve the requirements for this specific functionality. The intuition behind this result is that if a secret is to be shared between n parties with maximal privacy (i.e., so that no strict subset of the parties can recover the secret), then even a single party can prevent a subsequent reconstruction of the secret. This is due to the fact that any one party can just withhold its share and refuse to participate in the secret reconstruction phase. Since the remaining $n-1$ parties should not be able to recover the secret in the case that they are corrupted, they cannot recover it when they are honest here as well. Thus, it is impossible to construct a protocol for secret sharing that achieves security (or security with abort) against $n-1$ *semi-honest* (and of course malicious) parties, and full security (including guaranteed output delivery) against just a single malicious party. Our proof is motivated by the above-described intuition, but the details are somewhat different. In particular, our proof uses a non-standard formalization of secret-sharing as a *reactive* functionality. We note that some standard formulations of verifiable secret-sharing from the literature (e.g., the one from [27]) are non-reactive, and our proof does not hold for these formulations.

A positive result for standard functionalities. In light of the above, we restrict our attention from now on to the evaluation of standard (non-reactive) functionalities. In this case, we show the following positive result: any n-party functionality f can be computed by a protocol that *simultaneously* achieves the following two security properties: (1) full security in the presence of any minority of *malicious* parties; and (2) full security in the presence of any number of *semi-honest* parties (we obtain full security here even in the case of no honest majority because the adversary is semi-honest and so never aborts). Moreover, such a protocol can be implemented in a constant number of rounds. (The above results can be based on standard cryptographic assumptions.) Thus, in contrast to the case of reactive functionalities, our main question can be answered affirmatively for the case where privacy is only required to hold against semi-honest parties.

The above positive result is shown via the following simple protocol: first, the parties use a type $\mathcal{P}_n$ protocol to evaluate a randomized functionality $\hat{f}$ that outputs an *authenticated secret-sharing* of the output of f for which the shares of $< n/2$ colluding parties yields no information about the secret. If this stage terminates successfully (without aborting), then the parties exchange their outputs of $\hat{f}$, from which they can recover the output of f. In case the adversary aborted the computation of $\hat{f}$, the parties use a protocol of type $\mathcal{P}_{n/2}$ to evaluate f. This combination of the two types of protocols combines their relative advantages (in the sense discussed above), but falls short of providing privacy against $t < n$ *malicious* parties. We also describe an interesting variant of this protocol that can provide a relaxed form of full privacy even in the malicious case. Specifically, in addition to security against $s < n/2$ malicious parties, it can be guaranteed that any coalition of $t < n$ malicious parties can learn no more information than is implied by $t + 1$ invocations of the functionality. For some natural functionalities (like a voting functionality), this relaxed notion of privacy is equivalent to the standard notion of privacy (see the end of Section 3 for discussion). Thus, in some interesting cases, our main question can be answered affirmatively even for malicious adversaries, as long as only privacy (and not security) is sufficient.

A negative result for standard functionalities. We finally turn to the most challenging remaining case of standard functionalities and where the adversary can be *malicious* both in the case of an honest majority and in the case of no honest majority. Arguably, as far as the corruption model is concerned, this is the most interesting case. Unlike the previous cases, here we do not completely settle the question. Instead, we prove that there do not exist *constant round* protocols for general secure multiparty computation that are secure for the case of an honest majority and private in the case of no honest majority (we actually show that the number of rounds must grow with the security parameter). We note that constant-round protocols *do exist* for type $\mathcal{P}_{n/2}$ protocols and for type $\mathcal{P}_n$ protocols alone, as well as for the setting of our above positive result. Thus, our negative result here provides a nontrivial separation in terms of round complexity. We leave open the possibility of protocols that require a non-constant number of rounds in this setting.

Summary. We stress that all of our results apply to the *computational* model of security. They are also quite insensitive to whether or not a broadcast primitive is given. In particular, all negative results apply even when a broadcast channel is assumed, and a variant of the positive results (replacing $s < n/2$ by $s < n/3$) can be obtained also without broadcast or any setup assumptions. Finally, we stress again that we strengthen our negative results by showing that even privacy alone (rather than security-with-abort) cannot be achieved. Our main results are summarized in Table 1.

	Standard functionalities	Reactive functionalities
Privacy against $t < n$ semi-honest parties and security against $s < n/2$ malicious parties	Yes (constant-round) Theorem 1	No Theorem 4
Privacy against $t < n$ malicious parties and security against $s < n/2$ malicious parties	No constant-round Theorem 2	No Theorem 4

Table 1. Summary of main results: the existence and non-existence of protocols that simultaneously guarantee security against $s < n/2$ malicious parties and privacy against $t < n$ malicious or semi-honest parties. Negative results apply whenever $s + t \geq n$, and in particular when $s = 1$ and $t = n - 1$. Our positive result actually provides security for the semi-honest case.

1.3 Related Work

There is a significant body of work that deals with extending the basic feasibility results for MPC in various directions. We mention some of this work below and explain the differences from ours.

Multi-threshold MPC. Several previous works consider the possibility of refining known feasibility results for MPC by allowing different security thresholds for different types of corruption [23, 22, 24]. The focus of these works is very different from ours. In particular, they all fall into the setting of an honest majority, while we study the situation where honest majority cannot be guaranteed. Moreover, in contrast to [23, 24], in this work we only deal with *computational* security and do not insist on any kind of unconditional security.

Fair MPC. Recall that a secure MPC protocol is said to be *fair* if at the end of the protocol either all parties learn the output or no party learns the output. (See [30] a for a comprehensive discussion of fairness and its variants.) It is known that even simple functionalities such as coin-flipping or XOR cannot be fairly computed in the case of no honest majority [14]. In contrast, in this work, in the setting of no honest majority we settle for privacy or security-with-abort which do not imply fairness. Thus, our negative results are not implied by impossibility results that relate to fairness. In light of the fact that "full fairness" cannot be achieved without an honest majority, there has been a considerable amount of work suggesting possible alternatives. The *gradual release* technique [21, 7, 26] can guarantee the following partial form of fairness: if the adversary quits the protocol early, then the adversary and the honest parties must invest comparable amounts of time in order to recover the outputs. This approach departs from traditional cryptographic protocols in that it does not set an a-priori bound on the resources invested by *honest* parties. In contrast, in this work, we consider the traditional setting in which the running time of honest parties is bounded by some fixed polynomial in the security parameter. Other approaches for getting around the impossibility of fairness involve relaxed notions of fairness [37, 4, 29], or employing semi-trusted third parties or physical assumptions [1, 8, 35]. None of these approaches applies to the standard model we consider here.

Organization. In Section 2, we provide some necessary definitions. In Section 3, we present our positive result for the case of standard (non-reactive) functionalities. In Section 4, we present a negative result for standard functionalities and, in Section 5, we present stronger negative results for the case of reactive functionalities. Due to lack of space, some of the material (including proofs) were omitted from this extended abstract and can be found in the full version.

2 Preliminaries

The Model. Our default network model consists of n parties, $P_1, \ldots, P_n$, who interact in synchronous rounds via authenticated secure point-to-point channels and a broadcast medium. We note, however, that the question we consider and our results are quite insensitive to the network model and in particular to the availability of broadcast, as discussed in the end of Section 1.2. Similarly, our positive results hold even when the adversary is allowed *rushing* (i.e., sending its messages at a given round only after receiving all messages sent by honest parties in the same round), whereas our negative results hold even if all messages of a given round are delivered simultaneously. This should be contrasted with the known impossibility result for coin-flipping [14], which crucially depends on the adversary's rushing capability. Indeed, flipping coins becomes a trivial task if the network can enforce simultaneous message exchange.

Throughout the paper, we consider *computational* security against computationally bounded adversaries. The security parameter will be denoted by k. We assume for simplicity that the adversary is *static*, namely that the set of corrupted parties is chosen at the onset of the protocol in a non-adaptive manner. This strengthens our negative results, and is not essential for our positive results.

Finally, we consider both *malicious* adversaries, who have total control over the behavior of corrupted parties and may thus arbitrarily deviate from the protocol specification, and *semi-honest* adversaries, who can record all information viewed by corrupted parties but do not otherwise modify their behavior. We will sometimes also refer to *fail-stop* adversaries which can only deviate from a semi-honest behavior by aborting, namely stopping to send messages. Unless stated otherwise, adversaries are assumed to be malicious.

Defining security. We assume some familiarity of the reader with the basic simulation-based approach for defining secure computation, as described in detail in [9, 27]. Generally speaking, this definitional approach compares the *real-life* interaction of an adversary with the protocol and its interaction with an *ideal* function evaluation process in which a trusted party is employed. Security is then defined by requiring that whatever can be achieved in the real model could have also been achieved (or *simulated*) in the ideal model. In other words, for every adversary A attacking the real execution of the protocol there exists an adversary A', sometimes referred to as a *simulator*, which can "achieve the same effect" by attacking the ideal function evaluation process. The two main types of security considered in this work differ in how the ideal model is defined.

Ideal Model – without abort. This variant of the ideal model corresponds to the strong notion of security (with guaranteed output delivery) that is achieved by type $\mathcal{P}_{n/2}$ protocols. In this case, the interaction of the simulator A' with the ideal evaluation of f is very simple:

- Honest parties send their input x_i to the trusted party. Corrupted parties may send the trusted party arbitrary inputs as instructed by A'. Denote by x'_i the value sent by party P_i. In the case of a semi-honest adversary, we require that $x'_i = x_i$.
- The trusted party computes $f(x'_1, \ldots, x'_n) = (y_1, \ldots, y_n)$ and sends the value y_i to party P_i. (If f is randomized, this computation involves random coins that are generated by the trusted party.) Any missing or "invalid" value x'_i is substituted by a valid default value (say 0) before evaluating f.

Ideal Model – with abort. In this case, a malicious (or even a fail-stop) adversary can abort the computation in the ideal model *after learning its outputs*.[4] We use $\perp$ to represent the output of honest parties resulting from the adversary aborting the protocol and let $I \subset [n]$ denote the set of corrupted parties. The following definition of the ideal model follows that of [34]:

- The parties send their inputs to the trusted party. As before, we let x'_i denote the (possibly modified) value sent by party P_i.
- The trusted party computes $f(x'_1, \ldots, x'_n) = (y_1, \ldots, y_n)$ and sends to each corrupted party P_i, $i \in I$, its output y_i.
- Based on the values $(y_i)_{i \in I}$, received in the previous step, the adversary chooses whether to continue the ideal function evaluation process or to abort. In the former case, the trusted party sends to each uncorrupted party P_i, $i \notin I$, its output value y_i. In the latter case, each party P_i such that $i \notin I$ receives the special value $\perp$ indicating that the adversary chose to abort.

When referring to constant-round protocols, we allow simulators to run in *expected* polynomial time (cf. [34]); our negative results hold even if the simulator's running time is unbounded.

Security versus privacy. In the literature on secure computation, privacy is often synonymous with security against a semi-honest adversary. Here we also refer to privacy against *malicious* adversaries, in which case we consider the privacy property in isolation. Privacy requires that the adversary's view when attacking the real model can be simulated in the ideal model. This should be contrasted with the (stronger) standard security definitions, which consider the joint distribution of the adversary's view and the outputs of uncorrupted

[4] It is possible to realize a somewhat stronger notion of security in which the adversary can abort the protocol *after learning the outputs* only if it corrupts some predefined party, say P_1. In this case, the adversary should be given the opportunity to abort the protocol *before learning the output* even when the corrupted parties do not include P_1. Since the distinction between these two variants does not effect our results, we prefer the simpler notion.

parties in both the real and ideal models. We note that considering privacy on its own is often not sufficient. In particular, as discussed in [9], privacy alone for randomized functionalities does not capture our intuition of what privacy *should* mean. Moreover, privacy against a *malicious* adversary (in contrast to privacy against a semi-honest adversary) does not compose. Nevertheless, it is a meaningful notion of security that captures the adversary's inability to learn more than it should about the *inputs* of the honest parties. Furthermore, in this work we mainly use privacy for our negative results, thereby strengthening them.

Authenticated secret sharing. Our positive results will make use of *authenticated secret sharing* [41, 16] (also known as "honest-dealer VSS" or "weak VSS"). An s-secure authenticated secret sharing scheme allows an honest dealer to distribute a secret σ among n players such that an adversary corrupting up to s players learns nothing about σ from its shares. Moreover, even by modifying its shares, the adversary cannot prevent honest players from later reconstructing the correct value σ, except with negligible probability. An authenticated secret sharing scheme is defined by a pair $\mathcal{S} = (\mathcal{D}, \mathcal{R})$ of efficient algorithms, where $\mathcal{D}(1^k, 1^n, \sigma)$ is a randomized secret distribution algorithm outputting an n-tuple of shares, and $\mathcal{R}(\sigma_1, \ldots, \sigma_n)$ is a reconstruction function recovering the secret from the (possibly modified) shares. We will also need to rely on the existence of an efficient *share completion* algorithm which, given a secret σ and a set of $s' \leq s$ shares, can sample the remaining $n - s'$ shares conditioned on the given information. A formal definition of authenticated secret-sharing can be found in the full version. We rely on the following fact.

Fact 1 *(implicit in [41]) There exists an s-secure n-party authenticated secret sharing scheme for any $s < n/2$.*

3 Positive Result for Standard Functionalities

In this section, we present a protocol demonstrating the possibility of obtaining the best of both worlds for standard (non-reactive) functionalities, as long as the adversary is semi-honest in the case of no honest majority. That is, we obtain a protocol that provides *full security* against a minority of malicious parties, and *full security* against any number of semi-honest parties.

Theorem 1. *Assume the existence of enhanced trapdoor permutations. Then, for any probabilistic polynomial-time computable (non-reactive) functionality f, there exists a* single *protocol π such that:*

1. *π securely computes f in the presence of an s-bounded* malicious *adversary, for any $s < n/2$;*
2. *π securely computes f in the presence of a t-bounded* semi-honest *adversary, for any $t < n$.*

Furthermore, if collision resistant hash functions exist, the protocol π can be constructed with just a constant number of rounds.

We prove the theorem by presenting a protocol that achieves the desired security features. Assume, without loss of generality, that the (possibly randomized) functionality f has a single output that is given to all parties (see Remark 1 below). We also assume without loss of generality that the output length of f, denoted by ℓ, is determined by k. We use the fact that known protocols that securely compute f with abort in the presence of an $(n-1)$-bounded malicious adversary are fully secure in the presence of an $(n-1)$-bounded semi-honest adversary. Our protocol proceeds as follows.

Protocol 1 multiparty computation secure against a minority of malicious parties and against any number of semi-honest parties:

- **Inputs:** *Each party* P_i, $1 \leq i \leq n$, *has an input* x_i. *All parties hold a security parameter* k.
- **Output:** *Each party* P_i *would like to obtain* $y = f(x_1, \ldots, x_n)$.
- **Parameters:** *Security thresholds* $s = \lfloor (n-1)/2 \rfloor < n/2$ *and* $t = n - 1$.
- **The protocol:**

  Phase I:

  1. *The parties begin by computing an authenticated secret-sharing of* $f(x_1, \ldots, x_n)$. *Specifically, let* $\mathcal{S} = (\mathcal{D}, \mathcal{R})$ *be an* s*-secure authenticated secret sharing scheme (see Section 2), and let* $\hat{f}$ *be the randomized functionality defined by*

     $$\hat{f}(1^k, x_1, \ldots, x_n) = \mathcal{D}(1^k, 1^n, f(x_1, \ldots, x_n)).$$

     Let π_{n} *be a protocol that securely computes* $\hat{f}$ *with abort in the presence of* t*-bounded malicious adversaries (and is fully secure in the presence of* t*-bounded semi-honest adversaries). The parties invoke* π_{n} *on their inputs* x_i*, obtaining local outputs* y_i.
  2. *If Phase I terminated successfully (with honest parties outputting values* $y_i \neq \perp$*), each party* P_i *sends* y_i *to all other parties and outputs* $y = \mathcal{R}(y_1, \ldots, y_n)$. *Otherwise (if Phase I terminated with "abort"), proceed to Phase II below.*

  Phase II:

  Let $\pi_{\mathrm{n}/2}$ *be protocol that securely computes* f *in the presence of* s*-bounded malicious adversaries. The parties invoke* $\pi_{\mathrm{n}/2}$ *on their original inputs* x_i*, and output their outputs in this invocation.*

Notice that even a single malicious party can abort the computation of π_{n} in the first phase. However, in such a case all the honest parties output $y_i = \perp$ and proceed to the next phase where the protocol is fully secure in the presence of s-bounded malicious adversaries. The key observation used in constructing the protocol is that *either* the adversary is semi-honest, in which case the protocol will terminate in phase I, *or* the adversary is malicious, but then we know that it is s-bounded. Thus, it will learn nothing in the first phase, and the protocol in the second phase will yield the desired result.

Remark 1 (Single output vs. multiple outputs). In Protocol 1, we assume that the functionality f has a single output y which is given to all parties. To handle general functionalities, we can use the standard reduction where the output includes a concatenation of n values $y_i \oplus r_i$, where y_i is the local output of P_i, and r_i is a mask randomly selected by P_i and included in its input. It is not difficult to show that this reduction is secure for any number of corruptions.

In the full version of this paper, we prove the following:

Lemma 1. *Protocol 1 is secure against semi-honest t-bounded adversaries for any $t < n$.*

Lemma 2. *Protocol 1 is secure against malicious s-bounded adversaries, for any $s < n/2$.*

To conclude the proof of Theorem 1, we note that protocols $\pi_{\mathrm{n}/2}, \pi_{\mathrm{n}}$ as needed indeed exist. That is, we can take $\pi_{\mathrm{n}/2}$ to be the protocol of [28] and π_{n} to be the protocol of [41] (obtaining a non-constant number of rounds), or we can take $\pi_{\mathrm{n}/2}$ to be the protocol of [38] and π_{n} to be the protocol of [5] (obtaining a constant number of rounds, but additionally requiring the existence of collision-resistant hash functions). □

In the full version we discuss extensions of Theorem 1 to the cases of adaptive security and UC security.

Remark 2 (On strengthening semi-honest security). In the case of no honest majority ($n/2 \le t < n$), Theorem 1 only refers to security against a semi-honest adversary. However, the security guarantee provided by Protocol 1 in this case is somewhat stronger. In particular, for any *non-aborting adversary* (i.e., an adversary that does not cause the honest parties to abort), the computation will be completed in Phase I, and privacy is thus preserved in this case. Since an adversary that aborts may be identified as misbehaving, real adversarial parties may have a lot to lose by causing an abort in Phase I. Note, however, that Protocol 1 *does not* generally satisfy security-with-abort, or even privacy, against $t < n$ *malicious* parties. Indeed, a malicious majority can learn the entire input in Phase II. In Section 4 we show that a similar type of insecurity is inherent to constant-round protocols.

An interesting protocol variant. A useful property of most known protocols of type $\mathcal{P}_n$ (e.g., the protocols in [27] or [34]) is that if the protocol aborts, then a corrupted party is identified (where the same corrupted party is identified by all honest parties). This feature gives rise to the following variant of Protocol 1, that obtains a meaningful notion of security even against a *majority of malicious* parties (in addition to full security against a minority of malicious parties):

> Repeatedly run Phase I, disqualifying a bad party at each iteration until no aborts occur. (If a party is disqualified, its input is replaced by a default value.) Then, all remaining parties exchange their authenticated shares and reconstruct the output of f as in Step 2 of Protocol 1.

We first note that this protocol may run at most $t + 1$ iterations, as one corrupted party is eliminated in each repetition. For this variant of Protocol 1, an interesting security guarantee can be made. First, similarly to Protocol 1, the protocol is secure against $s < n/2$ malicious parties. Moreover, even if a malicious adversary corrupts $t < n$ parties, this adversary can be simulated in a relaxed ideal model where the simulator can invoke the functionality up to $t+1$ times. In each such invocation, the simulator gets to learn the outputs first, and then decides whether to abort (and continue to the next invocation) or to deliver the outputs to the uncorrupted parties (in which case the protocol terminates). If it gets to invocation $t + 1$, then it must deliver the output.

For many natural functionalities the above type of security is quite reasonable, and in particular may imply privacy. For instance, in the case of the basic voting functionality (sum of 0/1 values), the above protocol is fully private against a malicious adversary, because t invocations of the functionality are equivalent to a single invocation in terms of the information learned by the simulator. This holds because in a single invocation the adversary can learn exactly how many honest parties voted 1 and how many voted 0. Future invocations, where the adversary changes the votes of the corrupted parties, will yield no additional information. Thus, using this approach, we actually get a voting protocol with security against $s < n/2$ and privacy against $t < n$ *malicious* parties. We stress that even for this specific functionality the above protocol falls short of providing full security, or even security with abort, against a malicious majority. For instance, corrupted parties can force the vote tally to be even, or force a tie between the candidates. Note, however, that obtaining such an unfair advantage is possible only when a *majority* of the voters are maliciously corrupted. In such a case, protecting the *privacy* of the honest minority is typically the most important concern. This concern is completely resolved by our protocol, without compromising full security in the (more likely) case of a malicious minority.

4 Negative Result for Standard Functionalities

In this section we prove our main negative result for standard (non-reactive) functionalities. Specifically, we demonstrate the existence of a 3-party functionality for which there does not exist any *constant-round* protocol that is both secure against one malicious party and private against two malicious parties. (A generalization to n parties is presented later.) The proof of this result is quite subtle; we first describe the general idea at a high level.

Denote the 3 parties by A, B, C and their respective inputs by a, b, c. We consider a functionality f which, on randomly chosen inputs, allows a coalition of B and another party (A or C) to learn *partial* information about the input of the remaining party, but not everything. Let q be a round number such that the following holds: After q rounds, A and B can learn from the honest execution of the protocol the *maximal* amount of information about c allowed by the the ideal evaluation of f. On the other hand, after $q-1$ rounds B and C can only obtain a substantially smaller amount of information about a. (If no such such q exists, we can exchange the roles of A and C and the rest of the argument proceeds symmetrically.) An adversary corrupting A and B can now break the privacy of the protocol as follows. It acts honestly for the first $q-1$ rounds. Starting from round q, it makes A stop sending any messages but keeps B acting honestly. At this stage, the adversary has already learned the maximal amount of information about c allowed by the ideal model. However, the protocol must continue as if only A is corrupted, in which case the outputs of B and C should correspond to substituting the input of A by some default value. (Here we use the fact that the protocol should be fully secure in the case that only A is corrupted.) The crucial point is that B and C do not have enough information about a to guarantee that it is always used as the default value. Thus, with some noticeable probability, B learns an output of f on (a', b, c) for some $a' \neq a$. This new information can be used by the adversary to learn more about c than is allowed in the ideal model.

We now proceed with the formal statement of the result and its proof.

Theorem 2. *There exists a finite 3-party functionality f for which there is no constant-round protocol π such that:*

- *π securely computes f in the presence of a malicious adversary corrupting one party;*
- *π privately computes f in the presence of a malicious adversary corrupting two parties.*

This holds even if the malicious adversary is restricted to be of a fail-stop type.

The proof of Theorem 2 relies on a technical definition of information yielding defined below. We will consider honest executions of the protocol on random inputs, and would like to measure the "amount of information" some partial view yields on an input. For this, it will be convenient to use the following measurement. Intuitively, it measures the maximal probability of guessing the input correctly *without being wrong* (except with probability that tends to 0 as k tends to infinity), but possibly outputting "don't know".

Definition 1 *Let $(X(k), V(k))$ be an ensemble of (possibly) correlated random variables and let $0 \leq p \leq 1$ be a constant. We say that V yields X with probability (at least) p if there exists a polynomial-size prediction algorithm P such that $\Pr[P(V(k)) = X(k)] \geq p - \epsilon(k)$, and $\Pr[P(V(k)) \notin \{X(k),$ "don't know"$\}] \leq \epsilon(k)$ for some $\epsilon(k)$ that tends to 0 as k tends to infinity.*

In the following proof we will use Definition 1 with X being an input to a protocol and V a corresponding view obtained by running the protocol with security parameter k. (In our specific example, X will not depend on k.)

Proof Sketch: We prove Theorem 2 by exhibiting a specific functionality f as required. This functionality involves three parties, A, B, C, where A's input is denoted a, C's input is c and B has no input. The functionality is defined by $f(a, \perp, c) = (\perp, out, \perp)$, where $out = c$ if $a = c$, and $out = *$ otherwise. The inputs a, c are taken from the set $\{0, 1, 2\}$. We will argue that there is no constant-round protocol that computes f with full security against a single malicious party and privacy against two malicious parties.

Suppose, by way of contradiction, that π is a protocol for f that obtains both privacy against two parties and security against one. Now, consider the honest execution of π on inputs a and c that are selected uniformly and independently at random in the domain $\{0,1,2\}$. For an honest run of the protocol, we denote by q_{AB} the first round after which the view of the coalition of A and B on the protocol yields c (in the sense of Definition 1) with probability $1/3$. We know that $q_{AB} > 0$, since initially it is impossible for A and B to guess c without error. We also know that q_{AB} is no larger than the (constant) number of rounds in the protocol, since eventually even the view of B alone must yield c with probability $1/3$. Thus, q_{AB} is well defined. Define q_{BC} symmetrically, i.e., the first round in which the view of B and C yields a with probability $1/3$. Assume, without loss of generality, that $q_{AB} \leq q_{BC}$. In other words, A and B learn c at no later than B and C learn a. The proof for the case $q_{AB} \geq q_{BC}$ proceeds symmetrically.

Let $q = q_{AB}$, i.e., after q rounds the joint views of A and B yields c with probability $p_1 = 1/3$, where the probability is taken over the choice of a, c and the random coin tosses of all parties. Let P be a corresponding prediction algorithm (of c from the view of A and B), and denote by G_1 the event that P does not output "don't know", given the view of A and B after q rounds. By the definition of q and P, and relying only on the fact that $\Pr[P(V(k)) = X(k)] \geq p - \epsilon(k)$, it follows that

$$\Pr[G_1] \geq 1/3 - \epsilon(k) \tag{1}$$

After $q-1$ rounds, the view of B and C does not yield the value of a with probability $1/3$. This means that, for any poly-size predictor P' that is wrong with probability at most $\epsilon'(k)$ (for some ϵ' that tends to 0 as k tends to infinity), there is some constant $p_2 < 1/3$ and infinitely many k's such that the probability that P''s output is not "don't know", based on the view of B and C in the first $q-1$ rounds, is at most p_2. (To be consistent with Definition 1, note that p_2 equals some $p' - \epsilon'(k)$ where $p' < 1/3$.)

Note that the assumption that p_2 is bounded away from $1/3$ is where we use the assumption that the protocol has a constant number of rounds. A more fundamental reason for the restriction on the number of rounds is the following. We will need the information obtained in round q to be significantly better than the information obtained in round $q-1$, and furthermore will need the difference between the two amounts of information to be bigger than the difference between round q and the maximal amount of information. The difficult case of non-constant-round protocols, for which the above property cannot be met, is when after m rounds the view yields the input with probability $1/3 - 2^{-m}$. This suggests that our general proof approach cannot go beyond a logarithmic number of rounds.

From here on, we consider the following experiment. Pick the two inputs a, c uniformly and independently at random. The adversary attacks the protocol, when invoked on uniformly random inputs, as follows. It corrupts A and B, and after $q-1$ rounds makes A abort. (That is, the last message sent by A is that sent in Round $q-1$.) Define the event G_1 as above, based on the view of A and B in the first q rounds (note that the joint view of A and B depends only on C's messages to A and B in the first q rounds and these are all obtained even though A halts after $q-1$ rounds). We now define another event G_2 on the probability space induced by the same execution of the protocol. By the security guarantee, and since B is still following the protocol, the protocol must continue as if only A is corrupted. In particular, B should eventually output c with probability (roughly) $1/3$. Let G_2 denote the event that B outputs some value in $\{0,1,2\}$ (in which case, as follows from the definition of f, it also learns the value of the input c). Note, however, that there is no guarantee that when G_2 occurs, the output of B is necessarily equal to a (since A is corrupted).

In the following, $\epsilon(k)$ should be interpreted as *some* function that tends to 0 as k tends to infinity (not necessarily negligible in k) as in Definition 1. Using the above notations, the following facts can now be deduced. First,

$$\Pr[G_1 \text{ and } a \neq c] < \epsilon(k). \tag{2}$$

This is because when $a \neq c$ then even the full view does not allow A and B to predict c with "small" error (where from here on, small error means with error $\epsilon(k)$ for some function $\epsilon(\cdot)$ that tends to 0 as its input tends to infinity). Thus, in the ideal model, it is impossible to predict with small error. This means that if Eq. (2) did not hold, then the assumed predictor could be used to distinguish the output of the adversary in a real protocol execution from its output in an ideal one.

In addition to the above, we have that in the ideal model B must output c with probability $1/3$. Thus, by the security guarantee

$$\Pr[G_2] \geq 1/3 - \epsilon(k). \tag{3}$$

(Note that Eq. (3) actually holds for a *negligible* function $\epsilon(k)$. Nevertheless, it suffice to use a function $\epsilon(k)$ that tends to 0 as k tends to infinity.) Finally, we claim that the probability that B generates output in $\{0,1,2\}$, but the predictor outputs "don't know" is at most $\epsilon(k)$; that is:

$$\Pr[G_2 \text{ and } \mathrm{not}(G_1)] \leq \epsilon(k). \tag{4}$$

This last inequality is shown by way of contradiction. If Equation 4 does not hold, then there is some constant $p > 0$ such that $\Pr[G_2 \text{ and } \mathrm{not}(G_1)] > p$ for infinitely many k. If that happens, we can derive a contradiction to the guaranteed privacy of the protocol π. To show that privacy does not hold, we show that there is some constant $p' > 1/3$ such that the adversary (that corrupts A and B) can guess c with probability greater than p' for infinitely many k's (while outputting a wrong guess with vanishingly small probability). This cannot happen in the ideal model. The adversary uses the following strategy: Corrupt A and B; Run the protocol normally for $q-1$ rounds; Invoke the predictor P on the partial view of A and B, and if it outputs some value c then output the same value and halt; Otherwise, pretend as if A is dead and continue to run the protocol until it terminates (with B acting honestly); If B outputs some value c then output the same value and halt; Otherwise, output "don't know". Note that a guess c output by the algorithm is wrong with only small probability, and the event of outputting some value c is the union of G_1 and G_2 which, using Equation 1 and the assumption above, is at least $1/3 - \epsilon(k) + p$ for infinitely many k's. Since p is a positive constant and $\epsilon(k)$ goes to zero, we obtain a contradiction.

So, we assume Equation 4 does hold and obtain a contradiction to the property of round q; i.e., to the fact that the view of B and C does not yield the value a with probability $1/3$ after $q - 1$ rounds. We construct a new predictor that yields the value of a from the view of B and C at round $q - 1$ in the following manner. It takes the view of B and C and continues to run the protocol until the end assuming that A aborted. Since A is not involved from Round q and on, this can be done based on the partial view of B and C alone (in the first $q - 1$ rounds). Note that this run is distributed exactly like the experiment previously described, where A aborts and B and C continue to run the protocol on their own. The experiment and distribution remain identical even though we previously thought of A and B as trying to obtain the value c and now we think of B and C as trying to obtain the value a. The predictor outputs whatever B outputs. By Equation 3, the probability that B outputs a value in $\{0, 1, 2\}$ is roughly $1/3$. By Equation 4, whenever this happens it is almost always the case that G_1 occurs. Namely, if we had run the predictor P on the view of A and B, it would have predicted the value c correctly. But if G_1 occurs then, by Equation 2, we get that $a = c$. To sum up, the output of B generated from the view of B and C in round $q - 1$ yields a with probability (almost) $1/3$ and is wrong with only small probability, contradicting the definition of the round q. □

Note that while we allow the simulator an arbitrary malicious behavior in the ideal world, the real-world adversary described above is only a fail-stop one. This strengthens our result.

Using standard player-partitioning arguments, the result for the above 3-party functionality f can be generalized to rule out general constant-round protocols for any s, t such that $s + t \geq n$.

Theorem 3. *Let n, s, t be such that $s + t \geq n$. Then there is an n-party functionality for which there is no constant-round protocol that is simultaneously:*
- *secure against an s-bounded malicious adversary;*
- *private against a t-bounded malicious adversary.*

5 Negative Result for Reactive Functionalities

For the case of reactive functionalities, we obtain the following strong negative result.

Theorem 4. *There exists a finite, reactive 3-party functionality f for which there does not exist a protocol π such that:*
- *π securely computes f in the presence of a malicious adversary corrupting one party;*
- *π privately computes f in the presence of a semi-honest adversary corrupting two parties.*

This holds even if the malicious adversary is restricted to be of a fail-stop type.

In the full version of the paper we prove Theorem 4 using a two-phase functionality, which roughly corresponds to the task of committing to a secret and later decommitting it. Similarly to Theorem 3, this negative result can also be generalized to any s, t such that $s + t \geq n$.

References

[1] N. Asokan, V. Shoup, and M. Waidner. Optimistic Fair Exchange of Digital Signatures (Extended Abstract). In *Proc. EUROCRYPT 1998*, pages 591-606.

[2] B. Barak, and A. Sahai. How To Play Almost Any Mental Game Over The Net - Concurrent Composition via Super-Polynomial Simulation. In *Proc. of 46th FOCS*, pp. 543-552, 2005.

[3] J. Bar-Ilan and D. Beaver. Non-cryptographic fault-tolerant computing in a constant number of rounds. In *Proc. 8th ACM PODC*, pages 201–209, 1989.

[4] D. Beaver, S. Goldwasser. Multiparty Computation with Faulty Majority. In *Proc. of FOCS 1989*, pp. 468-473.

[5] D. Beaver, S. Micali and P. Rogaway. The Round Complexity of Secure Protocols. In 22*nd STOC*, pages 503–513, 1990.

[6] M. Ben-Or, S. Goldwasser and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In 20*th STOC,* pages 1–10, 1988.

[7] D. Boneh and M. Naor. Timed Commitments. In *CRYPTO 2000*, pp. 236-254.

[8] C. Cachin and J. Camenisch. Optimistic Fair Secure Computation. In *Proc. CRYPTO 2000*, pages 93-111.

[9] R. Canetti. Security and Composition of Multiparty Cryptographic Protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[10] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. FOCS 2001: 136-145.

[11] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively Secure Multi-Party Computation. In 28*th STOC*, pages 639-648, 1996.

[12] R. Canetti, Y. Lindell, R. Ostrovsky and A. Sahai. Universally Composable Two-Party and Multi-Party Computation. In 34*th STOC*, pages 494–503, 2002.

[13] D. Chaum, C. Crépeau and I. Damgård. Multi-party Unconditionally Secure Protocols. In 20*th STOC*, pages 11–19, 1988.

[14] R. Cleve. Limits on the Security of Coin Flips when Half the Processors Are Faulty. In *Proc. of STOC 1986*, pp. 364-369.

[15] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient Multiparty Computations Secure Against an Adaptive Adversary. In *Proc. EUROCRYPT* 1999, pages 311-326.

[16] R. Cramer, I. Damgård, and S. Fehr. On the Cost of Reconstructing a Secret, or VSS with Optimal Reconstruction Phase. In *Proc. CRYPTO* 2001, pages 503-523.

[17] R. Cramer, I. Damgård, and U. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *EUROCRYPT '00*, pp. 316-334.

[18] R. Cramer, I. Damgård, and J. Nielsen. Multiparty computation from threshold homomorphic encryption. In *EUROCRYPT '01*, pp. 280-299.

[19] I. Damgård and J. Nielsen. Multiparty Computation from Threshold Homomorphic Encryption. CRYPTO 2003: 247-264.

[20] I. Damgård and Y. Ishai. Constant-Round Multiparty Computation Using a Black-Box Pseudorandom Generator. In *CRYPTO 2005*, pp. 378–394.

[21] S. Even, O. Goldreich and A. Lempel. A Randomized Protocol for Signing Contracts. In *Communications of the ACM,* 28(6):637–647, 1985.

[22] M. Fitzi, M. Hirt, T. Holenstein, and J. Wullschleger. Two-Threshold Broadcast and Detectable Multi-party Computation. EUROCRYPT 2003: 51-67.

[23] M. Fitzi, M. Hirt, and U. M. Maurer. Trading Correctness for Privacy in Unconditional Multi-Party Computation (Extended Abstract). CRYPTO 1998: 121-136

[24] M. Fitzi, T. Holenstein, and J. Wullschleger. Multi-party Computation with Hybrid Security. EUROCRYPT 2004: 419-438

[25] J. A. Garay, P. D. MacKenzie, and K. Yang. Efficient and Universally Composable Committed Oblivious Transfer and Applications. TCC 2004.

[26] J. A. Garay, P. D. MacKenzie, M. Prabhakaran, and K. Yang. Resource Fairness and Composability of Cryptographic Protocols. *Proc. 3rd TCC*, 2006. Also appears in Cryptology ePrint Archive, Report 2005/370.

[27] O. Goldreich. *Foundations of Cryptography: Volume 2 – Basic Applications.* Cambridge University Press, 2004.

[28] O. Goldreich, S. Micali and A. Wigderson. How to Play any Mental Game – A Completeness Theorem for Protocols with Honest Majority. In 19*th STOC,* pages 218–229, 1987. For details see [27].

[29] S. Goldwasser and L. Levin. Fair Computation of General Functions in Presence of Immoral Majority. In *CRYPTO '90,* pp. 77–93.

[30] S. Goldwasser, and Y. Lindell. Secure Multi-Party Computation without Agreement. J. Cryptology 18(3): 247-287 (2005). Preliminary version in DISC 2002.

[31] M. Hirt and U. M. Maurer. Robustness for Free in Unconditional Multi-party Computation. CRYPTO 2001: 101-118.

[32] Y. Ishai and E. Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *Proc. 41st FOCS*, pp. 294–304, 2000.

[33] J. Katz and R. Ostrovsky. Round-Optimal Secure Two-Party Computation. In *CRYPTO 2004*, pages 335-354.

[34] J. Katz, R. Ostrovsky, and A. Smith. Round Efficiency of Multi-party Computation with a Dishonest Majority. In *EUROCRYPT 2003*, pages 578-595.

[35] M. Lepinski, S. Micali, C. Peikert, and A. Shelat. Completely fair SFE and coalition-safe cheap talk. In *Proc. PODC 2004*, pages 1-10.

[36] Y. Lindell. Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation. *J. Cryptology* 16(3): 143-184 (2003). Preliminary version in Crypto 2001.

[37] M. Luby, S. Micali, and C. Rackoff. How to Simultaneously Exchange a Secret Bit by Flipping a Symmetrically-Biased Coin. In *24th FOCS*, pp. 11-21, 1983.

[38] R. Pass. Bounded-Concurrent Secure Multi-Party Computation With a Dishonest Majority. In *Proc. STOC 2004*, pages 232-241.

[39] R. Pass and A. Rosen. Bounded-Concurrent Secure Two-Party Computation in a Constant Number of Rounds. In *Proc. FOCS 2003.*, pp. 404-413, 2005.

[40] B. Pfitzmann and M. Waidner. Information-Theoretic Pseudosignatures and Byzantine Agreement for $t \geq n/3$. IBM Research Report RZ 2882 (#90830), IBM Research Division, Zurich, 1996.

[41] T. Rabin and M. Ben-Or. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. In *Proc.* 21*st STOC*, pages 73–85. ACM, 1989.

[42] A. Shamir. How to share a secret. *Commun. ACM*, 22(6):612–613, June 1979.

[43] A. Yao. How to Generate and Exchange Secrets. In 27*th FOCS*, pages 162–167, 1986.