On Achieving the "Best of Both Worlds" in Secure Multiparty Computation*

Jonathan Katz†

Abstract

Two settings are typically considered for secure multiparty computation, depending on whether or not a majority of the parties are assumed to be honest. Protocols designed under this assumption provide "full security" (and, in particular, guarantee output delivery and fairness) when this assumption is correct; however, if half or more of the parties are dishonest then security is completely compromised. On the other hand, protocols tolerating arbitrarily-many faults do not provide fairness or guaranteed 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. Ishai, et al. (Crypto 2006) recently addressed this question, and ruled out *constant-round* protocols of this type.

As our main result, we completely settle the question by ruling out protocols using any (expected) polynomial number of rounds. Given this stark negative result, we then ask what can be achieved if we are willing to assume *simultaneous message transmission* (or, equivalently, a non-rushing adversary). In this setting, we show that impossibility still holds for logarithmic-round protocols. We also show, for any polynomial p, a protocol (whose round complexity depends on p) that can be simulated to within closeness $O(1/p)$.

*This research was done while visiting IPAM, and was supported by NSF CAREER award #0447075 and US-Israel Binational Science Foundation grant #2004240.

†Dept. of Computer Science, University of Maryland. `jkatz@cs.umd.edu`.

1 Introduction

Protocols for secure multiparty computation (MPC) [26, 16, 4, 7] allow a set of parties to compute an arbitrary function of their inputs while preserving (to the extent possible) the privacy of the parties' inputs as well as the (global) correctness of their outputs. Formally, security of such protocols is defined 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 must also be guaranteed in a real-world execution of the protocol.

Existing work on secure MPC, for the most part, can be divided neatly into the study of two, very different settings (here and in the rest of the paper, we assume the existence of a broadcast channel or a trusted preprocessing phase sufficient for implementing broadcast): the case when a majority of the parties are honest, and the case when an arbitrary number of parties may be malicious. These settings differ not only in the approaches that are used to construct secure protocols, but also in the results that can be achieved. In further detail:

- In the presence of an honest majority, MPC protocols achieving so-called *full security* are possible [4, 7, 24, 1]. Roughly speaking, "full security" means that such protocols guarantee not only privacy and correctness but also fairness (i.e., if one party receives its output then all parties do) and output delivery (i.e., honest parties will receive their outputs).

- If half or more of the players are malicious (with the 2-party setting as a special case), then it is impossible [8] to construct protocols achieving full security in the sense described above. Specifically, while privacy and correctness are still attainable, it is impossible (in general) to guarantee fairness or output delivery. The relaxed notion of security that can be obtained in this setting is sometimes called *security with abort*, and protocols realizing this notion are known for any number of corrupted parties [26, 16, 2, 17, 18] assuming the existence of enhanced trapdoor permutations.

An unfortunate drawback of most existing protocols for each of the above settings is that they do not provide any security beyond what is implied by the definitions. Specifically, 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, 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 the problem, consider trying to decide which type of protocol is more appropriate to implement secure voting. Since we would like privacy of individuals' votes (and, perhaps even more so, correctness[1]) to hold 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) can perform a denial-of-service attack that prevents all honest parties from learning the outcome. Neither option is very appealing.

In light of the above, a natural question is the following:

> *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 case a majority of the parties are honest, and security with abort otherwise?*

[1] Although one might be tempted to think that correctness is not important in an election once a majority are dishonest, this is not so: we still want all honest parties to agree on the outcome, and the malicious half should not be able to manipulate the outcome so that it appears unanimous, for example.

The current lack of an answer to the above question represents a serious gap in our understanding of stand-alone secure computation, and is one of the few remaining open questions regarding *feasibility* (rather than efficiency) in this domain.

Ishai, et al. [19] recently addressed the above question, and showed a number of positive and negative results. Most relevant here is their result ruling out *constant-round* protocols achieving the best of both worlds. (They also show the impossibility of realizing *reactive* functionalities in any number of rounds. In this work, we consider standard, non-reactive functionalities only.) Their work left open the possibility that protocols with polynomially-many rounds (using, e.g., some variant of the ideas of [21, 2, 9, 17]) might still give a positive answer to the above question.

1.1 Our Results

As our main contribution, we settle the above question by showing that there exist functionalities for which *no* polynomial-round (or even expected polynomial-round) protocol can achieve the best of both worlds. Actually, our result is even stronger: we show a functionality for which any protocol that is fully-secure for a minority of fail-stop adversaries cannot even achieve *privacy*[2] in the presence of a fail-stop adversary corrupting half or more of the players. Our result generalizes to show that any protocol with full security when t parties (out of a total of n parties) are corrupted will not guarantee privacy when $n - t$ parties are corrupted; "corrupted" can be taken to mean "fail-stop" in each case.

This (somewhat unexpected) negative result is unfortunate, as it leaves protocol designers in a state of uncertainty about which type of security to aim for in a scenario when it is unclear what to assume about the number of corrupted players. (Think again of the voting example mentioned earlier.) An important question thus becomes:

What weaker (yet meaningful) security guarantees **can** *be achieved for our problem?*

Ishai, et al. [19] have already explored various answers to this question:

1. For any t, s with $t+s < n$ and $t < n/2$, they show that a variant of a standard MPC protocol achieves full security for t malicious parties, and security with abort for s malicious parties.

2. They also construct a protocol that guarantees full security against either (1) $t < n/2$ malicious parties or (2) any number of *semi-honest* (i.e., honest-but-curious) parties.[3]

3. A variant of the above protocol achieves full security in the presence of a malicious minority; furthermore, the actions of any $s < n$ malicious players can be simulated by an ideal-world adversary who is allowed to invoke the functionality s times. For certain functionalities — with voting as a prime example — this latter notion implies privacy and correctness (but not independence of inputs).

Here, we explore a different relaxation: we ask what can be attained if *simultaneous message transmission* is assumed. This is equivalent to considering a non-rushing adversary, and indeed our impossibility result mentioned earlier holds only for the case of a rushing adversary. From a theoretical point of view it is quite natural to consider simultaneous message transmission since other, similar impossibility results (e.g., the impossibility of fair coin tossing without an honest majority [8]) can be overcome in this model. On the other hand, as we will see, this assumption

[2] We formally define what is meant by this term in Section 2.

[3] A semi-honest party acts honestly, but tries to learn additional information from the execution of the protocol. Full security is possible here (for an arbitrary number of faults) because a semi-honest adversary never aborts.

does not trivialize things, either. Finally, simultaneous message transmission can be obtained (at least heuristically) by relying on *timed commitments* [5]. Other physical settings may also exist where it is reasonable to assume that simultaneous message transmission can be achieved.

In the simultaneous-message model, we show additional negative and positive results. On the negative side, we show that obtaining the "best of both worlds" is still impossible for *logarithmic-round* protocols. This improves on the work of Ishai, et al. [19] whose impossibility result for constant-round protocols, mentioned earlier, holds even for non-rushing adversaries. On the positive side, we show that for any polynomial p it is possible to construct a protocol (with polynomial round complexity depending on p) that is fully secure against a malicious minority and also "$O(1/p)$-secure with abort" for any number of *malicious* adversaries. Roughly speaking, this latter notion means that the actions of any PPT adversary in the real world can be simulated by a PPT adversary in the ideal world such that the distributions of the resulting outcomes cannot be distinguished with probability better than $O(1/p)$. (The protocol provides additional security guarantees as well; see Section 5 for further discussion.)

1.2 Future Directions

Our work resolves the main question in this area, in that it demonstrates the impossibility of attaining the "best of both worlds" as originally hoped. Nevertheless, it is important to determine what *can* be achieved, and the following are the most compelling questions left open by our work:

- Do there exist super-logarithmic round protocols in the simultaneous-message model that attain the "best of both worlds"?
- Recall that our positive result obtains full security against a malicious minority and $O(1/p)$-security with abort for arbitrarily-many malicious parties. Is it possible to achieve this without assuming simultaneous message transmission? In either network setting, is it possible to achieve $O(1/p)$ *full* security for any number of corrupted parties?

We remark that the answers may be different for general functionalities and so-called *finite* functionalities (i.e., functions whose domain and range do not grow with the security parameter).[4] In any case, it is clear that more work is needed.

1.3 Comparison to Related Work

We briefly discuss some work that is related to our own, stressing the differences between the goals being considered as well as the results achieved.

MPC with varying thresholds. Other works [11, 12, 13] have also studied protocols with different security guarantees depending on the number of corrupted parties. Although the motivation is related, we differ in particular from these works in that they always assume an honest majority.

Fair MPC. Our results neither imply, nor are they implied by, existing results ruling out fairness in the case of no honest majority [8]. Our positive result in the simultaneous-message model is related to existing work aimed at achieving various partial notions of fairness [21, 2, 9, 17]; however, to the best of our knowledge none of the prior work in this direction has suggested a simulation-based definition of exactly what sort of fairness is achieved (see [15, Section 7.7.1] for a comment to this effect), nor do existing techniques appear to satisfy the definition of $O(1/p)$-security we use here.

Related to the above is work on *gradual release* [10, 5, 23, 14] that, informally, guarantees the following: if the adversary aborts early, then both the adversary and the honest players can obtain

[4] Both impossibility results in this paper apply even to finite functionalities.

their outputs by investing a "similar" amount of work. This approach deviates from "standard" notions of protocol design in that there is no a priori polynomial bound on the running time of honest parties. In this work, we assume the traditional requirement that the running time of honest parties is bounded (at least in expectation) by some fixed polynomial in the security parameter.

2 Security Definitions

We use the standard definitions of security for multiparty computation; see, e.g., [15]. For convenience, we provide a brief review of these definitions here. We also define the less-standard notion of *privacy*, following [19].

Computational indistinguishability. A distribution ensemble $X = \{X(k,a)\}_{k\in\mathbb{N},a\in\{0,1\}^*}$ is an infinite sequence of probability distributions indexed by k and a. Two distribution ensembles X, Y are computationally indistinguishable, denoted $X \stackrel{c}{\equiv} Y$, if there exists a negligible function ε such that for every polynomial-time algorithm D and all a we have

$$\left|\Pr[D(1^k, a, X(k,a)) = 1] - \Pr[D(1^k, a, Y(k,a)) = 1]\right| \leq \varepsilon(k).$$

The real model. We assume the standard communication model where n parties communicate in synchronous rounds using pairwise private and authenticated channels. We also assume that these parties have access to a broadcast channel; since a broadcast channel can be realized using a PKI and digital signatures for any number of corrupted parties, this assumption is inessential to our results. We will consider both rushing adversaries (who may delay sending messages on behalf of corrupted parties in a given round until the messages sent by all honest parties in that round have been received) as well as non-rushing adversaries (who must decide on what messages to send in a given round independent of the messages sent by honest parties in that same round). For simplicity, we assume a static adversary who may only corrupt parties in advance of the protocol execution; this strengthens our negative results and seems inessential for our positive result.

Malicious adversarial parties may deviate in an arbitrary manner from the protocol specification, while semi-honest adversaries are assumed to follow the protocol faithfully. Intermediate between these two are *fail-stop* adversaries, who are assumed to follow the protocol as specified except that they may abort (and refuse to send any more messages) at any time, depending on their view.

At the beginning of a real execution of a protocol, each party P_i holds the security parameter 1^k and its input x_i. We let f denote a poly-time computable, randomized function that maps players' inputs $\vec{x} \stackrel{\text{def}}{=} (x_1, \ldots, x_n)$ to a vector of outputs $(y_1, \ldots, y_n)$, where y_i denotes the output intended for P_i. We deal only with standard, non-reactive functionalities here.

The adversary $\mathcal{A}$ takes as input 1^k, the set $I \subset [n]$ of corrupted parties, the inputs of the corrupted parties, and an auxiliary input z. The interaction of $\mathcal{A}$ with a protocol π defines a random variable $\text{REAL}_{\pi,\mathcal{A},I}(k, \vec{x}, z)$ whose value is determined by the coin tosses of the adversary and the honest players. This random variable contains the output of the adversary (which may be an arbitrary function of its view) as well as the outputs of the uncorrupted parties. We let $\text{REAL}_{\pi,\mathcal{A},I}$ denote the distribution ensemble $\{\text{REAL}_{\pi,\mathcal{A},I}(k, \vec{x}, z)\}_{k\in\mathbb{N},\langle\vec{x},z\rangle\in\{0,1\}^*}$.

The ideal model — full security. Here the parties interact with a trusted party implementing f. Each honest party P_i holds an input x_i as before; the adversary $\mathcal{A}'$ is again given 1^k, the set I of corrupted parties, the inputs of all the corrupted parties, and an auxiliary input z. Each honest party P_i sets $x'_i = x_i$ and sends x'_i to the trusted party; each corrupted party P_j sends an arbitrary input x'_j to the trusted party as directed by $\mathcal{A}'$. In case some corrupted party P_j does not send an input, x'_j is set to a default value. The trusted party computes $(y_1, \ldots, y_n) \leftarrow f(x'_1, \ldots, x'_n)$, choosing a uniformly random tape for f in case it is randomized, and sends y_i to party P_i.

The interaction of $\mathcal{A}'$ with the trusted party defines a random variable $\text{IDEAL}_{f,\mathcal{A}',I}(k, \vec{x}, z)$ whose value is determined by the random coins of the adversary and those used by the trusted party in evaluating f. This random variable contains the output of $\mathcal{A}'$ (which may be an arbitrary function of its view) as well as the outputs of the uncorrupted parties. We let $\text{IDEAL}_{f,\mathcal{A}',I}$ denote the distribution ensemble $\left\{\text{IDEAL}_{f,\mathcal{A}',I}(k, \vec{x}, z)\right\}_{k\in\mathbb{N},\langle\vec{x},z\rangle\in\{0,1\}^*}$.

The ideal model — security with abort.[5] As in the previous case, parties again interact with a trusted party implementing f. Parties send inputs $x'_1, \ldots, x'_n$ to this trusted party, who then computes $(y_1, \ldots, y_n) \leftarrow f(x'_1, \ldots, x'_n)$ as before. Now, however, output delivery occurs in two phases. First, the trusted party sends to $\mathcal{A}'$ the outputs $\{y_i\}_{i\in I}$. The adversary can then decide whether to abort the trusted party, or whether to allow it to continue. In the former case, the trusted party sends the special symbol $\perp$ to all honest parties as their output, where $\perp$ is assumed not to lie in the range of f. In the latter case, the trusted party sends the correct output y_i to each honest party P_i.

The interaction of $\mathcal{A}'$ with the trusted party defines a random variable $\text{IDEAL}_{f_\perp,\mathcal{A}',I}(k, \vec{x}, z)$ as above, and we let $\text{IDEAL}_{f_\perp,\mathcal{A}',I}$ denote the distribution ensemble $\left\{\text{IDEAL}_{f_\perp,\mathcal{A}',I}(k, \vec{x}, z)\right\}_{k\in\mathbb{N},\langle\vec{x},z\rangle\in\{0,1\}^*}$. (The subscript "$\perp$" indicates that the adversary can abort computation of f.)

Security definitions. We may now define what it means for a protocol to be secure.

Definition 1 Let f be an n-party randomized functionality, and π be an n-party protocol. Then π t-securely computes f if for any PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{A}'$ such that for any $I \subset [n]$ with $|I| \leq t$:

$$\text{REAL}_{\pi,\mathcal{A},I} \stackrel{c}{\equiv} \text{IDEAL}_{f,\mathcal{A}',I}.$$

Similarly, π t-securely computes f with abort if for any PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{A}'$ such that for any $I \subset [n]$ with $|I| \leq t$:

$$\text{REAL}_{\pi,\mathcal{A},I} \stackrel{c}{\equiv} \text{IDEAL}_{f_\perp,\mathcal{A}',I}.$$

As in [19], we define also a notion of *privacy* for malicious adversaries. Informally, privacy implies that an adversary does not learn anything about the inputs of the honest parties that is not implied by its own inputs and outputs (as could be obtained by interacting with a trusted party implementing f). Let $\text{OUTPUT}_{\mathcal{A}}\left(\text{REAL}_{\pi,\mathcal{A},I}(k, \vec{x}, z)\right)$ denote the output of the adversary $\mathcal{A}$ in the indicated random variable, and let $\text{OUTPUT}_{\mathcal{A}'}\left(\text{IDEAL}_{f,\mathcal{A}',I}(k, \vec{x}, z)\right)$ correspondingly denote the output of $\mathcal{A}'$. We denote by $\text{OUTPUT}_{\pi,\mathcal{A},I}$ (resp., $\text{OUTPUT}_{f,\mathcal{A}',I}$) the corresponding distribution ensembles in the natural way. (We remark that since we only consider the output of $\mathcal{A}'$ in the second case, it does not matter whether we are in the ideal world allowing abort or not.)

Definition 2 Let f be an n-party randomized functionality, and π be an n-party protocol. Then π t-privately computes f in the presence of malicious adversaries if for any PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{A}'$ such that for any $I \subset [n]$ with $|I| \leq t$:

$$\text{OUTPUT}_{\pi,\mathcal{A},I} \stackrel{c}{\equiv} \text{OUTPUT}_{f,\mathcal{A}',I}.$$

We stress that considering privacy alone yields a relatively weak notion of security which has many shortcomings (see discussion in [19]); however, we use it only in the context of proving impossibility.

[5] Alternative approaches to defining the ideal model in this case exist [18], though the differences are unimportant for our work once we assume a broadcast channel.

3 Main Impossibility Result

We now show our main result, which rules out any protocol achieving the "best of both worlds":

Theorem 1 *Let n, t, s be such that $t + s = n$ and $t \geq 1$. Then there exists a finite, deterministic functionality $\tilde{f}$ for which there is no polynomial-round protocol π that simultaneously t-securely computes $\tilde{f}$ (for rushing adversaries) and s-privately computes $\tilde{f}$ in the presence of malicious, rushing adversaries. This holds even if we consider only fail-stop adversaries in each case.*

The theorem is tight: as discussed in the full version of [19] (obtained from the authors), if $t+s < n$ and $t < n/2$ then for any poly-time functionality f there exists (under standard cryptographic assumptions) a protocol π that t-securely computes f and s-securely computes f with abort. Note also that the theorem is only interesting when $t < n/2$, otherwise we already know that there exist functionalities that cannot be t-securely computed (i.e., without allowing abort).

We prove the theorem in two steps: we first present a finite, deterministic functionality f for which any protocol π that t-securely computes f does not s-securely compute f with abort. Extending this counterexample, we then give a (slightly different) functionality $\tilde{f}$ and show that any protocol π that t-securely computes $\tilde{f}$ does not even s-*privately* compute $\tilde{f}$.

3.1 Ruling out Security with Abort

Fix n, s, and $1 \leq t < n/2$ as in the theorem (as mentioned, the theorem is already known to hold if $t \geq n/2$). Define f as follows: players P_1 and P_n each provide a bit b_1 and b_n, respectively, as input, and each receive as output $b_1 \oplus b_n$ (no other players receive output). Let π be any protocol that t-securely computes f for fail-stop adversaries. We assume that π operates in a fixed number of *segments*, each exactly n rounds long, where party P_i, and this party only, sends a message in the i^{th} round of a segment (i.e., in a given segment first P_1 speaks, then P_2, etc. until P_n speaks and then the next segment begins). If π is secure against a rushing adversary, then it can always be transformed, albeit by increasing the round complexity, so that it has this form. (We remark also that the assumption that only one party speaks in any given round is not essential for our proof, but merely serves to simplify things conceptually.) Let $r = r(k)$ denote the number of segments of the protocol. We assume that if π is run (honestly) to completion, then the outputs of P_1 and P_n are correct (and, in particular, agree) with probability[6] at least 7/8.

Define $A \stackrel{\text{def}}{=} \{P_1, \ldots, P_t\}$, $B \stackrel{\text{def}}{=} \{P_{t+1}, \ldots, P_{n-t}\}$, and $C \stackrel{\text{def}}{=} \{P_{n-t+1}, \ldots, P_n\}$. Consider the experiment in which P_1 and P_n choose their inputs uniformly at random and then run the protocol honestly, except for (possibly) aborting at some round. All other parties in A and C run the protocol honestly except that they, too, may possibly abort. Parties in B run the protocol honestly and never abort. Let v_1^i, with $1 \leq i \leq r$, denote the final output of P_1 when parties in C all simultaneously abort in segment $i+1$ (i.e., segment i is the last segment in which parties in C send any messages). Extending this notation, let v_1^0 denote the output of P_1 if parties in C abort immediately without sending any messages. Define v_n^i similarly to be the output of P_n when all parties in A simultaneously abort in segment $i+1$ (i.e., send messages for the final time in segment i), and v_n^0 to be the output of P_n when parties in A abort immediately. Note that v_1^i can be computed from the joint view of all parties in $A \cup B$ as soon as all players in C have sent their segment-i messages, and analogously for v_n^i.

[6]Security (or even just correctness) of π actually implies that this holds with all but negligible probability. However, we will make use of the relaxed requirement stated here in Section 3.3.

Security of π implies that, for all i, we have $v_1^i \in \{0,1\}$ (and, in particular, $v_1^i \neq \perp$) with all but negligible probability, and similarly for v_n^i. This is true since π provides full security against t fail-stop adversaries, and at most t players abort in the experiment defining v_1^i, v_n^i. In what follows, we will assume for simplicity that $v_1^i, v_n^i \in \{0,1\}$ with probability 1.

Consider the following summation:

$$\begin{aligned}
&\left(\Pr\left[v_1^0 = 1 \bigwedge v_n^0 = 1\right] + \Pr\left[v_1^0 = 0 \bigwedge v_n^r = 1\right] - \frac{1}{2}\right) && (1)\\
+\ &\left(\Pr\left[v_1^0 = 0 \bigwedge v_n^0 = 0\right] + \Pr\left[v_1^0 = 1 \bigwedge v_n^r = 0\right] - \frac{1}{2}\right) && (2)\\
+\ &\sum_{i=0}^{r-1}\Bigg[\left(\Pr\left[v_1^i = 0 \bigwedge v_n^{i+1} = 0\right] + \Pr\left[v_1^i = 1 \bigwedge v_n^i = 0\right] - \frac{1}{2}\right) && (3)\\
&+\left(\Pr\left[v_1^i = 1 \bigwedge v_n^{i+1} = 1\right] + \Pr\left[v_1^i = 0 \bigwedge v_n^i = 1\right] - \frac{1}{2}\right) && (4)\\
&+\left(\Pr\left[v_1^{i+1} = 0 \bigwedge v_n^{i+1} = 0\right] + \Pr\left[v_1^i = 0 \bigwedge v_n^{i+1} = 1\right] - \frac{1}{2}\right) && (5)\\
&+\left(\Pr\left[v_1^{i+1} = 1 \bigwedge v_n^{i+1} = 1\right] + \Pr\left[v_1^i = 1 \bigwedge v_n^{i+1} = 0\right] - \frac{1}{2}\right)\Bigg], && (6)
\end{aligned}$$

which evaluates to:

$$\begin{aligned}
&\Pr\left[v_1^0 = 1 \bigwedge v_n^0 = 1\right] + \Pr\left[v_1^0 = 0 \bigwedge v_n^r = 1\right]\\
&+\ \Pr\left[v_1^0 = 0 \bigwedge v_n^0 = 0\right] + \Pr\left[v_1^0 = 1 \bigwedge v_n^r = 0\right] - 1\\
&+\ \sum_{i=0}^{r-1}\Bigg[\Pr\left[v_1^i = 1 \bigwedge v_n^i = 0\right] + \Pr\left[v_1^i = 0 \bigwedge v_n^i = 1\right]\\
&\quad + \Pr\left[v_1^{i+1} = 0 \bigwedge v_n^{i+1} = 0\right] + \Pr\left[v_1^{i+1} = 1 \bigwedge v_n^{i+1} = 1\right] - 1\Bigg]\\
=\ &\Pr\left[v_1^0 = 1 \bigwedge v_n^0 = 1\right] + \Pr\left[v_1^0 = 0 \bigwedge v_n^r = 1\right] + \Pr\left[v_1^0 = 0 \bigwedge v_n^0 = 0\right]\\
&+ \Pr\left[v_1^0 = 1 \bigwedge v_n^r = 0\right] + \Pr\left[v_1^0 = 1 \bigwedge v_n^0 = 0\right] + \Pr\left[v_1^0 = 0 \bigwedge v_n^0 = 1\right]\\
&+ \Pr\left[v_1^r = 0 \bigwedge v_n^r = 0\right] + \Pr\left[v_1^r = 1 \bigwedge v_n^r = 1\right] - 2\\
=\ &\Pr\left[v_1^0 = 0 \bigwedge v_n^r = 1\right] + \Pr\left[v_1^0 = 1 \bigwedge v_n^r = 0\right] + \Pr\left[v_1^r = v_n^r\right] - 1\\
\geq\ &\Pr\left[v_1^0 = 0 \bigwedge b_1 \neq b_n\right] - \Pr\left[v_n^r \neq 1 \bigwedge b_1 \neq b_n\right]\\
&+ \Pr\left[v_1^0 = 1 \bigwedge b_1 = b_n\right] - \Pr\left[v_n^r \neq 0 \bigwedge b_1 = b_n\right] + \Pr\left[v_1^r = v_n^r\right] - 1\\
\geq\ &\Pr\left[b_1 \neq b_n \mid v_1^0 = 0\right] \cdot \Pr[v_1^0 = 0] + \Pr\left[b_1 = b_n \mid v_1^0 = 1\right] \cdot \Pr[v_1^0 = 1] + \frac{3}{4} - 1,
\end{aligned}$$

using the assumed correctness of π when run honestly to completion. Since v_1^0 is independent of P_n's input b_n, we have $\Pr\left[b_1 \neq b_n \mid v_1^0 = 0\right] = \Pr\left[b_1 = b_n \mid v_1^0 = 1\right] = \frac{1}{2}$. It follows that the above sum is at least $\frac{1}{4}$, and so at least one of the summands (1)–(6) is at least $p(k) \stackrel{\text{def}}{=} \frac{1}{4\cdot(4r(k)+2)}$, which is noticeable. We show that this implies that π does not s-securely compute f with abort, even for fail-stop adversaries.

Case 1(a). Say $\Pr\left[v_1^0 = 1 \bigwedge v_n^0 = 1\right] + \Pr\left[v_1^0 = 0 \bigwedge v_n^r = 1\right] - \frac{1}{2} \geq p(k)$, and consider the adversary who corrupts players in $A \cup B$ and does the following: it chooses input for P_1 as well as random tapes for all players in A and B uniformly at random, and then computes v_1^0. If $v_1^0 = 1$, it aborts all players in A immediately and simply has players in B run π honestly with (the honest parties) C. If $v_1^0 = 0$, the adversary simply has all players in $A \cup B$ run the entire protocol honestly.

Note that $|A \cup B| = s$. Furthermore, the probability that P_n outputs 1 in a real execution of the protocol is exactly

$$\Pr\left[v_1^0 = 1 \bigwedge v_n^0 = 1\right] + \Pr\left[v_1^0 = 0 \bigwedge v_n^r = 1\right] \geq \frac{1}{2} + p(k).$$

However, in an ideal execution with any adversary corrupting players in $A \cup B$, the honest player P_n will not output 1 with probability greater than $\frac{1}{2}$ (given that its input is chosen uniformly at random).[7] It follows that π does not s-securely compute f with abort.

Case 1(b). Say $\Pr\left[v_1^0 = 0 \bigwedge v_n^0 = 0\right] + \Pr\left[v_1^0 = 1 \bigwedge v_n^r = 0\right] - \frac{1}{2} \geq p(k)$. An argument as above gives an adversary who corrupts parties in $A \cup B$ and forces P_n to output 0 with probability noticeably greater than 1/2. This again implies that π does not s-securely compute f with abort.

Case 2(a). Say there exists an index $i \in \{0, \ldots, r(k) - 1\}$ for which

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

Consider the adversary given auxiliary input $z = i$ who corrupts the players in A and B and acts as follows: it chooses random input for P_1 and then 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 players in $A \cup B$ jointly have enough information to compute v_1^i. If $v_1^i = 1$, then all parties in A abort and send no more messages. If $v_1^i = 0$, 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 entire rest of the protocol honestly.

Now, the probability that P_n outputs 0 in a real execution of the protocol is exactly

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

However, in an ideal execution with any adversary corrupting players in $A \cup B$, the honest player P_n will not output 0 with probability greater than $\frac{1}{2}$ given that its input is chosen uniformly at random. Thus, π does not s-securely compute f with abort.

Case 2(b). If there exists an i such that $\Pr\left[v_1^i = 1 \bigwedge v_n^{i+1} = 1\right] + \Pr\left[v_1^i = 0 \bigwedge v_n^i = 1\right] - \frac{1}{2} \geq p(k)$, an argument as above gives an adversary corrupting players 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

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

(note that all indices have been shifted by 1 for convenience). Consider the adversary given auxiliary input $z = i$ who corrupts players in $B \cup C$ and acts as follows: it chooses random input for P_n

[7] Note that in the ideal model with abort, an adversary corrupting s players can bias the value of P_n's output *conditioned on* P_n *not outputting* $\perp$, but still cannot force P_n to output 1 with probability more than $\frac{1}{2}$.

and then 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 player with lowest index who is in C.) At this point, the players in $B \cup C$ can jointly compute v_n^i. If $v_n^i = 1$, then all players in C abort in this segment and refuse to 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 players in C send their (honestly-generated) messages in segment i but will 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

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

However, in an ideal execution with any adversary corrupting players in $B \cup C$, the honest player 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 π does not s-securely compute f with abort.

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

The observant reader will note that the argument above is very similar to that given by Cleve [8] in showing the impossibility of fair coin tossing without honest majority. A key difference is that in Cleve's setting honest parties are required to output a bit no matter what (in particular, no matter how many other parties abort); in our setting, if half or more of the parties abort then honest parties can simply output $\perp$. This introduces additional technical complications in the argument.

3.2 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 to obtain a t-secure protocol π that at least obtains *s-privacy* (in the presence of malicious adversaries). By building on the result of the previous section we unfortunately rule out this possibility as well.

Given n, t, s as before, we define a function $\tilde{f}$ that takes inputs from P_1 and P_n, and returns output to P_1, P_n, and also (and here we differ from before) P_{t+1}. On input $(b_1, \alpha_0, \alpha_1)$ from P_1 and (b_n, β_0, β_1) from P_n, where $b_1, b_n, \alpha_0, \alpha_1, \beta_0, \beta_1 \in \{0,1\}$, functionality $\tilde{f}$ computes $v = b_1 \oplus b_n$, gives v to P_1 and P_n, and gives (v, α_v, β_v) to P_{t+1}. That is:

$$\begin{aligned}&\tilde{f}((b_1, \alpha_0, \alpha_1), \lambda, \ldots, \lambda, (b_n, \beta_0, \beta_1)) = \\ &\qquad (b_1 \oplus b_n, \lambda, \ldots, \lambda, \underbrace{(b_1 \oplus b_n, \alpha_{b_1 \oplus b_n}, \beta_{b_1 \oplus b_n})}_{\text{output of } P_{t+1}}, \lambda, \ldots, \lambda, b_1 \oplus b_n),\end{aligned}$$

where we let λ denote an empty input/output.

Let π be a protocol that t-securely computes $\tilde{f}$. Let A, B, C be a partition of the players as in the previous section, and recall that $P_{t+1} \in B$. Consider an experiment in which P_1 and P_n choose their inputs uniformly at random and all parties run protocol π honestly, except that players in A or C may possibly abort. An argument exactly as in the previous section shows that there exists a real-world adversary $\mathcal{A}$ 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_1 to output some bit v with probability noticeably greater than 1/2.

Assume without loss of generality that the first case holds, and there is an adversary $\mathcal{A}$ who corrupts players 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 $\mathcal{A}$ only causes the t players in A to abort, and the remaining corrupted players in B act entirely honestly. Since π is fully-secure for up to t malicious players, all players in $B \cup C$ should receive their outputs (except possibly with negligible probability) even if all players in A abort. Moreover, t-security of π implies that the output of the honest-looking P_{t+1} should be consistent with the input and output of P_n (except with negligible probability). Taken together, this means that the view of $\mathcal{A}$ — which includes the output honestly generated by P_{t+1} — includes β_0 with probability at least $1/2 + p'(k)$ for some noticeable function p', and furthermore $\mathcal{A}$ knows when this occurs (since the output of P_{t+1} includes $v \stackrel{\text{def}}{=} b_1 \oplus b_n$ in addition to β_v). Thus, $\mathcal{A}$ can output a guess for β_0 which is correct with probability at least

$$\frac{1}{2} + p'(k) + \frac{1}{2} \cdot \left(\frac{1}{2} - p'(k)\right) = \frac{3}{4} + \frac{p'(k)}{2}\,.$$

In contrast, no ideal-world adversary $\mathcal{A}'$ corrupting $A \cup B$ can output a guess for β_0 which is correct with probability better than 3/4 when P_n chooses its inputs uniformly at random. This shows that π does not s-privately compute $\tilde{f}$.

3.3 Protocols with Expected Polynomial Round Complexity

A simple extension of the arguments given above applies to protocols having *expected* polynomial round complexity (and, say, potentially running for an unbounded number of rounds). We sketch the main idea here, with respect to the proof given in Section 3.1. Say we have a protocol π that t-securely computes f and for which the expected number of *segments* in π is $p(k)$. Security (or even just correctness) of π implies that if π is run honestly to completion, then the outputs of P_1 and P_n are correct (and, in particular, agree) with all but negligible probability. Setting $r(k) = 8 \cdot r'(k)$, this means that except with probability negligibly close to 1/8 the outputs of P_1 and P_n are correct *and are computed by the end of segment $r(k)$*. The remainder of the proof is as before.

4 A Lower Bound for Non-Rushing Adversaries

The stark impossibility result of the preceding section forces us to examine relaxations of our original goal. Here we consider the case of *non-rushing* adversaries or, equivalently, the case when simultaneous message transmission is available. This setting is natural to consider since, for example, fair coin flipping is impossible for rushing adversaries [8] but trivial to achieve for non-rushing adversaries; one might hope for something similar here. Moreover, as noted earlier, it appears that simultaneous message transmission can be implemented under plausible cryptographic assumptions.

Unfortunately, eliminating rushing does not seem to make the problem significantly easier; we show that no *logarithmic-round* protocol can achieve the best of both worlds even in this setting.

Theorem 2 *Let n, t, s be such that $t + s = n$ and $t \geq 1$. Then there exists a finite n-party functionality f for which there is no protocol π whose round complexity is logarithmic in the security parameter and such that π simultaneously t-securely computes f (for non-rushing adversaries) and* *s-privately computes f in the presence of malicious, non-rushing adversaries. This holds even if we consider only fail-stop adversaries in each case.*

For simplicity here, we look only at the case $n = 3$, $t = 1$, $s = 2$ (the argument generalizes in a straightforward manner). As in [19], we consider the case of three parties A, B, C computing the following functionality f: both A and C provides inputs $a, c \in \{1, 2, 3\}$; party B is given the output

$$f(a, \lambda, c) \stackrel{\text{def}}{=} \begin{cases} a & \text{if } a = c \\ \star & \text{otherwise} \end{cases},$$

where λ denotes an empty input and $\star$ denotes a distinguished output that does not lie in $\{1, 2, 3\}$.

Let π be a protocol that 1-securely computes f, and let $r(k) = O(\log k)$ be the number of rounds in π. In contrast to the previous section, here we allow for all parties to send messages in a single round. Consider the experiment in which A and C each choose their inputs a, c uniformly at random, and all parties run π honestly except that A or C may possibly abort the protocol early. Define A_i to be the output that B generates if the last message sent by C was in round i (and A, B continue to run the protocol honestly until the end), with A_0 denoting the output of B in case C never sends any messages. Similarly, define C_i to be the output that B generates if the last message sent by A was in round i (and B, C continue to run the protocol honestly until the end). Note that if an adversary controls both A and B, then A_i can be computed immediately after C sends its round-i message. (And analogously for C_i.) Also, with all but negligible probability we have $A_i \in \{a, \star\}$ and $C_i \in \{c, \star\}$ since π is fully secure when only one party aborts, and thus the output of the (honest-looking) B must be consistent with the input of the remaining honest party. For convenience, we will simply assume that this occurs with probability 1.

Define Sum_i for odd $i \in \{1, \dots, r\}$ as follows:

$$\begin{aligned} \mathsf{Sum}_i \stackrel{\text{def}}{=} & \left(\Pr\left[A_i = a \bigwedge a = c\right] - \Pr\left[A_i = a \bigwedge a \neq c\right] + \Pr\left[A_i = \star \bigwedge C_{i-1} = c\right] - \frac{1}{3}\right) \\ & + \left(\frac{1}{3} - \Pr\left[A_i = \star \bigwedge C_{i-1} = c\right] - \Pr\left[A_i = a \bigwedge C_{i-1} = c\right]\right) \\ & + 2 \cdot \left(\Pr\left[A_i = a \bigwedge a \neq c\right] + \Pr\left[A_i = a \bigwedge a = c\right] - \frac{1}{3}\right). \end{aligned}$$

Sum_i for even $i \in \{1, \dots, r\}$ is defined analogously by interchanging the roles of A and C:

$$\begin{aligned} \mathsf{Sum}_i \stackrel{\text{def}}{=} & \left(\Pr\left[C_i = c \bigwedge a = c\right] - \Pr\left[C_i = c \bigwedge a \neq c\right] + \Pr\left[C_i = \star \bigwedge A_{i-1} = a\right] - \frac{1}{3}\right) \\ & + \left(\frac{1}{3} - \Pr\left[C_i = \star \bigwedge A_{i-1} = a\right] - \Pr\left[C_i = c \bigwedge A_{i-1} = a\right]\right) \\ & + 2 \cdot \left(\Pr\left[C_i = c \bigwedge a \neq c\right] + \Pr\left[C_i = c \bigwedge a = c\right] - \frac{1}{3}\right). \end{aligned}$$

Define also $X_i = \left(\frac{1}{3} - \Pr[A_i = a \bigwedge a = c]\right)$ for i odd, and $X_i = \left(\frac{1}{3} - \Pr[C_i = c \bigwedge a = c]\right)$ for i even (now including 0). The following claim follows by simple algebraic manipulation (see Appendix A):

Claim 1 *For all $i \in \{1, \dots, r\}$ it holds that $3 \cdot X_i + \mathsf{Sum}_i \geq X_{i-1}$.*

By induction, it follows that for any $j \leq i$ we have $3^j \cdot X_i + \sum_{k=i-j+1}^{i} 3^{k-(i-j+1)} \cdot \mathsf{Sum}_k \geq X_{i-j}$ and so in particular

$$
\begin{aligned}
3^r \cdot X_r + \sum_{i=1}^{r} 3^{i-1} \cdot \mathsf{Sum}_i &\geq X_0 \\
&= \left(\frac{1}{3} - \Pr\left[C_0 = c \bigwedge a = c\right]\right) \\
&= \left(\frac{1}{3} - \frac{1}{3} \cdot \Pr[C_0 = c]\right) \geq \frac{2}{9} - \varepsilon,
\end{aligned} \tag{7}
$$

for ε approaching 0 as k grows large. The second equality uses the facts that C_0 is independent of a, and a, c are chosen uniformly in $\{1, 2, 3\}$. The final inequality is by the 1-security of π (since if $\Pr[C_0 = c] > \frac{1}{3}$ then an adversary corrupting A who aborts immediately causes the honest player B to output c with probability greater than 1/3, something that cannot occur in the ideal world).

Since $r = r(k)$ is logarithmic in k, Eq. (7) implies that either $X_r \geq \frac{1}{\mathsf{poly}(k)}$ or, for some i, $\mathsf{Sum}_i \geq \frac{1}{\mathsf{poly}(k)}$. Since $a = c$ with probability 1/3 and $A_r = C_r = a = c$ with all but negligible probability when this occurs (this follows from correctness of π), the former cannot be the case. As in Section 3.1, we show that the latter contradicts security of π. Without loss of generality, assume $\mathsf{Sum}_i \geq \frac{1}{\mathsf{poly}(k)}$ for an odd index i. Then one of the three summands in the definition of Sum_i is noticeable. Consider the three possibilities:

Case 1. Say $\left(\Pr\left[A_i = a \bigwedge a = c\right] - \Pr\left[A_i = a \bigwedge a \neq c\right] + \Pr\left[A_i = \star \bigwedge C_{i-1} = c\right] - \frac{1}{3}\right)$ is noticeable. Consider the adversary $\mathcal{A}$ with auxiliary input $z = i$ who corrupts A and B, chooses input a uniformly at random, and runs the protocol honestly except that it aborts A at round i (so A's last message is sent in round $i-1$). Player B runs the entire protocol honestly, and so the output of (the adversarially-controlled player) B is exactly C_{i-1}. Furthermore, using the message sent by C in round i as well as the internal states of players A and B at that stage of the protocol, $\mathcal{A}$ can also compute A_i. If $A_i \neq \star$, then $\mathcal{A}$ outputs A_i. If $A_i = \star$ and $C_{i-1} \neq \star$ then $\mathcal{A}$ outputs C_{i-1}. Finally, if $A_i = \star = C_{i-1}$ then $\mathcal{A}$ outputs "don't know."

Viewing $\mathcal{A}$ as outputting a guess for the value of c (and allowing $\mathcal{A}$ to output "don't know"), we see that $\mathcal{A}$ is correct with probability $\Pr\left[A_i = a \bigwedge a = c\right] + \Pr\left[A_i = \star \bigwedge C_{i-1} = c\right]$ (recall that $C_{i-1} \in \{c, \star\}$) and incorrect with probability $\Pr\left[A_i = a \bigwedge a \neq c\right]$. Thus, its *bias* in guessing c (defined as the probability that it guesses correctly minus the probability that it guesses incorrectly) is noticeably better than 1/3. Yet in the ideal model, no adversary corrupting A and B can guess c with bias better than 1/3 when C's input is chosen uniformly at random: the best strategy is to send an arbitrary input a to the trusted party computing f, output a if the trusted party returns a as output, and output a random guess otherwise (or simply output "don't know" in this case). It is easy to turn this observation into a method for distinguishing, with noticeable probability, between the output of $\mathcal{A}$ (in the real world) and the output of any ideal-world adversary.[8] We conclude that π does not 2-privately compute f in the presence of malicious adversaries.

Case 2(a). Say $\left(\frac{1}{3} - \Pr\left[A_i = \star \bigwedge C_{i-1} = c\right] - \Pr\left[A_i = a \bigwedge C_{i-1} = c\right]\right)$ is noticeable, and note that this means that $\Pr[C_{i-1} = c] \leq \frac{1}{3} - 1/\mathsf{poly}(k)$. Then consider the adversary who corrupts A, chooses input a at random, and runs honestly until round $i-1$ at which point it aborts. Then the output of the honest party B is equal to C's input c with probability noticeably different from 1/3. But no

[8]For example, consider the distinguisher that with probability half outputs 1 if the guess for c is correct (and 0 otherwise), and with probability half outputs 0 if the guess for c is incorrect (and 1 otherwise).

adversary corrupting A in the ideal world (without abort) can cause B to output c with probability different from $1/3$ when c is chosen uniformly at random. Thus, this possibility cannot occur since it would contradict the assumption that π 1-securely computes f.

Case 2(b). The case when $\left(\Pr\left[A_i = a \bigwedge a \neq c\right] + \Pr\left[A_i = a \bigwedge a = c\right] - \frac{1}{3}\right)$ is noticeable is analogous to the above, in that we can construct a real-world adversary who corrupts C and causes B to output A's input a with probability noticeably different from $1/3$. This would again contradict the assumption that π 1-securely computes f.

5 A Positive Result

The previous section shows that even if we allow simultaneous message transmission, positive results will be hard to come by (at least in the sense that the round complexity for any protocol achieving the "best of both worlds" must be super-logarithmic). In fact, we conjecture (though, obviously, have not yet been able to prove) that even protocols with polynomially-many rounds cannot achieve the best of both worlds in that setting, and so some additional relaxation is needed.

Here, we show a positive result for our problem: for any p, we show a protocol with round complexity $O(pn^2)$ which is (1) fully secure in the presence of an honest majority, even for the case of a rushing adversary and (2) can be simulated to within $O(1/p)$ (in a sense we will soon precisely define) in the presence of arbitrarily-many malicious (but non-rushing) parties. The protocol is also fully-secure for any number of semi-honest parties; see further discussion below.

We now define the notion to which we have informally referred above. Say two distribution ensembles X, Y are indistinguishable to within δ, denoted $X \overset{\delta}{\approx} Y$, if there exists a negligible function ε such that for every PPT algorithm D and all a we have

$$\left|\Pr[D(1^k, a, X(k,a)) = 1] - \Pr[D(1^k, a, Y(k,a)) = 1]\right| \leq \delta(k) + \varepsilon(k).$$

If f is an n-party functionality and π is an n-party protocol, then π t-securely computes f with abort to within δ if for any PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{A}'$ such that for any $I \subset [n]$ with $|I| \leq t$ we have $\textsc{real}_{\pi,\mathcal{A},I} \overset{\delta}{\approx} \textsc{ideal}_{f,\mathcal{A}',I}$. (We could define a similar notion for the case of full security but we will not need it.) The main result of this section is the following:

Theorem 3 *Under suitable cryptographic assumptions (inherited from [20, 22]), for any n, any $t < n/2$, any $s < n$, any n-party poly-time functionality f, and any polynomial p, there exists an $O(pt^2)$-round protocol that t-securely computes f (for rushing adversaries) and s-securely computes f with abort to within $O(1/p)$ (for non-rushing adversaries). The protocol also s-securely computes f for semi-honest adversaries.*

Our protocol does not weaken the security obtained when fewer than half the parties are corrupted (namely, we achieve the standard notion of security in that setting, even for rushing adversaries); thus, the protocol is an improvement over "standard" protocols achieving security for honest majority in that at least not all is lost in case half or more of the parties are corrupted. Furthermore, our protocol is a strict improvement of the main positive result from [19] since we also achieve (full) security against an arbitrary number of *semi-honest* adversaries. Though not explicitly mentioned in the theorem, our protocol is also a strict improvement of the second positive result from [19] since our protocol also has the property that the view of a malicious adversary corrupting any $s < n$ parties can be simulated by an ideal-world adversary that is allowed to invoke the functionality $s + 1$ times (this holds even when rushing is allowed).

We now present the details. Fix n, and let f be a poly-time n-party functionality. We assume for ease of presentation that f is a *single-output* functionality; this is without loss of generality since protocols computing such functionalities can be used in the standard way to compute functionalities where each party gets a possibly different output. We also fix $t = \lfloor (n-1)/2 \rfloor$, though reducing the number of rounds (as claimed) for smaller values of t is straightforward.

Assume that the output of f can be viewed as lying in some field $\mathbb{F}$ which also contains $[n]$. (Actually, if the output of f is v bits long we need a field which can represent strings of length at least $v+1$.) Let $h = \lfloor n/2 \rfloor + 1$. For a polynomial p given as a parameter, define the following randomized functionality F for parties $H \subseteq [n]$ and inputs $\{x_i\}_{i \in H}$:

> For each $i \notin H$, set x_i to some fixed default value (as determined by f)
> Compute $y \leftarrow f(x_1, \ldots, x_n)$
> Choose random $s^* \in \mathbb{F}$ and random $i^* \in \{1, \ldots, pn\}$
> Generate an h-out-of-$|H|$ sharing of s^*
> For $1 \leq i < i^*$, generate an $|H|$-out-of-$|H|$ sharing of $s^i \stackrel{\text{def}}{=} 0$
> For $i^* \leq i \leq pn$, generate an $|H|$-out-of-$|H|$ sharing of $s^i \stackrel{\text{def}}{=} 1\|(y + s^*)$
> Each party in H receives as output its share of each of the above $pn + 1$ values

When we refer to "secret sharing," we always intend the classical scheme by Shamir [25]. We also require implicitly that the shares given to the parties are authenticated, e.g., by being signed with respect to a public key generated by F and given to all parties. (We do not explicitly mention this in what follows.)

We now have the following protocol to compute f. Each party begins by setting $H = \{P_1, \ldots, P_n\}$. (Intuitively, this represents the set of parties currently believed to be honest.) Then:

Phase 1. Do the following until it either completes successfully or until $|H| < h$:

1. Run an $|H|$-party protocol π_F among parties in H that $(|H|-1)$-securely computes F with abort (even for rushing adversaries) on inputs $\{x_i\}_{i \in H}$.
2. If all parties receive their output, proceed to phase 2. Otherwise, if honest parties abort because of detected cheating[9] by a party P_i, remove P_i from H and return to step 1.

If $|H| < h$ at any point, parties output $\perp$ and terminate the entire protocol.

Phase 2. In round i, for $i = 1, \ldots, pn$, each party $P_j \in H$ broadcasts its share of s^i. This occurs until either s^i is of the form $1\|y'$ or until some player cheats in round i (a player is considered cheating if they do not broadcast their share, or if they broadcast an incorrect value; recall our assumption that all shares are authenticated). Note that if $i > 1$ then all parties can compute the secret s^{i-1} from the shares correctly broadcast in the previous round. Then:

- If $s^i = 1\|y'$, then each party broadcasts its share of s^*. If at least h parties broadcast their share of s^* correctly, then reconstruct s^* and output $y' - s^*$; otherwise, output $\perp$ and terminate the protocol.
- If the above does no occur, and some player cheats in round i (and so $i = 1$ or $s^{i-1} = 0$), remove all parties who were caught cheating from H. If $|H| < h$ then output $\perp$ and terminate the protocol. Otherwise, return to phase 1.

Using [20, 22], there exists a constant-round protocol π_F with the necessary security properties. Thus, the worst-case round complexity of the above protocol is $pn \cdot \lfloor (n-1)/2 \rfloor = O(pn^2)$.

We sketch the claimed security properties for the above protocol in Appendix B.

[9] We assume π_F has the property that cheaters are identified; known protocols have this property.

Acknowledgments

I am very grateful to Adam Smith for numerous highly informative conversations on this topic (some going back many years...). I would also like to thank Yuval Ishai, Chiu-Yuen Koo, and Yehuda Lindell for discussions at the initial stages of this research, and Chiu and Dov Gordon for proofreading earlier versions of this manuscript and pointing out some typos. Thanks go again to Yuval for suggesting that timed commitments might be used to implement simultaneous message transmission.

References

[1] D. Beaver. Multiparty Protocols Tolerating Half Faulty Processors. Crypto '89.

[2] D. Beaver and S. Goldwasser. Multiparty Computation with Faulty Majority. FOCS '89.

[3] D. Beaver, S. Micali, and P. Rogaway. The Round Complexity of Secure Protocols. STOC '90.

[4] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. STOC '88.

[5] D. Boneh and M. Naor. Timed Commitments. Crypto 2000.

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

[7] D. Chaum, C. Crépeau, and I. Damgård. Multiparty Unconditionally-Secure Protocols. STOC '88.

[8] R. Cleve. Limits on the Security of Coin Flips when Half the Processors are Faulty. STOC '86.

[9] R. Cleve. Controlled Gradual Disclosure Schemes for Random Bits and Their Applications. Crypto '89.

[10] S. Even, O. Goldreich, and A. Lempel. A Randomized Protocol for Signing Contracts. *Comm. ACM* 28(6): 637–647, 1985.

[11] M. Fitzi, M. Hirt, and U. Maurer. Trading Correctness for Privacy in Unconditional Multiparty Computation. Crypto '98.

[12] M. Fitzi, M. Hirt, T. Holenstein, and J. Wullschleger. Two-Threshold Broadcast and Detectable Multiparty Computation. Eurocrypt 2003.

[13] M. Fitzi, M. Hirt, and J. Wullschleger. Multiparty Computation with Hybrid Security. Eurocrypt 2004.

[14] J. Garay, P. MacKenzie, M. Prabhakaran, and K. Yang. Resource Fairness and Composability of Cryptographic Protocols. TCC 2006.

[15] O. Goldreich. *Foundations of Cryptography, vol. 2: Basic Applications.* Cambridge University Press, 2004.

[16] O. Goldreich, S. Micali, and A. Wigderson. How to Play any Mental Game — A Completeness Theorem for Protocols with Honest Majority. STOC '87.

[17] S. Goldwasser and L. Levin. Fair Computation of General Functions in Presence of Immoral Majority. Crypto '90.

[18] S. Goldwasser and Y. Lindell. Secure Multiparty Computation without Agreement. *J. Cryptology* 18(3): 247–287, 2005.

[19] Y. Ishai, E. Kushilevitz, Y. Lindell, and E. Petrank. On Combining Privacy with Guaranteed Output Delivery in Secure Multiparty Computation. Crypto 2006.

[20] J. Katz, R. Ostrovsky, and A. Smith. Round Efficiency of Multi-Party Computation with a Dishonest Majority. Eurocrypt 2003.

[21] M. Luby, S. Micali, and C. Rackoff. How to Simultaneously Exchange a Secret Bit by Flipping a Symmetrically-Biased Coin. FOCS '83.

[22] R. Pass. Bounded-Concurrent Secure Multi-Party Computation with a Dishonest Majority. STOC 2004.

[23] B. Pinkas. Fair Secure Two-Party Computation. Eurocrypt 2003.

[24] T. Rabin and M. Ben-Or. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. STOC '89.

[25] A. Shamir. How to Share a Secret. *Comm. ACM* 22(6): 612–613, 1979.

[26] A. Yao. How to Generate and Exchange Secrets. FOCS '86.

A Proof of Claim 1

We consider the case of i odd; the case when i is even follows similarly. Note first that the expression defining Sum_i simplifies to

$$\mathsf{Sum}_i = 3 \cdot \Pr\left[A_i = a \bigwedge a = c\right] - \Pr\left[A_i = a \bigwedge C_{i-1} = c\right] + \Pr\left[A_i = a \bigwedge a \neq c\right] - \frac{2}{3}.$$

We then have

$$\begin{aligned}
&3 \cdot X_i + \mathsf{Sum}_i \\
&= 3 \cdot \left(\frac{1}{3} - \Pr\left[A_i = a \bigwedge a = c\right]\right) + \mathsf{Sum}_i \\
&= \Pr\left[A_i = a \bigwedge a \neq c\right] - \Pr\left[A_i = a \bigwedge C_{i-1} = c\right] + \frac{1}{3} \\
&= \Pr\left[A_i = a \bigwedge a \neq c \bigwedge C_{i-1} = c\right] + \Pr\left[A_i = a \bigwedge a \neq c \bigwedge C_{i-1} \neq c\right] \\
&\quad - \Pr\left[A_i = a \bigwedge C_{i-1} = c \bigwedge a = c\right] - \Pr\left[A_i = a \bigwedge C_{i-1} = c \bigwedge a \neq c\right] + \frac{1}{3} \\
&= \Pr\left[A_i = a \bigwedge a \neq c \bigwedge C_{i-1} \neq c\right] - \Pr\left[A_i = a \bigwedge C_{i-1} = c \bigwedge a = c\right] + \frac{1}{3} \\
&\geq \frac{1}{3} - \Pr\left[C_{i-1} = c \bigwedge a = c\right] = X_{i-1}.
\end{aligned}$$

B Proof of Theorem 3 (Sketch)

We consider each of the claimed security properties in turn.

B.1 Security for $n-1$ Semi-Honest Adversaries

We first show that the protocol of Section 5 is fully secure for any number of semi-honest adversaries. This is quite easy to see. Consider a semi-honest adversary corrupting $s < n$ parties. Phase 1 will complete successfully with $H = \{P_1, \ldots, P_n\}$. Security of π_F implies that the adversary learns no information in phase 1 other than its output, which consists of the relevant shares of the secrets $s^*, s^1, \ldots, s^{pn}$. Its shares may completely determine s^*, but this is just a random value. In phase 2, the adversary learns the values $s^1, \ldots, s^{i^*}, s^*$; the first $i^* - 1$ of these are just the value "0," while s^{i^*} and s^* are random subject to the fact that their xor is equal to the correct output y. All honest players also output y, which is distributed correctly since it was computed by π_F in phase 1.

B.2 Security for $t < n/2$ Malicious, Rushing Adversaries

Here, we show that the protocol of Section 5 is fully secure for any $t < n/2$ malicious, rushing adversaries. Intuitively, we may argue as follows: Regardless of the adversary's actions in phase 1, it learns nothing more than t shares of secrets $s^*, s^1, \ldots, s^{pn}$. Since the threshold for each of these secrets is strictly greater than t, the adversary learns nothing about the actual secrets.

In phase 2, the adversary is essentially reduced to a fail-stop adversary since all shares are authenticated. If the adversary never aborts in a particular iteration of phase 2, then all parties learn a (correctly-distributed) output y in that iteration. It is also clear that honest parties will eventually compute some correctly-distributed output y that takes into account the inputs of all honest parties. We need only show that the adversary does not learn more than it is supposed to.

Consider what happens if some set of adversarially-controlled parties aborts in round i of an iteration of phase 2. There are two cases:

- If $i = 1$ or $s^{i-1} = 0$, the adversary may learn the value $y' = s^* + y$ (from the shares broadcast by the honest parties in round i) but this value is random and independent of the correct output y (the key point is that the adversary learns nothing about s^* and so y' is indeed random). Then the protocol continues by eliminating some corrupted parties from H and again running phase 1.
- If $s^{i-1} = 1\|y'$, then all parties reconstruct y and terminate the protocol (and the adversary learns only the output value y).

Formally, note that the protocol described above can be viewed naturally as a protocol π running in the F-hybrid model. We will prove that π defined in this way t-securely computes f in the F-hybrid model; that is (roughly speaking), that for every PPT adversary $\mathcal{A}$ corrupting at most t parties and running π in the F-hybrid model, there exists an adversary $\mathcal{A}'$ corrupting at most t parties in the ideal world with access to a trusted party computing f such that the outcomes of these two executions (which includes, as in Section 2, both the output of the adversary as well as the outputs of the honest parties) are indistinguishable. The composition theorem of Canetti [6] then implies that the entire real-world protocol (in which access to F is emulated using a secure protocol π_F) is secure.

Let $\mathcal{A}$ be a hybrid-model adversary controlling corrupted parties in I. Consider the following ideal-world adversary $\mathcal{A}'$ that interacts with a trusted party computing f. $\mathcal{A}'$ begins by setting

$H = \{P_1, \ldots, P_n\}$. Then it operates in the following phases: (Note that it will always hold that $|H| \geq h$ since we have an honest majority here; we use this to simplify what follows.)

Phase 1. $\mathcal{A}'$ does the following:

1. If $I \cap H = \emptyset$, then choose a random index $i^* \in \{1, \ldots, pn\}$ and proceed to phase 2.

2. Otherwise, obtain from $\mathcal{A}$ inputs $\{x_i\}_{i \in I \cap H}$ for F on behalf of all parties in $I \cap H$. Adversary $\mathcal{A}'$ records these values, but *does not yet send these to the trusted party computing f.*

3. $\mathcal{A}'$ provides to $\mathcal{A}$, on behalf of each party in $I \cap H$, randomly-generated shares of $pn + 1$ random secrets. (These shares are also authenticated in the obvious way.) At this point, $\mathcal{A}$ is allowed to determine whether to abort computation of F or whether to allow computation of F to continue. In the former case, $\mathcal{A}'$ chooses a random index $i^* \in \{1, \ldots, pn\}$ and proceeds to phase 2. In the latter case, at least one party in $I \cap H$ is identified as[10] malicious; this party is removed from H and $\mathcal{A}'$ returns to step 1.

Phase 2. Entering phase 2, $\mathcal{A}'$ holds inputs $\{x_i\}_{i \in I \cap H}$ and an index i^*. For each round i of (the current iteration of) this phase, $\mathcal{A}'$ does the following:

Case 1: $i < i^*$. Do:

1. Broadcast (on behalf of the honest parties) random shares consistent with the appropriate shares given to $\mathcal{A}$ in phase 1 (if any) and the secret $s^i = 0$.

2. After receiving the above, $\mathcal{A}$ broadcasts its round-i messages on behalf of players in $I \cap H$. If these messages are exactly the appropriate shares given to $\mathcal{A}$ in phase 1 (or if $I \cap H = \emptyset$), continue to the next round. Otherwise, identify the cheating parties in $I \cap H$, remove them from H, and return to phase 1.

Case 2: $i = i^*$. Do:

1. Broadcast (on behalf of the honest parties) random shares consistent with the appropriate shares given to $\mathcal{A}$ in phase 1 (if any) and the secret $s^{i^*} = 1 \| y'$ for random y'.

2. After receiving the above, $\mathcal{A}$ broadcasts its round-i^* messages on behalf of players in $I \cap H$. If these messages are exactly the appropriate shares given to $\mathcal{A}$ in phase 1 (or if $I \cap H = \emptyset$), then:

   - Send $\{x_i\}_{i \in I \cap H}$ to the trusted party computing f; obtain in return an output y.
   - Broadcast (on behalf of the honest parties) random shares consistent with the appropriate shares given to $\mathcal{A}$ in phase 1 (if any) and the secret $s^* = y' - y$.

   Otherwise, identify the cheating parties in $I \cap H$, remove them from H, and return to phase 1.

It is not hard to see that the above provides a good simulation.

[10]Technically, we need to augment the ideal world for the case of security with abort so that the adversary is allowed to choose which malicious players are identified as malicious.

B.3 Security for $s < n$ Malicious, Non-Rushing Adversaries

We now show that the protocol in Section 5 is secure with abort to within $O(1/p)$ for $s < n$ malicious, non-rushing adversaries. Intuitively, we may argue as follows: in phase 1 the adversary learns nothing, even if it aborts execution of π_F after receiving its own outputs (but before honest players receive theirs). In a given iteration of phase 2, the adversary learns "more than it is supposed to" only if it aborts in round i^*: if it aborts in round $i < i^*$ then (as in the previous case) it learns nothing and the protocol continues with another execution of phase 1; if it aborts in round $i > i^*$ then (similar to the previous case) it learns the output y but all honest parties either learn y also or else terminate the protocol with output $\perp$.

Thus, the only difficulty is in case there exists an iteration of phase 2 in which some adversarial party aborts in round i^*. Note that in the previous section (when the number of corrupted parties is less than $n/2$), the adversarial parties did not learn the value of s^* and so learning y' did not leak any information. Here, however, the adversary may have corrupted h or more parties in which case it learns s^* in addition to y'.

However, the adversary can abort in round i^* only with probability at most $1/pn$ in any given iteration because it has no information about the value of i^* (since it corrupts strictly fewer than n parties). We remark that here is where we rely on the fact that the adversary is non-rushing: it must decide whether or not to abort in a given round *before* learning whether or not that round corresponds to round i^*. In contrast, a rushing adversary could always wait until the honest parties send their round-i messages; determine whether the secret s^i reconstructs to $1\|y'$, and abort if and only if this is the case.

Since there are at most $O(n)$ iterations of phase 2, and the adversary guesses i^* correctly with probability only $1/pn$ in any given iteration, a union bound shows that the adversary can correctly guess i^* in *some* iteration only with probability at most $O(1/p)$. This observation leads naturally to an ideal-world adversary which generates an execution that is indistinguishable from a real execution to within $O(1/p)$.

Formally, we again view the main protocol as a protocol π running in the F-hybrid model. As before, we prove that π defined in this way s-securely computes f (with abort) to within $O(1/p)$ in the F-hybrid model. Although the composition theorem of Canetti [6] no longer applies directly (since we prove security to within $O(1/p)$ rather than security to within a negligible factor), it is not hard to see that the proof from [6] extends to imply that the entire real-world protocol (in which access to F is emulated using a secure protocol π_F) is similarly secure.

Let $\mathcal{A}$ be a hybrid-model adversary controlling corrupted parties in I, where we assume $|I| \geq n/2$ since otherwise the proof given previously applies. Consider the following ideal-world adversary $\mathcal{A}'$ that interacts with a trusted party computing f. $\mathcal{A}'$ begins by setting $H = \{P_1, \ldots, P_n\}$. Then it operates in the following phases:

Phase 1.

1. If $|H| < h$, $\mathcal{A}'$ sends arbitrary inputs to the trusted party computing f, receives output (that it ignores), and aborts the trusted party before it gives output to any of the honest parties.

2. Otherwise, $\mathcal{A}'$ obtains from $\mathcal{A}$ inputs $\{x_i\}_{i \in I \cap H}$ for F on behalf of all parties in $I \cap H$. Adversary $\mathcal{A}'$ records these values, but *does not yet send these to the trusted party computing f.*

3. $\mathcal{A}'$ chooses random s^*, and provides to $\mathcal{A}$ (on behalf of each party in $I \cap H$) randomly-generated shares of s^* as well as pn other, random secrets. (These shares are authenticated in the obvious way.) At this point, $\mathcal{A}$ is allowed to determine whether to abort computation

of F or whether to allow computation of F to continue. In the former case, $\mathcal{A}'$ chooses a random index $i^* \in \{1, \ldots, pn\}$ and proceeds to phase 2. In the latter case, at least one party in $I \cap H$ is identified as malicious (see footnote 10); this party is removed from H and $\mathcal{A}'$ returns to step 1.

Phase 2. Entering phase 2, $\mathcal{A}'$ holds inputs $\{x_i\}_{i \in I \cap H}$ and an index i^*. For each round i of (the current iteration of) this phase, $\mathcal{A}'$ does the following:

Case 1: $i < i^*$. Do:

1. $\mathcal{A}$ broadcasts its round-i messages on behalf of players in $I \cap H$. (Recall that we are assuming a non-rushing adversary, and so $\mathcal{A}$ must decide on its round-i messages before it observes the round-i messages of the honest parties.)

2. $\mathcal{A}'$ broadcasts (on behalf of the honest parties) random shares consistent with the appropriate shares given to $\mathcal{A}$ in phase 1 (if any) and the secret $s^i = 0$.

3. If the round-i messages sent by $\mathcal{A}$ are exactly the appropriate shares given to $\mathcal{A}$ in phase 1, continue to the next round. Otherwise, identify the cheating parties in $I \cap H$, remove them from H, and return to phase 1.

Case 2: $i = i^*$. Do:

1. $\mathcal{A}$ broadcasts its round-i^* messages on behalf of players in $I \cap H$.

2. If the round-i^* messages sent by $\mathcal{A}$ are exactly the appropriate shares given to $\mathcal{A}$ in phase 1, then

   - Send $\{x_i\}_{i \in I \cap H}$ to the trusted party computing f; obtain in return an output y.
   - Broadcast (on behalf of the honest parties) random shares consistent with the appropriate shares given to $\mathcal{A}$ in phase 1 and the secret $s^{i^*} = 1 \| (y + s^*)$.
   - $\mathcal{A}$ then broadcasts its supposed shares of s^* and $\mathcal{A}'$ broadcasts the honest parties' shares of s^*. If at least h (correct) shares of s^* were broadcast in total, A' tells the trusted party computing f to continue; otherwise, it tells the trusted party computing f to abort.

   Otherwise (i.e., at least one round-i^* message broadcast by $\mathcal{A}$ is incorrect), $\mathcal{A}'$ terminates the simulation (unsuccessfully) and outputs fail.

It is not too difficult to see that when $\mathcal{A}'$ does not output fail, it outputs a good simulation. Furthermore, it outputs fail with probability at most $O(n)/pn = O(1/p)$.