Complete Fairness in Multi-Party Computation Without an Honest Majority

S. Dov Gordon* Jonathan Katz*

Abstract

Gordon et al. recently showed that certain (non-trivial) functions can be computed with complete fairness in the *two-party* setting. Motivated by their results, we initiate a study of complete fairness in the *multi-party* case and demonstrate the first completely-fair protocols for non-trivial functions in this setting. We also provide evidence that achieving fairness is "harder" in the multi-party setting, at least with regard to round complexity.

*Dept. of Computer Science, University of Maryland. Email: {gordon,jkatz}@cs.umd.edu. This work was supported by NSF CAREER award #0447075 and US-Israel Binational Science Foundation grant #2004240.

1 Introduction

In the setting of secure computation, a group of parties wish to run a protocol for computing some function of their inputs while preserving, to the extent possible, security properties such as privacy, correctness, input independence and others. These requirements are formalized by comparing a real-world execution of the protocol to an *ideal world* where there is a trusted entity who performs the computation on behalf of the parties. Informally, a protocol is "secure" if for any real-world adversary $\mathcal{A}$ there exists a corresponding ideal-world adversary $\mathcal{S}$ (corrupting the same parties as $\mathcal{A}$) such that the result of executing the protocol in the real world with $\mathcal{A}$ is computationally indistinguishable from the result of computing the function in the ideal world with $\mathcal{S}$.

One desirable property is *fairness* which, intuitively, means that either *everyone* receives the output, or else *no one* does. Unfortunately, it has been shown by Cleve [4] that complete fairness is impossible *in general* without a majority of honest parties. Until recently, Cleve's result was interpreted to mean that *no* non-trivial functions could be computed with complete fairness without an honest majority. A recent result of Gordon et al. [7], however, shows that this folklore is wrong; there exist non-trivial functions than *can* be computed with complete fairness in the two-party setting. Their work demands that we re-evaluate our current understanding of fairness.

Gordon et al. [7] deal exclusively with the case of two-party computation, and leave open the question of fairness in the multi-party setting. Their work does *not* immediately extend to the case of more than two parties. (See also the discussion in the section that follows.) Specifically, an additional difficulty that arises in the multi-party setting is the need to ensure consistency between the outputs of the honest parties, even after a malicious abort. In the two-party setting, the honest player can respond to an abort by replacing the malicious player's input with a random value and computing the function locally (so long as it is "early" enough in the protocol that the malicious player has not yet learned anything about the output). This option is not immediately available in the multi-party setting, as the honest players do not know each other's input values, and cannot replace them with random values. This issue is compounded with the adversary's ability to adaptively abort the t malicious players in any order and at anytime, making fairness in the multi-party setting still harder to achieve.

In light of the above discussion, we initiate the study of complete fairness in the multi-party setting. We focus on the case when a private broadcast channel (or, equivalently, a PKI) is available to the parties; note that Cleve's impossibility result applies in this case as well. Although one can meaningfully ask what can be achieved in the absence of broadcast, we have chosen to assume private broadcast so as to separate the question of *fairness* from the question of *agreement* (which has already been well studied in the cryptographic and distributed systems literature). We emphasize that, as in [7], we are interested in obtaining *complete* fairness rather than some notion of *partial* fairness.

1.1 Our Results

A natural first question is whether two-party feasibility results [7] can be extended "easily" to the multi-party setting. More formally, say we have a function $f : \{0,1\} \times \cdots \times \{0,1\} \to \{0,1\}$ taking n boolean inputs. (We restrict to boolean inputs/outputs for simplicity only.) For any subset $\emptyset \subset I \subset [n]$, we can define the *partition* f_I of f to be the two-input function $f_I : \{0,1\}^{|I|} \times \{0,1\}^{n-|I|}$ defined as

$$f_I(y, z) = f(x),$$

where $x \in \{0,1\}^n$ is such that $x_I = y$ and $x_{\bar{I}} = z$. It is not hard to see that if there exists an I for which f_I *cannot* be computed with complete fairness in the two-party setting, then f cannot be computed with complete fairness in the multi-party setting. Similarly, the round complexity for computing f with complete fairness in the multi-party case must be at least the round complexity of fairly computing each f_I. What about the converse? We show the following negative result regarding such a "partition-based" approach to the problem:

Theorem 1 *(Under suitable cryptographic assumptions) there exists a 3-party function f all of whose partitions can be computed with complete fairness in $O(1)$ rounds, but for which any protocol computing f with complete fairness requires $\omega(\log k)$ rounds, where k is the security parameter.*

This seems to indicate that fairness in the multi-party setting is qualitatively harder than fairness in the two-party setting.

The function f for which we prove the above theorem is interesting in its own right: it is the 3-party *majority* function (i.e., voting). Although the $\omega(\log k)$-round lower bound may seem discouraging, we are able to show a positive result for this function; to the best of our knowledge, this represents the first non-trivial feasibility result for complete fairness in the multi-party setting.

Theorem 2 *(Under suitable cryptographic assumptions) there exists an $\omega(\log k)$-round protocol for securely computing 3-party majority with complete fairness.*

Unfortunately, our efforts to extend the above result to the case of n-party majority have been unsuccessful. One may therefore wonder whether there exists *any* (non-trivial) function that can be computed with complete fairness for general n. Indeed, there is:

Theorem 3 *(Under suitable cryptographic assumptions) for any number of parties n, there exists an $O(n)$-round protocol for securely computing boolean* OR *with complete fairness.*

OR is non-trivial in our context: OR is complete for multi-party computation (without fairness) [9], and cannot be computed with information-theoretic privacy even in the two-party setting [3].

Relation to prior work. At a superficial level, the proof of the $\omega(\log k)$-round lower bound of Theorem 1 uses an approach similar to that used to prove an analogous lower bound in [7]. We stress, however, that the theorem does not follow as a corollary of that work (indeed, it cannot since each of the partitions of f can be computed with complete fairness in $O(1)$ rounds). Furthermore, we need to introduce new ideas to prove the result in our setting; in particular, we rely in an essential way on the fact that the output of any two honest parties must agree (whereas this issue does not arise in the two-party setting considered in [7]).

Ishai et al. [8] propose a protocol for computing the *sum* of n inputs that is resilient to a dishonest majority in a weaker sense than that considered here. Specifically, they show that when $t < n$ parties are corrupted then a real execution of the protocol is as secure as an execution in the ideal world with complete fairness, but where the adversary can query the ideal functionality $O(t)$ times (using different inputs each time). We refer the reader to their work for further discussion.

1.2 Outline of the Paper

We include the standard definitions of secure multi-party computation in Appendix A. We stress that although the definitions are standard, what is *not* standard is that we are interested in attaining complete fairness even though we do not have an honest majority.

We begin with our negative result, showing that any completely-fair protocol for 3-party majority requires $\omega(\log k)$ rounds. Recall that what is especially interesting about this result is that it demonstrates a gap between the round complexities required for completely-fair computation of a function and its (two-party) partitions. In Section 3, we show feasibility of completely-fair computation of 3-party majority via an $\omega(\log k)$-round protocol. In Section 4 we describe our feasibility result for the case of boolean OR.

2 A Lower Bound on the Round Complexity of Majority

2.1 Proof Overview

In this section, we prove Theorem 1 taking as our function f the three-party majority function maj. That is, $\mathsf{maj}(x_1, x_2, x_3) = 0$ if at least two of the three values $\{x_1, x_2, x_3\}$ are 0, and is 1 otherwise. Note that any partition of maj is just (isomorphic to) the greater-than-or-equal-to function, where the domain of one input can be viewed as $\{0, 1, 2\}$ and the domain of the other input can be viewed as $\{0, 1\}$ (in each case, representing the number of '1' inputs held). Gordon et al. [7] show that, under suitable cryptographic assumptions, the greater-than-or-equal-to function on constant-size domains can be securely computed with complete fairness in $O(1)$ rounds.

We prove Theorem 1 by showing that any completely-fair 3-party protocol for maj requires $\omega(\log k)$ rounds. The basic approach is to argue that if Π is any protocol for securely computing maj, then eliminating the last round of Π results in a protocol Π' that still computes maj correctly "with high probability". Specifically, if the error probability in Π is at most μ (that we will eventually set to some negligible function of k), then the error probability in Π' is at most $c \cdot \mu$ for some constant c. If the original protocol Π has $r = O(\log k)$ rounds, then applying this argument inductively r times gives a protocol that computes maj correctly on all inputs with probability significantly better than guessing *without any interaction at all*. This gives the desired contradiction.

To prove that eliminating the last round of Π cannot affect correctness "too much", we consider a constraint that holds for the ideal-world evaluation of maj. (Recall, we are working in the ideal world where complete fairness holds.) Specifically, consider an adversary who corrupts two parties, and let the input of the honest party P be chosen uniformly at random. The adversary can learn P's input by submitting $(0, 1)$ or $(1, 0)$ to the trusted party. The adversary can also try to bias the output of maj to be the opposite of P's choice by submitting $(0, 0)$ or $(1, 1)$; this will succeed in biasing the result half the time. But the adversary cannot both learn P's input and *simultaneously* bias the result. (If the adversary submits $(0, 1)$ or $(1, 0)$, the output of maj is always equal to P's input; if the adversary submits $(0, 0)$ or $(1, 1)$ the the output of maj reveals nothing about P's input.) Concretely, for any ideal-world adversary, the *sum* of the probability that the adversary guesses P's input and the probability that the output of maj is not equal to P's input is at most 1. In our proof, we show that if correctness holds with significantly lower probability when the last round of Π is eliminated, then there exists a real-world adversary that violates this constraint.

2.2 Proof Details

We number the parties P_1, P_2, P_3, and work modulo 3 in the subscript. The input of P_j is denoted by x_j. The following claim formalizes the ideal-world constraint described informally above.

Claim 4 *For all $j \in \{1,2,3\}$ and any adversary $\mathcal{A}$ corrupting P_{j-1} and P_{j+1} in an ideal-world computation of* maj*, we have*

$$\Pr[\mathcal{A} \text{ correctly guesses } x_j] + \Pr[\text{OUTPUT}_j \neq x_j] \leq 1,$$

where the probabilities are taken over the random coins of $\mathcal{A}$ and random choice of $x_j \in \{0,1\}$.

Proof: Consider an execution in the ideal world, where P_j's input x_j is chosen uniformly at random. Let EQUAL be the event that $\mathcal{A}$ submits two equal inputs (i.e., $x_{j-1} = x_{j+1}$) to the trusted party. In this case, $\mathcal{A}$ learns nothing about P_j's input and so can guess x_j with probability at most 1/2. It follows that:

$$\Pr[\mathcal{A} \text{ correctly guesses } x_j] \leq \frac{1}{2}\Pr[\text{EQUAL}] + \Pr[\overline{\text{EQUAL}}].$$

Moreover,

$$\Pr[\text{OUTPUT}_j \neq x_j] = \frac{1}{2}\Pr[\text{EQUAL}]$$

since $\text{OUTPUT}_j \neq x_j$ occurs only if $\mathcal{A}$ submits $x_{j-1} = x_{j+1} = \bar{x}_j$ to the trusted party. Therefore:

$$\begin{aligned}\Pr[\mathcal{A} \text{ correctly guesses } x_j] + \Pr[\text{OUTPUT}_j \neq x_j] &\leq \frac{1}{2}\Pr[\text{EQUAL}] + \Pr[\overline{\text{EQUAL}}] + \frac{1}{2}\Pr[\text{EQUAL}] \\ &= \Pr[\text{EQUAL}] + \Pr[\overline{\text{EQUAL}}] = 1,\end{aligned}$$

proving the claim. ■

Let Π be a protocol that securely computes maj using $r = r(k)$ rounds. Consider an execution of Π in which all parties run the protocol honestly except for possibly aborting in some round. We denote by $b_j^{(i)}$ the value that P_{j-1} and P_{j+1} both[1] output if P_j aborts the protocol after sending its round-i message (and then P_{j-1} and P_{j+1} honestly run the protocol to completion). Similarly, we denote by $b_{j-1}^{(i)}$ (resp., $b_{j+1}^{(i)}$) the value output by P_j and P_{j+1} (resp., P_j and P_{j-1}) when P_{j-1} (resp., P_{j+1}) aborts after sending its round-i message. Note that an adversary who corrupts, e.g., both P_{j-1} and P_{j+1} can compute $b_j^{(i)}$ immediately after receiving the round-i message of P_j.

Since Π securely computes maj with complete fairness, the ideal-world constraint from the previous claim implies that for all $j \in \{1,2,3\}$, any inverse polynomial $\mu(k)$, and any poly-time adversary $\mathcal{A}$ controlling players P_{j-1} and P_{j+1}, we have:

$$\Pr_{x_j \leftarrow \{0,1\}}[\mathcal{A} \text{ correctly guesses } x_j] + \Pr_{x_j \leftarrow \{0,1\}}[\text{OUTPUT}_j \neq x_j] \leq 1 + \mu(k) \tag{1}$$

for k sufficiently large. Security of Π also guarantees that if the inputs of the honest parties agree, then with all but negligible probability their output must be their common input regardless of when a malicious P_j aborts. That is, for k large enough we have

$$x_{j+1} = x_{j-1} \;\Rightarrow\; \Pr\left[b_j^{(i)} = x_{j+1} = x_{j-1}\right] \geq 1 - \mu(k) \tag{2}$$

[1] It is not hard to see that security of Π implies that the outputs of P_{j-1} and P_{j+1} in this case must be equal with all but negligible probability. For simplicity we assume this to hold with probability 1 but our proof can be modified easily to remove this assumption.

for all $j \in \{1,2,3\}$ and all $i \in \{0,\ldots,r(k)\}$.

The following claim represents the key step in our lower bound.

Claim 5 *Fix a protocol Π, a function μ, and a value k such that Equations (1) and (2) hold, and let $\mu = \mu(k)$. Say there exists an i, with $1 \le i \le r(k)$, such that for all $j \in \{1,2,3\}$ and all $c_1, c_2, c_3 \in \{0,1\}$ it holds that:*

$$\Pr\left[b_j^{(i)} = \mathsf{maj}(c_1,c_2,c_3) \mid (x_1,x_2,x_3) = (c_1,c_2,c_3)\right] \ge 1-\mu. \tag{3}$$

Then for all $j \in \{1,2,3\}$ and all $c_1, c_2, c_3 \in \{0,1\}$ it holds that:

$$\Pr\left[b_j^{(i-1)} = \mathsf{maj}(c_1,c_2,c_3) \mid (x_1,x_2,x_3) = (c_1,c_2,c_3)\right] \ge 1-5\mu. \tag{4}$$

Proof: When $j = 1$ and $c_2 = c_3$, the desired result follows from Equation (2); this is similarly true for $j = 2$, $c_1 = c_3$ as well as $j = 3$, $c_1 = c_2$.

Consider the real-world adversary $\mathcal{A}$ that corrupts P_1 and P_3 and sets $x_1 = 0$ and $x_3 = 1$. Then:

- $\mathcal{A}$ runs the protocol honestly until it receives the round-i message from P_2.
- $\mathcal{A}$ then locally computes the value of $b_2^{(i)}$.
  - If $b_2^{(i)} = 0$, then $\mathcal{A}$ aborts P_1 *without sending its round-i message* and runs the protocol (honestly) on behalf of P_3 until the end. By definition, the output of P_2 will be $b_1^{(i-1)}$.
  - If $b_2^{(i)} = 1$, then $\mathcal{A}$ aborts P_3 *without sending its round-i message* and runs the protocol (honestly) on behalf of P_1 until the end. By definition, the output of P_2 will be $b_3^{(i-1)}$.
- After completion of the protocol, $\mathcal{A}$ outputs $b_2^{(i)}$ as its guess for the input of P_2.

Consider an experiment in which the input x_2 of P_2 is chosen uniformly at random, and then $\mathcal{A}$ runs protocol Π with P_2. Using Equation (3), we have:

$$\begin{aligned}\Pr\left[\mathcal{A} \text{ correctly guesses } x_2\right] &= \Pr\left[b_2^{(i)} = x_2\right] \\ &= \Pr\left[b_2^{(i)} = f(0,x_2,1)\right] \;\ge\; 1-\mu\,. \end{aligned} \tag{5}$$

We also have:

$$\begin{aligned}\Pr\left[\textsc{output}_2 \ne x_2\right] &= \frac{1}{2}\cdot\Pr\left[\textsc{output}_2 = 1 \mid (x_1,x_2,x_3) = (0,0,1)\right] \\ &\quad + \frac{1}{2}\cdot\Pr\left[\textsc{output}_2 = 0 \mid (x_1,x_2,x_3) = (0,1,1)\right] \\ &= \frac{1}{2}\Big(\Pr\left[b_1^{(i-1)} = 1 \wedge b_2^{(i)} = 0 \mid (x_1,x_2,x_3) = (0,0,1)\right] \\ &\quad + \Pr\left[b_3^{(i-1)} = 1 \wedge b_2^{(i)} = 1 \mid (x_1,x_2,x_3) = (0,0,1)\right] \\ &\quad + \Pr\left[b_3^{(i-1)} = 0 \wedge b_2^{(i)} = 1 \mid (x_1,x_2,x_3) = (0,1,1)\right] \\ &\quad + \Pr\left[b_1^{(i-1)} = 0 \wedge b_2^{(i)} = 0 \mid (x_1,x_2,x_3) = (0,1,1)\right]\Big)\,. \end{aligned} \tag{6}$$

¿From Equation (1), we know that the sum of Equations (5) and (6) is upper-bounded by $1+\mu$. Looking at the first summand in (6), this implies that

$$\Pr\left[b_1^{(i-1)} = 1 \wedge b_2^{(i)} = 0 \mid (x_1, x_2, x_3) = (0,0,1)\right] \leq 4\mu. \tag{7}$$

Probabilistic manipulation gives

$$\begin{aligned}
&\Pr\left[b_1^{(i-1)} = 1 \wedge b_2^{(i)} = 0 \mid (x_1, x_2, x_3) = (0,0,1)\right] \\
&= 1 - \Pr\left[b_1^{(i-1)} = 0 \vee b_2^{(i)} = 1 \mid (x_1, x_2, x_3) = (0,0,1)\right] \\
&\geq 1 - \Pr\left[b_1^{(i-1)} = 0 \mid (x_1, x_2, x_3) = (0,0,1)\right] - \Pr\left[b_2^{(i)} = 1 \mid (x_1, x_2, x_3) = (0,0,1)\right] \\
&\geq 1 - \Pr\left[b_1^{(i-1)} = 0 \mid (x_1, x_2, x_3) = (0,0,1)\right] - \mu\,;
\end{aligned}$$

where the last inequality is due to the assumption of the claim. Combined with (7), this implies:

$$\Pr\left[b_1^{(i-1)} = 0 \mid (x_1, x_2, x_3) = (0,0,1)\right] \geq 1 - 5\mu.$$

Applying an analogous argument starting with the third summand in (6) gives

$$\Pr\left[b_3^{(i-1)} = 1 \mid (x_1, x_2, x_3) = (0,1,1)\right] \geq 1 - 5\mu.$$

Repeating the entire argument, but modifying the adversary to consider all possible pairs of corrupted parties and all possible settings of their inputs, completes the proof of the claim. ■

Theorem 6 *Any protocol* Π *that securely computes* maj *with complete fairness (assuming one exists*[2] *at all) requires* $\omega(\log k)$ *rounds.*

Proof: Assume to the contrary that there exists a protocol Π that securely computes maj with complete fairness using $r = \mathcal{O}(\log k)$ rounds. Let $\mu(k) = \frac{1}{4 \cdot 5^{r(k)}}$, and note that μ is noticeable. By the assumed security of Π, the conditions of Claim 3 hold for k large enough; Equation (3), in particular, holds for $i = r(k)$. Fixing this k and applying the claim iteratively $r(k)$ times, we conclude that P_{j-1} and P_{j+1} can correctly compute the value of the function, on all inputs, with probability at least 3/4 *without interacting with* P_j *at all.* This is clearly impossible. ■

3 Fair Computation of Majority for Three Players

In this section we describe a completely-fair protocol for computing maj for the case of $n = 3$ parties. The high-level structure of our protocol is as follows: the protocol consists of two phases. In the first phase, the parties run a secure-with-abort protocol to generate (authenticated) shares of certain values; in the second phase some of these shares are exchanged, round-by-round, for a total of m iterations. A more detailed description of the protocol follows.

In the first phase of the protocol the parties run a protocol π implementing a functionality ShareGen that computes certain values and then distributes authenticated 3-out-of-3 shares of

[2] In the following section we show that such a protocol does, indeed, exist.

ShareGen

Inputs: Let the inputs to ShareGen be $x_1, x_2, x_3 \in \{0,1\}$. (If one of the received inputs is not in the correct domain, then a default value of 1 is used for that player.) The security parameter is k.

Computation:

1. Define values $b_1^{(1)}, \ldots, b_1^{(m)}$, $b_2^{(1)}, \ldots, b_2^{(m)}$ and $b_3^{(1)}, \ldots, b_3^{(m)}$ in the following way:
   - Choose $i^* \geq 1$ according to a geometric distribution with parameter $\alpha = 1/5$ (see text).
   - For $i = 0$ to $i^* - 1$ and $j \in \{1,2,3\}$ do:
     - Choose $\hat{x}_j \leftarrow \{0,1\}$ at random.
     - Set $b_j^{(i)} = \mathsf{maj}(x_{j-1}, \hat{x}_j, x_{j+1})$.
   - For $i = i^*$ to m and $j \in \{1,2,3\}$, set $b_j^{(i)} = \mathsf{maj}(x_1, x_2, x_3)$.
2. For $0 \leq i \leq m$ and $j \in \{1,2,3\}$, choose $b_{j|1}^{(i)}$, $b_{j|2}^{(i)}$ and $b_{j|3}^{(i)}$ as random three-way shares of $b_j^{(i)}$. (E.g., $b_{j|1}^{(i)}$ and $b_{j|2}^{(i)}$ are random and $b_{j|3}^{(i)} = b_{j|1}^{(i)} \oplus b_{j|2}^{(i)} \oplus b_j^{(i)}$.)
3. Let $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$. For $0 \leq i \leq m$, and $j, j' \in \{1,2,3\}$, let $\sigma_{j|j'}^{(i)} = \mathsf{Sign}_{sk}(i \| j \| j' \| b_{j|j'}^{(i)})$.

Output:

1. Send to each P_j the public key pk and the values $\left\{(b_{1|j}^{(i)}, \sigma_{1|j}^{(i)}), (b_{2|j}^{(i)}, \sigma_{2|j}^{(i)}), (b_{3|j}^{(i)}, \sigma_{3|j}^{(i)})\right\}_{i=0}^{m}$. Additionally, for each $j \in \{1,2,3\}$ parties P_{j-1} and P_{j+1} receive the value $b_{j|j}^{(0)}$.

Figure 1: Functionality ShareGen.

these values to the parties. (See Figure 1.) Three sets of values $\{b_1^{(i)}\}_{i=0}^m, \{b_2^{(i)}\}_{i=0}^m$, and $\{b_3^{(i)}\}_{i=0}^m$ are computed; looking ahead, $b_j^{(i)}$ denotes the value that parties P_{j-1} and P_{j+1} are supposed to output in case party P_j aborts after iteration i of the second phase; see below. The values $b_j^{(i)}$ are computed probabilistically, in the same manner as in [7]. That is, a round i^* is first chosen according to a geometric distribution with parameter $\alpha = 1/5$.[3] (We will set m so that $i^* \leq m$ with all but negligible probability.) Then, for $i < i^*$ the value of $b_j^{(i)}$ is computed using the true inputs of P_{j-1} and P_{j+1} but a random input for P_j; for $i \geq i^*$ the value $b_j^{(i)}$ is set equal to the correct output (i.e., it is computed using the true inputs of all parties). Note that even an adversary who knows all the parties' inputs and learns, sequentially, the values (say) $b_1^{(1)}, b_1^{(2)}, \ldots$ cannot determine definitively when round i^* occurs.

We choose the protocol π computing ShareGen to be secure-with-designated-abort [5] for P_1. Roughly speaking, this means privacy and correctness are ensured no matter what, and output delivery and (complete) fairness are guaranteed unless P_1 is corrupted; see Appendix A.3.

The second phase of the protocol proceeds in a sequence of $m = \omega(\log n)$ iterations. (See Figure 2.) In each iteration i, each party P_j broadcasts its share of $b_j^{(i)}$. (We stress that we allow rushing, and do not assume synchronous broadcast.) Observe that, after this is done, parties P_{j-1} and P_{j+1} *jointly* have enough information to reconstruct $b_j^{(i)}$, but neither party has any information about $b_j^{(i)}$ on its own. If all parties behave honestly until the end of the protocol, then in the final

[3] This is the distribution on $\mathbb{N} = \{1, 2, \ldots\}$ given by flipping a biased coin (that is heads with probability α) until the first head appears.

Protocol 1

Inputs: Party P_i has input $x_i \in \{0,1\}$. The security parameter is k.

The protocol:

1. **Preliminary phase:**
   (a) Parties P_1, P_2 and P_3 run a protocol π for computing $\mathsf{ShareGen}$. Each player uses their respective inputs, x_1, x_2 and x_3, and security parameter k.
   (b) If P_2 and P_3 receive $\perp$ from this execution, then P_2 and P_3 run a two-party protocol π_{OR} to compute the logical-or of their inputs.
   Otherwise, continue to the next stage.

   In what follows, parties always verify signatures; invalid signatures are treated as an abort.

2. **For $i = 1, \ldots, m-1$ do:**
   Broadcast shares:
   (a) Each P_j broadcasts $(b_{j|j}^{(i)}, \sigma_{j|j}^{(i)})$.
   (b) If (only) P_j aborts:
      i. P_{j-1} and P_{j+1} broadcast $(b_{j|j-1}^{(i-1)}, \sigma_{j|j-1}^{(i-1)})$ and $(b_{j|j+1}^{(i-1)}, \sigma_{j|j+1}^{(i-1)})$, respectively.
      ii. If one of P_{j-1}, P_{j+1} aborts in the previous step, the remaining player outputs its own input value. Otherwise, P_{j-1} and P_{j+1} both output $b_j^{(i-1)} = b_{j|1}^{(i-1)} \oplus b_{j|2}^{(i-1)} \oplus b_{j|3}^{(i-1)}$. (Recall that if $i = 1$, parties P_{j-1} and P_{j+1} received $b_{j|j}^{(0)}$ as output from π.)
   (c) If two parties abort, the remaining player outputs its own input value.

3. **In round $i = m$ do:**
   (a) Each P_j broadcasts $b_{1|j}^{(m)}, \sigma_{1|j}^{(m)}$.
   (b) If no one aborts, then all players output $b_1^{(m)} = b_{1|1}^{(m)} \oplus b_{1|2}^{(m)} \oplus b_{1|3}^{(m)}$. If (only) P_j aborts, then P_{j-1} and P_{j+1} proceed as in step 2b. If two players abort, the remaining player outputs its own input value as in step 2c.

Figure 2: A protocol for computing majority.

iteration all parties reconstruct $b_1^{(m)}$ and output this value.

If a single party P_j aborts in some iteration i, then the remaining players P_{j-1} and P_{j+1} jointly reconstruct the value $b_j^{(i-1)}$ and output this value. (Recall that these two parties jointly have enough information to do this.) If two parties abort in some iteration i (whether at the same time, or one after the other) then the remaining party simply outputs its own input.

We refer to Figures 1 and 2 for the formal specification of the protocol. We now prove that this protocol securely computes maj with complete fairness.

Theorem 7 *Assume that* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is a secure signature scheme, that* π *securely computes* $\mathsf{ShareGen}$ *with designated abort, and that* π_{OR} *securely computes* OR *with complete fairness.*[4] *Then the protocol in Figure 2 securely computes* maj *with complete fairness.*

Proof: Let Π denote the protocol of Figure 2. Observe that Π yields the correct output with all but negligible probability when all players are honest. This is because, with all but negligible probability, $i^* \leq m$, and then $b_j^{(m)} = \mathsf{maj}(x_1, x_2, x_3)$. We thus focus on security of Π.

[4]It is shown in [7] that such a protocol exists under standard assumptions.

We note that when no parties are corrupt, the proof of security is straightforward, since we assume the existence of a private broadcast channel. We therefore consider separately the cases when a single party is corrupted and when two parties are corrupted. Since the entire protocol is symmetric except for the fact that P_1 may choose to abort π, without loss of generality we may analyze the case when the adversary corrupts P_1 and the case when the adversary corrupts $\{P_1, P_2\}$. In each case, we prove security of Π in a hybrid world where there is an ideal functionality computing ShareGen (with abort) as well as an ideal functionality computing OR (with complete fairness). Applying the composition theorem of [2] then gives the desired result. The case where only P_1 is corrupted is much simpler, and is therefore handled in Section B.1.

Claim 8 *For every non-uniform, poly-time adversary $\mathcal{A}$ corrupting P_1 and P_2 and running Π in a hybrid model with access to ideal functionalities computing* ShareGen *(with abort) and* OR *(with completes fairness), there exists a non-uniform, poly-time adversary $\mathcal{S}$ corrupting P_1 and P_2 and running in the ideal world with access to an ideal functionality computing* maj *(with complete fairness), such that*

$$\left\{\text{IDEAL}_{\mathsf{maj},\mathcal{S}}(x_1,x_2,x_3,k)\right\}_{x_i\in\{0,1\},k\in\mathbb{N}} \stackrel{\text{s}}{\equiv} \left\{\text{HYBRID}^{\mathsf{ShareGen},\mathsf{OR}}_{\Pi,\mathcal{A}}(x_1,x_2,x_3,k)\right\}_{x_i\in\{0,1\},k\in\mathbb{N}}.$$

Proof: This case is significantly more complex than the case when only a single party is corrupted, since here $\mathcal{A}$ learns $b_3^{(i)}$ in each iteration i of the second phase. As in [7], we must deal with the fact that $\mathcal{A}$ might abort exactly in iteration i^*, after learning the correct output but before P_3 has enough information to compute the correct output.

We now describe a simulator $\mathcal{S}$ who corrupts P_1 and P_2 and runs $\mathcal{A}$ as a black-box. For ease of exposition in what follows, we sometimes refer to the actions of P_1 and P_2 when more formally we mean the action of $\mathcal{A}$ on behalf of those parties.

1. $\mathcal{S}$ invokes $\mathcal{A}$ on the inputs x_1 and x_2, the auxiliary input z, and the security parameter k.
2. $\mathcal{S}$ receives x_1' and x_2' from P_1 and P_2, respectively, as input to ShareGen. If $x_1' \notin \{0,1\}$ (resp., $x_2' \notin \{0,1\}$), then $\mathcal{S}$ sets $x_1' = 1$ (resp., $x_2' = 1$).
3. $\mathcal{S}$ computes $(sk, pk) \leftarrow \mathsf{Gen}(1^k)$, and then generates shares as follows:
   (a) Choose $\left\{b_{1|1}^{(i)}, b_{2|1}^{(i)}, b_{3|1}^{(i)}, b_{1|2}^{(i)}, b_{2|2}^{(i)}, b_{3|2}^{(i)}\right\}_{i=0}^{m}$ uniformly at random.
   (b) Choose $\hat{x}_3 \leftarrow \{0,1\}$ and set $b_3^{(0)} = \mathsf{maj}(x_1', x_2', \hat{x}_3)$. Set $b_{3|3}^{(0)} = b_3^{(0)} \oplus b_{3|1}^{(0)} \oplus b_{3|2}^{(0)}$.

   $\mathcal{S}$ then hands $\mathcal{A}$ the public key pk, the values $\left\{b_{1|1}^{(i)}, b_{2|1}^{(i)}, b_{3|1}^{(i)}, b_{1|2}^{(i)}, b_{2|2}^{(i)}, b_{3|2}^{(i)}\right\}_{i=0}^{m}$ (along with their appropriate signatures), and the value $b_{3|3}^{(0)}$ as the outputs of P_1 and P_2 from ShareGen.
4. If P_1 aborts execution of ShareGen, then $\mathcal{S}$ extracts x_2'' from P_2 as its input to OR. It then sends $(1, x_2'')$ to the trusted party computing maj, outputs whatever $\mathcal{A}$ outputs, and halts.
5. Otherwise, if P_1 does not abort, then $\mathcal{S}$ picks a value i^* according to a geometric distribution with parameter $\alpha = \frac{1}{5}$.

   In what follows, for ease of description, we will use x_1 and x_2 in place of x_1' and x_2', keeping in mind that that $\mathcal{A}$ could of course have used substituted inputs. We also ignore the presence

of signatures from now on, and leave the following implicit in what follows: (1) $\mathcal{S}$ always computes an appropriate signature when sending any value to $\mathcal{A}$; (2) $\mathcal{S}$ treats an incorrect signature as an abort; and (3) if $\mathcal{S}$ ever receives a valid signature on a previously unsigned message (i.e., a *forgery*), then $\mathcal{S}$ outputs fail and halts.

Also, from here on we will say that $\mathcal{S}$ *sends* b *to* $\mathcal{A}$ *in round* i if $\mathcal{S}$ sends a value $b_{3|3}^{(i)}$ such that $b_{3|3}^{(i)} \oplus b_{3|1}^{(i)} \oplus b_{3|2}^{(i)} = b_3^{(i)} = b$.

6. For round $i = 1, \ldots, i^* - 1$, the simulator $\mathcal{S}$ computes and then sends $b_3^{(i)}$ as follows:
   (a) Select $\hat{x}_3 \leftarrow \{0,1\}$ at random.
   (b) $b_3^{(i)} = \mathsf{maj}(x_1, x_2, \hat{x}_3)$.
7. If P_1 aborts in round $i < i^*$, then $\mathcal{S}$ sets $\hat{x}_2 = x_2$ and assigns a value to $\hat{x}_1$ according to the following rules that depend on the values of (x_1, x_2) and on the value of $b_3^{(i)}$:
   (a) If $x_1 = x_2$, then $\mathcal{S}$ sets $\hat{x}_1 = x_1$ with probability $\frac{3}{8}$ (and sets $\hat{x}_1 = \bar{x}_1$ otherwise).
   (b) If $x_1 \neq x_2$ and $b_3^{(i)} = x_1$, then $\mathcal{S}$ sets $\hat{x}_1 = x_1$ with probability $\frac{1}{4}$ (and sets $\hat{x}_1 = \bar{x}_1$ otherwise).
   (c) If $x_1 \neq x_2$ and $b_3^{(i)} = x_2$, then $\mathcal{S}$ sets $\hat{x}_1 = x_1$ with probability $\frac{1}{2}$ (and sets $\hat{x}_1 = \bar{x}_1$ otherwise).

   $\mathcal{S}$ then finishes the simulation as follows:
   (a) If $\hat{x}_1 \neq \hat{x}_2$, then $\mathcal{S}$ submits $(\hat{x}_1, \hat{x}_2)$ to the trusted party computing maj. Denote the output it receives from the trusted party by b_{out}. Then $\mathcal{S}$ sets $b_1^{(i-1)} = b_{\mathsf{out}}$, computes $b_{1|3}^{(i-1)} = b_1^{(i-1)} \oplus b_{1|1}^{(i-1)} \oplus b_{1|2}^{(i-1)}$, sends $b_{1|3}^{(i-1)}$ to P_2 (on behalf of P_3), outputs whatever $\mathcal{A}$ outputs, and halts.
   (b) If $\hat{x}_1 = \hat{x}_2$, then $\mathcal{S}$ sets $b_1^{(i-1)} = \hat{x}_1 = \hat{x}_2$, computes $b_{1|3}^{(i-1)} = b_1^{(i-1)} \oplus b_{1|1}^{(i-1)} \oplus b_{1|2}^{(i-1)}$, and sends $b_{1|3}^{(i-1)}$ to P_2 (on behalf of P_3). (We stress that this is done *before* sending anything to the trusted party computing maj.) If P_2 aborts, then $\mathcal{S}$ sends $(0,1)$ to the trusted party computing maj. Otherwise, it sends $(\hat{x}_1, \hat{x}_2)$ to the trusted party computing maj. In both cases it outputs whatever $\mathcal{A}$ outputs, and then halts.

   If P_2 aborts in round $i < i^*$, then $\mathcal{S}$ acts analogously but swapping the roles of P_1 and P_2 as well as x_1 and x_2.

   If both parties abort in round $i < i^*$ (at the same time), then $\mathcal{S}$ sends $(0,1)$ to the trusted party computing maj, outputs whatever $\mathcal{A}$ outputs, and halts.
8. In round i^*:
   (a) If $x_1 \neq x_2$, then $\mathcal{S}$ submits (x_1, x_2) to the trusted party. Let $b_{\mathsf{out}} = \mathsf{maj}(x_1, x_2, x_3)$ denote the output.
   (b) If $x_1 = x_2$, then $\mathcal{S}$ simply sets $b_{\mathsf{out}} = x_1 = x_2$ *without querying the trusted party* and continues. (Note that in this case, $b_{\mathsf{out}} = \mathsf{maj}(x_1, x_2, x_3)$ even though $\mathcal{S}$ did not query the trusted party.)

9. In rounds $i^*, \ldots, m-1$, the simulator $\mathcal{S}$ sends b_{out} to $\mathcal{A}$.

   If $\mathcal{A}$ aborts P_1 and P_2 simultaneously, then $\mathcal{S}$ submits $(1,0)$ to the trusted party (if he hasn't already done so in step 8a), outputs whatever $\mathcal{A}$ outputs, and halts.

   If $\mathcal{A}$ aborts P_1 (only), then $\mathcal{S}$ sets $b_1^{(i-1)} = b_{\mathsf{out}}$, computes $b_{1|3}^{(i-1)} = b_1^{(i-1)} \oplus b_{1|1}^{(i-1)} \oplus b_{1|2}^{(i-1)}$, and sends $b_{1|3}^{(i-1)}$ to P_2 (on behalf of P_3). Then:

   Case 1: $x_1 \neq x_2$. Here $\mathcal{S}$ has already sent (x_1, x_2) to the trusted party. So $\mathcal{S}$ simply outputs whatever $\mathcal{A}$ outputs and ends the simulation.

   Case 2: $x_1 = x_2$. If P_2 does not abort, then $\mathcal{S}$ sends (x_1, x_2) to the trusted party. If P_2 aborts, then $\mathcal{S}$ sends $(0,1)$ to the trusted party. In both cases $\mathcal{S}$ then outputs whatever $\mathcal{A}$ outputs and halts.

   If $\mathcal{A}$ aborts P_2 (only), then $\mathcal{S}$ acts as above but swapping the roles of P_1, P_2 and x_1, x_2. If $\mathcal{A}$ does not abort anyone through round m, then $\mathcal{S}$ sends (x_1, x_2) to the trusted party (if he hasn't already done so), outputs what $\mathcal{A}$ outputs, and halts.

Due to space limitations, the analysis of $\mathcal{S}$ is given in Appendix B.2. ■

■

4 Completely-Fair Computation of Boolean OR

The protocol in the previous section enables completely-fair computation of 3-party majority; unfortunately, we were not able to extend the approach to the case of $n > 3$ parties. In this section, we demonstrate feasibility of completely-fair computation of a non-trivial function for an arbitrary number of parties n, any $t < n$ of whom are corrupted. Specifically, we show how to compute boolean OR with complete fairness.

The basic idea behind our protocol is to have the parties repeatedly try to compute OR *on committed inputs* using a protocol that is secure-with-designated-abort where only the lowest-indexed party can force an abort. (See Appendix A.3.) The key observation is that, in case of an abort, the dishonest players only "learn something" about the inputs of the honest players in case all the malicious parties use input 0. (If any of the malicious players holds input 1, then the output is always 1 regardless of the inputs of the honest parties.) So, if the lowest-indexed party is corrupt and aborts the computation of the committed OR, then the remaining parties simply recompute the committed OR using '0' as the effective input for any parties who have already been eliminated. The parties repeatedly proceed in this fashion, eliminating dishonest parties at each iteration. Eventually, when the lowest-indexed player is honest, the process terminates and all honest players receive (correct) output.

The actual protocol follows the above intuition, but is a bit more involved. A formal description of the protocol is given in Figure 3, and the "committed OR" functionality is defined in Figure 4.

Theorem 9 *Assume* Com *is a computationally-hiding, statistically-binding commitment scheme, and that $\pi_{\mathcal{P}}$ securely computes* $\mathsf{CommittedOR}_{\mathcal{P}}$ *(with abort). Then the protocol of Figure 3 computes* OR *with complete fairness.*

Protocol 2

Inputs: Each party P_i holds input $x_i \in \{0,1\}$, and the security parameter is k.

Computation:

1. Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the set of all players.
2. Each player P_i chooses random coins r_i and broadcasts $c_i = \mathsf{Com}(1^k, x_i, r_i)$, where Com denotes a computationally-hiding, statistically-binding commitment scheme. If any party P_i does not broadcast anything (or otherwise broadcasts an invalid value), then all honest players output 1. Otherwise, let $\vec{c} = (c_1, \ldots, c_n)$.
3. All players $P_i \in \mathcal{P}$ run a protocol $\pi_{\mathcal{P}}$ for computing $\mathsf{CommittedOR}_{\mathcal{P}}$, with party P_i using $(x_i, r_i, \vec{c}_{\mathcal{P}})$ as its input where $\vec{c}_{\mathcal{P}} \stackrel{\text{def}}{=} (c_i)_{i:P_i \in \mathcal{P}}$.
4. If players receive $\perp$ from the execution of $\mathsf{CommittedOR}_{\mathcal{P}}$, they set $\mathcal{P} = \mathcal{P} \setminus \{P^*\}$, where $P^* \in \mathcal{P}$ is the lowest-indexed player in $\mathcal{P}$, and return to step 3.
5. If players receive a set $\mathcal{D} \subset \mathcal{P}$ from the execution of $\mathsf{CommittedOR}_{\mathcal{P}}$, they set $\mathcal{P} = \mathcal{P} \setminus \mathcal{D}$ and return to step 3.
6. If players receive a binary output from the execution of $\mathsf{CommittedOR}_{\mathcal{P}}$, they output this value and end the protocol.

Figure 3: A protocol computing OR for n players.

$\mathsf{CommittedOR}_{\mathcal{P}}$

Inputs: The functionality is run by parties in $\mathcal{P}$. Let the input of player $P_i \in \mathcal{P}$ be $(x_i, r_i, \vec{c}^{\,i})$ where $\vec{c}^{\,i} = (c^i_j)_{j:P_j \in \mathcal{P}}$. The security parameter is k.

For each party $P_i \in \mathcal{P}$, determine its output as follows:

1. Say P_j *disagrees with* P_i if either (1) $\vec{c}^{\,j} \neq \vec{c}^{\,i}$ or (2) $\mathsf{Com}(1^k, x_j, r_j) \neq c^i_j$. (Note that disagreement is not a symmetric relation.)
2. Let $\mathcal{D}_i$ be the set of parties who disagree with P_i.
3. If there exist any parties that disagree with each other, return $\mathcal{D}_i$ as output to P_i. Otherwise, return $\bigvee_{j:P_j \in \mathcal{P}} x_j$ to all parties.

Figure 4: Functionality $\mathsf{CommittedOR}_{\mathcal{P}}$, parameterized by a set $\mathcal{P}$

Proof: Let Π denote the protocol of Figure 3. For simplicity we assume Com is perfectly binding, though statistical binding suffices. For any non-uniform, polynomial time adversary $\mathcal{A}$ in the hybrid world, we demonstrate a non-uniform polynomial-time adversary $\mathcal{S}$ corrupting the same parties as $\mathcal{A}$ and running in the ideal world with access to an ideal functionality computing OR (with complete fairness), such that

$$\left\{\text{IDEAL}_{\mathsf{OR},\mathcal{S}}(x_1, \ldots, x_n, k)\right\}_{x_i \in \{0,1\}, k \in \mathbb{N}} \stackrel{\text{c}}{\equiv} \left\{\text{HYBRID}^{\mathsf{CommittedOR}_{\mathcal{P}}}_{\Pi,\mathcal{A}}(x_1, \ldots, x_n, k)\right\}_{x_i \in \{0,1\}, k \in \mathbb{N}}.$$

Applying the composition theorem of [2] then proves the theorem.

We note that when no players are corrupt, the proof of security is easy, due to the assumed existence of a private broadcast channel. We now describe the execution of $\mathcal{S}$:

1. Let $\mathcal{C} \neq \emptyset$ be the corrupted players, and let $\mathcal{H} = \{P_1, \ldots P_n\} \setminus \mathcal{C}$ denote the honest players. Initialize $\mathcal{I} = \mathcal{C}$. Looking ahead, $\mathcal{I}$ denotes the set of corrupted parties who have not yet been eliminated from the protocol.

2. $\mathcal{S}$ invokes $\mathcal{A}$ on the inputs $\{x_i\}_{i:P_i \in \mathcal{C}}$, the auxiliary input z, and the security parameter k.

3. For $P_i \in \mathcal{H}$, the simulator $\mathcal{S}$ gives to $\mathcal{A}$ a commitment $c_i = \mathsf{Com}(1^k, x_i, r_i)$ to $x_i = 0$ using randomness r_i. $\mathcal{S}$ then records the commitment c_i that is broadcast by $\mathcal{A}$ on behalf of each party $P_i \in \mathcal{C}$. If any corrupted player fails to broadcast a value c_i, then $\mathcal{S}$ submits 1's to the trusted party on behalf of all corrupted parties, outputs whatever $\mathcal{A}$ outputs, and halts.

4. If $\mathcal{I} = \emptyset$, $\mathcal{S}$ submits (on behalf of all the corrupted parties) 0's to the trusted party computing OR (unless it has already done so). It then outputs whatever $\mathcal{A}$ outputs, and halts. If $\mathcal{I} \neq \emptyset$, continue to the next step.

5. $\mathcal{S}$ sets $\mathcal{P} = \mathcal{H} \cup \mathcal{I}$ and obtains inputs $\{(r_i, x_i, \vec{c}^{\,i})\}_{i:P_i \in \mathcal{I}}$ for the computation of $\mathsf{CommittedOR}_{\mathcal{P}}$. For each $P_i \in \mathcal{P}$, the simulator $\mathcal{S}$ computes the list of players $\mathcal{D}_i$ that disagree with P_i (as in Figure 4), using as the inputs of the honest parties the commitments defined in Step 3, and assuming that honest parties provide correct decommitments. Observe that if $P_i, P_j \in \mathcal{H}$ then $\mathcal{D}_i = \mathcal{D}_j \subseteq \mathcal{I}$. Let $\mathcal{D}_{\mathcal{H}} \subseteq \mathcal{I}$ be the set of parties that disagree with the honest parties.

   Let P^* be the lowest-indexed player in $\mathcal{P}$. If no parties disagree with each other, go to step 6. Otherwise:

   (a) If $P^* \in \mathcal{I}$, then $\mathcal{A}$ is given $\{\mathcal{D}_i\}_{i:P_i \in \mathcal{I}}$. If P^* aborts, then $\mathcal{S}$ sets $\mathcal{I} = \mathcal{I} \setminus \{P^*\}$ and goes to step 4. If P^* does not abort, then $\mathcal{S}$ sets $\mathcal{I} = \mathcal{I} \setminus \mathcal{D}_{\mathcal{H}}$ and goes to step 4.

   (b) If $P^* \notin \mathcal{I}$, then $\mathcal{A}$ is given $\{\mathcal{D}_i\}_{i:P_i \in \mathcal{I}}$. Then $\mathcal{S}$ sets $\mathcal{I} = \mathcal{I} \setminus \mathcal{D}_{\mathcal{H}}$ and goes to step 4.

6. $\mathcal{S}$ computes the value $b = \bigvee_{P_i \in \mathcal{I}} x_i$.

   (a) If $b = 0$, and $\mathcal{S}$ has not yet queried the trusted party computing OR, then $\mathcal{S}$ submits 0's (on behalf of all the corrupted parties) to the trusted party and stores the output of the trusted party as b_{out}. $\mathcal{S}$ gives b_{out} to $\mathcal{A}$ (either as just received from the trusted party, or as stored in a previous execution of this step).

   (b) If $b = 1$, then $\mathcal{S}$ gives the value 1 to $\mathcal{A}$ *without* querying the trusted party.

   $\mathcal{S}$ now continues as follows:

   (a) If $P^* \in \mathcal{I}$ and P^* aborts, then $\mathcal{S}$ sets $\mathcal{I} = \mathcal{I} \setminus \{P^*\}$ and goes to step 4.

   (b) If $P^* \notin \mathcal{I}$, or if P^* does not abort, then $\mathcal{S}$ submits 1's to the trusted party if it has not yet submitted 0's. It outputs whatever $\mathcal{A}$ outputs, and halts.

We refer the reader to B.3 for an analysis of the above simulation. ■

References

[1] D. Beaver. Foundations of secure interactive computing. In *Advances in Cryptology — Crypto '91*, volume 576 of *Lecture Notes in Computer Science*, pages 377–391. Springer, 1992.

[2] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[3] B. Chor and E. Kushilevitz. A zero-one law for boolean privacy. *SIAM Journal of Discrete Math*, 4(1):36–47, 1991.

[4] R. Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proc. 18th Annual ACM Symposium on Theory of Computing (STOC)*, pages 364–369, 1986.

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

[6] S. Goldwasser and L. Levin. Fair computation and general functions in the presence of immoral majority. In *Advances in Cryptology — Crypto '90*, volume 537 of *Lecture Notes in Computer Science*. Springer, 1991.

[7] D. Gordon, C. Hazay, J. Katz, and Y. Lindell. Complete fairness in secure two-party computation. In *Proc. 40th Annual ACM Symposium on Theory of Computing (STOC)*, 2008.

[8] Y. Ishai, E. Kushilevitz, Y. Lindell, and E. Petrank. On combining privacy with guaranteed output delivery in secure multiparty computation. In *Advances in Cryptology — Crypto 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 483–500. Springer, 2006.

[9] J. Kilian, E. Kushilevitz, S. Micali, and R. Ostrovsky. Reducibility and completeness in private computations. *SIAM J. Computing*, 29(4):1189–1208, 2000.

[10] S. Micali and P. Rogaway. Secure computation. In *Advances in Cryptology — Crypto '91*, volume 576 of *Lecture Notes in Computer Science*, pages 392–404. Springer, 1992.

A Standard Definitions

A.1 Preliminaries

We denote the security parameter by k. A function $\mu(\cdot)$ is negligible if for every positive polynomial $p(\cdot)$ and all sufficiently large k it holds that $\mu(k) < 1/p(k)$. A distribution ensemble $X = \{X(a,k)\}_{a\in\mathcal{D}_k, k\in\mathbb{N}}$ is an infinite sequence of random variables indexed by $a \in \mathcal{D}_k$ and $k \in \mathbb{N}$, where $\mathcal{D}_k$ is a set that may depend on k. (Looking ahead, $\mathcal{D}_k$ will denote the domain of the parties' inputs.) Two distribution ensembles $X = \{X(a,k)\}_{a\in\mathcal{D}_k, k\in\mathbb{N}}$ and $Y = \{Y(a,k)\}_{a\in\mathcal{D}_k, k\in\mathbb{N}}$ are computationally indistinguishable, denoted $X \stackrel{\text{c}}{\equiv} Y$, if for every non-uniform polynomial-time algorithm D there exists a negligible function $\mu(\cdot)$ such that for every k and every $a \in \mathcal{D}_k$

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

The statistical difference between two distributions $X(a,k)$ and $Y(a,k)$ is defined as

$$\mathsf{SD}\left(X(a,k),\, Y(a,k)\right) = \frac{1}{2} \cdot \sum_s \left|\Pr[X(a,k) = s] - \Pr[Y(a,k) = s]\right|,$$

where the sum ranges over s in the support of either $X(a,k)$ or $Y(a,k)$. Two distribution ensembles $X = \{X(a,k)\}_{a\in\mathcal{D}_k, k\in\mathbb{N}}$ and $Y = \{Y(a,k)\}_{a\in\mathcal{D}_k, k\in\mathbb{N}}$ are *statistically close*, denoted $X \stackrel{\text{s}}{\equiv} Y$, if there is a negligible function $\mu(\cdot)$ such that for every k and every $a \in \mathcal{D}_k$, it holds that $\mathsf{SD}\left(X(a,k), Y(a,k)\right) \leq \mu(k)$.

A.2 Secure Multi-Party Computation with Complete Fairness

Multi-party computation. A multi-party protocol for parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ and computing a (possibly randomized) functionality $f = (f_1, \ldots, f_n)$, is a protocol satisfying the following functional requirement: if each P_i begins by holding 1^k and input x, and all parties run the protocol honestly, then the joint distribution of the outputs of the parties is statistically close to $(f_1(x_1, \ldots, x_n), \ldots, f_n(x_1, \ldots, x_n))$.

In what follows, we define what we mean by a *secure* protocol. Our definition follows the standard definition of [5] (based on [6, 10, 1, 2]), *except that we require complete fairness* even though we do not have honest majority. We consider active adversaries, who may deviate from the protocol in an arbitrary manner, and static corruptions.

Security of protocols (informal). The security of a protocol is analyzed by comparing what an adversary can do in a real protocol execution to what it can do in an ideal scenario that is secure by definition. This is formalized by considering an *ideal* computation involving an incorruptible *trusted party* to whom the parties send their inputs. The trusted party computes the functionality on the inputs and returns to each party its respective output. Loosely speaking, a protocol is secure if any adversary interacting in the real protocol (where no trusted party exists) can do no more harm than if it was involved in the above-described ideal computation.

Execution in the ideal model. The parties are $\mathcal{P} = \{P_1, \ldots, P_n\}$, and there is an adversary $\mathcal{S}$ who has corrupted some subset $\mathcal{I} \subset \mathcal{P}$ of them. An ideal execution for the computation of f proceeds as follows:

Inputs: Each party P_i holds its input x_i and the security parameter k. The adversary $\mathcal{S}$ also receives an auxiliary input z.

Send inputs to trusted party: The honest parties send their inputs to the trusted party. $\mathcal{S}$ may substitute any values it likes on behalf of the corrupted parties. We denote by x'_i the value sent to the trusted party on behalf of P_i.

Trusted party sends outputs: If any x'_i is not in the correct domain, the trusted party sets $x'_i = \hat{x}_i$ for some default value $\hat{x}_i$. Then, the trusted party chooses r uniformly at random and sends $f_i(x'_1, \ldots, x'_n; r)$ to each P_i.

Outputs: The honest parties output whatever they were sent by the trusted party, the corrupted parties output nothing, and $\mathcal{S}$ outputs an arbitrary function of its view.

We let $\text{IDEAL}_{f,\mathcal{S}(z)}(x_1, \ldots, x_n, k)$ be the random variable consisting of the output of the adversary and the output of the honest parties following an execution in the ideal model as described above.

Execution in the real model. Here a multi-party protocol π is executed by $\mathcal{P}$, and there is no trusted party. In this case, the adversary $\mathcal{A}$ gets the inputs of the corrupted parties (as well as an auxiliary input z) and sends all messages on behalf of these parties, using an arbitrary polynomial-time strategy. The honest parties follow the instructions of π.

Let f be as above and let π be a multi-party protocol computing f. Let $\mathcal{A}$ be a non-uniform probabilistic polynomial-time machine with auxiliary input z. We let $\text{REAL}_{\pi,\mathcal{A}(z)}(x_1, \ldots, x_n, k)$ be the random variable consisting of the view of the adversary and the output of the honest parties, following an execution of π where P_i begins holding its input x_i and the security parameter k.

Security as emulation of an ideal execution. Having defined the ideal and real models, we can now define security of a protocol. Loosely speaking, the definition says that a secure protocol (in the real model) emulates the ideal model (in which a trusted party exists).

Definition 10 *Let f be as above. Protocol π is said to* securely compute f with complete fairness *if for every non-uniform probabilistic polynomial-time adversary $\mathcal{A}$ in the real model, there exists a non-uniform probabilistic polynomial-time adversary $\mathcal{S}$ in the ideal model such that*

$$\left\{\text{IDEAL}_{f,\mathcal{S}(z)}(\vec{x},k)\right\}_{\vec{x}\in(\{0,1\}^*)^n,\, z\in\{0,1\}^*,\, k\in\mathbb{N}} \stackrel{\text{c}}{\equiv} \left\{\text{REAL}_{\pi,\mathcal{A}(z)}(\vec{x},k)\right\}_{\vec{x}\in(\{0,1\}^*)^n,\, z\in\{0,1\}^*,\, k\in\mathbb{N}}.$$

A.3 Secure Multi-Party Computation With Designated Abort

This definition is standard for secure multi-party computation without an honest majority [5]. It allows *early abort* (i.e., the adversary may receive its own outputs even though the honest parties do not), but only if P_1 is corrupted.

We again let $\mathcal{P} = \{P_1, \ldots P_n\}$ denote the parties, and consider an adversary $\mathcal{S}$ who has corrupted a subset $\mathcal{I} \subset \mathcal{P}$ of them. The only change from the definition in Section A.2 is with regard to the ideal model for computing f, which is now defined as follows:

Inputs: As previously.

Send inputs to trusted party: As previously.

> If any x'_i is not in the correct domain, the trusted party sets $x'_i = \hat{x}_i$ for some default value $\hat{x}_i$. Then, the trusted party chooses r uniformly at random and sets $z_i = f_i(x'_1, \ldots, x'_n; r)$.

Trusted party sends outputs, P_1 honest: The trusted party sends z_i to each P_i.

Trusted party sends outputs, P_1 corrupt: The trusted party sends $\{z_i\}_{i:P_i\in\mathcal{I}}$ to $\mathcal{S}$. Then $\mathcal{S}$ sends either abort or continue to the trusted party. In the former case the trusted party sends $\perp$ to all honest parties, and in the latter case the trusted party sends z_i to each honest P_i.

> Note that an adversary corrupting P_1 can always abort the protocol, even if $|\mathcal{I}| < n/2$.

Outputs: As previously.

We let $\text{IDEAL}^{\mathsf{abort}}_{f,\mathcal{S}(z)}(\vec{x},k)$ be the random variable consisting of the output of the adversary and the output of the honest parties following an execution in the ideal model as described above.

Definition 11 *Let f be a functionality, and let π be a protocol computing f. Protocol π is said to* securely compute f with designated abort *if for every non-uniform probabilistic polynomial-time adversary $\mathcal{A}$ in the real model, there exists a non-uniform probabilistic polynomial-time adversary $\mathcal{S}$ in the ideal model such that*

$$\left\{\text{IDEAL}^{\mathsf{abort}}_{f,\mathcal{S}(z)}(\vec{x},k)\right\}_{\vec{x}\in(\{0,1\}^*)^n,\, z\in\{0,1\}^*,\, k\in\mathbb{N}} \stackrel{\text{c}}{\equiv} \left\{\text{REAL}_{\pi,\mathcal{A}(z)}(\vec{x},k)\right\}_{\vec{x}\in(\{0,1\}^*)^n,\, z\in\{0,1\}^*,\, k\in\mathbb{N}}.$$

B Proofs

B.1 Proof of Security for Majority With a Single Corrupted Party

Claim 12 *For every non-uniform, poly-time adversary $\mathcal{A}$ corrupting P_1 and running Π in a hybrid model with access to ideal functionalities computing* ShareGen *(with abort) and* OR *(with complete fairness), there exists a non-uniform, poly-time adversary $\mathcal{S}$ corrupting P_1 and running in the ideal world with access to an ideal functionality computing* maj *(with complete fairness), such that*

$$\left\{\text{IDEAL}_{\mathsf{maj},\mathcal{S}}(x_1,x_2,x_3,k)\right\}_{x_i\in\{0,1\},k\in\mathbb{N}} \stackrel{\text{s}}{\equiv} \left\{\text{HYBRID}^{\mathsf{ShareGen},\mathsf{OR}}_{\Pi,\mathcal{A}}(x_1,x_2,x_3,k)\right\}_{x_i\in\{0,1\},k\in\mathbb{N}}.$$

Proof:

Fix some polynomial-time adversary $\mathcal{A}$ corrupting P_1. We now describe a simulator $\mathcal{S}$ that also corrupts P_1 and runs $\mathcal{A}$ as a black box.

1. $\mathcal{S}$ invokes $\mathcal{A}$ on the input x_1, the auxiliary input z, and the security parameter k.
2. $\mathcal{S}$ receives input $x_1' \in \{0,1\}$ on behalf of P_1 as input to ShareGen.
3. $\mathcal{S}$ computes $(sk, pk) \leftarrow \mathsf{Gen}(1^k)$, and gives to $\mathcal{A}$ the public key pk and values $b^{(0)}_{2|2}$, $b^{(0)}_{3|3}$, and $\left\{b^{(i)}_{1|1}, b^{(i)}_{2|1}, b^{(i)}_{3|1}\right\}_{i=0}^{m}$ (along with their appropriate signatures) chosen uniformly at random.
4. If $\mathcal{A}$ aborts execution of ShareGen, then $\mathcal{S}$ sends 1 to the trusted party computing maj, outputs whatever $\mathcal{A}$ outputs, and halts. Otherwise, $\mathcal{S}$ picks a value i^* according to a geometric distribution with parameter $\alpha = \frac{1}{5}$.

   For simplicity in what follows, we ignore the presence of signatures and leave the following implicit from now on: (1) $\mathcal{S}$ always computes an appropriate signature when sending any value to $\mathcal{A}$; (2) $\mathcal{S}$ treats an incorrect signature as an abort; and (3) if $\mathcal{S}$ ever receives a valid signature on a previously unsigned message, then $\mathcal{S}$ outputs fail and halts.
5. $\mathcal{S}$ now simulates the rounds of the protocol one-by-one: for $i = 1$ to $m-1$, the simulator chooses random $b^{(i)}_{2|2}$ and $b^{(i)}_{3|3}$ and sends these to $\mathcal{A}$. During this step, an abort by $\mathcal{A}$ (on behalf of P_1) is treated as follows:

   (a) If P_1 aborts in round $i \le i^*$, then $\mathcal{S}$ chooses a random value $\hat{x}_1$ and sends it to the trusted party computing maj.

   (b) If P_1 aborts in round $i > i^*$, then $\mathcal{S}$ submits x_1' to the trusted party computing maj.

   In either case, $\mathcal{S}$ then outputs whatever $\mathcal{A}$ outputs and halts.
6. If P_1 has not yet aborted, $\mathcal{S}$ then simulates the final round of the protocol. $\mathcal{S}$ sends x_1' to the trusted party, receives $b_{\mathsf{out}} = \mathsf{maj}(x_1', x_2, x_3)$, and chooses $b^{(m)}_{1|2}$ and $b^{(m)}_{1|3}$ at random subject to $b^{(m)}_{1|2} \oplus b^{(m)}_{1|3} \oplus b^{(m)}_{1|1} = b_{\mathsf{out}}$. $\mathcal{S}$ then gives these values to $\mathcal{A}$, outputs whatever $\mathcal{A}$ outputs, and halts.

Due to the security of the underlying signature scheme, the probability that $\mathcal{S}$ outputs fail is negligible in k. Note that the view of P_1 is otherwise statistically close in both worlds. Indeed, until round m the view of P_1 is independent of the inputs of the other parties in both the real and ideal worlds. In round m itself, P_1 learns the (correct) output b_{out} in the ideal world and learns this value with all but negligible probability in the real world.

We therefore only have to argue that outputs of the two honest parties in the real and ideal worlds are statistically close. Clearly this is true if P_1 never aborts. As for the case when P_1 aborts at some point during the protocol, we divide our analysis into the following cases:

- If P_1 aborts the execution of $\mathsf{ShareGen}$ in step 4, then $\mathcal{S}$ submits '1' on behalf of P_1 to the trusted party computing maj. Thus, in the ideal world, the outputs of P_2 and P_3 will be $\mathsf{maj}(1, x_2, x_3)$. In the real world, if P_1 aborts computation of $\mathsf{ShareGen}$, the honest parties output $\mathsf{OR}(x_2, x_3)$. Since $\mathsf{maj}(1, x_2, x_3) = \mathsf{OR}(x_2, x_3)$, their outputs are the same.
- If P_1 aborts in round i of the protocol (cf. step 5), then in both the real and ideal worlds the following holds:
  - If $i \leq i^*$, then P_2 and P_3 output $\mathsf{maj}(\hat{x}_1, x_2, x_3)$ where $\hat{x}_1$ is chosen uniformly at random.
  - If $i > i^*$, then P_2 and P_3 output $\mathsf{maj}(x'_1, x_2, x_3)$

  Since i^* is identically distributed in both worlds, the outputs of P_2 and P_3 in this case are identically distributed as well.
- If P_1 aborts in round m (cf. step 6), then in the ideal world the honest parties will output $\mathsf{maj}(x'_1, x_2, x_3)$. In the real world the honest parties output $\mathsf{maj}(x'_1, x_2, x_3)$ as long as $i^* \leq m - 1$, which occurs with all but negligible probability.

This completes the proof. ■

B.2 Completing the Proof of Claim 8

We first note that the probability $\mathcal{S}$ outputs fail is negligible, due to the security of the underling signature scheme. We state the following claim:

Claim 13 *If P_1 and P_2 both abort, then $\mathcal{S}$ always sends $(0, 1)$ or $(1, 0)$ to the trusted party.*

We leave verification to the reader. We must prove that for any set of fixed inputs, the joint distribution over the possible views of $\mathcal{A}$ and the output of P_3 is equal in the ideal and hybrid worlds:

$$(\text{VIEW}_{\mathsf{hyb}}(x_1, x_2, x_3), \text{OUT}_{\mathsf{hyb}}(x_1, x_2, x_3)) \equiv (\text{VIEW}_{\mathsf{ideal}}(x_1, x_2, x_3), \text{OUT}_{\mathsf{ideal}}(x_1, x_2, x_3)) \tag{8}$$

We begin by noting that this is trivially true when no players ever abort. It is also easy to verify that this is true when P_1 aborts during the execution of $\mathsf{ShareGen}$. From here forward, we therefore assume that $\mathcal{A}$ aborts player P_1 at some point after the execution of $\mathsf{ShareGen}$. We consider what happens when $\mathcal{A}$ aborts P_2 as well, but for simplicity we will only analyze the cases where P_1 is aborted first, and when they are aborted at the same time. The analysis when $\mathcal{A}$ aborts only P_2, or when he aborts P_1 sometime after P_2 is symmetric and is not dealt with here. We will break up the view of $\mathcal{A}$ into two parts: the view before P_1 aborts, where a particular instance of this view is

denoted by $\vec{a}_i$, and the single message intended for P_2 that $\mathcal{A}$ receives after P_1 aborts, denoted by $b_1^{(i-1)}$. Letting i denote the round in which P_1 aborts, and b_{out} the value output by P_3, we wish to prove:

$$\Pr\left[(\text{VIEW}_{\mathsf{hyb}}, \text{OUT}_{\mathsf{hyb}}) = (\vec{a}_i, b_1^{(i-1)}, b_{\mathsf{out}})\right] = \Pr\left[(\text{VIEW}_{\mathsf{ideal}}, \text{OUT}_{\mathsf{ideal}}) = (\vec{a}_i, b_1^{(i-1)}, b_{\mathsf{out}})\right]$$

(we drop explicit mention of the inputs to improve readability). Towards proving this, we first prove the following two claims.

Claim 14 *For all fixed inputs and all feasible adversarial views $(\vec{a}_i, b_1^{(i-1)})$,*

$$\Pr\left[(\text{VIEW}_{\mathsf{hyb}}, \text{OUT}_{\mathsf{hyb}}) = (\vec{a}_i, b_1^{(i-1)}, b_{\mathsf{out}}) \bigwedge i > i^*\right] = \Pr\left[(\text{VIEW}_{\mathsf{ideal}}, \text{OUT}_{\mathsf{ideal}}) = (\vec{a}_i, b_1^{(i-1)}, b_{\mathsf{out}}) \bigwedge i > i^*\right]$$

Proof: We denote by $P_2^{\perp}$ the event that P_2 aborts the protocol (either at the same time as P_1, or after P_1 aborts, during the exchange of the shares of $b_1^{(i-1)}$). We also replace the event $\left(\text{VIEW}_{\mathsf{hyb}} = (\vec{a}_i, b_1^{(i-1)}) \bigwedge i > i^*\right)$ by E_{hyb}, and the event $\left(\text{VIEW}_{\mathsf{ideal}} = (\vec{a}_i, b_1^{(i-1)}) \bigwedge i > i^*\right)$ by $\text{E}_{\mathsf{ideal}}$ in order to shorten notation. We have the following:

$$\begin{aligned}
&\Pr\left[(\text{VIEW}_{\mathsf{hyb}}, \text{OUT}_{\mathsf{hyb}}) = (\vec{a}_i, b_1^{(i-1)}, b_{\mathsf{out}}) \bigwedge i > i^*\right] \\
&= \Pr\left[\text{OUT}_{\mathsf{hyb}} = b_{\mathsf{out}} \bigwedge P_2^{\perp} \bigwedge \text{E}_{\mathsf{hyb}}\right] \\
&\quad + \Pr\left[\text{OUT}_{\mathsf{hyb}} = b_{\mathsf{out}} \bigwedge \neg P_2^{\perp} \bigwedge \text{E}_{\mathsf{hyb}}\right] \\
&= \Pr\left[\text{OUT}_{\mathsf{hyb}} = b_{\mathsf{out}} \mid P_2^{\perp} \bigwedge \text{E}_{\mathsf{hyb}}\right] \cdot \Pr\left[P_2^{\perp} \bigwedge \text{E}_{\mathsf{hyb}}\right] \\
&\quad + \Pr\left[\text{OUT}_{\mathsf{hyb}} = b_{\mathsf{out}} \mid \neg P_2^{\perp} \bigwedge \text{E}_{\mathsf{hyb}}\right] \cdot \Pr\left[\neg P_2^{\perp} \bigwedge \text{E}_{\mathsf{hyb}}\right]
\end{aligned}$$

and that the same is true in the ideal world. It follows from the descriptions of the protocol and the simulator that

$$\Pr\left[P_2^{\perp} \bigwedge \text{VIEW}_{\mathsf{hyb}} = (\vec{a}_i, b_1^{(i-1)}) \bigwedge i > i^*\right] = \Pr\left[P_2^{\perp} \bigwedge \text{VIEW}_{\mathsf{ideal}} = (\vec{a}_i, b_1^{(i-1)}) \bigwedge i > i^*\right]$$

and similarly that

$$\Pr\left[\neg P_2^{\perp} \bigwedge \text{VIEW}_{\mathsf{hyb}} = (\vec{a}_i, b_1^{(i-1)}) \bigwedge i > i^*\right] = \Pr\left[\neg P_2^{\perp} \bigwedge \text{VIEW}_{\mathsf{ideal}} = (\vec{a}_i, b_1^{(i-1)}) \bigwedge i > i^*\right].$$

The above two equalities hold because the protocol is designed such that any view $\vec{a}_i$ occurs with the same probability in both worlds. Furthermore, given that $i > i^*$, it holds that $b_1^{(i-1)} = f(x_1, x_2, x_3)$, independent of $\vec{a}_i$. P_2 decides whether to abort based only on these two variables, so the decision is the same in both worlds. We therefore need only to prove that

$$\begin{aligned}
&\Pr\left[\text{OUT}_{\mathsf{hyb}} = b_{\mathsf{out}} \mid P_2^{\perp} \bigwedge \text{VIEW}_{\mathsf{hyb}} = (\vec{a}_i, b_1^{(i-1)}) \bigwedge i > i^*\right] \\
&= \Pr\left[\text{OUT}_{\mathsf{ideal}} = b_{\mathsf{out}} \mid P_2^{\perp} \bigwedge \text{VIEW}_{\mathsf{ideal}} = (\vec{a}_i, b_1^{(i-1)}) \bigwedge i > i^*\right] \qquad (9)
\end{aligned}$$

and that

$$\begin{aligned}
&\Pr\left[\text{OUT}_{\mathsf{hyb}} = b_{\mathsf{out}} \mid \neg P_2^{\perp} \bigwedge \text{VIEW}_{\mathsf{hyb}} = (\vec{a}_i, b_1^{(i-1)}) \bigwedge i > i^*\right] \\
&= \Pr\left[\text{OUT}_{\mathsf{ideal}} = b_{\mathsf{out}} \mid \neg P_2^{\perp} \bigwedge \text{VIEW}_{\mathsf{ideal}} = (\vec{a}_i, b_1^{(i-1)}) \bigwedge i > i^*\right] \qquad (10)
\end{aligned}$$

Both equations follow easily again from the protocol and simulator descriptions. To see Equation 9, note that in the hybrid world when both P_1 and P_2 abort, P_3 always outputs his own input, $b_{\mathsf{out}} = x_3$. In the ideal world, recall from claim 13 that anytime P_1 and P_2 both abort, and in particular in round $i > i^*$, $\mathcal{S}$ submits either $(0,1)$ or $(1,0)$ to the trusted party, resulting in $b_{\mathsf{out}} = x_3$. For Equation 10, note that in the hybrid world when P_1 aborts in round $i > i^*$, and P_2 does not, P_3 outputs $b_{\mathsf{out}} = f(x_1, x_2, x_3)$. In the ideal world, this is also true, as $\mathcal{S}$ submits (x_1, x_2) to the trusted party (either in step 8a or in step 9). ■

We proceed now to the more difficult claim, in the case when $i \leq i^*$:

Claim 15 *For all fixed inputs, for all outputs b_{out}, and for all feasible adversarial views $(\vec{a}_i, b_1^{(i-1)})$,*

$$\Pr\left[(\text{VIEW}_{\mathsf{hyb}}, \text{OUT}_{\mathsf{hyb}}) = (\vec{a}_i, b_1^{(i-1)}, b_{\mathsf{out}}) \bigwedge i \leq i^*\right] = \Pr\left[(\text{VIEW}_{\mathsf{ideal}}, \text{OUT}_{\mathsf{ideal}}) = (\vec{a}_i, b_1^{(i-1)}, b_{\mathsf{out}}) \bigwedge i \leq i^*\right]$$

Proof: We denote by $P_2^{\perp}$, as before, the event that P_2 aborts the protocol. We now replace the event $\left(\text{VIEW}_{\mathsf{hyb}} = (\vec{a}_i, b_1^{(i-1)}) \bigwedge i \leq i^*\right)$ by E_{hyb}, and the event $\left(\text{VIEW}_{\mathsf{ideal}} = (\vec{a}_i, b_1^{(i-1)}) \bigwedge i \leq i^*\right)$ by $\text{E}_{\mathsf{ideal}}$ to shorten notation. We again have that

$$\begin{aligned}
&\Pr\left[(\text{VIEW}_{\mathsf{hyb}}, \text{OUT}_{\mathsf{hyb}}) = (\vec{a}_i, b_1^{(i-1)}, b_{\mathsf{out}}) \bigwedge i \leq i^*\right] \\
&= \Pr\left[\text{OUT}_{\mathsf{hyb}} = b_{\mathsf{out}} \bigwedge P_2^{\perp} \bigwedge \text{E}_{\mathsf{hyb}}\right] \\
&\quad + \Pr\left[\text{OUT}_{\mathsf{hyb}} = b_{\mathsf{out}} \bigwedge \neg P_2^{\perp} \bigwedge \text{E}_{\mathsf{hyb}}\right] \\
&= \Pr\left[\text{OUT}_{\mathsf{hyb}} = b_{\mathsf{out}} \mid P_2^{\perp} \bigwedge \text{E}_{\mathsf{hyb}}\right] \cdot \Pr\left[P_2^{\perp} \bigwedge \text{E}_{\mathsf{hyb}}\right] \\
&\quad + \Pr\left[\text{OUT}_{\mathsf{hyb}} = b_{\mathsf{out}} \mid \neg P_2^{\perp} \bigwedge \text{E}_{\mathsf{hyb}}\right] \cdot \Pr\left[\neg P_2^{\perp} \bigwedge \text{E}_{\mathsf{hyb}}\right]
\end{aligned}$$

and again, the same probabilistic argument holds in the ideal world. Rewriting the above, therefore, we equivalently must prove that

$$\begin{aligned}
&\Pr\left[\text{OUT}_{\mathsf{hyb}} = b_{\mathsf{out}} \mid P_2^{\perp} \bigwedge \text{E}_{\mathsf{hyb}}\right] \cdot \Pr\left[P_2^{\perp} \mid \text{E}_{\mathsf{hyb}}\right] \cdot \Pr\left[\text{E}_{\mathsf{hyb}}\right] \\
&\quad + \Pr\left[\text{OUT}_{\mathsf{hyb}} = b_{\mathsf{out}} \mid \neg P_2^{\perp} \bigwedge \text{E}_{\mathsf{hyb}}\right] \cdot \Pr\left[\neg P_2^{\perp} \mid \text{E}_{\mathsf{hyb}}\right] \cdot \Pr\left[\text{E}_{\mathsf{hyb}}\right] \\
&= \Pr\left[\text{OUT}_{\mathsf{ideal}} = b_{\mathsf{out}} \mid P_2^{\perp} \bigwedge \text{E}_{\mathsf{ideal}}\right] \cdot \Pr\left[P_2^{\perp} \mid \text{E}_{\mathsf{ideal}}\right] \cdot \Pr\left[\text{E}_{\mathsf{ideal}}\right] \\
&\quad + \Pr\left[\text{OUT}_{\mathsf{ideal}} = b_{\mathsf{out}} \mid \neg P_2^{\perp} \bigwedge \text{E}_{\mathsf{ideal}}\right] \cdot \Pr\left[\neg P_2^{\perp} \mid \text{E}_{\mathsf{ideal}}\right] \cdot \Pr\left[\text{E}_{\mathsf{ideal}}\right]
\end{aligned}$$

Note that trivially have

$$\Pr\left[\neg P_2^{\perp} \mid \text{E}_{\mathsf{hyb}}\right] = \Pr\left[\neg P_2^{\perp} \mid \text{E}_{\mathsf{ideal}}\right]$$

and that

$$\Pr\left[P_2^{\perp} \mid \mathrm{E}_{\mathsf{hyb}}\right] = \Pr\left[P_2^{\perp} \mid \mathrm{E}_{\mathsf{ideal}}\right]$$

Furthermore, by the definition of the protocol, if P_2 aborts, P_3 outputs $b_{\mathsf{out}} = x_3$ (just as in the previous claim). It is easy to see that this is true in the ideal world as well, so we have

$$\Pr\left[\textsc{out}_{\mathsf{hyb}} = b_{\mathsf{out}} \mid P_2^{\perp} \bigwedge \mathrm{E}_{\mathsf{hyb}}\right] = \Pr\left[\textsc{out}_{\mathsf{ideal}} = b_{\mathsf{out}} \mid P_2^{\perp} \bigwedge \mathrm{E}_{\mathsf{ideal}}\right]$$

When P_2 does not abort, in both worlds $b_{\mathsf{out}} = b_1^{(i-1)}$. So as long as we can prove that

$$\Pr\left[\mathrm{E}_{\mathsf{hyb}}\right] = \Pr\left[\mathrm{E}_{\mathsf{ideal}}\right] \tag{11}$$

it will then follow that

$$\Pr\left[\textsc{out}_{\mathsf{ideal}} = b_{\mathsf{out}} \mid \neg P_2^{\perp} \bigwedge \mathrm{E}_{\mathsf{ideal}}\right] = \Pr\left[\textsc{out}_{\mathsf{hyb}} = b_{\mathsf{out}} \mid \neg P_2^{\perp} \bigwedge \mathrm{E}_{\mathsf{hyb}}\right]$$

which will complete the proof of our claim. Before proceeding, we make one final simplification of Equation 11. Recall that any view $\vec{a}_i$ of $\mathcal{A}$ (after the completion of $\mathsf{ShareGen}$) consists simply of the values $b_3^{(1)}, \ldots, b_3^{(i)}, b_1^{(i-1)}$. Letting $\textsc{view}_{\mathsf{hyb}}^{i-1}$ (respectively $\textsc{view}_{\mathsf{ideal}}^{i-1}$) denote the values received by $\mathcal{A}$ in the first $i-1$ rounds of the protocol in the hybrid (resp. ideal) world, and $\textsc{view}_{\mathsf{hyb}}^{i}$ (resp. $\textsc{view}_{\mathsf{ideal}}^{i}$) denote the round i message, along with the following final message received by $\mathcal{A}$ after it aborts P_1 in round i, we note that:

$$\begin{aligned}
&\Pr\left[\mathrm{E}_{\mathsf{hyb}}\right] \\
&\quad = \Pr\left[\textsc{view}_{\mathsf{hyb}}^{i} = (b_3^{(i)}, b_1^{(i-1)}) \mid \textsc{view}_{\mathsf{hyb}}^{i-1} = \vec{a}_{i-1} \bigwedge i \leq i^*\right] \cdot \Pr\left[\textsc{view}_{\mathsf{hyb}}^{i-1} = \vec{a}_{i-1} \bigwedge i \leq i^*\right]
\end{aligned}$$

and, equivalently in the ideal world:

$$\begin{aligned}
&\Pr\left[\mathrm{E}_{\mathsf{ideal}}\right] \\
&\quad = \Pr\left[\textsc{view}_{\mathsf{ideal}}^{i} = (b_3^{(i)}, b_1^{(i-1)}) \mid \textsc{view}_{\mathsf{ideal}}^{i-1} = \vec{a}_{i-1} \bigwedge i \leq i^*\right] \cdot \Pr\left[\textsc{view}_{\mathsf{ideal}}^{i-1} = \vec{a}_{i-1} \bigwedge i \leq i^*\right]
\end{aligned}$$

It is trivially true from the protocol and simulator descriptions that

$$\Pr\left[\textsc{view}_{\mathsf{hyb}}^{i-1} = \vec{a}_{i-1} \bigwedge i \leq i^*\right] = \Pr\left[\textsc{view}_{\mathsf{ideal}}^{i-1} = \vec{a}_{i-1} \bigwedge i \leq i^*\right]$$

Furthermore, conditioned $i \leq i^*$, we know that $\textsc{view}_{\mathsf{hyb}}^{i}$ (resp., $\textsc{view}_{\mathsf{ideal}}^{i}$) is independent of $\textsc{view}_{\mathsf{hyb}}^{i-1}$ (resp., $\textsc{view}_{\mathsf{ideal}}^{i-1}$). Therefore, to prove Equation 11, and thus Theorem 7, it suffices to prove that

$$\Pr\left[\textsc{view}_{\mathsf{hyb}}^{i} = (b_3^{(i)}, b_1^{(i-1)}) \mid i \leq i^*\right] = \Pr\left[\textsc{view}_{\mathsf{ideal}}^{i} = (b_3^{(i)}, b_1^{(i-1)}) \mid i \leq i^*\right]$$

We proceed now to do this by looking at every possible set of inputs (x_1, x_2, x_3).

If $(\mathbf{x_1 = x_2 = x_3})$:

$$\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{ideal}} = (x_1, x_1)\right] = \Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{hyb}} = (x_1, x_1)\right] = 1$$

In both worlds, $b_3^{(i)}$ is always x_1. When P_1 aborts in the ideal world, in accordance with step 7a, $\mathcal{S}$ chooses $\hat{x}_1 = x_1 = x_2$ with probability $\frac{3}{8}$ and sends $b_1^{(i-1)} = x_1$ to $\mathcal{A}$. If $\mathcal{S}$ chooses $\hat{x}_1 \neq x_1$, then it submits $(\hat{x}_1, x_2)$ for $\hat{x}_1 \neq x_2$ to the trusted party, and $b_{\mathsf{out}} = x_3 = x_1$, so again $b_1^{(i-1)} = x_1$. The analysis is even simpler in the hybrid world, as both values are always x_1.

If $(\mathbf{x_1 = x_2 \neq x_3})$:

$$\begin{aligned}
\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{ideal}} = (x_1, x_1)\right] &= \left((1-\alpha)\cdot\frac{3}{8}\right) + \alpha = \frac{1}{2} \\
\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{hyb}} = (x_1, x_1)\right] &= \left((1-\alpha)\cdot\frac{1}{2}\right) + \left(\alpha\cdot\frac{1}{2}\right) = \frac{1}{2} \\
\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{ideal}} = (x_1, \overline{x}_1)\right] &= (1-\alpha)\cdot\frac{5}{8} = \frac{1}{2} \\
\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{hyb}} = (x_1, \overline{x}_1)\right] &= \left((1-\alpha)\cdot\frac{1}{2}\right) + \left(\alpha\cdot\frac{1}{2}\right) = \frac{1}{2}
\end{aligned}$$

If $(\mathbf{x_3 = x_1 \neq x_2})$:

$$\begin{aligned}
\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{ideal}} = (x_1, x_1)\right] &= \left(\frac{1}{2}(1-\alpha)\cdot\frac{1}{4}\right) + \alpha = \frac{3}{10} \\
\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{hyb}} = (x_1, x_1)\right] &= \left(\frac{1}{2}(1-\alpha)\cdot\frac{1}{2}\right) + \left(\alpha\cdot\frac{1}{2}\right) = \frac{3}{10} \\
\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{ideal}} = (x_1, \overline{x}_1)\right] &= \frac{1}{2}(1-\alpha)\cdot\frac{3}{4} = \frac{3}{10} \\
\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{hyb}} = (x_1, \overline{x}_1)\right] &= \left(\frac{1}{2}(1-\alpha)\cdot\frac{1}{2}\right) + \left(\alpha\cdot\frac{1}{2}\right) = \frac{3}{10} \\
\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{ideal}} = (\overline{x}_1, x_1)\right] &= \Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{hyb}} = (\overline{x}_1, x_1)\right] = \\
\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{ideal}} = (\overline{x}_1, \overline{x}_1)\right] &= \Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{hyb}} = (\overline{x}_1, \overline{x}_1)\right] = \frac{1}{2}(1-\alpha)\cdot\frac{1}{2} = \frac{1}{5}
\end{aligned}$$

If $(\mathbf{x_1 \neq x_2 = x_3})$:

$$\begin{aligned}
\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{ideal}} = (x_1, x_1)\right] &= \Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{hyb}} = (x_1, x_1)\right] = 0 \\
\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{ideal}} = (x_1, \overline{x}_1)\right] &= \Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{hyb}} = (x_1, \overline{x}_1)\right] = \frac{1}{2}(1-\alpha) = \frac{2}{5} \\
\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{ideal}} = (\overline{x}_1, x_1)\right] &= \Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{hyb}} = (\overline{x}_1, x_1)\right] = 0 \\
\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{ideal}} = (\overline{x}_1, \overline{x}_1)\right] &= \Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{hyb}} = (\overline{x}_1, \overline{x}_1)\right] = \frac{1}{2}(1-\alpha) + \alpha = \frac{3}{5}
\end{aligned}$$

The key observation with this last set of inputs is that when $x_2 = x_3$, and $i < i^*$, regardless of what value $\mathcal{S}$ chooses for $\hat{x}_1$, $b_1^{(i-1)} = x_2 = x_3$, just as in the hybrid world.

■

B.3 Completing the Proof of Theorem 9

We first demonstrate that the view of $\mathcal{A}$ in the hybrid world is computationally indistinguishable from its view in the ideal world. In step 3, the commitments by the simulator are all commitments to 0 values rather than to the actual inputs of the honest players. However, it is easy to see that if $\mathcal{A}$ could distinguish between the two worlds in this step, he could violate the security of the underlying commitment scheme. We now show that, except for Step 3 of the simulation, the ideal world view generated by the simulator and the hybrid world view are identically distributed. Recall that at the start of Step 5 of the simulation, we let $\mathcal{P} = \mathcal{I} \cup \mathcal{H}$ denote the set of remaining players. We first note that when the outputs of $\mathsf{CommittedOR}_{\mathcal{P}}(r_i, x_i, \vec{c}^{\,i})_{i:P_i \in \mathcal{I} \cup \mathcal{H}}$ are disagreement lists (rather than the OR of the remaining inputs), then in Step 5, $\mathcal{S}$ is capable both of correctly detecting this, and of computing the disagreement lists, independently of the honest input values. In this step, then, the view of $\mathcal{A}$ will be identical to his view in the hybrid world. If all remaining players are consistent, in which case the output of $\mathsf{CommittedOR}_{\mathcal{P}}$ is the binary OR of the remaining inputs, there are two possibilities. If the input x_i for some $P_i \in \mathcal{I}$ is 1, then the adversarial view created by $\mathcal{S}$ in Step 6b is exactly as in the hybrid world; the output of $\mathsf{CommittedOR}_{\mathcal{I} \cup \mathcal{H}}$ in the hybrid world is always 1 in this case, regardless of the honest players' inputs. When all input values x_i for $P_i \in \mathcal{I}$ are 0, then the hybrid world output of $\mathsf{CommittedOR}_{\mathcal{I} \cup \mathcal{H}}$ will depend on the inputs of the honest players, and $\mathcal{S}$ must query the trusted party to determine this value. Note, however, that in this case the output of $\mathsf{CommittedOR}_{\mathcal{I} \cup \mathcal{H}}$ in all subsequent calls in the hybrid world will remain unchanged, regardless of which players are later excluded, and thus the view generated by $\mathcal{S}$ in Step 6a is correct every time.

We next consider the joint distribution of the honest players' outputs with the view of $\mathcal{A}$. We claim that the output of the honest players (in both worlds) is exactly $b = \bigvee_{P_i} x_i$ for P_i that are never eliminated, where for honest parties, these are simply their original input values, and for malicious parties these are the values they first committed to (either in Step 3 in the ideal world, or in Step 2 in the hybrid world). In the hybrid world, this claim follows trivially from the protocol description. In the ideal world, there are two possible submissions that $\mathcal{S}$ can make to the trusted party: $\mathcal{S}$ can submit all 0's or all 1's. $\mathcal{S}$ submits 0's to the trusted party when all (remaining) inputs x_i for $P_i \in \mathcal{I}$ are 0 (in Step 6a), or when $\mathcal{I} = \emptyset$ (in Step 4). In this case the output of the honest parties is

$$b = \bigvee_{P_i \in \mathcal{I} \cup \mathcal{H}} x_i = \bigvee_{P_i \in \mathcal{I}' \cup \mathcal{H}} x_i$$

for any $\mathcal{I}' \subseteq \mathcal{I}$. Therefore, regardless of which players from $\mathcal{I}$ are eliminated in the future, the output of the honest parties is equal to the OR of the inputs of the non-eliminated players, as claimed. The only time $\mathcal{S}$ submits 1's to the trusted party is in Step 6b, after it has verified that no more parties will abort, and that (at least) one of the remaining inputs is a 1. Here too, then, the output of the honest players is consistent with the inputs of the non-eliminated players. Finally, notice that the set of players still participating at the end of the protocol depends only on the view of $\mathcal{A}$. Since we have already argued that the distributions on $\mathcal{A}$'s views in the two worlds are computationally indistinguishable, it follows that the distribution on possible sets of non-eliminated players, $\mathcal{I} \cup \mathcal{H}$ are computationally indistinguishable as well. This completes the proof sketch.