Complete Fairness in Multi-Party Computation Without an Honest Majority

Samuel Dov Gordon[*]

Abstract

A well-known result of Cleve shows that complete fairness is impossible, in general, without an honest majority. Somewhat surprisingly, Gordon et al. recently showed that certain (non-trivial) functions *can* be computed with complete fairness in the *two-party* setting. Motivated by their result, we show here the first completely-fair protocols (for non-trivial functions) in the *multi-party* setting. Specifically, we show that boolean OR can be computed fairly for any number of parties n, and that voting can be computed fairly for $n = 3$ (in each case, we tolerate an arbitrary number of corruptions). Our protocol for voting requires $\omega(\log k)$ rounds, where k is the security parameter, and we prove this is optimal if complete fairness is desired.

1 Introduction

In the setting of secure computation, a set 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 *no one* receives the output. Unfortunately, it has been shown by Cleve [4] that complete fairness is impossible *in general* when a majority of parties is not honest. Until recently, it was thought that *no* non-trivial functions could be computed with complete fairness without an honest majority. A recent result of Gordon et al. [10], however, shows that this perception is wrong, at least in the two-party setting, and forces a re-evaluation of our current understanding of fairness.

Gordon et al. [10] deal exclusively with the case of two-party computation and leave open the question of fairness in the multi-party setting. Extending their work, we initiate a study of fairness in the setting of $n > 2$ parties, and demonstrate the first completely-fair protocols (for non-trivial functionalities) tolerating any $t < n$ corrupted players. We stress that we are interested in obtaining *complete fairness* for certain functionalities, exactly as in the case of an honest majority. This is in contrast to work aimed at achieving weaker notions of partial fairness [5, 9, 2, 14, 7] (but for all functionalities). Our results assume the existence of a broadcast channel (or, equivalently, a PKI).

[*]Dept. of Computer Science, University of Maryland. Email: {gordon}@cs.umd.edu.

Although one could meaningfully ask what can be achieved in the absence of broadcast, we have chosen to assume broadcast so as to separate questions of *fairness* from questions of *agreement.*

Our first result shows that fairness can be obtained for some non-trivial functions for an arbitrary number of players n. Specifically, we show:

Theorem *(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.*

We stress that OR is a non-trivial function; in particular, Kilian et al. [12] show that OR is complete in the sense that black-box access to the OR functionality is sufficient for the secure computation of any other functionality (without complete fairness).

For our next result, we turn to a specific function of interest: boolean voting (or, equivalently, the majority function). Here, we are able to show a positive result only for the case $n = 3$:

Theorem *(Under suitable cryptographic assumptions) there exists a protocol for securely computing the majority function for $n = 3$ with complete fairness.*

Ishai et al. [11] also present a protocol for computing the voting functionality (for any n) without an honest majority. Their protocol, however, achieves only a weak notion of security when an honest majority is not present. (Indeed, their goal was to present a protocol that achieves the standard notion of security when an honest majority is present, and achieves *some* meaningful notion of security when this is not the case.) Specifically, when t parties are corrupted they can prove, informally, that a real execution of the protocol is as secure as an execution in the ideal world when the adversary is allowed to query the ideal functionality $t + 1$ times. We refer to their work for further explanation.

Our protocol for voting requires $\omega(\log k)$ rounds, where k is the security parameter. Our final result shows that this is, in fact, optimal.

Theorem *Any protocol for securely computing the majority function for $n = 3$ with complete fairness requires $\omega(\log k)$ rounds.*

1.1 Outline of the Paper

Standard notions of security for secure multi-party computation are reviewed in Appendix A. We stress that although the definitions are standard, what is *non-standard* is that we consider the definition of completely-fair secure computation even when we do not have an honest majority. We will refer to the weaker security definition, where dishonest parties are allowed to abort the computation even in the ideal world, as *security with abort.* When considering protocols that are secure with abort, will we assume that malicious parties can cause an abort *only if P_1 is malicious* [8].

In Section 2 we describe our feasibility result for the case of boolean OR, and in Section 3 we show our protocol for computing the 3-party voting functionality. Our lower bound on the number of rounds required for completely-fair computing of voting is described in Section 4. Due to space limitations, most of the proofs have been deferred to the appendices.

2 Fair Computation of the OR Function for n Players

In this section we provide a protocol for securely computing the OR functionality with complete fairness for n parties, any $t < n$ of whom may be corrupted. The basic idea behind the protocol is

Protocol Π_1

Inputs: Let the inputs to OR be $x_1, \ldots, x_n$, where $x_i \in \{0, 1\}$

Computation:

1. Let $\mathcal{P}$ be the set of all players.
2. Each player P_i chooses random coins r_i and uses a computationally-hiding commitment scheme to compute and broadcast $c_i = \mathsf{Com}(x_i, r_i)$. If some P_i does not broadcast anything, all honest players output 1. Otherwise, let $\vec{c} = (c_1, \ldots, c_n)$.
3. All players $P_i \in \mathcal{P}$ run an ideal computation (with abort) for $\mathsf{ORWithVerify}$, where party P_i uses $(r_i, x_i, \vec{c})$ as its input.
4. Let $P \in \mathcal{P}$ be the player with the lowest index in $\mathcal{P}$. If players receive $\perp$ from the execution of $\mathsf{ORWithVerify}$, they set $\mathcal{P} = \mathcal{P} \setminus \{P\}$ and return to step 3.[a]
5. If players receive a set $\mathcal{I}$ of faulty players from the execution of $\mathsf{ORWithVerify}$, they set $\mathcal{P} = \mathcal{P} \backslash \mathcal{I}$ and return to step 3.
6. If players receive a binary output from the execution of $\mathsf{ORWithVerify}$, they output this value and end the protocol.

[a]Recall we assume that only the player with the lowest index can abort.

Figure 1: Functionality OR for n players

to have the parties securely compute OR *with abort*, and then recover in case an abort occurs. The key observation is that the dishonest players only learn something from this computation if they all hold input 0, since if any of the malicious players hold input 1 then they trivially know the result in advance. Therefore, if P_1 is corrupt and chooses to abort after receiving the output (but before allowing the honest parties to receive output)[1], the honest parties can safely assume that his input is 0. They then exclude P_1 and restart the computation of OR with abort without him, which is equivalent (with respect to the output) to substituting a 0 for his input. The protocol is presented in Figure 1.

One technical detail that needs to be addressed in the above description is how we ensure that the malicious parties use the same inputs every time they restart the computation of OR with abort. To force this behavior, we begin the protocol by having every party broadcast a commitment to their input bit. Then, instead of computing a simple OR with abort, the parties compute the functionality $\mathsf{ORWithVerify}$, described in 2. This functionality computes OR, but only after checking that all parties are still using the input values they originally committed to. To achieve this, in addition to providing their original inputs (and random coins), each party will also provide the list of commitments that they originally received from other players. If all players participate correctly, then all inputs will result in exactly those commitments. However, if any two parties disagree on the original commitments, or if any player's input does not result in the expected commitment, the functionality returns a list of all discrepancies, and the honest parties will discover a set of dishonest parties. They then restart the computation without those parties, which, again, is equivalent to substituting a 0 for their inputs. We proceed to give the formal statement and proof of our theorem.

Theorem 1 *Assume* Com *is a computationally-hiding commitment scheme, and that* π *securely computes* $\mathsf{ORWithVerify}$ *with abort. Then protocol* Π_1 *computes* OR *with complete fairness.*

[1]Recall that we are using the convention that only P_1 can instruct the trusted party to abort after receiving output. We refer the reader to Appendix A.3 for more details.

ORWithVerify

Inputs: Let the input of player P_i be $(x_i, r_i, \vec{c}_i)$ where $\vec{c}_i = (c_{i,1}, \ldots, c_{i,\ell})$.

For each party P_j, determine its output as follows:

1. Say P_i *is inconsistent with* P_j if either (1) $\vec{c}_i \neq \vec{c}_j$ or (2) $\mathsf{Com}(x_i, r_i) \neq c_{j,i}$. (Note that this is not a symmetric relation.)
2. Let $\mathcal{I}_j$ be the set of parties inconsistent with P_j.
3. If there exist any parties inconsistent with each other, return $\mathcal{I}_j$ as output to P_j. Otherwise, return $\bigvee_i x_i$ to all parties.

Figure 2: Functionality ORWithVerify for n players

Proof: 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

$$\left\{\text{IDEAL}_{\mathsf{OR},\mathcal{S}}(x_1, \ldots x_n)\right\}_{x_i \in \{0,1\}} \stackrel{\text{c}}{\equiv} \left\{\text{HYBRID}^{\mathsf{ORWithVerify}}_{\Pi_1,\mathcal{A}}(x_1, \ldots, x_n)\right\}_{x_i \in \{0,1\}} .$$

For simplicity we assume Com is perfectly binding, though statistical binding suffices.

1. Initialize $\mathcal{I}$ to be the set of corrupted player, and let $\mathcal{H} = \mathcal{P} \setminus \mathcal{I}$ denote the honest players. For $P_j \in \mathcal{H}$, the simulator $\mathcal{S}$ creates a commitment c_j to a random bit, and broadcasts c_j to the corrupted parties. $\mathcal{S}$ then records the commitment c_i broadcast by each corrupted P_i. 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, and outputs whatever $\mathcal{A}$ outputs.

2. If $\mathcal{I} = \emptyset$ then $\mathcal{S}$ submits 0's to the trusted party on behalf of all corrupted parties. Otherwise, $\mathcal{S}$ continues.

3. All players in $\mathcal{I}$ provide values $(\hat{x}_i, \hat{r}_i, \vec{c}')$ for an ideal computation of ORWithVerify. If $\mathcal{S}$ receives a binary output from ORWithVerify, it continues with Step 4. Otherwise, let $P \in \mathcal{H} \cup \mathcal{I}$ be the player with the lowest index in $\mathcal{H} \cup \mathcal{I}$, and let $\mathcal{I}'$ be the list of malicious players output by ORWithVerify:

   (a) If $P \in \mathcal{I}$, the list $\mathcal{I}'$ is given to P. If P aborts, $\mathcal{S}$ sets $\mathcal{I} = \mathcal{I} \setminus \{P\}$. If P does not abort, then $\mathcal{S}$ sets $\mathcal{I} = \mathcal{I} \setminus \mathcal{I}'$. In both cases, $\mathcal{S}$ returns to Step 2.

   (b) if $P \notin \mathcal{I}$, $\mathcal{S}$ sets $\mathcal{I} = \mathcal{I} \setminus \mathcal{I}'$ and repeats step 2.

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

   (a) If $b = 0$, $\mathcal{S}$ submits all 0's to the trusted party for OR on behalf of $\mathcal{A}$. Denote by b_{out} the output of the trusted party. $\mathcal{S}$ gives b_{out} to $\mathcal{A}$ and outputs what $\mathcal{A}$ outputs.

   (b) if $b = 1$, $\mathcal{S}$ gives the value 1 to $\mathcal{A}$ *without* querying the trusted party. If $P \notin \mathcal{I}$, or if $\mathcal{P}$ does not choose to abort, then $\mathcal{S}$ submits all 1s to the trusted party and outputs whatever $\mathcal{A}$ outputs. If $P \in \mathcal{I}$ and P aborts, $\mathcal{S}$ sets $\mathcal{I} = \mathcal{I} \setminus \{P\}$ and returns to step 2.

■

3 Fair Computation of Voting for Three Players

Players first compute the secure computation (with abort) that is described in $\mathsf{ShareGen}$. This execution results in each player obtaining shares of a three-way secret sharing of three different bit sequences: $(b_1^{(1)}, \ldots, b_1^{(m)})$, $(b_2^{(1)}, \ldots, b_2^{(m)})$ and $(b_3^{(1)}, \ldots, b_3^{(m)})$. The value $b_1^{(i)}$ is the bit that players P_2 and P_3 should output if P_1, aborts in round $i+1$. $b_2^{(i)}$ and $b_3^{(i)}$ are defined symmetrically. In the second part of the protocol, after these shares have been distributed, the players simultaneously broadcast their shares as follows. In round i, P_1 broadcasts its share of $b_1^{(i)}$, P_2 broadcasts its share of $b_2^{(i)}$ and P_3 broadcasts its share of $b_3^{(i)}$. In other words, in round i party P_j sends a share of the value that the *other* parties should output if he ceases to participate in the next round. Notice that if all (or even just two) parties are honest, nobody can reconstruct any of the $b_j^{(i)}$ values upon receiving P_j's share in round i. If P_j does not send a valid, signed message in round $i+1$, only then the other two players, who we refer to as P_{j+1} and P_{j-1}, exchange their own shares of $b_j^{(i)}$, and, using the share received from P_j in the prior round, they reconstruct $b_j^{(i)}$ and output the resulting bit. If all players act honestly, then in the final round they jointly reconstruct $b_1^{(m)} = b_2^{(m)} = b_3^{(m)} = \mathsf{Majority}(x_1, x_2, x_3)$.

A few subtleties arise. One is that we must ensure that the values of $b_j^{(1)}, \ldots, b_j^{(m)}$ are all outputs that are consistent with the inputs of P_{j-1} and P_{j+1}. In particular, if P_{j-1} and P_{j+1} have the same input value, then all values of $b_j^{(i)}$ must be equal to this value. For example, we cannot allow it to occur (due to the correctness requirement) that when both of their inputs are 0, and P_j aborts, they output 1. We provide the details of how these values are determined in protocoI $\mathsf{ShareGen}$. Another difficult arises in the case that two parties are corrupted. Say, players P_{j-1} and P_{j+1} are both controlled by a malicious adversary, then the adversary will learn each value of $b_j^{(i)}$ in round i. To ensure that this extra information is unhelpful, we use the approach taken in [10]. A round i^* is randomly selected according to a geometric distribution with parameter $\alpha = 1/5$, and the value of i^* is kept hidden from the participants. In rounds $i < i^*$ the values of $b_j^{(i)}$ reveal no information: they provide output that uses a randomly chosen input for player P_j. In round $i \geq i^*$, the $b_j^{(i)}$ will reveal the correct output. As we will demonstrate below, the biggest difficulty in proving security will be handling an adversary that happens to abort exactly in round i^*: this adversary will learn the correct output in round i^*, and then abort before the honest party learns the same value. The security of the protocol will rely on the fact that it is difficult for such an adversary to consistently and correctly identify round i^*.

Theorem 2 *Assume that* $(\mathsf{SigGen}, \mathsf{Sig}, \mathsf{Vrfy})$ *is a secure digital signature scheme, and that* π *securely computes* $\mathsf{ShareGen}$ *with abort. Then Protocol* Π_2*, with* $\alpha = 1/5$*, securely computes* $\mathsf{Majority}$ *with complete fairness.*

Proof: Note first that the protocol yields the correct output of f when all players are honest. This is because, with all but negligible probability, $i^* \leq m$, and $b_j^{(m)} = \mathsf{Majority}(x_1, x_2, x_3)$. Note also that once the execution of $\mathsf{ShareGen}$ is complete, the rest of protocol Π_2 is symmetric. We refer the reader to [8] for more details, and simply note that protocols for secure computation with abort exist in which only P_1 has the ability to abort. Therefore, if P_1 is honest, we are assured that π completes successfully (though perhaps with substituted inputs for one or both of the other players if they aborted during its execution). It suffices, then, to prove, without loss of generality, ShareGen

Inputs: Let the inputs to $\mathsf{ShareGen}$ 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^* according to a geometric distribution with parameter α (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 $b_j^{(i)} = \mathsf{Majority}(x_{j-1}, \hat{x}_j, x_{j+1})$
   - For $i = i^*$ to m, and $j \in \{1,2,3\}$ set $b_j^{(i)} = \mathsf{Majority}(x_1, x_2, x_3)$.
2. For $0 \le i \le m$, $j \in \{1,2,3\}$, 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 $b_{j|1}^{(i)} \oplus b_{j|2}^{(i)} \oplus b_{j|3}^{(i)} = b_j^{(i)}$. For $i = 0$, we will always let the share $b_{j|j}^{(0)} = 0$, and choose $b_{j|j-1}^{(0)}$ and $b_{j|j+1}^{(0)}$ at random such that $b_{j|j-1}^{(0)} \oplus b_{j|j+1}^{(0)} = b_j^{(0)}$.)
3. Compute $(pk, sk) \leftarrow \mathsf{SigGen}(1^k)$. For $0 \le i \le m$, and $l, j \in \{1,2,3\}$, let $\sigma_{l|j}^{(i)} = \mathsf{Sig}_{sk}(i \| j \| b_{l|j}^{(i)})$.

Output:

1. Send to P_j the values $\left\{(b_{1|j}^{(i)}, \sigma_{1|j}^{(i)}), (b_{2|j}^{(i)}, \sigma_{2|j}^{(i)}), (b_{3|j}^{(i)}, \sigma_{3|j}^{(i)})\right\}_{i=0}^{m}$ and public key pk.

Figure 3: Functionality $\mathsf{ShareGen}$, parameterized by a value α.

that Π_2 is secure when P_1 alone is corrupt, and when both P_1 and P_2 are corrupted. The following two claims and their proofs will therefore prove our theorem. In what follows, we drop the matter of signatures to simplify reading. We assume that the simulator in both claims will send signed messages and will verify received messages. When we say a corrupted player aborts, this includes the case where he sends an incorrectly signed message.

Claim 1 *For every non-uniform, polynomial-time adversary $\mathcal{A}$ corrupting P_1 and running Π_2 in a hybrid model with access to an ideal functionality computing* $\mathsf{ShareGen}$ *(with abort), there exists a non-uniform, polynomial-time adversary $\mathcal{S}$ corrupting P_1 and running in the ideal world with access to an ideal functionality computing f (with complete fairness), such that*

$$\left\{\mathrm{IDEAL}_{f,\mathcal{S}}(x_1, x_2, x_3, k)\right\}_{x_i \in \{0,1\}, k \in \mathbb{N}} \overset{\mathrm{c}}{\equiv} \left\{\mathrm{HYBRID}_{\Pi_2,\mathcal{A}}^{\mathsf{ShareGen}}(x_1, x_2, x_3, k)\right\}_{x_i \in \{0,1\}, k \in \mathbb{N}}.$$

Proof: The proof of Claim 1 is similar to and simpler than the proof of Claim 2 that follows. We leave the proof of Claim 1 to Appendix B. ■

Claim 2 *For every non-uniform, polynomial-time adversary $\mathcal{A}$ corrupting P_1 and P_2 and running Π_2 in a hybrid model with access to an ideal functionality computing* $\mathsf{ShareGen}$ *(with abort), and one computing* OR *without abort, there exists a non-uniform, polynomial-time adversary $\mathcal{S}$ corrupting P_1 and P_2 and running in the ideal world with access to an ideal functionality computing f (with complete fairness), such that*

$$\left\{\mathrm{IDEAL}_{f,\mathcal{S}}(x_1, x_2, x_3, k)\right\}_{x_i \in \{0,1\}, k \in \mathbb{N}} \overset{\mathrm{c}}{\equiv} \left\{\mathrm{HYBRID}_{\Pi_2,\mathcal{A}}^{\mathsf{ShareGen}}(x_1, x_2, x_3, k)\right\}_{x_i \in \{0,1\}, k \in \mathbb{N}}.$$

Protocol Π_2

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

The protocol:

1. **Preliminary phase:**
   Parties P_1, P_2 and P_3 run an ideal execution with abort for $\mathsf{ShareGen}$, where P_1 can abort if he is malicious. Each player uses their respective inputs, x_1, x_2 and x_3, and security parameter k. z

   (a) If players receive $\perp$, P_1 is "blamed" (see text), and P_2 and P_3 run an ideal computation *with complete fairness* for OR (see section 2). (Note that this is equivalent to using a substituted input value of 1 for P_1.) Otherwise:

   (b) Denote the output of P_j from $\mathsf{ShareGen}$ by $\left\{(b^{(i)}_{1|j}, \sigma^{(i)}_{1|j}), (b^{(i)}_{2|j}, \sigma^{(i)}_{2|j}), (b^{(i)}_{3|j}, \sigma^{(i)}_{3|j})\right\}_{i=0}^{m}$ and pk.

2. **For $i = 1, \ldots, m-1$ do:**
   Player broadcast:

   (a) All three players simultaneously broadcast: P_j sends $(b^{(i)}_{j|j}, \sigma^{(i)}_{j|j})$

   (b) Players verify each others' broadcasted shares using pk. Invalid signatures are treated as though the sending party aborted the protocol.

   (c) If (only) P_j failed to send a valid message:

      i. P_{j+1} and P_{j-1} respectively broadcast their shares: $(b^{(i-1)}_{j|j+1}, \sigma^{(i-1)}_{j|j+1})$ and $(b^{(i-1)}_{j|j-1}, \sigma^{(i-1)}_{j|j-1})$.

      ii. Each player verifies that the message sent by the other is valid. Invalid messages are treated as an abort. If one player aborts, the remaining honest player outputs their own input value. Otherwise, P_{j-1} and P_{j+1} both output $b^{(i-1)}_j = b^{(i-1)}_{j|1} \oplus b^{(i-1)}_{j|2} \oplus b^{(i-1)}_{j|3}$. (Recall that for the special case of $i = 1$, $b^{(0)}_{j|j} = 0$, so the two remaining players can still reconstruct $b^{(0)}_j$.)

   (d) If two players fail to send valid messages, the remaining honest player outputs its own input value.

3. **In round $i = m$ do:**

   (a) All players send their shares (and signatures) of $b^{(m)}_1$ and verify as in step 2b. If P_j aborts or sends a bad message, P_{j-1} and P_{j+1} proceed as in step 2c, and output the resulting value. If two players abort, the remaining honest player outputs its input value as in step 2d.

   (b) If verification passes, all players output $b^{(m)}_1 = \mathsf{Majority}(x_1, x_2, x_3) = b^{(m)}_{1|1} \oplus b^{(m)}_{1|2} \oplus b^{(m)}_{1|3}$.

Figure 4: Generic protocol for computing 3 party voting on 2 candidates.

Proof: The proof in this case is subtle, and the simulator $\mathcal{S}$ will be more complex than the one described in Appendix B. The difficulty now is that P_1 and P_2 will learn the value of $b^{(i)}_3$ in round i, which will yield some information about the input of P_3. Note, very much like in the proof of security found in [10], the complication arises because the adversary might abort in round i^*, after learning the correct output, but before P_3 has generated correct output. For ease of explanation in what follows, we will sometimes refer to the actions or views of P_1 and P_2, when more formally we mean the action of $\mathcal{A}$ on behalf of those parties.

By symmetry of the protocol, we will assume that P_1 aborts before P_2. To make this assumption, we will have to consider inputs $(x_1, x_2) = (0, 1)$ independently from inputs $(x_1, x_2) = (1, 0)$. We proceed with the description of $\mathcal{S}$.

0. If in any of the following, $\mathcal{S}$ receives a forged signature from P_1, $\mathcal{S}$ outputs fail and ends the simulation.

1. $\mathcal{S}$ receives x_1' and x_2' from P_1 and P_2 respectively. If $x_1' \notin \{0, 1\}$ (resp. $x_2' \notin \{0, 1\}$) $\mathcal{S}$ continues the simulation setting $x_1' = 1$ (resp. $x_2' = 1$).

2. $\mathcal{S}$ runs $(sk, pk) \leftarrow \mathsf{SigGen}(1^k)$, generates random shares:
$$\left((b_{1|1}^{(1)}, b_{2|1}^{(1)}, b_{3|1}^{(1)}), (b_{1|2}^{(1)}, b_{2|2}^{(1)}, b_{3|2}^{(1)}), \ldots, (b_{1|1}^{(m)}, b_{2|1}^{(m)}, b_{3|1}^{(m)}), (b_{1|2}^{(m)}, b_{2|2}^{(m)}, b_{3|2}^{(m)})\right)$$
and sends these and pk to $\mathcal{A}$. If P_1 aborts, then $\mathcal{S}$extracts x_2'' from P_2, as intended for the ideal OR functionality, and sends $(1, x_2'')$ to the trusted party.

3. If P_1 (and thus P_2) has not yet aborted, $\mathcal{S}$ 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 when in fact we mean x_1' and x_2', remembering, of course that $\mathcal{A}$ could have used substituted inputs.

4. For round $i = 1, \ldots, (i^* - 1)$, $\mathcal{S}$ chooses a value for $b_3^{(i)}$ as follows:
   (a) If $x_1 = x_2 = 0$, he chooses $b_3^{(i)} = 0$
   (b) If $x_1 = x_2 = 1$, he chooses $b_3^{(i)} = 1$
   (c) If $x_1 \neq x_2$, he chooses $b_3^{(i)} = \begin{cases} 0 & \text{with probability } \frac{1}{2} \\ 1 & \text{with probability } \frac{1}{2} \end{cases}$
   He then sends $b_{3|3}^{(i)}$ to $\mathcal{A}$ such that $b_3^{(i)} = b_{3|3}^{(i)} \oplus b_{3|1}^{(i)} \oplus b_{3|2}^{(i)}$.

   For simplicity, from here forward, we will say that $\mathcal{A}$ *received* b in round i (or, equivalently that $\mathcal{S}$ *sent* b) if $b_{3|3}^{(i)} \oplus b_{3|1}^{(i)} \oplus b_{3|2}^{(i)} = b_3^{(i)} = b$.

5. 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 $\hat{x}_1 = x_1$ with probability $\frac{3}{8}$
   (b) If $x_1 \neq x_2$ and $b_3^{(i)} = x_1$ then $\hat{x}_1 = x_1$ with probability $\frac{1}{4}$
   (c) If $x_1 \neq x_2$ and $b_3^{(i)} = x_2$ then $\hat{x}_1 = x_1$ with probability $\frac{1}{2}$
   Note that when $x_1 = x_2$, $b_3^{(i)} = x_1$. $\mathcal{S}$ now finishes the simulation as follows.
   (d) If $\hat{x}_1 \neq \hat{x}_2$, then $\mathcal{S}$ submits $(\hat{x}_1, \hat{x}_2)$ to the trusted party. Denote the output it receives from the trusted party by $b_{\mathsf{out}} = f(\hat{x}_1, \hat{x}_2, x_3)$. $\mathcal{S}$ set $b_1^{(i-1)} = b_{\mathsf{out}}$, sends $b_1^{(i-1)}$ to P_2, and outputs whatever $\mathcal{A}$ outputs.
   (e) If $\hat{x}_1 = \hat{x}_2$, then $\mathcal{S}$ sends $b_{\mathsf{out}} = b_1^{(i-1)} = \hat{x}_1 = \hat{x}_2$ to P_2 *before* sending anything to the trusted party. If P_2 aborts, it sends $(0, 1)$ to the trusted party. Otherwise, it sends $(\hat{x}_1, \hat{x}_2)$ to the trusted party. In both cases it outputs whatever $\mathcal{A}$ outputs.

6. In round i^*:

   (a) If $x_1 \neq x_2$, $\mathcal{S}$ submits (x_1, x_2) to the trusted party. Denote the value he receives by $b_{\mathsf{out}} = f(x_1, x_2, x_3)$.

   (b) If $x_1 = x_2$, he simply sets $b_{\mathsf{out}} = x_1 = x_2$ *without querying the trusted party* and continues (note that in this case, $b_{\mathsf{out}} = f(x_1, x_2, x_3)$ even though $\mathcal{S}$ did not query the trusted party).

7. In rounds $i^*, \ldots, m$, $\mathcal{S}$ sends b_{out} to $\mathcal{A}$. If $\mathcal{A}$ aborts P_1 and P_2 simultaneously, $\mathcal{S}$ submits $(1, 0)$ to the trusted party (if he hasn't already done so in step 6a). If $\mathcal{A}$ aborts P_1 (alone), $\mathcal{S}$ sets $b_1^{(i-1)} = b_{\mathsf{out}}$ and sends the final share, $b_{1|3}^{(i-1)}$, to P_2.

   (a) If $x_1 \neq x_2$, then $\mathcal{S}$ has already sent (x_1, x_2) to the trusted party. He simply outputs whatever $\mathcal{A}$ outputs and ends the simulation.

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

In the interest of saving space, we refer the reader to Appendix C for the analysis of $\mathcal{S}$.

■

■

4 A Lower Bound on the Round Complexity of Voting

We prove here that any 3-party protocol for voting requires$\omega(\log(n))$ rounds. In our proof, we use a constraint from the ideal world to demonstrate that the last round of the protocol is "unnecessary". We then apply the argument a logarithmic number of times to obtain a protocol that reveals the correct output without any message exchange, yielding a contradiction. As we will see, the argument cannot be applied more than a logarithmic number of times, so the proof will not rule out polynomial-round protocols (such as the one demonstrated in Section 3).

The ideal world constraint that we use for our proof relates to an adversary's ability with respect to two different goals: guessing the honest party's input, and biasing the output away from that value. If an adversary that controls two parties submits inputs $(0, 1)$ or $(1, 0)$ to the trusted party, he can guess the honest party's input with probability 1, as the output in this case is always the same as the honest input. However, for the same reason, the probability that he biases the output away from the honest input is 0. If, on the other hand, the adversary submits $(0, 0)$ or $(1, 1)$, then in interacting with an honest player that chooses input at random, he will bias the output against the honest input with probability $\frac{1}{2}$. Now, however, he obtains no information about the honest input, and the probability that he guesses its value is at most $\frac{1}{2}$. In both cases, the *sum* of these probabilities is 1.

Under the assumption that a secure, completely fair, $O(\log k)$ round protocol for voting exists, we demonstrate a real world adversary that either breaks the ideal world constraint mentioned above (contradicting the assumption that the protocol is secure), or it allows us to prove that the last message of the protocol is unnecessary. Before describing the adversary, we first reintroduce some notation, and formalize the above mentioned constraints.

The notation is the same as that used in Section 3, but we review it here so that the section is self-contained. We assume there exists a secure, completely fair 3-party protocol for voting, and since much of our proof is symmetric (though not necessarily the protocol itself), we refer to the players as P_{j-1}, P_j and P_{j+1}, each respectively holding inputs x_{j-1}, x_j and x_{j+1}. Our proof then will be applied for $j \in \{1, 2, 3\}$, where addition is modulo 3. We denote by $b_j^{(i)}$ the value output by P_{j-1} and P_{j+1} when the last message sent by P_j is in round i. 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 the last message sent by P_{j-1} (resp. P_{j+1}) is in round i.

Theorem 3 *Any protocol* Π *that securely computes* Majority *for three parties with complete fairness requires* $r = \omega(\log k)$ *rounds.*

We begin the proof by demonstrating the following ideal world constraint.

Claim 3 *For all* $j \in \{1, 2, 3\}$ *and any ideal-world adversary* $\mathcal{A}$ *controlling parties* P_{j-1} *and* P_{j+1} *it holds that*

$$\Pr[\mathcal{A} \text{ outputs } x_j] + \Pr[\textsc{output}_j \neq x_j] \leq 1,$$

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

Proof: We begin by letting EQUAL be the event that an ideal-world adversary $\mathcal{A}$ submits two equal inputs (i.e., $x_{j-1} = x_{j+1}$) to the trusted party. It is easy to see that

$$\Pr[\mathcal{A} \text{ outputs } x_j] \leq \frac{1}{2}\Pr[\textsc{equal}] + \Pr\left[\overline{\textsc{equal}}\right],$$

since when EQUAL occurs the adversary learns nothing about the value of x_j. Also,

$$\Pr[\textsc{output}_j \neq x_j] = \frac{1}{2}\Pr[\textsc{equal}]$$

since the only way $\textsc{output}_j \neq x_j$ can occur is if $\mathcal{A}$ submits $x_{j-1} = x_{j+1} = \bar{x}_j$ to the trusted party. The claim is proved by summing these two probabilities:

$$\begin{aligned} &\Pr[\mathcal{A} \text{ outputs } x_j] + \Pr[\textsc{output}_j \neq x_j] \\ &\leq \frac{1}{2}\Pr[\textsc{equal}] + \Pr\left[\overline{\textsc{equal}}\right] + \frac{1}{2}\Pr[\textsc{equal}] \\ &= \Pr[\textsc{equal}] + \Pr\left[\overline{\textsc{equal}}\right] = 1. \end{aligned}$$

■

This ideal-world constraint results in the following real-world constraint. For all $j \in \{1, 2, 3\}$, any inverse polynomial $\mu(n)$, and any adversary $\mathcal{A}$ controlling players P_{j-1} and P_{j+1}

$$\Pr[\mathcal{A} \text{ outputs } x_j] + \Pr[\textsc{output}_j \neq x_j] \leq 1 + \mu(n) \qquad (1)$$

for n sufficiently large. Consider now the following real world adversary $\mathcal{A}_i$ running a completely-fair, $r(n)$-round protocol for voting. $\mathcal{A}_i$ corrupts P_{j-1} and P_{j+1} (for any $j \in \{1, 2, 3\}$) and randomly chooses inputs x_{j-1}, x_{j+1} subject to the constraint $x_{j-1} \neq x_{j+1}$. Then:

- $\mathcal{A}_i$ runs the protocol honestly until he receives a round i message from player P_j (but does not yet send any round i messages).
- $\mathcal{A}_i$ locally computes the value of $b_j^{(i)}$ as though P_j aborted:
  - If $b_j^{(i)} = x_{j-1}$, $\mathcal{A}_i$ aborts P_{j-1} *without sending his round i message* and continues the protocol (honestly) on behalf of P_{j+1} until the end. Note that P_j outputs $b_{j-1}^{(i-1)}$.
  - If $b_j^{(i)} = x_{j+1}$, $\mathcal{A}_i$ aborts P_{j+1} *without sending his round i message* and continues the protocol (honestly) on behalf of P_{j-1} until the end. Note that P_j outputs $b_{j+1}^{(i-1)}$.
- $\mathcal{A}_i$ outputs $b_j^{(i)}$

We use this adversary and the constraint of Equation (1) to prove the following claim.

Claim 4 *Fix μ and n such that Equation (1) holds, and let $\mu = \mu(n)$. Say that there exists an i, with $1 \le i \le r(n)$, such that for all $j \in \{1, 2, 3\}$ and all inputs (x_{j-1}, x_j, x_{j+1}) it holds that:*

$$\Pr\left[b_j^{(i)} = \mathsf{Majority}(x_{j-1}, x_j, x_{j+1})\right] \ge 1 - \mu. \tag{2}$$

Then for all $j \in \{1, 2, 3\}$ and all inputs (x_{j-1}, x_j, x_{j+1}) it holds that:

$$\Pr\left[b_j^{(i-1)} = \mathsf{Majority}(x_{j-1}, x_j, x_{j+1})\right] \ge 1 - 5\mu. \tag{3}$$

Proof: We first note that when $x_{j-1} = x_j = x_{j+1}$, the claim follows trivially by the correctness and fairness properties of the protocol. P_{j-1} and P_{j+1}, for example, must output $\mathsf{Majority}$ regardless of when P_j aborts. So we assume that *some* two inputs differ. Without loss of generality, we will say that $x_{j-1} \ne x_{j+1}$. We begin by analyzing the probability that $\mathcal{A}_i$ outputs x_j, as well as the probability that P_j outputs $\bar{x}_j$. Recall that in the description of $\mathcal{A}_i$, we state that $\mathcal{A}_i$ chooses inputs $x_{j-1} \ne x_{j+1}$, which is also our assumption here. We leave that assumption implicit from here forward.

$$\Pr[\mathcal{A}_i \text{ outputs } x_j] = \Pr\left[b_j^{(i)} = x_j\right] = \Pr\left[b_j^{(i)} = \mathsf{Majority}(x_{j-1}, x_j, x_{j+1})\right] \ge 1 - \mu \tag{4}$$

where the last equality holds because when $x_{j-1} \ne x_{j+1}$, $x_j = \mathsf{Majority}(x_{j-1}, x_j, x_{j+1})$, and the inequality follows from the assumption in our claim (equation 2). We now compute the probability that P_j does not output its own vote, x_j.

$$\begin{aligned}
&\Pr[P_j \text{ outputs } \bar{x}_j] \\
&= \left(\frac{1}{2} \cdot \Pr[P_j \text{ outputs } \bar{x}_j \mid x_{j-1} = x_j]\right) + \left(\frac{1}{2} \cdot \Pr[P_j \text{ outputs } \bar{x}_j \mid x_j = x_{j+1}]\right) \\
&= \frac{1}{2}\Big(\Pr\left[b_{j-1}^{(i-1)} = \bar{x}_j \wedge b_j^{(i)} = x_j \mid x_{j-1} = x_j\right] + \Pr\left[b_{j+1}^{(i-1)} = \bar{x}_j \wedge b_j^{(i)} = \bar{x}_j \mid x_{j-1} = x_j\right] \\
&\quad + \Pr\left[b_{j+1}^{(i-1)} = \bar{x}_j \wedge b_j^{(i)} = x_j \mid x_j = x_{j+1}\right] + \Pr\left[b_{j-1}^{(i-1)} = \bar{x}_j \wedge b_j^{(i)} = \bar{x}_j \mid x_j = x_{j+1}\right]\Big)
\end{aligned} \tag{5}$$

From the constraint in equation 1, we know that the sum of equations 4 and 5 is bounded by $1+\mu$:

$$\begin{aligned}
1+\mu \geq 1-\mu+\frac{1}{2}\Big(&\Pr\left[b_{j-1}^{(i-1)}=\bar{x}_j \wedge b_j^{(i)}=x_j \mid x_{j-1}=x_j\right]+\Pr\left[b_{j+1}^{(i-1)}=\bar{x}_j \wedge b_j^{(i)}=\bar{x}_j \mid x_{j-1}=x_j\right]\\
&+\Pr\left[b_{j+1}^{(i-1)}=\bar{x}_j \wedge b_j^{(i)}=x_j \mid x_j=x_{j+1}\right]+\Pr\left[b_{j-1}^{(i-1)}=\bar{x}_j \wedge b_j^{(i)}=\bar{x}_j \mid x_j=x_{j+1}\right]\Big)\\
\geq\quad & 1-\mu+\frac{1}{2}\Pr\left[b_{j-1}^{(i-1)}=\bar{x}_j \wedge b_j^{(i)}=x_j \mid x_{j-1}=x_j\right]\\
\geq\quad & 1-\mu+\frac{1}{2}\left(1-\Pr\left[b_{j-1}^{(i-1)}=x_j \mid x_{j-1}=x_j\right]-\Pr\left[b_j^{(i)}=\bar{x}_j \mid x_{j-1}=x_j\right]\right)\\
\geq\quad & 1-\mu+\frac{1}{2}\left(1-\Pr\left[b_{j-1}^{(i-1)}=x_j \mid x_{j-1}=x_j\right]-\mu\right)\\
\Rightarrow\quad & \Pr\left[b_{j-1}^{(i-1)}=x_j=x_{j-1} \mid x_{j-1}=x_j\right]\geq 1-5\mu. \qquad (6)
\end{aligned}$$

Furthermore, if $x_j = x_{j+1}$, then by correctness and fairness of the protocol, $\Pr\left[b_{j-1}^{(i-1)}=x_{j+1}=x_j\right] \geq 1-\mu$ for all i. Following the same argument as above, we also have:

$$\begin{aligned}
1+\mu \quad \geq \quad & 1-\mu+\frac{1}{2}\left(1-\Pr\left[b_{j+1}^{(i-1)}=x_j \mid x_j=x_{j+1}\right]-\mu\right)\\
\Rightarrow \quad & \Pr\left[b_{j+1}^{(i-1)}=x_j=x_{j+1} \mid x_j=x_{j+1}\right]\geq 1-5\mu, \qquad (7)
\end{aligned}$$

and when $x_{j-1}=x_j$, we have by the correctness and fairness properties that

$$\Pr\left[b_{j+1}^{(i-1)}=x_{j-1}=x_j\right]\geq 1-\mu.$$

Therefore, under the assumption of Equation (2) and that $x_{j-1}\neq x_{j+1}$, regardless of the value of x_j, it follows that

$$\Pr\left[b_{j-1}^{(i-1)}=\mathsf{Majority}(x_{j-1},x_j,x_{j+1})\right]\geq 1-5\mu$$
$$\Pr\left[b_{j+1}^{(i-1)}=\mathsf{Majority}(x_{j-1},x_j,x_{j+1})\right]\geq 1-5\mu$$

It remains only to prove that

$$\Pr\left[b_{j}^{(i-1)}=\mathsf{Majority}(x_{j-1},x_j,x_{j+1})\right]\geq 1-5\mu$$

To see this, note that if $x_{j-1}\neq x_{j+1}$, either $x_{j-1}\neq x_j$ or $x_{j+1}\neq x_j$. In the first case, we can redefine the adversary $\mathcal{A}_i$ to corrupt parties P_{j-1} and P_j, and in the second case we redefine it to corrupt P_{j+1} and P_j. The rest of the argument is completely symmetric, and both cases leads to the required conclusion. This completes the proof of the claim. ■

We now use Claim 2 to complete the proof of the theorem. Let $p(n)=5^{r(n)}$, and $\mu(n)=3/4p(n)$. For sufficiently large n, by the assumed correctness of Π, Claim 2 holds when $i=r(n)$. We can therefore apply the claim $r(n)$ times to conclude that for all inputs (x_{j-1},x_j,x_{j+1}), and all $j\in\{1,2,3\}$,

$$\Pr\left[b_{j}^{(0)}=\mathsf{Majority}(x_{j-1},x_j,x_{j+1})\right]\geq 1-5^{r(n)}\mu(n)=3/4$$

which implies that for any inputs, any two players can correctly compute the correct output of Π without ever interacting with the third player. This is clearly impossible, and we conclude that no such protocol Π exists.

References

[1] D. Beaver. Foundations of secure interactive computing. In Feigenbaum [6], pages 377–391.

[2] D. Boneh and M. Naor. Timed commitments. In *CRYPTO '00: Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, pages 236–254, London, UK, 2000. Springer-Verlag.

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

[4] R. Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *STOC*, pages 364–369. ACM, 1986.

[5] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

[6] J. Feigenbaum, editor. *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*. Springer, 1992.

[7] J. A. Garay, P. D. MacKenzie, M. Prabhakaran, and K. Yang. Resource fairness and composability of cryptographic protocols. In S. Halevi and T. Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 404–428. Springer, 2006.

[8] O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 2004.

[9] S. Goldwasser and L. A. Levin. Fair computation of general functions in presence of immoral majority. In A. Menezes and S. A. Vanstone, editors, *CRYPTO*, volume 537 of *Lecture Notes in Computer Science*, pages 77–93. Springer, 1990.

[10] S. Gordon, C. Hazay, J. Katz, and Y. Lindell. Complete fairness in secure two-party computation. "STOC 2008, to appear".

[11] Y. Ishai, E. Kushilevitz, Y. Lindell, and E. Petrank. On combining privacy with guaranteed output delivery in secure multiparty computation. In C. Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 483–500. Springer, 2006.

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

[13] S. Micali and P. Rogaway. Secure computation (abstract). In Feigenbaum [6], pages 392–404.

[14] B. Pinkas. Fair secure two-party computation. In E. Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 87–105. Springer, 2003.

A Standard Definitions

A.1 Preliminaries

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

$$\big|\Pr[D(X(a,n)) = 1] - \Pr[D(Y(a,n)) = 1]\big| \leq \mu(n).$$

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

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

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

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 functionality $\mathcal{F} = \{f_n\}$, with $f_n = (f_n^1, \ldots, f_n^m)$, is a protocol satisfying the following functional requirement: if all parties hold the same value n, and P_i begins by holding input $x_i \in X_n^i$, then at the conclusion of the protocol party P_i obtains $f_n^i(x_1, \ldots, x_m; r)$ for a uniformly-chosen random string r, with all but negligible probability (in n).

In what follows, we define what we mean by a *secure* protocol. Our definition follows the standard definition of [8] (based on [9, 13, 1, 3]), *except that we require complete fairness* even though we are in the two-party setting. (Thus, our definition is equivalent to the one in [8] for the case of an honest majority, even though we will *not* have an 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 $\mathcal{P} = \{P_1, \ldots, P_m\}$, and there is an adversary $\mathcal{A}$ who has corrupted some subset $\mathcal{I} \subset \mathcal{P}$ of them. An ideal execution for the computation of $\mathcal{F} = \{f_n\}$ proceeds as follows:

Inputs: All players P_i hold the same value of n, and their inputs $x_i \in X_n^i$. The adversary $\mathcal{A}$ receives an auxiliary input z.

Send inputs to trusted party: The honest parties send their inputs to the trusted party. $\mathcal{A}$ may substitute any set of (coordinated)values for the corrupted parties that it controls and sends them to the trusted party. Denote the values sent to the trusted party by $\bar{x}'$.

Trusted party sends outputs: If $x_i' \notin X_n^i$ the trusted party sets $x' = \hat{x}_i$ for some default value $\hat{x}_i$. Then, the trusted party chooses r uniformly at random and sends $f_n^i(\bar{x}'; r)$ to P_i.

Outputs: The honest parties output whatever they were sent by the trusted party, the corrupted parties output nothing, and $\mathcal{A}$ outputs any arbitrary (probabilistic polynomial-time computable) function of its view.

We let $\text{IDEAL}_{\mathcal{F},\mathcal{A}(z)}(x, y, n)$ 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. We next consider the real model in which 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 and sends all messages on behalf of these parties, using an arbitrary polynomial-time strategy. The honest parties follow the instructions of π.

Let $\mathcal{F}$ be as above and let π be a multi-party protocol computing $\mathcal{F}$. Let $\mathcal{A}$ be a non-uniform probabilistic polynomial-time machine with auxiliary input z. We let $\text{REAL}_{\pi,\mathcal{A}(z)}(x, y, n)$ be the random variable consisting of the view of the adversary and the output of the honest parties, following an execution of π where all parties begin by holding n, and P_i begins by holding and input x_i.

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

Definition 4 *Let $\mathcal{F}$ be as above. Protocol π is said to* securely compute $\mathcal{F}$ 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*

$$\left\{\text{IDEAL}_{\mathcal{F},\mathcal{S}(z)}(\bar{x}, n)\right\}_{\bar{x}\in(X_n^1\times\cdots\times X_n^m),\, z\in\{0,1\}^*,\, n\in\mathbb{N}} \stackrel{\text{c}}{\equiv} \left\{\text{REAL}_{\pi,\mathcal{A}(z)}(\bar{x}, n)\right\}_{\bar{x}\in(X_n^1\times\cdots\times X_n^m),\, z\in\{0,1\}^*,\, n\in\mathbb{N}} .$$

A.3 Secure Multi-Party Computation With Abort

This definition is the standard one for secure multi-party computation without an honest majority [8] in that it allows *early abort*; i.e., the adversary may receive its own outputs even though the honest parties do not. We again let $\mathcal{P} = \{P_1, \ldots P_m\}$ denote the parties, and consider an adversary $\mathcal{A}$ 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 $\mathcal{F} = \{f_n\}$, which is now defined as follows:

Inputs: As previously.

Send inputs to trusted party: As previously.

Trusted party sends output to corrupted party: If for some i, $x'_i \notin X^i_n$, the trusted party sets $x'_i = \hat{x}_i$ for some default value $\hat{x}_i$. Otherwise, the trusted party chooses r uniformly at random and computes $z_i = f^i_n(\bar{x}'; r)$ If P_1 is corrupted, then for every party P_i that is corrupted, the trusted party sends z_i to this party (i.e., to the adversary $\mathcal{A}$). If P_1 is honest, then the trusted party sends z_i to every party.

Adversary decides whether to abort: If P_1 is dishonest, then after receiving its output (as described above), the adversary $\mathcal{A}$ either sends 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_j to P_j.

Outputs: As previously.

We let $\text{IDEAL}^{\mathsf{abort}}_{\mathcal{F},\mathcal{A}(z)}(\bar{x}, n)$ 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 5 *Let $\mathcal{F}$ be a functionality, and let π be a protocol computing $\mathcal{F}$. Protocol π is said to* securely compute $\mathcal{F}$ with 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*

$$\left\{\text{IDEAL}^{\mathsf{abort}}_{\mathcal{F},\mathcal{S}(z)}(\bar{x}, n)\right\}_{\bar{x}\in(X^1_n\times\cdots\times X^m_n),\, z\in\{0,1\}^*,\, n\in\mathbb{N}} \stackrel{\mathsf{c}}{\equiv} \left\{\text{REAL}_{\pi,\mathcal{A}(z)}(\bar{x}, n)\right\}_{\bar{x}\in(X^1_n\times\cdots\times X^m_n),\, z\in\{0,1\}^*,\, n\in\mathbb{N}}$$

A.4 The Hybrid Model

The hybrid model combines both the real and ideal models. Specifically, an execution of a protocol π in the hybrid model for some ideal functionality $\mathcal{G}$ involves the parties sending regular messages to each other (as in the real model). However, in addition, the parties have access to a trusted party computing $\mathcal{G}$. The parties communicate with this trusted party in exactly the same way as in the ideal models described above; the question of which ideal model is taken (that with or without abort) must be specified. In this paper, we will always consider a hybrid model where the functionality $\mathcal{G}$ is computed according to the ideal model *with abort.* Let $\mathcal{G}$ be a functionality and let π be a multi-party protocol that includes real messages between the parties, and ideal subroutine calls to $\mathcal{G}$. Let $\mathcal{A}$ be a non-uniform probabilistic polynomial-time machine with auxiliary input z. We let $\text{HYBRID}^{\mathcal{G}}_{\pi,\mathcal{A}(z)}(\bar{x}, n)$ be the random variable consisting of the view of the adversary and the output of the honest parties, following an execution of π (with ideal calls to $\mathcal{G}$) where all players begin by holding n, and P_i begins by holding input x_i.

The hybrid model gives a powerful tool for proving the security of protocols. Specifically, it enables us to construct a protocol π for securely computing some functionality $\mathcal{F}$, while using a subprotocol that securely computes $\mathcal{G}$. Furthermore, it is possible to analyze the security of π considering an ideal execution of $\mathcal{G}$ (rather than a real protocol that securely computes it). We denote by π the protocol that uses ideal calls to $\mathcal{G}$ and by π^ρ the composed protocol where *sequential* calls to $\mathcal{G}$ are replaced by *sequential* executions of the real protocol ρ. (Sequentiality here means that while a given execution of ρ is being run, no other messages are sent.) This is summarized in the following theorem, a close variant of which was proved in [3].

Theorem 6 *Let ρ be a protocol that securely computes $\mathcal{G}$ with abort, and let π be a protocol that securely computes $\mathcal{F}$ with complete fairness when given access to a trusted party who computes $\mathcal{G}$*

for the parties (according to the ideal model with abort). Then, the protocol π^ρ securely computes $\mathcal{F}$ with complete fairness.

B Proof of Claim 1

Notice that the view of P_1 in the hybrid world reveals nothing until the final round of the protocol. In particular, the output it receives from $\mathsf{ShareGen}$ is independent of all inputs, and contains nothing but random bits and signatures on these bits. The shares sent in the intermediate rounds have this same property. We now provide a simulator $\mathcal{S}$.

0. If in any of the following, $\mathcal{S}$ receives a forged signature from P_1, $\mathcal{S}$ outputs fail and ends the simulation.

1. $\mathcal{S}$ receives input x_1' from P_1 intended for trusted functionality $\mathsf{ShareGen}$. If $x_1' \notin \{0,1\}$, $\mathcal{S}$ sets $x_1' = 1$ and continues the simulation as though P_1's input is 1.

2. $\mathcal{S}$ runs $(pk, sk) \leftarrow \mathsf{SigGen}(1^k)$, creates random shares $\Big((b_{1|1}^{(1)}, b_{2|1}^{(1)}, b_{3|1}^{(1)}), \ldots, (b_{1|1}^{(m)}, b_{2|1}^{(m)}, b_{3|1}^{(m)})\Big)$, and gives these and pk to P_1. If P_1 aborts, $\mathcal{S}$ submits 1 to the trusted party for f and outputs whatever P_1 outputs.

3. $\mathcal{S}$ picks a value i^* according to a geometric distribution with parameter $\alpha = \frac{1}{5}$.

4. $\mathcal{S}$ now simulates round i for $1 \le i < m-1$ by generating and sending to P_1 (signed) shares of $b_{2|2}^{(i)}$ and $b_{3|3}^{(i)}$ in round i.

   (a) If P_1 aborts in round $i \le i^*$, then $\mathcal{S}$ submits
   $$\hat{x} = \begin{cases} 0 & \text{with probability } \frac{1}{2} \\ 1 & \text{with probability } \frac{1}{2} \end{cases}$$
   to the trusted party and outputs whatever P_1 outputs.

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

5. If P_1 has not yet aborted in round m, then $\mathcal{S}$ sends x_1' to the trusted party, receives $b_{\mathsf{out}} = \mathsf{Majority}(x_1', x_2, x_3)$, and creates $b_{1|2}^{(m)}, b_{1|3}^{(m)}, b_{2|2}^{(m)}, b_{2|3}^{(m)}, b_{3|2}^{(m)}$ and $b_{3|3}^{(m)}$ such that, together with the shares given to P_1 at the beginning of the simulation, for $j \in \{1,2,3\}$, $b_{j|1}^{(m)} \oplus b_{j|2}^{(m)} \oplus b_{j|3}^{(m)} = b_{\mathsf{out}}$. He gives these shares to P_1, and outputs whatever he outputs.

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 identical in both worlds. This is because the shares are independent of all inputs; they are random values whether generated by $\mathcal{S}$ or by the trusted functionality for $\mathsf{ShareGen}$. We only have to argue, therefore, that the other two players output identically distributed values in both worlds when P_1 aborts (if P_1 never aborts, then clearly the simulation is successful). We break our analysis into the following cases:

- If P_1 gives incorrect input in either step 1 of the simulation or step 1 of the protocol in the hybrid world, then in both worlds, signed shares are created with a default value of 1 in place of its actual input. The rest of the protocol and simulation then proceed as though this were the value submitted to $\mathsf{ShareGen}$.

- If P_1 aborts in Step 2, then $\mathcal{S}$ submits $(1, x_2'')$ to the trusted party, and the output of P_2 and P_3 is $\mathsf{Majority}(1, x_2'', x_3) = \mathsf{OR}(x_2'', x_3)$. In the hybrid world, they execute an ideal computation of the OR functionality. It follows that the joint distribution over the view of $\mathcal{A}$ and the outputs of P_2 and P_3 is the same in both worlds. In the following cases, we assume that P_1 aborts after the simulation of $\mathsf{ShareGen}$ is complete, in round i of the message exchange.

- If $\boldsymbol{x_2 = x_3}$
  In the ideal world, the value sent by $\mathcal{S}$ to the trusted party is irrelevant when $x_2 = x_3$. The honest parties both output $f(1, x_2, x_3) = f(0, x_2, x_3) = x_2 = x_3$. In the hybrid world, both P_2 and P_3 output $b_1^{(i-1)} = f(\hat{x}, x_2, x_3) = x_2 = x_3$ if $i \le i^*$, and $b_1^{(i-1)} = f(x_1', x_2, x_3) = x_2 = x_3$ if $i > i^*$. (Note that this relies on the properties of our particular functionality.)

- If $\boldsymbol{x_2 \neq x_3}$
  For $i \le i^*$, in both worlds the output of P_2 and P_3 is

$$b_{\mathsf{out}} = \begin{cases} f(0, x_2, x_3) = 0 & \text{with probability } \frac{1}{2} \\ f(1, x_2, x_3) = 1 & \text{with probability } \frac{1}{2} \end{cases}$$

  by both the specification of $\mathsf{ShareGen}$ and the description of $\mathcal{S}$. For $i > i^*$, in the hybrid world, $b_1^{(i-1)} = f(x_1', x_2, x_3)$, which is exactly what P_2 and P_3 receive in the ideal world, since $\mathcal{S}$ submits x_1' in round i^*.

C Completing the Proof of Claim 2

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 5 *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:

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

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 $\mathsf{ShareGen}$. We therefore assume that $\mathcal{A}$ aborts player P_1 at some point after the execution of $\mathsf{ShareGen}$. 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:

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

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

Claim 6 *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:

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

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

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

and that

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

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 5 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 6a or in step 7b). ■

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

Claim 7 *For all fixed inputs, for all outputs b_{out}, and for all feasible adversarial views $(\vec{a}_i, b_1^{(i-1)})$,*

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

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

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

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

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

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

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

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

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

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 $\mathrm{VIEW}_{\mathsf{hyb}}^{i-1}$ (respectively $\mathrm{VIEW}_{\mathsf{ideal}}^{i-1}$) denote the values received by $\mathcal{A}$ in the first $i-1$ rounds of the protocol in the hybrid (resp. ideal) world, and $\mathrm{VIEW}_{\mathsf{hyb}}^{i}$ (resp. $\mathrm{VIEW}_{\mathsf{ideal}}^{i}$) 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:

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

and, equivalently in the ideal world:

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

It is trivially true from the protocol and simulator descriptions that

$$\Pr\left[\mathrm{VIEW}_{\mathsf{hyb}}^{i-1} = \vec{a}_{i-1} \bigwedge i \le i^*\right] = \Pr\left[\mathrm{VIEW}_{\mathsf{ideal}}^{i-1} = \vec{a}_{i-1} \bigwedge i \le i^*\right]$$

Furthermore, conditioned $i \le i^*$, we know that $\mathrm{VIEW}_{\mathsf{hyb}}^{i}$ (resp., $\mathrm{VIEW}_{\mathsf{ideal}}^{i}$) is independent of $\mathrm{VIEW}_{\mathsf{hyb}}^{i-1}$ (resp., $\mathrm{VIEW}_{\mathsf{ideal}}^{i-1}$). Therefore, to prove Equation 11, and thus Theorem 2, it suffices to prove that

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

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 5a, $\mathcal{S}$ 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 $\hat{x}_1 \ne x_1$, then it submits $(\hat{x}_1, x_2)$ for $\hat{x}_1 \ne x_2$ 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 $(\mathbf{x_1 = x_2 \neq x_3})$:

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

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

$$\begin{aligned}
\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{ideal}} = (x_1, x_1)\right] &= \left(\frac{1}{2}(1-\alpha)\cdot\frac{1}{4}\right) + \alpha = \frac{3}{10} \\
\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{hyb}} = (x_1, x_1)\right] &= \left(\frac{1}{2}(1-\alpha)\cdot\frac{1}{2}\right) + \left(\alpha\cdot\frac{1}{2}\right) = \frac{3}{10} \\
\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{ideal}} = (x_1, x_2)\right] &= \frac{1}{2}(1-\alpha)\cdot\frac{3}{4} = \frac{3}{10} \\
\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{hyb}} = (x_1, x_2)\right] &= \left(\frac{1}{2}(1-\alpha)\cdot\frac{1}{2}\right) + \left(\alpha\cdot\frac{1}{2}\right) = \frac{3}{10} \\
\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{ideal}} = (x_2, x_1)\right] &= \Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{hyb}} = (x_2, x_1)\right] = \\
\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{ideal}} = (x_2, x_2)\right] &= \Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{hyb}} = (x_2, x_2)\right] = \frac{1}{2}(1-\alpha)\cdot\frac{1}{2} = \frac{1}{5}
\end{aligned}$$

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

$$\begin{aligned}
\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{ideal}} = (x_1, x_1)\right] &= \Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{hyb}} = (x_1, x_1)\right] = 0 \\
\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{ideal}} = (x_1, x_2)\right] &= \Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{hyb}} = (x_1, x_2)\right] = \frac{1}{2}(1-\alpha) = \frac{2}{5} \\
\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{ideal}} = (x_2, x_1)\right] &= \Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{hyb}} = (x_2, x_1)\right] = 0 \\
\Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{ideal}} = (x_2, x_2)\right] &= \Pr\left[(b_3^{(i)}, b_1^{(i-1)})_{\mathsf{hyb}} = (x_2, x_2)\right] = \frac{1}{2}(1-\alpha) + \alpha = \frac{3}{5}
\end{aligned}$$

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.

■