Three Party Secure Computation with Friends and Foes

Bar Alon* alonbar08@gmail.com
Amos Beimel† amos.beimel@gmail.com
Eran Omri* omrier@ariel.ac.il

October 17, 2023

Abstract

In secure multiparty computation (MPC), the goal is to allow a set of mutually distrustful parties to compute some function of their private inputs in a way that preserves security properties, even in the face of adversarial behavior by some of the parties. However, classical security definitions do not pose any privacy restrictions on the view of honest parties. Thus, if an attacker adversarially leaks private information to *honest* parties, it does not count as a violation of privacy. This is arguably undesirable, and in real-life scenarios, it is hard to imagine that possible users would agree to have their private information revealed, even if only to other honest parties.

To address this issue, Alon et al. [CRYPTO 20] introduced the notion of *security with friends and foes* (FaF security). In essence, (t, h)-FaF security requires that a malicious adversary corrupting up to t parties cannot help a coalition of h semi-honest parties to learn anything beyond what they can learn from their inputs and outputs (combined with the input and outputs of the malicious parties). They further showed that (t, h)-FaF security with n parties is achievable for any functionality if $2t + h < n$, and for some functionality, (t, h)-FaF security is impossible assuming $2t + h \geq n$. A remaining important open problem is to characterize the set of n-party functionalities that can be computed with (t, h)-FaF security assuming $2t + h \geq n$.

In this paper, we focus on the special, yet already challenging, case of $(1, 1)$-FaF security for three-party, 2-ary (two inputs), symmetric (all parties output the same value) functionalities. We provide several positive results, a lower bound on the round complexity, and an impossibility result. In particular, we prove the following.

1. We identify a large class of three-party Boolean symmetric 2-ary functionalities that can be computed with $(1, 1)$-FaF full security.
2. We identify a large class of three-party (possibly non-Boolean) symmetric 2-ary functionalities, for which no $O(\log \kappa)$-round protocol computes them with $(1, 1)$-FaF full security. This matches the round complexity of our positive results for various interesting functionalities, such as equality of strings.

Keywords: MPC with friends and foes; full security; lower bounds; protocols

*Department of Computer Science, Ariel University. Ariel Cyber Innovation Center (ACIC).
†Department of Computer Science, Ben Gurion University.

Contents

1 Introduction

In secure multiparty computation (MPC), the goal is to allow a set of mutually distrustful parties to compute some function of their private inputs in a way that preserves security properties, even despite adversarial behavior by some of the parties. Some of the most basic security properties that may be desired are correctness, privacy, independence of inputs, fairness, and guaranteed output delivery. The notion of full security captures all of the above security properties.[1] Classical security definitions (cf., [12]) assume the existence of a single adversarial entity controlling the set of corrupted parties. A malicious adversary may deviate from the protocol in any way. In particular, it may send non-prescribed messages to honest parties. Such messages could potentially leak private information to *honest* parties, e.g., the secret input of some other honest party. Since the classical definitions pose *no* restrictions on the view of honest parties in the protocol, they do not count this as a violation of privacy. Moreover, even the protocol itself may instruct all parties to send their inputs to other honest parties, if say, all possible corrupted parties have been previously revealed (e.g., in the protocol of [18]). Again, this would still not count as a violation of privacy according to the classical security definition. This is arguably undesirable in many situations that fall into the MPC framework. Furthermore, when considering MPC solutions for real-life scenarios, it is hard to imagine that possible users would agree to have their private inputs revealed to honest parties (albeit not to malicious ones).

To address this issue, Alon et al. [1] introduced a new security definition called *security with friends and foes* (FaF security) that, in addition to standard security requirement, poses a privacy requirement on the view of subsets of honest parties. In essence, (t, h)-FaF security requires that for every malicious adversary $\mathcal{A}$ corrupting t parties, and for any disjoint subset of h parties, both the view of the adversary and the joint view of the additional h parties can be simulated (separately) in the ideal model. The security of the protocol should hold even if the malicious adversary sends to some h (semi-)honest parties non-prescribed messages. In fact, the adversary is allowed to send messages after the protocol is terminated.

Alon et al. [1] accompanied the new security notion with several feasibility and impossibility results. They showed that achieving (t, h)-FaF security with n parties against computational adversaries is achievable for any functionality if and only if $2t + h < n$. That is, if $2t + h < n$ then for any n-party functionality there exists a (t, h)-FaF secure protocol computing it, and conversely, if $2t + h \geq n$, then there exists a functionality that cannot be computed with (t, h)-FaF security. Note that this does not rule out the existence of n-party functionalities that can still be computed with (t, h)-FaF security, even when $2t + h \geq n$. Indeed, Alon et al. [1] also presented interesting examples of such functionalities. This includes n-party coin tossing with (t, h)-FaF security assuming $t < n/2$ and $h \leq n - t$, and three-party XOR with $(1, 1)$-FaF security, both of which are known to be impossible to securely compute without an honest majority (with standard security requirements) [9]. This raises the following natural question:

Which n-party functionalities can be computed with (t, h)-FaF security assuming $2t + h \geq n$?

[1] Formally, security is defined via the real vs. ideal paradigm, where a (real-world) protocol is required to emulate an ideal setting, in which the adversary is limited to selecting inputs for the corrupted parties and receiving their outputs.

1.1 Our Results

In this paper, we are interested in the special, yet already challenging, three-party setting where all parties output the same value and are interested in achieving $(1,1)$-FaF security[2]. We show several positive results, a lower bound on the round complexity required for achieving FaF security, and an impossibility result. We next review our results, starting with describing the positive results. Before doing so, we introduce a *dealer model*, which simplifies the proofs and descriptions of our protocols.

The dealer model. The following dealer model serves as a middle ground between the ideal world for FaF security and real-world protocols. It is useful for constructing protocols as it abstracts away technical implementation issues. In particular, this allows our protocols to admit information-theoretic security in the dealer model. Furthermore, in the dealer model we define, the adversary receives no messages, and the only attacks it can perform are to change its input and abort prematurely. This makes the security analysis of such protocols much simpler. Importantly, we show a general compilation from protocols in the dealer model to protocols in the real world and vice versa. The second direction, where we compile a real-world FaF secure protocol into a FaF secure protocol in the dealer model, helps us describe impossibility results in a clear way. It additionally gives more intuition into the impossibility result of Alon et al. [1], where the attacker aborts by selecting a round independently from its view in the protocol. The above compilation shows that indeed an attack cannot rely on the view of the adversary, apart from the round number.

In this dealer model, parties interact in rounds via a trusted dealer, and the malicious adversary is only allowed to abort in each round. In more detail, the interaction proceeds as follows. First, the parties send their inputs to the dealer. The dealer then computes *backup values* for each pair of parties for each round. These values will later be used as the output of two parties in case the remaining third party aborts. Then, in each round, the dealer approaches the parties in a certain order, without revealing any information to the approached party, besides the round number. The party being approached responds with either $\mathsf{continue}$ or abort. If it sends abort, then the dealer sends to the remaining pair of parties a backup value (that depends on the round number). The two parties output this value and halt. Additionally, the dealer also sends to each honest party the appropriate backup values corresponding to the honest party and the aborting party (this models FaF security where the malicious adversary may send its real world view to the other parties). If no abort occurred then the dealer sends the output of the function to all parties.

Theorem 1.1 (Informal)**.** *Assume that secure protocols for oblivious transfer exist. Let $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \to \mathcal{W}$ be a three-party functionality. Then f can be computed with $(1,1)$-FaF security if and only if it can be computed with $(1,1)$-FaF security in the dealer model.*

Possibility results for $(1,1)$-FaF security. We focus on $(1,1)$-FaF security in the three-party setting, assuming that only two parties hold inputs, and that all parties receive the same output (i.e., symmetric functionalities). We provide several positive results in this setting.

In our first result, we show that if a 2-ary function (two inputs) f has a two-party protocol that computes it with both (standard) malicious security and with (standard) semi-honest security, then f can be computed as a three-party functionality with $(1,1)$-FaF security, with all three parties

[2]The security notion was called FaF full security in [1].

obtaining the output. It is instructive to note that even if a two-party protocol is secure against malicious adversaries, it may still *not* be secure against semi-honest adversaries [4].

Theorem 1.2 (Informal)**.** *Assume that secure protocols for oblivious transfer exist. Let $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{W}$ be a 2-ary function. Assume that there exists a protocol π for computing f as a symmetric two-party functionality, providing both (standard) malicious and semi-honest security. Then f can be computed as a symmetric three-party functionality with $(1,1)$-FaF security.*

Note that simply letting the two parties holding inputs run the secure protocol between themselves, and then having them send the output to the remaining third party does not work. This is due to the fact that a corrupt party can lie about the outcome, and then the third party has no way of detecting who is lying.

As an application, consider Boolean functionalities, namely, the output of the parties is a single bit. Asharov et al. [3] characterized all two-party symmetric Boolean functionalities that can be securely computed. We observe that the protocol they constructed also admits semi-honest security. Thus, we may apply Theorem 1.2 to the class of functionalities captured by the (positive) result of [3], and obtain the following result for three-party FaF-secure computation. First, for a deterministic function $f : \mathcal{X} \times \mathcal{Y} \times \{\lambda\} \to \{0,1\}$ we associate with it a matrix $M_f \in \{0,1\}^{|\mathcal{X}|\times|\mathcal{Y}|}$ defined as $M_f(x,y) = f(x,y)$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. Then we have the following.

Corollary 1.3. *Assume that secure protocols for oblivious transfer exist. Let $f : \mathcal{X} \times \mathcal{Y} \times \{\lambda\} \to \{0,1\}$ be a three-party Boolean symmetric functionality. Assume that either the all-one vector or the all-zero vector is an affine combination[3] of either the rows or the columns of M_f. Then f can be computed with $(1,1)$-FaF security.*

We now turn to our second positive result, providing several sufficient conditions for the existence of $(1,1)$-FaF secure 3-party protocols for Boolean functionalities. For a Boolean function f we let $\overline{M}_f$ be the negated matrix, defined as $\overline{M}_f(x,y) = 1 - f(x,y)$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.

Theorem 1.4 (Informal)**.** *Assume that secure protocols for oblivious transfer exist. Let $f : \mathcal{X} \times \mathcal{Y} \times \{\lambda\} \to \{0,1\}$ be a three-party Boolean symmetric functionality. Assume that at least one of the following holds.*

1. *Both M_f and $\overline{M}_f$ have a trivial kernel, or both M_f^T and $\overline{M}_f^T$ have a trivial kernel, i.e., the kernel contains only the all-zero vector.*
2. *The all-one vector is a linear combination of either the rows or columns of M_f, where all coefficients are strictly positive.*

Then f can be computed with $(1,1)$-FaF security.

The round complexity of the protocol we construct is $\omega(\log \kappa)$, where κ is the security parameter. Below we present a lower bound on the round complexity that matches the upper bound for several functionalities.

Observe that the class of functionalities captured by Theorem 1.4 is different from the class of functionalities captured by Corollary 1.3. Indeed, for an integer $m \geq 2$, consider the equality function $\mathsf{EQ} : [m]^2 \times \{\lambda\} \to \{0,1\}$, defined as $\mathsf{EQ}(x,y) = 1$ if $x = y$, and $\mathsf{EQ}(x,y) = 0$ if $x \neq y$. Then the associated matrix M_{EQ} is the $m \times m$ identity matrix, which clearly satisfies Item 1, hence it can be computed with $(1,1)$-FaF security. However, it cannot be computed as a two-party functionality as it implies coin tossing. We provide a more general theorem alongside its proof in Section 4.2.

[3] A affine combination is a linear combination where the sum of the coefficients is 1.

Negative results. We now turn to our negative results. Our first result is a lower bound on the number of rounds required for FaF security. We identify a class of functionalities such that, in order to compute any of them with $(1,1)$-FaF security, would require many rounds of interactions. To simplify the presentation in this introduction, we limit the statement to Boolean functions (see Theorem 5.2 for the generalization to non-Boolean functions).

Theorem 1.5 (Informal)**.** *Let $f : \mathcal{X} \times \mathcal{Y} \times \{\lambda\} \to \{0,1\}$ be a deterministic three-party Boolean functionality. Assuming that the matrix M_f has no constant rows, no constant columns, and that no row or column has its negation appearing in M_f. Then there is no $O(\log \kappa)$-round protocol computing f with $(1,1)$-FaF security.*

Observe that the equality function $\mathsf{EQ} : [m]^2 \times \{\lambda\} \to \{0,1\}$, where $m \geq 3$, satisfies the conditions in Theorem 1.5. Note that this matches the round complexity of the protocol from Theorem 1.4.

Our final result states there exists a three-party non-Boolean functionality that depends on two inputs, which cannot be computed with FaF security.

Theorem 1.6 (Informal)**.** *Assume the existence of one-way permutations. Then there exists a three-party 2-ary symmetric functionality that cannot be computed with $(1,1)$-FaF security.*

We do not know if such impossibility results hold for a Boolean functionality, and we leave it as an interesting open question.

1.2 Our Techniques

In this section, we provide an overview of our techniques. Let us first recall the definition of $(1,1)$-FaF security. We say that a protocol computes a functionality f with $(1,1)$-FaF security, if for any adversary $\mathcal{A}$ (statically) corrupting a party P the following holds: (i) there exists a simulator Sim that can simulate (in the ideal-world[4]) $\mathcal{A}$'s view in the real-world (so far, this is standard security), and (ii) for any uncorrupted party $\mathsf{Q} \neq \mathsf{P}$, there exists a "semi-honest" simulator Sim_Q, such that, given the parties' inputs and Sim's ideal-world view (i.e., its randomness, inputs, auxiliary input, and output received from the trusted party), can generate a view that is indistinguishable form the real-world view of Q, i.e., $(\text{VIEW}^{\mathsf{real}}_{\mathsf{Q}}, \text{OUT}^{\mathsf{real}})$ is indistinguishable from $(\text{VIEW}^{\mathsf{ideal}}_{\mathsf{Sim}_\mathsf{Q}}, \text{OUT}^{\mathsf{ideal}})$.

We now proceed to describe our techniques. Throughout the rest of the section, we denote the parties by A, B, and C, holding inputs x, y, and z, respectively.

Proof of Theorem 1.1. We show that a functionality can be computed with $(1,1)$-FaF security if and only if it can be computed with in an appropriate dealer model. Let us begin with a more detailed description of a dealer-model protocol. An r-round protocol in the dealer model for $(1,1)$-FaF security is described as follows. First, the parties send their inputs to the dealer. The dealer then computes *backup value* $\mathsf{ab}_0, \ldots, \mathsf{ab}_r$, $\mathsf{ac}_0, \ldots, \mathsf{ac}_r$, and $\mathsf{bc}_0, \ldots, \mathsf{bc}_r$. Then, for $i = 1$ to r, the dealer does the following.

1. If no abort was ever sent, approach party A, which responds with either $\mathsf{continue}$ or abort.
2. If A responds with abort, then send x and bc_{i-1} to B and C, sends $\mathsf{ab}_0, \ldots, \mathsf{ab}_{i-1}$ to B and $\mathsf{ac}_0, \ldots, \mathsf{ac}_{i-1}$ to C, and halts. Parties B and C then output bc_{i-1}.

[4] All the adversary can do in the ideal-world is to select its input for the computation and receive the output. Specifically, it cannot prevent the output from other parties or learn anything other than the output.

3. If A responds with continue, approach party B, which responds with either continue or abort.

4. If B responds with abort, then sends y and ac_{i-1} to A and C, sends $\mathsf{ab}_0, \ldots, \mathsf{ab}_{i-1}$ to A and $\mathsf{bc}_0, \ldots, \mathsf{bc}_i$ to C, and halts. Parties A and C then output ac_{i-1}.

5. If B responds with continue, approach party C, which responds with either continue or abort.

6. If C responds with abort, then sends z and ab_{i-1} to A and B, sends $\mathsf{ac}_0, \ldots, \mathsf{ac}_i$ to A and $\mathsf{bc}_0, \ldots, \mathsf{bc}_i$ to B, and halts. Parties A and B then output ab_{i-1}.

If no abort was ever sent, then the dealer sends the last backup values (which must equal to $f(x, y, z)$ with high probability to ensure correctness), and the parties output the value they received. Showing that the protocol in the dealer model can be emulated by a real world protocol (without the dealer) is done using standard techniques. Specifically, the parties compute a 3-out-of-3 secret sharing of the backup values, each signed using a signature scheme. This computation is done using a FaF secure-with-identifiable-abort protocol. That is, the malicious and semi-honest adversaries may learn the output first, and may prevent the honest party from receiving the output at the cost of revealing the identity of the malicious party. Then, in every round, the parties send their shares for the backup value of the other two parties. If a party changes its share (which is captured with overwhelming probability using the signature scheme) or does not send any message at all, then the remaining two parties reconstruct and output the last backup value that they can reconstruct. See Section 3 for more details.

As for the other direction, we compile a real-world FaF secure protocol into a FaF secure protocol in the dealer model. Here, the dealer samples randomness for the parties and executes the protocol in its head. For each round i, it computes the value that a pair of parties output in case the remaining third party aborts after sending i messages (honestly). It then uses these values to define the backup values that it gives to the parties in the protocol.

Proof of Theorem 1.2. Recall that we are given a function f, for which there is a two-party protocol π_2 that computes f with both malicious security and with semi-honest security. We show that f can be computed with $(1,1)$-FaF security in the three-party setting when all parties receive the output. Let r denote the number of rounds in π_2. We assume without loss of generality that the interaction in π_2 is as follows. Each round $i \in [r]$ is composed of two messages, the first sent by A and the second sent by B.[5] A malicious party may send any message that it wants, or send no message at all. In the latter case, the honest party must output some value from the range of the function (recall that π is fully secure). These values are called *backup values*. We denote by $\mathsf{a}_0, \ldots, \mathsf{a}_r$ and $\mathsf{b}_0, \ldots, \mathsf{b}_r$ the backup values of the parties A and B, respectively. Specifically, we let a_i be the output of A assuming that B sent the messages of the first i rounds honestly but did not send the $(i+1)^{\text{th}}$ message, and we let b_i be the output of B assuming that A sent the messages of the first i rounds honestly but did not send the $(i+1)^{\text{th}}$ message.

We next construct a FaF secure three-party protocol π_3. By Theorem 1.1, it suffices to do so in the dealer model, i.e., it suffices to describe how the dealer computes the backup values. For every $i \in [r]$, the dealer sets $\mathsf{ab}_i = f(x, y)$, $\mathsf{ac}_i = \mathsf{a}_i$, and $\mathsf{bc}_i = \mathsf{b}_i$. Intuitively, a corrupt C cannot affect the output of A and B. Moreover, as π_2 admits semi-honest security, the backup values they receive

[5] Note that transforming a protocol into one with this structure might double the number of rounds.

reveal no information to them.[6] As for a malicious A (a malicious B is completely symmetric), note that A has no view. Therefore, to simulate an adversary $\mathcal{A}_3$ corrupting A, we only need to define an appropriate distribution over the inputs (sent by the simulator to the trusted party), so that the output in both the real and ideal world are indistinguishable. To do this, we emulate $\mathcal{A}_3$ using an adversary $\mathcal{A}_2$ for the two-party protocol π_2. The adversary $\mathcal{A}_2$ behaves honestly until the round where $\mathcal{A}_3$ aborts, and aborts at the same round. By the assumed security of π_2, this attack can be simulated in the two-party ideal world. This defines a distribution over the inputs of A. Using the same distribution in the three-party ideal world results in the same distribution for the output. Now, consider a semi-honest party Q in the three-party protocol; the challenge in the FaF model is to construct a view consistent with the input chosen by the malicious adversary controlling A, and the messages B gets from the dealer. Let i denote the round where $\mathcal{A}$ aborts. If $\mathsf{Q} = \mathsf{B}$, then the only information it receives in the real world is $\mathsf{ab}_0, \ldots, \mathsf{ab}_{i-1}$ and the output $\mathsf{bc}_{i-1} = \mathsf{b}_{i-1}$. Since $\mathsf{ab}_j = f(x, y)$ for all j, this can be simulated in the ideal world, since the simulator for the semi-honest B receives the input of the malicious party A. On the other hand, if $\mathsf{Q} = \mathsf{C}$, then in the real world it receives $\mathsf{ac}_0 = \mathsf{a}_0, \ldots, \mathsf{ac}_{i-1} = \mathsf{a}_{i-1}$. These values are generated by the simulator for $\mathcal{A}_2$ in the two-party setting. Moreover, they are generated consistently with the output $\mathsf{bc}_{i-1} = \mathsf{b}_{i-1}$. Therefore, this simulator can be used to simulate the view of C.

Proof of Theorem 1.4. We now turn to our second positive result. Here we are given a three-party Boolean symmetric functionality $f : \mathcal{X} \times \mathcal{Y} \times \{\lambda\} \to \{0, 1\}$ satisfying one of two conditions. We show that it can be computed with $(1, 1)$-FaF security. Similarly to the previous result, we may describe only the backup values for the protocol in the dealer model. We construct a protocol inspired by the protocols of [14, 3], which follow the special round paradigm, however, the proof of security follows a new construction for the simulator.

Roughly, a special round i^* (whose value is unknown to all parties) is sampled at random according to a geometric distribution with a sufficiently small parameter $\alpha > 0$. Before round i^* is reached, the backup values ac_i of A and C, and bc_i of B and C, are random and independent. After i^* the backup values are equal to $f(x, y)$. In more detail, for every $i < i^*$ we let $\mathsf{ac}_i = f(x, \tilde{y}_i)$, where $\tilde{y}_i \leftarrow \mathcal{Y}$ is sampled uniformly at random, and for every $i < i^* + 1$ we let $\mathsf{bc}_i = f(\tilde{x}_i, y)$, where $\tilde{x}_i$ is chosen according to some distribution that depends on the function. All other backup values are equal to $f(x, y)$. Finally, the backup values for A and B are all equal to $f(x, y)$.[7]

First, observe that a corrupt C cannot attack the protocol, since it cannot prevent A and B from outputting $f(x, y)$, nor can it provide them with any new information. Next, similarly to [14, 3], a corrupt B cannot attack since C learns the backup value ac_i before it learns bc_i. Thus, if B aborts at round i^* or afterwards, then A and C output $f(x, y)$. Otherwise, if B aborts before i^*, then A and C output an independent random value. Additionally, B cannot help either of the other parties to obtain any additional information. We are left with the case where A is malicious, which can generate an advantage for C by guessing $i^* + 1$ and aborting in this round. This causes B and C to output $\mathsf{bc}_{i^*} = f(\tilde{x}_{i^*}, y)$, which is a random value. However, C receives $\mathsf{ac}_{i^*} = f(x, y)$ from the dealer.

We show that a simulator exists; we do so by constructing a different simulator than the one

[6] Note that here are using the fact that π_2 is secure against semi-honest adversaries. Indeed, since A and B are semi-honest, to properly simulate them in the ideal world we need to use a simulator that does not change its input.

[7] The choice of setting bc_i to equal $f(x, y)$ only from round $i^* + 1$ is so that A and C learn the output before B and C. Another approach could be to modify the dealer model so that the dealer approaches B before A.

constructed by [14, 3]. There, the malicious simulator generates the view exactly the same as in the real world, and the advantage of the adversary is simulated by sending to the trusted party an input sampled according to a carefully chosen distribution. For our protocol, we let the malicious simulator send an input according to the "expected" distribution, i.e., the one used in the real world, which is either a random input before i^*+1 or the real input from i^*+1 onward.

We are now left with simulating the advantage that a semi-honest C has over the honest party B. We define its simulator by sampling the backup values *differently* from the real world. In more detail, let i denote the round where the malicious adversary aborted (set to $r+1$ if no such round exists). For every round $j < i$ the simulator generates a backup value ac_j according to the same distribution used in the real world, that is, ac_j is a random value if $j < i^*$, and $\mathsf{ac}_j = f(x,y)$ if $j \geq i^*$ (note that since $i > j$ it follows that $i \geq i^*+1$ in this case, hence the simulator received $f(x,y)$ from the trusted party). At round i, if $i > i^*$ we let the simulator set $\mathsf{ac}_i = f(x,y)$. Otherwise, if $i \leq i^*$ then the simulator samples the backup value according to a carefully chosen distribution. We show that under our assumptions on f, there exists a distribution such that the joint distribution of the view generated by the simulator and the output of honest parties is indistinguishable from the real world. We refer the reader to Section 4.2 for more details.

Proof of Theorem 1.5. We now sketch the proof of our lower bound on the round complexity required for FaF secure computation. Recall that we fix a three-party functionality $f : \mathcal{X} \times \mathcal{Y} \times \{\lambda\} \to \{0,1\}$, for which the matrix M_f has no constant rows, no constant columns, and that no row or column has its negation appearing in M_f. We show there is no $O(\log \kappa)$-round protocol computing f with $(1,1)$-FaF security. We assume that f is such that M_f has no duplicated rows and columns. This is without loss of generality since duplicating rows and columns, and removing duplications, does not compromise the FaF security of the protocol.

Assume towards contradiction there exists an $r = O(\log \kappa)$-round protocol computing f with $(1,1)$-FaF security. We assume without loss of generality that the protocol is in the dealer model (note that the transformation from a real world FaF secure protocol to a FaF secure protocol in the dealer model preserves the number of rounds). To gain some intuition, let us first consider a malicious adversary $\mathcal{B}$ corrupting B that sends $\mathsf{continue}$ to the dealer until round r. The adversary then aborts, causing A and C to output ac_{r-1}, and causing the dealer to send $\mathsf{bc}_r = f(x,y)$ to C.

First, we claim that in order to simulate the attack, the malicious simulator $\mathsf{Sim}_{\mathcal{B}}$ must send y to the trusted party, except with negligible probability. Intuitively, this follows from the following observation. Since $M_f(\cdot, y)$ is not constant, does not appear as duplication, and since the negation of $M_f(\cdot, y)$ does not appear anywhere else in M_f, for any $y' \neq y$ there exists $x_1, x_2 \in \mathcal{X}$ such that $M_f(x_1, y) \neq M_f(x_2, y)$ and $M_f(x_1, y') = M_f(x_2, y')$. Pictorially, the 2×2 matrix

$$\begin{array}{c} \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{cc} y & y' \end{array} \\ \begin{pmatrix} a & b \\ b & b \end{pmatrix} \end{array}$$

where $a \neq b \in \{0,1\}$, is embedded in M_f restricted to y and y' (in particular M_f contains an embedded OR). Now, suppose that the malicious simulator $\mathsf{Sim}_{\mathcal{B}}$ sends y' to the trusted party. Consider the semi-honest simulator $\mathsf{Sim}_{\mathcal{B},\mathsf{C}}$ for a semi-honest C. Note that it will not be able to distinguish between the case where A has input x_1 from the case it has input x_2. However, in the real world C is able to distinguish between them since it receives the output $f(x,y)$.

Next, given that $\mathsf{Sim}_{\mathcal{B}}$ does send y to the trusted party, this implies that in the ideal world the output of the honest party A is $f(x,y)$. Therefore, the same must hold in the real world, except with negligible probability. Recall that the malicious B aborted after receiving r messages from A, thus the output of A is ac_{r-1}. This implies that $\mathsf{ac}_{r-1} = f(x,y)$ except with negligible probability.

We can now continue with the same argument as before, this time applied to a malicious adversary corrupting A and aborting after receiving $r-1$ messages from B. We then apply this argument inductively for all r rounds, each time accumulating another error (from when comparing the real and ideal world). Similarly to the lower bound due to [14], we note that when formalizing this argument, the error that is being accumulated each round is multiplicative, with the error each time being $O(|\mathcal{X}| \cdot |\mathcal{Y}|)$. Therefore, after applying the argument $r = O(\frac{\log \kappa}{\log |\mathcal{X}| + \log |\mathcal{Y}|})$ times, we conclude that with constant probability the parties can compute f without any interaction at all, which is a clear contradiction. We stress that our overall strategy is substantially different from [14] in that we analyze what the simulator can send to the trusted party. We refer the reader to Section 5 for a formal analysis.

Proof of Theorem 1.6. We now show there exists a three-party functionality that depends on two inputs and cannot be computed with $(1,1)$-FaF security. The functionality we consider and the proof of impossibility are nearly identical to that of [1]. Let f be a one-way permutation. We consider the following functionality. Party A holds two strings a and y_{B}, and party B holds two string b and y_{A}. Party C holds no input. The output of all parties is (a,b) if $f(a) = y_{\mathsf{A}}$ and $f(b) = y_{\mathsf{B}}$, and $\perp$ otherwise.

Assume towards contradiction there exists a $(1,1)$-FaF secure protocol computing the function. We may assume the protocol to be in the dealer model. Consider an execution where the strings a and b are sampled uniformly and independently, and that $y_{\mathsf{A}} = f(a), y_{\mathsf{B}} = f(b)$. An averaging argument yields that there must exist a round i, where two parties, say A together with C, can recover (a,b) with significantly higher probability than B together with C. Our attacker corrupts A, sends its original inputs a and y_{B} to the dealer, and sends $\mathsf{continue}$ until round $i+1$. At round $i+1$ it sends abort.

Intuitively, in order to have the output of the honest party B in the ideal world distributed as in the real world (where it is $\perp$ with noticeable probability), the malicious simulator has to change its input (sent to the trusted party) with high enough probability. However, in this case, the semi-honest simulator for C, receives $\perp$ from the trusted party. Since the only information it has on b is $f(b)$, by the assumed security of f, the simulator for B will not be able to recover b with non-negligible probability. Hence, B's simulator will fail to generate a valid view for B. The detailed proof appears in Section 6.

1.3 Related Work

Understanding which functionalities can be computed with full security is the subject of many papers in the standard setting. This started with the seminal result of Cleve [9], who showed that fair two-party coin tossing is impossible. Surprisingly, Gordon et al. [14] showed that many two-party functionalities can be computed with full security. In particular, they showed a functionality containing an embedded XOR that can be computed with full security. This led to a series of works trying to characterize which two-party functionalities can be computed with full security [2, 3, 21, 11, 22]. In particular, [3] characterized the set of symmetric Boolean functionalities that are computable with full security.

In the multiparty setting much less is known. In the honest majority setting, if the parties are given secure point-to-point channels and a broadcast channel, then any functionality can be computed with full security without any cryptographic assumptions [23]. The dishonest majority setting was first considered by [13]. They showed that the three-party majority functionality, and n-party OR can be computed securely, for any number of corruptions. The case where exactly half of the parties can be corrupted was considered by Asharov et al. [3] . The setting of a non-constant number of parties was considered in Dachman-Soled [10]. The "Best-of-both-worlds security" definition [17, 19, 20] requires full security to hold in case of an honest majority, however, if at least half of the parties are corrupted, then the same protocol should be secure-with-abort. Finally, Halevi et al. [15] were the first to consider the solitary output setting, where only one party obtains the output.

1.4 Organization

We present the preliminaries in Section 2. We describe the dealer model in Section 3. Then, in Section 4 we present our positive results. In Section 5 we show our lower bound on the round complexity of $(1,1)$-FaF secure protocols. Finally, in Section 6 we show an impossibility for a 2-ary three-party functionality.

2 Preliminaries

2.1 Notations

We use calligraphic letters to denote sets, uppercase for random variables and distributions, lowercase for values, and we use bold characters to denote vectors. For $n \in \mathbb{N}$, let $[n] = \{1, 2 \dots n\}$. For a set $\mathcal{S}$ we write $s \leftarrow \mathcal{S}$ to indicate that s is selected uniformly at random from $\mathcal{S}$. Given a random variable (or a distribution) X, we write $x \leftarrow X$ to indicate that x is selected according to X. We let λ be the empty string. For a randomized function (or an algorithm) f we write $f(x)$ to denote the random variable induced by the function on input x, and write $f(x; \mathsf{rnd})$ to denote its value when the randomness of f is fixed to rnd.

To define security of protocols, we need to define computational indistinguishability between two distribution ensembles (i.e., the distributions of the real and ideal world). 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 domain that might depend on n. A PPT algorithm is probabilistic polynomial time, and a PPTM is a polynomial time (interactive) Turing machine. A PPT algorithm is non-uniform if it receives an advice as an additional input. A function $\mu \colon \mathbb{N} \to [0,1]$ is called negligible, if for every positive polynomial $p(\cdot)$ and all sufficiently large n, it holds that $\mu(n) < 1/p(n)$. We let $\mathrm{neg}(n)$ denote an unspecified negligible function. Computational indistinguishability is defined as follows.

Definition 2.1. *Let $X = (X_{a,n})_{a \in \mathcal{D}_n, n \in \mathbb{N}}$ and $Y = (Y_{a,n})_{a \in \mathcal{D}_n, n \in \mathbb{N}}$ be two ensembles, and let $\varepsilon = \varepsilon(\cdot)$. We say that X and Y are* ε-computationally indistinguishable*, denoted $X \overset{\mathrm{C}}{\equiv}_{\varepsilon} Y$, if for every non-uniform* PPT *distinguisher D such that for all sufficiently large n and for all $a \in \mathcal{D}_n$, it holds that*

$$\left|\Pr\left[\mathsf{D}(X_{a,n}) = 1\right] - \Pr\left[\mathsf{D}(Y_{a,n}) = 1\right]\right| < \varepsilon(n).$$

We say that X and Y are computationally indistinguishable*, denoted $X \overset{\mathrm{C}}{\equiv} Y$, if they are n^{-c}-computationally indistinguishable for all $c \in \mathbb{N}$.*

Secret sharing schemes. A (threshold) secret-sharing scheme [25, 7] is a method in which a dealer distributes shares of some secret to n parties such that t colluding parties do not learn anything about the secret, and any subset of $t+1$ parties can fully reconstruct the secret. We let $\mathcal{P} = \{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$ denote the set of participating parties. As a convention, for a secret s and a party $\mathsf{P}_i \in \mathcal{P}$, we let $s[i]$ be the share received by P_i. For a subset $\mathcal{S} \subseteq \mathcal{P}$ we denote $s[\mathcal{S}] = (s[i])_{i \in \mathcal{S}}$.

Definition 2.2 (Secret sharing)**.** *A $(t+1)$-out-of-n secret-sharing scheme over a message space $\mathcal{M}$ consists of a pair of algorithms $(\mathsf{Share}, \mathsf{Recon})$ satisfying the following properties:*

1. $(t+1)$**-reconstructability:** *For every secret $s \in \mathcal{M}$ and every subset $\mathcal{I} \subseteq [n]$ of size $|\mathcal{I}| \geq t+1$, if $(s[1], \ldots, s[n]) \leftarrow \mathsf{Share}(s)$ then $s = \mathsf{Recon}(s[\mathcal{I}])$.*

2. t**-privacy:** *For every two secrets $s_1, s_2 \in \mathcal{M}$, and every subset $\mathcal{I} \subseteq [n]$ of size $|\mathcal{I}| \leq t$, the distribution of the shares $s_1[\mathcal{I}]$ of s_1 is identical to that of $s_2[\mathcal{I}]$ of s_2, where $(s_1[1], \ldots, s_1[n]) \leftarrow \mathsf{Share}(s_1)$ and $(s_2[1], \ldots, s_2[n]) \leftarrow \mathsf{Share}(s_2)$.*

In this work, we only consider 3-out-of-3 additive secret sharing schemes. Here, the message space $\mathcal{M}$ is an additive group $\mathbb{G}$, and $\mathsf{Share}(s)$ samples $s[1], s[2] \leftarrow \mathbb{G}$ independently, and sets $s[3] = s - s[1] - s[2]$. The reconstruction algorithm simply adds all shares.

2.2 The Model of Computation

We follow the standard *ideal vs. real* paradigm for defining security [12, 8]. Intuitively, security is defined by describing an ideal functionality, in which both the corrupted and non-corrupted parties interact with a trusted entity. A real-world protocol is secure if an adversary in the real world cannot cause more harm than an adversary in the ideal world. In the classical definition, this is captured by showing that an ideal-world adversary (simulator) can simulate the full view of the real world malicious adversary. For FaF security, we further require that the view of a subset of the uncorrupted parties can be simulated in the ideal world (including the interaction with the adversary). We next give a more detailed definition, tailored to the three-party setting.

The FaF Real Model

A three-party protocol π is defined by a set of three PPT interactive Turing machines $\{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$. Each Turing machine (party) holds at the beginning of the execution the common security parameter 1^κ, a private input, and random coins.

Throughout the entire work, we will assume the parties execute the protocol over a synchronous network. That is, the execution proceeds in rounds: each round consists of a *send phase* (where parties send their messages for this round) followed by a *receive phase* (where they receive messages from other parties). We consider a fully connected point-to-point network, where every pair of parties is connected by a communication line. We will consider the *secure-channels* model, where the communication lines are assumed to be ideally private (and thus the adversary cannot read or modify messages sent between two honest parties). Additionally, we assume the parties have access to a broadcast channel, allowing each party to faithfully send the same message to all other parties.

An adversary is a *non-uniform* PPT interactive Turing machine. It starts the execution with an input that contains the identity of the corrupted party, its input, and an additional auxiliary input $\mathsf{aux} \in \{0,1\}^*$. We will only consider static adversaries that can choose the subset of parties to corrupt prior to the execution of the protocol. At the end of the protocol's execution, the

adversary outputs some function of its view (which consists of its random coins, its auxiliary input, the input of the corrupted party, and the messages it sees during the execution of the protocol, and specifically, including possibly non-prescribed messages sent to it by a malicious adversary).

We consider two adversaries. The first adversary we consider is a malicious adversary $\mathcal{A}$ that controls a single party $\mathsf{P} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$. We will refer to P as the malicious party. The adversary has access to the full view of the corrupted party. Additionally, the adversary may instruct the corrupted party to deviate from the protocol in any way it chooses. The adversary can send messages (even if not prescribed by the protocol) to any uncorrupted party – in every round of the protocol, and can do so after all messages for this round were sent. The adversary can also send messages to the uncorrupted parties *after* the protocol is terminated. The adversary is also given an auxiliary input $\mathsf{aux}_{\mathcal{A}}$.

The second adversary is a semi-honest adversary $\mathcal{A}_{\mathsf{Q}}$ that controls a party $\mathsf{Q} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\} \setminus \{\mathsf{P}\}$ of the remaining parties (for the sake of clarity, we will only refer to P as corrupted). Similarly to $\mathcal{A}$, this adversary also has access to the full view of its party. However, $\mathcal{A}_{\mathsf{Q}}$ *cannot* instruct the party to deviate from the prescribed protocol in any way, but may try to infer information about the remaining non-corrupted party, given its view in the protocol (which includes the joint view of P and Q). This adversary is given an auxiliary input $\mathsf{aux}_{\mathsf{Q}}$. We will refer to Q as the semi-honest party.

We next define the real-world global view for security parameter $\kappa \in \mathbb{N}$, an input tuple (x, y, z), and auxiliary inputs $\mathsf{aux}_{\mathcal{A}}, \mathsf{aux}_{\mathsf{Q}} \in \{0,1\}^*$ with respect to adversaries $\mathcal{A}$ and $\mathcal{A}_{\mathsf{Q}}$ controlling the parties P and Q respectively. Let $\text{OUT}^{\mathsf{real}}_{\pi,\mathcal{A}}(\kappa,(x,y,z))$ denote the outputs of the uncorrupted parties (i.e., those in $\{\mathsf{A},\mathsf{B},\mathsf{C}\} \setminus \{\mathsf{P}\}$) in a random execution of π, with $\mathcal{A}$ corrupting the party P. Further let $\text{VIEW}^{\mathsf{real}}_{\pi,\mathcal{A}}(\kappa,(x,y,z))$ be the output of the malicious adversary $\mathcal{A}$ during an execution of π. In addition, we let $\text{VIEW}^{\mathsf{real}}_{\pi,\mathcal{A},\mathcal{A}_{\mathsf{Q}}}(\kappa,(x,y,z))$ be the output of $\mathcal{A}_{\mathsf{Q}}$ during an execution of π when running alongside $\mathcal{A}$.

We let

$$\text{REAL}_{\pi,\mathcal{A}(\mathsf{aux}_{\mathcal{A}}),}(\kappa,(x,y,z)) = \left(\text{VIEW}^{\mathsf{real}}_{\pi,\mathcal{A}}(\kappa,(x,y,z)),\ \text{OUT}^{\mathsf{real}}_{\pi,\mathcal{A}}(\kappa,(x,y,z))\right),$$

denote the view of the malicious adversary and the output of the uncorrupted parties, and we let

$$\text{REAL}_{\pi,\mathcal{A}(\mathsf{aux}_{\mathcal{A}}),\mathcal{A}_{\mathsf{Q}}(\mathsf{aux}_{\mathsf{Q}})}(\kappa,(x,y,z)) = \left(\text{VIEW}^{\mathsf{real}}_{\pi,\mathcal{A},\mathcal{A}_{\mathsf{Q}}}(\kappa,(x,y,z)),\ \text{OUT}^{\mathsf{real}}_{\pi,\mathcal{A}}(\kappa,(x,y,z))\right),$$

denote the view of the semi-honest adversary and the output of the uncorrupted parties.

The FaF Ideal Model

We next describe the interaction in the *FaF security ideal model*, which specifies the requirements for fully secure FaF computation of the function f with security parameter κ. Let $\mathcal{A}$ be an adversary in the ideal world, which is given an auxiliary input $\mathsf{aux}_{\mathcal{A}}$ and corrupts a party P called *corrupted*. Further let $\mathcal{A}_{\mathsf{Q}}$ be a semi-honest adversary, which controls a party $\mathsf{Q} \in \{\mathsf{A},\mathsf{B},\mathsf{C}\} \setminus \{\mathsf{P}\}$ and is given an auxiliary input $\mathsf{aux}_{\mathsf{Q}}$. We stress that the classical formulation of the ideal model does not contain the second adversary.

The ideal model roughly follows the standard ideal model, where the parties send their inputs to a trusted party that does the computation and sends them the output. Additionally, we give the semi-honest adversary $\mathcal{A}_{\mathsf{Q}}$ the ideal-world view of $\mathcal{A}$ (i.e., its input, randomness, auxiliary input, and output received from the trusted party). This is done due to the fact that in the real world,

we cannot prevent the adversary from sending its entire view to the uncorrupted parties. Formally, the ideal world is described as follows.

The FaF ideal model – Full security.

Inputs: Party A holds 1^κ and $x \in \{0,1\}^*$, party B holds 1^κ and $y \in \{0,1\}^*$, and party C holds 1^κ and $z \in \{0,1\}^*$. The adversaries $\mathcal{A}$ and $\mathcal{A}_\mathsf{Q}$ are given each an auxiliary input $\mathsf{aux}_\mathcal{A}, \mathsf{aux}_\mathsf{Q} \in \{0,1\}^*$ respectively, and the inputs of the party controlled by them. The trusted party T holds 1^κ.

Parties send inputs: Each uncorrupted party (including the semi-honest party) sends its input to T. The malicious adversary $\mathcal{A}$ sends a value v' as the input for corrupted party P. If the adversary does not send any input, the trusted party replaces its input with a default value. Write (x', y', z') for the tuple of inputs received by the trusted party.

The trusted party performs computation: The trusted party T selects a random string rnd and computes $(w_\mathsf{A}, w_\mathsf{B}, w_\mathsf{C}) = f(x', y', z'; \mathsf{rnd})$, and sends w_A to A, sends w_B to B, and sends w_C to C.

The malicious adversary sends its (ideal-world) view: $\mathcal{A}$ sends to $\mathcal{A}_\mathsf{Q}$ its randomness, inputs, auxiliary input, and the output received from T.

Outputs: Each uncorrupted party (i.e., not P) outputs whatever output it received from T, party P output nothing. $\mathcal{A}$ and $\mathcal{A}_\mathsf{Q}$ output some function of their respective views.

We next define the ideal-world global view for security parameter $\kappa \in \mathbb{N}$, an input tuple (x, y, z), and auxiliary inputs $\mathsf{aux}_\mathcal{A}, \mathsf{aux}_\mathsf{Q} \in \{0,1\}^*$ with respect to adversaries $\mathcal{A}$ and $\mathcal{A}_\mathsf{Q}$ controlling the parties P and Q respectively. Let $\text{OUT}^{\mathsf{ideal}}_{f,\mathcal{A}}(\kappa, (x,y,z))$ denote the output of the uncorrupted parties (those in $\{\mathsf{A},\mathsf{B},\mathsf{C}\}\setminus\{\mathsf{P}\}$) in a random execution of the above ideal-world process, with $\mathcal{A}$ corrupting P. Further let $\text{VIEW}^{\mathsf{ideal}}_{f,\mathcal{A}}(\kappa, (x,y,z))$ be the *output* of $\mathcal{A}$ in such a process (this output should simulate the real world view of P). In addition, we let $\text{VIEW}^{\mathsf{ideal}}_{f,\mathcal{A},\mathcal{A}_\mathsf{Q}}(\kappa, (x,y,z))$ be the view description being the *output* of $\mathcal{A}_\mathsf{Q}$ in such a process, when running alongside $\mathcal{A}$. We let

$$\text{IDEAL}_{f,\mathcal{A}(\mathsf{aux}_\mathcal{A})}(\kappa, (x,y,z)) = \left(\text{VIEW}^{\mathsf{ideal}}_{f,\mathcal{A}}(\kappa, (x,y,z)),\ \text{OUT}^{\mathsf{ideal}}_{f,\mathcal{A}}(\kappa, (x,y,z))\right),$$

and we let

$$\text{IDEAL}_{f,\mathcal{A}(\mathsf{aux}_\mathcal{A}),\mathcal{A}_\mathsf{Q}(\mathsf{aux}_\mathsf{Q})}(\kappa, (x,y,z), \mathcal{A}_\mathsf{Q}) = \left(\text{VIEW}^{\mathsf{ideal}}_{f,\mathcal{A},\mathcal{A}_\mathsf{Q}}(\kappa, (x,y,z)),\ \text{OUT}^{\mathsf{ideal}}_{f,\mathcal{A}}(\kappa, (x,y,z))\right).$$

Having defined the real and ideal models, we can now define FaF full security of protocols according to the real/ideal paradigm. For brevity, we will refer to it simply as FaF security. We define a more general security notion, where the distinguishing advantage between the real and ideal worlds, is required to be bounded by a function $\varepsilon(\kappa)$ (we use this in Section 5 to state a more general lower bound on the round complexity required for FaF secure computations).

Definition 2.3 (FaF security)**.** *Let π be a protocol for computing f, and let $\varepsilon = \varepsilon(\cdot)$ be a function of the security parameter. We say that π computes f with* $(1,1)$-FaF ε-security*, if the following holds. For every non-uniform* PPT *adversary $\mathcal{A}$, controlling at most one party $\mathsf{P} \in \{\mathsf{A},\mathsf{B},\mathsf{C}\}$ in the*

real world, there exists a non-uniform PPT *adversary* $\mathsf{Sim}_{\mathcal{A}}$ *controlling the same party (if there is any) in the ideal model and for every non-uniform* semi-honest PPT *adversary* $\mathcal{A}_{\mathsf{Q}}$ *controlling at most one party* $\mathsf{Q} \in \{\mathsf{A},\mathsf{B},\mathsf{C}\} \setminus \{\mathsf{P}\}$ *among the remaining parties, there exists a non-uniform* PPT *adversary* $\mathsf{Sim}_{\mathcal{A},\mathsf{Q}}$*, controlling the same party (if there is any) in the ideal-world, such that*

$$\left\{\mathrm{IDEAL}_{f,\mathsf{Sim}(\mathsf{aux}_{\mathcal{A}})}(\kappa,(x,y,z))\right\}_{\kappa\in\mathbb{N},x,y,z\in\{0,1\}^*,\mathsf{aux}_{\mathcal{A}}\in\{0,1\}^*} \stackrel{\mathrm{C}}{\equiv}_{\varepsilon} \left\{\mathrm{REAL}_{\pi,\mathcal{A}(\mathsf{aux}_{\mathcal{A}})}(\kappa,(x,y,z))\right\}_{\kappa\in\mathbb{N},x,y,z\in\{0,1\}^*,\mathsf{aux}_{\mathcal{A}}\in\{0,1\}^*}.$$

and

$$\left\{\mathrm{IDEAL}_{f,\mathsf{Sim}_{\mathcal{A}}(\mathsf{aux}_{\mathcal{A}}),\mathsf{Sim}_{\mathcal{A},\mathsf{Q}}(\mathsf{aux}_{\mathsf{Q}})}(\kappa,(x,y,z),\mathsf{Sim}_{\mathcal{A},\mathsf{Q}})\right\}_{\kappa\in\mathbb{N},x,y,z\in\{0,1\}^*,\mathsf{aux}_{\mathcal{A}},\mathsf{aux}_{\mathsf{Q}}\in\{0,1\}^*} \stackrel{\mathrm{C}}{\equiv}_{\varepsilon} \left\{\mathrm{REAL}_{\pi,\mathcal{A}(\mathsf{aux}_{\mathcal{A}}),\mathcal{A}_{\mathsf{Q}}(\mathsf{aux}_{\mathsf{Q}})}(\kappa,(x,y,z),\mathcal{A}_{\mathsf{Q}})\right\}_{\kappa\in\mathbb{N},x,y,z\in\{0,1\}^*,\mathsf{aux}_{\mathcal{A}},\mathsf{aux}_{\mathsf{Q}}\in\{0,1\}^*}.$$

We say that π *computed* f *with* $(1,1)$*-FaF security if for all* $c\in\mathbb{N}$*,* π *computes* f *with* $(1,1)$*-FaF* κ^{-c}*-security.*

Observe that the correctness of the computation (in an honest execution) is implicitly required by the above definition. Indeed, as we allow the adversary to corrupt at most one party, by considering adversaries that corrupt no party, the definition requires the output of all parties in the real world to be indistinguishable from $f(x,y,z)$.

We next define the notion of backup values, which are the values that honest parties output in case the third party aborts (after sending messages honestly). Note that the notions of backup values are well-defined for any $(1,1)$-FaF secure protocol.

Definition 2.4 (Backup values)**.** *Let* $f:\mathcal{X}\times\mathcal{Y}\times\mathcal{Z}\to\mathcal{W}$ *be a three-party functionality, and let* π *be an* r*-round protocol computing* f *with* $(1,1)$*-FaF security. Let* $i\in\{0,\ldots,r\}$*, sample the randomness of the parties, and consider an honest execution of* π *with the sampled randomness until all parties sent* i *messages. For two distinct parties* $\mathsf{P},\mathsf{Q}\in\{\mathsf{A},\mathsf{B},\mathsf{C}\}$*, the* i^{th} *backup value of the pair* $\{\mathsf{P},\mathsf{Q}\}$ *is the value that an honest* P *and* Q *output if the third party aborts after sending* i *messages honestly.*

2.3 FaF Security-With-Identifiable-Abort

Although the focus of this work is on full security, in some of our constructions we use protocols admitting *security-with-identifiable-abort*. In terms of the definition, the only requirement that is changed is that the ideal-world simulator operates in a *different ideal model*. We next describe the interaction in the *FaF-secure-with-identifiable-abort ideal model* for the computation of the function f with security parameter κ.

Unlike the full security ideal model, here the malicious adversary can instruct the trusted party not to send the output to the honest parties, however, in this case, the adversary must publish the identity of a corrupted party. In addition, since there is no guarantee that in the real world the semi-honest parties will not learn the output, we always let the them receive their output in the ideal execution. This allows us to simulate unfair protocols, where in addition to the malicious adversary learning the output, it can decide whether the semi-honest parties can learn the output as well.

Let $\mathcal{A}$ be a malicious adversary in the ideal world, which is given an auxiliary input $\mathsf{aux}_{\mathcal{A}}$ and corrupts a party $\mathsf{P} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$. Furthermore, let $\mathcal{A}_{\mathsf{Q}}$ be a semi-honest adversary, which controls a party $\mathsf{Q} \neq \mathsf{P}$ and is given an auxiliary input $\mathsf{aux}_{\mathsf{Q}}$.

The FaF ideal model – Security-with-identifiable-abort.

Inputs: Party A holds 1^{κ} and $x \in \{0,1\}^*$, party B holds 1^{κ} and $y \in \{0,1\}^*$, and party C holds 1^{κ} and $z \in \{0,1\}^*$. The adversaries $\mathcal{A}$ and $\mathcal{A}_{\mathsf{Q}}$ are given each an auxiliary input $\mathsf{aux}_{\mathcal{A}}, \mathsf{aux}_{\mathsf{Q}} \in \{0,1\}^*$ respectively, and the inputs of the party controlled by them. The trusted party T holds 1^{κ}.

Parties send inputs: Each uncorrupted party sends its input to T. The malicious adversary $\mathcal{A}$ sends a value v' as the input for corrupted party P. If the adversary does not send any input, the trusted party replaces its input with a default value. Write (x', y', z') for the tuple of inputs received by the trusted party.

The trusted party performs computation: The trusted party T selects a random string rnd and computes $(w_{\mathsf{A}}, w_{\mathsf{B}}, w_{\mathsf{C}}) = f(x', y', z'; \mathsf{rnd})$, and sends w_{P} to $\mathcal{A}$ and sends w_{Q} to $\mathcal{A}_{\mathsf{Q}}$.

The malicious adversary sends its (ideal-world) view: $\mathcal{A}$ sends to $\mathcal{A}_{\mathsf{Q}}$ its randomness, inputs, auxiliary input, and the output received from T.

Malicious adversary instructs trusted party to continue or halt: The adversary $\mathcal{A}$ sends either $\mathsf{continue}$ or $(\mathsf{abort}, \mathsf{P})$ to T. If it sent $\mathsf{continue}$, then for every uncorrupted party $\mathsf{P}' \neq \mathsf{P}$ the trusted party sends it $w_{\mathsf{P}'}$. Otherwise, if $\mathcal{A}$ sent $(\mathsf{abort}, \mathsf{P})$, then T sends $(\mathsf{abort}, \mathsf{P})$ to the all honest parties.

Outputs: Each uncorrupted party (i.e., not P) outputs whatever output it received from T, party P output nothing. $\mathcal{A}$ and $\mathcal{A}_{\mathsf{Q}}$ output some function of their respective views.

2.4 The Two-Party Model

In one of our results, we will be interested in the two-party setting with (standard) security against both a malicious adversary and a semi-honest adversary, corrupting one party. In terms of definition, both the real and ideal world in the two-party setting are defined analogously to the three-party setting. That is, in the real world, two parties A and B interact, and each holds a private input, the security parameter, and random coins. In the ideal world, the computation is done via a trusted party in a similar way to the three-party definition. In this paper, we consider both security against a malicious adversary, and security against a semi-honest adversary. We say that a two-party protocol is fully secure if it is secure against any malicious adversary, and we say that the protocol if it has semi-honest security if it is secure against any semi-honest adversary.

3 The Dealer Model

In the description of our positive results, it will be convenient to consider a model with a dealer. Here, the real world is augmented with a trusted dealer, which is a PPTM that can interact with the parties in a limited way. Furthermore, the adversary is also limited when compared to a real world adversary: the adversary is assumed to be fail-stop, namely, it acts honestly, however, it may

decide to abort prematurely. Additionally, it may change the input it sends to the dealer. This model, which we show below to be equivalent to $(1,1)$-FaF security, offers a much simpler way to analyze the security of protocols. Moreover, our constructions will achieve information-theoretic security in the dealer model. A similar model was already considered for standard security with a dishonest majority [2, 3, 6, 5].

We next describe a blueprint for an r-round protocol in the dealer model for the $(1,1)$-FaF security model. That is, the blueprint instructs the dealer to compute $3r+3$ backup values and does not specify how to compute these backup values. A protocol in the dealer model is obtained from the blueprint by defining $3r+3$ functions computing these backup values. We will show that such $(1,1)$-FaF secure protocols exist if and only if a $(1,1)$-FaF secure protocol exists in the real world (assuming secure protocols for OT). For simplicity, we assume the function to be symmetric, i.e., all parties obtain the same output.

Protocol 3.1.
Inputs: *Parties* A, B, *and* C *hold inputs* x, y, *and* z, *respectively.*
Common input: *All parties hold the security parameter* 1^κ.

1. *The honest parties send their inputs to the dealer. The malicious adversary sends a value as the input for the corrupted party. If the adversary does not send any input, the dealer replaces it with a default value.*
2. *The dealer computes backup values* $\mathsf{ab}_0,\ldots,\mathsf{ab}_r$, $\mathsf{ac}_0,\ldots,\mathsf{ac}_r$, *and* $\mathsf{bc}_0,\ldots,\mathsf{bc}_r$. *It is required that* ab_0, ac_0, *and* bc_0, *do not depend on the inputs of* C, B, *and* A, *respectively.*
3. *For* $i=1$ *to* r*:*
   (a) *The dealer approaches party* A, *which responds with either* $\mathsf{continue}$ *or* abort.
   (b) *If* A *responds with* abort, *then the dealer sends* x *and* bc_{i-1} *to* B *and* C, *sends* $\mathsf{ab}_0,\ldots,\mathsf{ab}_{i-1}$ *to* B *and* $\mathsf{ac}_0,\ldots,\mathsf{ac}_{i-1}$ *to* C, *and halts. Parties* B *and* C *then output* bc_{i-1}.
   (c) *The dealer approaches party* B, *which responds with either* $\mathsf{continue}$ *or* abort.
   (d) *If* B *responds with* abort, *then the dealer sends* y *and* ac_{i-1} *to* A *and* C, *sends* $\mathsf{ab}_0,\ldots,\mathsf{ab}_{i-1}$ *to* A *and* $\mathsf{bc}_0,\ldots,\mathsf{bc}_i$ *to* C, *and halts. Parties* A *and* C *then output* ac_{i-1}.
   (e) *The dealer approaches party* C, *which responds with either* $\mathsf{continue}$ *or* abort.
   (f) *If* C *responds with* abort, *then the dealer sends* z *and* ab_{i-1} *to* A *and* B, *sends* $\mathsf{ac}_0,\ldots,\mathsf{ac}_i$ *to* A *and* $\mathsf{bc}_0,\ldots,\mathsf{bc}_i$ *to* B, *and halts. Parties* A *and* B *then output* ab_{i-1}.
4. *If no party aborted, the dealer sends* ab_r *to* A, *sends* bc_r *to* B, *and sends* ac_r *to* C.
5. *Party* A *output* ab_r, *party* B *output* bc_r, *and party* C *output* ac_r.

We stress that the dealer is always honest in the above execution. The security of the protocol is defined by comparing the above execution to the ideal world defined previously. However, unlike the real world, here the malicious adversary is only fail-stop. Thus, we say the protocol in the dealer model is $(1,1)$-FaF security if it is $(1,1)$-FaF secure against fail-stop adversaries. Furthermore, note that if the protocol is correct, then it is secure against semi-honest adversaries. This is because

the only information the adversary receives is the last backup value, which equals to the output. Therefore, when proving security, it suffices to always consider the case where there is a malicious adversary corrupting a party. Removing the dealer (i.e., constructing a $(1,1)$-FaF secure protocol without the dealer) can be done using standard techniques. We next provide an intuitive description of the real-world protocol without the dealer. The formal protocol appears below.

At the beginning of the interaction, the parties compute a secret sharing of all the backup values computed by the dealer, using a 3-out-of-3 secret sharing scheme, and all shares are signed.[8] This computation is done using a $(1,1)$-FaF secure-with-identifiable-abort protocol. Then, in each round i, party C broadcasts its share of ab_i, then B broadcasts its share of ac_i, and finally, party A broadcasts its share of bc_i. If a party does not send its share or it sends a different share (which is caught using the signature scheme, except with negligible probability), then the remaining two parties reconstruct the last backup value for which they hold the aborting party's share.

Observe that the view of a corrupted party consists of only random independent shares. Thus, it aborts (or sends an incorrect share) in the real world if and only if it aborts in the dealer model. Additionally, the view of a semi-honest party consists of random shares, the backup value it computes with the remaining honest party, and the shares it can reconstruct if given the malicious party's view. Thus, any attack in the real world can be emulated in the dealer model.

Additionally, the converse is also true. That is, if there is a $(1,1)$-FaF secure protocol computing f in the real world, there is a $(1,1)$-FaF secure protocol computing f in the dealer model. Indeed, the dealer simply computes the backup values of every pair of parties and interacts with the parties as described in the above model. Thus, as the real world and the ideal model are essentially equivalent, we will sometimes refer to the dealer model as the real world. We next formalize the statement and its proof.

Theorem 3.2. *Let $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \to \mathcal{W}$ be a three-party functionality. Then, assuming secure protocols for OT exist, f can be computed with $(1,1)$-FaF security in the real world if and only if it can be computed with $(1,1)$-FaF security in the dealer model.*

We prove the theorem by proving two lemmas, each handling a different direction of the statement.

Lemma 3.3. *Let $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \to \mathcal{W}$ be a three-party functionality. Then, if secure protocols for OT exist and f can be computed with $(1,1)$-FaF security in the dealer model, then f can be computed with $(1,1)$-FaF security in the real world.*

Proof. Assume there is a protocol π^{D} computing f in the dealer model that is $(1,1)$-FaF secure against fail-stop adversaries. We construct a protocol π^{R} computing f with $(1,1)$-FaF security in the real world.

Fix a signature scheme $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ (since OT implies one-way functions [16] and one-way functions imply signature scheme [24], the assumption of the lemma implies signature schemes). Let ShrGen denote the three-party functionality that, given the parties' inputs, outputs a 3-out-of-3 secret sharing for each of the backup values computed by the dealer, each signed using the signature scheme. Formally, we define ShrGen as follows.

Algorithm 3.4 (ShrGen)**.**
Inputs: *Parties* A, B, *and* C *hold inputs x, y, and z, respectively.*
Common input: *The parties hold the security parameter* 1^κ.

[8] The signature key can be replaced with a one-time MAC for every share.

1. *Sample a signature scheme keys* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa)$.

2. *For every* $i \in \{0, \ldots, r\}$ *do the following:*

   (a) *Compute the backup values* ab_i, ac_i, *and* bc_i, *as the dealer computes them.*

   (b) *If* $i = 0$, *then share each backup value in a 2-out-of-2 additive sharing scheme. Otherwise, share each backup value in a 3-out-of-3 additive secret-sharing scheme.*

   (c) *If* $i \geq 1$, *then for each backup value of two parties, sign the share of the third party. That is, for every* $i \in [r]$ *compute the following values:*

   - $\sigma_{i,\mathsf{C}} \leftarrow \mathsf{Sign}_{\mathsf{sk}}(\mathsf{ab}_i[\mathsf{C}])$.
   - $\sigma_{i,\mathsf{B}} \leftarrow \mathsf{Sign}_{\mathsf{sk}}(\mathsf{ac}_i[\mathsf{B}])$.
   - $\sigma_{i,\mathsf{A}} \leftarrow \mathsf{Sign}_{\mathsf{sk}}(\mathsf{bc}_i[\mathsf{A}])$.

3. *Compute the following signatures:*

   - $\sigma_{\mathsf{ab},\mathsf{A}} \leftarrow \mathsf{Sign}_{\mathsf{sk}}(\mathsf{ab}_r[\mathsf{A}])$ *and* $\sigma_{\mathsf{ac},\mathsf{A}} \leftarrow \mathsf{Sign}_{\mathsf{sk}}(\mathsf{ac}_r[\mathsf{A}])$.
   - $\sigma_{\mathsf{ab},\mathsf{B}} \leftarrow \mathsf{Sign}_{\mathsf{sk}}(\mathsf{ab}_r[\mathsf{B}])$ *and* $\sigma_{\mathsf{bc},\mathsf{B}} \leftarrow \mathsf{Sign}_{\mathsf{sk}}(\mathsf{bc}_r[\mathsf{B}])$.
   - $\sigma_{\mathsf{ac},\mathsf{C}} \leftarrow \mathsf{Sign}_{\mathsf{sk}}(\mathsf{ac}_r[\mathsf{C}])$ *and* $\sigma_{\mathsf{bc},\mathsf{C}} \leftarrow \mathsf{Sign}_{\mathsf{sk}}(\mathsf{bc}_r[\mathsf{C}])$.

4. *The parties obtain the following output.*

   - A *receives the public key* pk, *the shares of the backup value* $(\mathsf{ab}_i[\mathsf{A}], \mathsf{ac}_i[\mathsf{A}])_{i=0}^r$ *and* $(\mathsf{bc}_i[\mathsf{A}])_{i=1}^r$, *and the signatures* $(\sigma_{i,\mathsf{A}})_{i=1}^r$, $\sigma_{\mathsf{ab},\mathsf{A}}$, *and* $\sigma_{\mathsf{ac},\mathsf{A}}$.
   - B *receives the public key* pk, *the shares of the backup value* $(\mathsf{ab}_i[\mathsf{B}], \mathsf{bc}_i[\mathsf{B}])_{i=0}^r$ *and* $(\mathsf{ac}_i[\mathsf{B}])_{i=1}^r$, *and the signatures* $(\sigma_{i,\mathsf{B}})_{i=1}^r$, $\sigma_{\mathsf{ab},\mathsf{B}}$, *and* $\sigma_{\mathsf{bc},\mathsf{B}}$.
   - C *receives the public key* pk, *the shares of the backup value* $(\mathsf{ac}_i[\mathsf{C}], \mathsf{bc}_i[\mathsf{C}])_{i=0}^r$ *and* $(\mathsf{ab}_i[\mathsf{C}])_{i=1}^r$, *and the signatures* $(\sigma_{i,\mathsf{C}})_{i=1}^r$, $\sigma_{\mathsf{ac},\mathsf{C}}$, *and* $\sigma_{\mathsf{bc},\mathsf{C}}$.

Additionally, for each party P, we let $f_{\text{-}\mathsf{P}}$ denote the two-party functionality between the other two parties, obtained from f by fixing the input of P to a default value (x_0 if $\mathsf{P} = \mathsf{A}$, y_0 if $\mathsf{P} = \mathsf{B}$, and z_0 if $\mathsf{P} = \mathsf{C}$). We consider the following three-party protocol π^{R} for computing f, described in the $\{\mathsf{ShrGen}, f_{\text{-}\mathsf{A}}, f_{\text{-}\mathsf{B}}, f_{\text{-}\mathsf{C}}\}$-hybrid model. By [1, Theorem 4.2] there exists a protocol computing ShrGen with $(1,1)$-FaF security-with-identifiable-abort. Moreover, each $f_{\text{-}\mathsf{P}}$ can be computed with semi-honest security [26]. Thus, by the composition theorem, this implies the existence of a $(1,1)$-FaF secure protocol for computing f in the real world.[9]

Protocol 3.5.
Inputs: *Parties* A, B, *and* C *hold inputs* x, y, *and* z, *respectively.*
Common input: *The parties hold the security parameter* 1^κ.

1. *The parties call* ShrGen *with* $(1,1)$*-FaF security-with-identifiable-abort, with their inputs.*

[9] Technically, the composition theorem in [1] doesn't handle a subprotocol with semi-honest security after an abort occurred. However, we note that since the aborting party receives no messages at all after it aborts, the proof of the composition theorem can be easily extended to our setting.

2. *If* P *aborts the execution, then the remaining two parties call* $f_{\text{-}\mathsf{P}}$ *with their inputs and output the result.*

3. *Otherwise, the parties do the following. For* $i = 1$ *to* r*:*
   (a) *Party* A *broadcasts* $(\mathsf{bc}_i[\mathsf{A}], \sigma_{i,\mathsf{A}})$.
   (b) *If* A *did not send any message or* $\mathsf{Ver}_{\mathsf{pk}}(\mathsf{bc}_i[\mathsf{A}], \sigma_{i,\mathsf{A}}) = \mathsf{Fail}$*, then* B *and* C *reconstruct and output* bc_{i-1}.
   (c) *Otherwise, party* B *broadcasts* $(\mathsf{ac}_i[\mathsf{B}], \sigma_{i,\mathsf{B}})$.
   (d) *If* B *did not send any message or* $\mathsf{Ver}_{\mathsf{pk}}(\mathsf{ac}_i[\mathsf{B}], \sigma_{i,\mathsf{B}}) = \mathsf{Fail}$*, then* A *and* C *reconstruct and output* ac_{i-1}.
   (e) *Otherwise, party* C *broadcasts* $(\mathsf{ab}_i[\mathsf{C}], \sigma_{i,\mathsf{C}})$.
   (f) *If* C *did not send any message or* $\mathsf{Ver}_{\mathsf{pk}}(\mathsf{ab}_i[\mathsf{C}], \sigma_{i,\mathsf{C}}) = \mathsf{Fail}$*, then* A *and* B *reconstruct and output* ab_{i-1}.

4. *If no abort occurred, then*
   - A *broadcasts* $(\mathsf{ab}_r, \sigma_{\mathsf{ab},\mathsf{A}})$ *and* $(\mathsf{ac}_r, \sigma_{\mathsf{ac},\mathsf{A}})$.
   - B *broadcasts* $(\mathsf{ab}_r, \sigma_{\mathsf{ab},\mathsf{B}})$ *and* $(\mathsf{bc}_r, \sigma_{\mathsf{bc},\mathsf{B}})$.
   - C *broadcasts* $(\mathsf{ac}_r, \sigma_{\mathsf{ac},\mathsf{C}})$ *and* $(\mathsf{bc}_r, \sigma_{\mathsf{bc},\mathsf{C}})$.

5. *Since there is at most a single malicious party, each uncorrupted party received 3 shares for at least one of the backup values (one from round* r*, one from the other honest party, and one that they hold). Each party outputs the lexicographically first one.*

First, note that correctness is immediately implied from the correctness of the protocol in the dealer model, stating that $\mathsf{ab}_r = \mathsf{bc}_r = \mathsf{ac}_r$. We next show the security of the protocol. Let $\mathcal{A}^{\mathsf{R}}$ be a malicious adversary corrupting party P in the real world. We assume without loss of generality that $\mathcal{A}^{\mathsf{R}}$ broadcasts its entire view after the protocol is terminated. First, assume that $\mathcal{A}^{\mathsf{R}}$ aborts during the call to ShrGen. Then the output of the uncorrupted parties is determined by $f_{\text{-}\mathsf{P}}$. Thus, the malicious simulator simply sends the default value to the trusted party. Now, fix a semi-honest adversary $\mathcal{A}^{\mathsf{R}}_{\mathsf{Q}}$ corrupting the party $\mathsf{Q} \neq \mathsf{P}$ in the real world. Recall that according to the definition of the ideal world with secure-with-identifiable-abort, $\mathcal{A}^{\mathsf{R}}_{\mathsf{Q}}$ obtains its output alongside the output of $\mathcal{A}^{\mathsf{R}}$. In the case of ShrGen, this amounts to receiving the shares and signatures for all backup values. Aside from the 0^{th} backup value of P and Q, all shares are from a 3-out-of-3 secret sharing scheme. Thus, the semi-honest simulator only needs to generate the 0^{th} backup value of P and Q. By our assumption on the dealer model, this backup value does not depend on the honest party's input. Therefore, the simulator can compute it.

We now assume that $\mathcal{A}^{\mathsf{R}}$ does not abort during the call to ShrGen. We construct an adversary $\mathcal{A}^{\mathsf{D}}$ corrupting P in the real world (Protocol 3.5). In each round, the adversary $\mathcal{A}^{\mathsf{D}}$ will generate independent random shares and signatures, and use them to query $\mathcal{A}^{\mathsf{R}}$ for its message. If $\mathcal{A}^{\mathsf{R}}$ aborts or changes its signed message, then $\mathcal{A}^{\mathsf{R}}$ replies to the dealer with abort, and outputs whatever $\mathcal{A}^{\mathsf{R}}$ outputs. Otherwise, it replies with $\mathsf{continue}$. By the assumed security of π^{D}, there exists a malicious simulator $\mathsf{Sim}^{\mathsf{R}}$ for $\mathcal{A}^{\mathsf{D}}$. We claim that $\mathsf{Sim}^{\mathsf{R}}$ also simulates the real world adversary $\mathcal{A}^{\mathsf{R}}$. Indeed, by the definition of $\mathcal{A}^{\mathsf{D}}$, it generates the entire view of $\mathcal{A}^{\mathsf{R}}$, thus they output identically distributed values.

Next, fix a semi-honest adversary $\mathcal{A}^{\mathsf{R}}_{\mathsf{Q}}$ corrupting a party $\mathsf{Q} \neq \mathsf{P}$ in the real world. Consider the following semi-honest adversary $\mathcal{A}^{\mathsf{D}}_{\mathsf{Q}}$ corrupting the same party in the dealer model. First, it sends to $\mathcal{A}^{\mathsf{R}}_{\mathsf{Q}}$ its entire view, and random signed values to represent shares of backup values that were not yet revealed. $\mathcal{A}^{\mathsf{D}}_{\mathsf{Q}}$ then outputs whatever $\mathcal{A}^{\mathsf{R}}_{\mathsf{Q}}$ outputs. By the assumed FaF security of the dealer model protocol, there is a semi-honest simulator $\mathsf{Sim}^{\mathsf{D}}_{\mathsf{Q}}$ for $\mathcal{A}^{\mathsf{D}}_{\mathsf{Q}}$. Similarly to the malicious case, by the definition of the adversaries in the dealer model, the output of $\mathcal{A}^{\mathsf{D}}_{\mathsf{Q}}$ and $\mathcal{A}^{\mathsf{R}}_{\mathsf{Q}}$ are identical. Thus, $\mathsf{Sim}^{\mathsf{D}}_{\mathsf{Q}}$ also simulates $\mathcal{A}^{\mathsf{R}}_{\mathsf{Q}}$.

We are left with the case where there is no malicious adversary, yet there is a semi-honest adversary (i.e., showing that semi-honest security holds). Security in this case follows from the same argument as in the previous case (where there is a malicious adversary). Fix a semi-honest adversary $\mathcal{A}^{\mathsf{R}}_{\mathsf{Q}}$ corrupting a party Q in the real world. Consider the following semi-honest adversary $\mathcal{A}^{\mathsf{D}}_{\mathsf{Q}}$ corrupting the same party in the dealer model. It sends to $\mathcal{A}^{\mathsf{R}}_{\mathsf{Q}}$ its entire view, and random signed values to represent shares of backup values that were not yet revealed. $\mathcal{A}^{\mathsf{D}}_{\mathsf{Q}}$ then outputs whatever $\mathcal{A}^{\mathsf{R}}_{\mathsf{Q}}$ outputs. By the assumed FaF security of the dealer model protocol, there is a semi-honest simulator $\mathsf{Sim}^{\mathsf{D}}_{\mathsf{Q}}$ for $\mathcal{A}^{\mathsf{D}}_{\mathsf{Q}}$. Similarly to the previous case, the output of $\mathcal{A}^{\mathsf{D}}_{\mathsf{Q}}$ and $\mathcal{A}^{\mathsf{R}}_{\mathsf{Q}}$ are identical. Thus, $\mathsf{Sim}^{\mathsf{D}}_{\mathsf{Q}}$ also simulates $\mathcal{A}^{\mathsf{R}}_{\mathsf{Q}}$. □

It is left to state and prove the second direction.

Lemma 3.6. *Let $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \to \mathcal{W}$ be a three-party functionality. If f can be computed with $(1,1)$-FaF security in the real world, then f can be computed with $(1,1)$-FaF security in the dealer model.*

Proof. Assume there is a $(1,1)$-FaF secure protocol π^{R} computing f in the real world. To construct a protocol in the dealer model, it suffices to describe the distribution of the backup values ab_i, ac_i, and bc_i, for all $i \in \{0, \ldots, r\}$. The dealer computes these values by executing π^{R} in its head, and evaluating the backup values of each pair, as defined in Definition 2.4. That is, it samples random coins for the parties and executes π^{R} in its head to compute the backup values.

We next show that the protocol is secure. Fix a malicious adversary $\mathcal{A}^{\mathsf{D}}$ corrupting a party P in the dealer model. We construct an adversary $\mathcal{A}^{\mathsf{R}}$ that attacks the real world protocol π^{R} by emulating $\mathcal{A}^{\mathsf{D}}$. First, it queries the dealer model adversary $\mathcal{A}^{\mathsf{D}}$ to obtain the input it sends to the dealer, which it then passes to the ShrGen functionality. If $\mathcal{A}^{\mathsf{R}}$ aborts during the call to ShrGen, then send a default value to the trusted party, output whatever $\mathcal{A}^{\mathsf{R}}$ outputs, and halt. Then, in each round, the real world adversary $\mathcal{A}^{\mathsf{R}}$ queries $\mathcal{A}^{\mathsf{D}}$ to receive either $\mathsf{continue}$ or abort. If $\mathcal{A}^{\mathsf{D}}$ sent $\mathsf{continue}$, then $\mathcal{A}^{\mathsf{R}}$ sends the next message as an honest party would. Otherwise, it aborts the protocol. Finally, after the execution of π^{R} has terminated, $\mathcal{A}^{\mathsf{R}}$ sends its entire view to all uncorrupted parties and outputs nothing if $\mathcal{A}^{\mathsf{D}}$ sent abort at some round, and otherwise, outputs whatever an honest party would have. By the assumed security of the real world protocol π^{R}, there exists a simulator $\mathsf{Sim}^{\mathsf{R}}$ for $\mathcal{A}^{\mathsf{R}}$. We define the malicious simulator $\mathsf{Sim}^{\mathsf{D}}$ for the dealer model adversary $\mathcal{A}^{\mathsf{D}}$ to simply send to the trusted party whatever $\mathsf{Sim}^{\mathsf{R}}$ sends, and output whatever $\mathsf{Sim}^{\mathsf{R}}$ outputs. Clearly, the output of the uncorrupted parties in the ideal world interacting with $\mathsf{Sim}^{\mathsf{D}}$ is identical to their output in the ideal world interacting with $\mathsf{Sim}^{\mathsf{R}}$. By the assumed security of π^{R}, this output is indistinguishable from the output of the uncorrupted parties in the real world. By the definition of the dealer and $\mathcal{A}^{\mathsf{D}}$, this output is identically distributed to the output in the dealer model. Thus, the output in the ideal world interacting with $\mathsf{Sim}^{\mathsf{D}}$ is indistinguishable from the output in the dealer model.

We now consider a semi-honest adversary $\mathcal{A}^{\mathsf{D}}_{\mathsf{Q}}$ corrupting a party $\mathsf{Q} \neq \mathsf{P}$ in the dealer model. Consider the semi-honest adversary $\mathcal{A}^{\mathsf{R}}_{\mathsf{Q}}$ corrupting the same party in the real world, which outputs the following. It first computes what an honest Q outputs in the protocol, and additionally outputs all backup values of the pair $\{\mathsf{P}, \mathsf{Q}\}$ until the round in which $\mathcal{A}^{\mathsf{D}}$ aborted (recall that $\mathcal{A}^{\mathsf{D}}$ published its view, hence the backup values can be computed). It then sends it to $\mathcal{A}^{\mathsf{D}}_{\mathsf{Q}}$ and outputs whatever it outputs. By the assumed $(1,1)$-FaF security of π^{R}, there exists a simulator $\mathsf{Sim}^{\mathsf{R}}_{\mathsf{Q}}$ in the ideal world, generating a view for $\mathcal{A}^{\mathsf{R}}_{\mathsf{Q}}$. We define the simulator $\mathsf{Sim}^{\mathsf{D}}_{\mathsf{Q}}$ for the dealer model semi-honest adversary $\mathcal{A}^{\mathsf{D}}_{\mathsf{Q}}$ to output exactly what $\mathsf{Sim}^{\mathsf{R}}_{\mathsf{Q}}$ outputs. Since the two malicious simulators $\mathsf{Sim}^{\mathsf{R}}$ and $\mathsf{Sim}^{\mathsf{R}}_{\mathsf{D}}$ send the same input to the trusted party, the two semi-honest simulators $\mathsf{Sim}^{\mathsf{R}}_{\mathsf{Q}}$ and $\mathsf{Sim}^{\mathsf{D}}_{\mathsf{Q}}$ output identical views. By the assumed security of π^{R}, this view is indistinguishable from the output of $\mathcal{A}^{\mathsf{R}}_{\mathsf{Q}}$. By the definition of the dealer and the real world adversaries, the output of $\mathcal{A}^{\mathsf{R}}_{\mathsf{Q}}$ is identically distributed to the output of $\mathcal{A}^{\mathsf{D}}_{\mathsf{Q}}$. Thus, the ideal world interacting with the two simulators $\mathsf{Sim}^{\mathsf{D}}_{\mathsf{Q}}$ and $\mathsf{Sim}^{\mathsf{D}}$ is indistinguishable from the dealer model.

□

4 Feasibility Results for Three-Party FaF Security

In this section, we present our positive results. In Section 4.1, we show that if a function can be computed by a secure *two-party* protocol, then it can be computed by a three-party $(1,1)$-FaF secure protocol. Then, in Section 4.2 we provide feasibility results for symmetric Boolean functions, where all parties output the same bit as output.

4.1 A Compiler from 2-Party Standard Security to 3-Party FaF-Security

The next theorem states that if a function can be computed as a two-party symmetric functionality (i.e., both parties receive the same output) with security against a single malicious adversary and with security against a single semi-honest adversary (and might be also $(1,1)$-FaF secure), then it can be computed with $(1,1)$-FaF security as a three-party symmetric functionality. Note that simply letting the two parties A and B run the secure protocol between themselves, and then having them send the output to C does not work (since the original protocol might not be $(1,1)$-FaF secure). Furthermore, even if the original two-party protocol is $(1,1)$-FaF secure, a corrupt party can lie about the outcome, and then C has no way of detecting whether A is lying or B is.

Theorem 4.1. *Let $g : \mathcal{X} \times \mathcal{Y} \to \mathcal{W}$ be a symmetric 2-party functionality, and let $f : \mathcal{X} \times \mathcal{Y} \times \{\lambda\} \to \mathcal{W}$ be the 3-party functionality symmetric variant of g, i.e., it is defined as $f(x, y, \lambda) = g(x, y)$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. Suppose that there exists a two-party protocol computing g that is both fully secure and has semi-honest security. Then, assuming secure protocols for OT exist, f can be computed with $(1,1)$-FaF security.*

Proof. Let π_2 be the secure protocol for computing g that is assumed to exist, and let r denote its number of rounds. We construct a three-party protocol π_3 in the dealer model, computing f with $(1,1)$-FaF security. By Theorem 3.2 this implies the existence of a $(1,1)$-FaF secure protocol in the real world (assuming secure protocols for OT). Further let $\mathsf{a}_0, \ldots, \mathsf{a}_r$ and $\mathsf{b}_0, \ldots, \mathsf{b}_r$ denote the backup values of A and B, respectively (obtained by sampling randomness for A and B and simulating them in π_2). We assume without loss of generality that in each round, B is the first to send a message. Thus, A obtains a_i before B obtains b_i. We next construct the three-party

protocol π_3. Recall that a protocol in the dealer model is given by $3r+3$ functions for computing the backup values for each pair of parties in each round. We define these backup values as follows. Given inputs x and y of A and B, respectively, for every $i \in \{0,1,\ldots,r\}$ let $\mathsf{ab}_i = f(x,y,\lambda)$, let $\mathsf{ac}_i = \mathsf{a}_i$, and let $\mathsf{bc}_i = \mathsf{b}_i$. Recall, a_0 is the output of A in π_2 if B sent no message, and thus is independent of y. Similarly, b_0 is independent of x. Thus, the 0^{th} backup value does not depend on the third party's input.

Correctness of π_3 follows from the correctness of π_2, which implies that $\mathsf{a}_r = \mathsf{b}_r = g(x,y) = f(x,y,\lambda)$, except with negligible probability. We next show that π_3 admits $(1,1)$-FaF security against any fail-stop adversary corrupting a single party. Fix an adversary $\mathcal{A}_3$ corrupting a party $\mathsf{P} \in \{\mathsf{A},\mathsf{B},\mathsf{C}\}$. We separate the analysis into cases, depending on the identity of P.

Case 1: $\mathsf{P} \in \{\mathsf{A},\mathsf{B}\}$. We consider only the case where $\mathsf{P} = \mathsf{A}$. The case where $\mathsf{P} = \mathsf{B}$ is similar. Consider the adversary $\mathcal{A}_2$ for the two-party protocol π_2 that corrupts A and behaves the same as $\mathcal{A}_3$. That is, in each round it approaches $\mathcal{A}_3$ to obtain either $\mathsf{continue}$ or abort. If it received $\mathsf{continue}$, then it sends the next message honestly. Otherwise, it aborts and outputs whatever $\mathcal{A}_3$ outputs. By the assumed security of π_2, there exists a simulator Sim_2 for $\mathcal{A}_2$ in the two-party ideal world of g. The simulator Sim_3 for $\mathcal{A}_3$ simply executes Sim_2 to obtain the input x^* it sends to the trusted party (in the two-party ideal world) and sends x^* to T (in the three-party ideal world) to obtain the output w. The simulator outputs nothing if $\mathcal{A}_3$ aborted at some round, and otherwise, it sends w to $\mathcal{A}_3$ and outputs whatever $\mathcal{A}_3$ outputs. Since the distribution of the input x^* that Sim_3 sends to the trusted party is identical to the distribution of the input that Sim_2 sends to its trusted party, the output of B and C in the three-party ideal world is identically distributed to their output in the two-party ideal world. Moreover, since $\mathcal{A}_3$ receives no messages in the dealer model (aside from the output if no abort occurred), it follows that the joint distribution of the view generated by Sim_3 and the output of B and C in the three-party ideal world is identical to the joint distribution of $\mathcal{A}_3$ and the output of B in the three-party real world. Finally, as $\mathcal{A}_2$ aborts at the same round as $\mathcal{A}_3$, it follows that the output of B in π_2 is the same as in π_3.

We next consider a semi-honest adversary $\mathcal{A}_\mathsf{Q}$ corrupting party $\mathsf{Q} \in \{\mathsf{B},\mathsf{C}\}$. Let us first consider the case $\mathsf{Q} = \mathsf{B}$. Let i denote the round in which the malicious adversary aborted, set to $r+1$ if no such round exists. Then the view of B consists of $\mathsf{ab}_0,\ldots,\mathsf{ab}_{i-1}$ and its output $\mathsf{bc}_{i-1} = \mathsf{b}_{i-1}$. Since $\mathsf{ab}_j = f(x,y,\lambda)$ for all j, it follows that the semi-honest simulator (holding both x and y) can simply generate them, send them to $\mathcal{A}_\mathsf{B}$, and output whatever it outputs. Clearly, the joint distribution of the view of B and the output of C in the ideal world is identical to the corresponding distribution in the real world.

Let us now consider a semi-honest C. Similarly to the previous case, we let i denote the round in which the malicious adversary aborted, set to $r+1$ if no such round exists. Then the view of C consists of $\mathsf{ac}_0 = \mathsf{a}_0,\ldots,\mathsf{ac}_{i-1} = \mathsf{a}_{i-1}$ and the output $\mathsf{bc}_{i-1} = \mathsf{b}_{i-1}$. We define its simulator as follows. Run the *malicious* simulator Sim_2 for the two-party setting, with the *same randomness used by* $\mathsf{Sim}_\mathcal{A}$. This generates backup values $\mathsf{a}_0,\ldots,\mathsf{a}_{i-1}$ that, by the security assumption of π_2, are consistent with the output b_{i-1}. That is, the distribution of $\mathsf{a}_0,\ldots,\mathsf{a}_{i-1}$ and b_{i-1} in π_2, is indistinguishable from the view generated by Sim_2 and the output of B in the two-party ideal world. Finally, the semi-honest simulator sends $\mathsf{a}_0,\ldots,\mathsf{a}_{i-1}$ to the semi-honest adversary $\mathcal{A}_\mathsf{C}$ and outputs whatever it outputs. As stated before, the backup values $\mathsf{a}_0,\ldots,\mathsf{a}_{i-1}$ generated by the simulator are consistent with the output $\mathsf{b}_{i-1} = \mathsf{bc}_{i-1}$. Therefore, the real world in the three-party setting is indistinguishable from the ideal world.

Case 2: $\mathsf{P} = \mathsf{C}$. First, observe that regardless of what the adversary does, the output of A and B in both worlds is $f(x, y, \lambda)$. Now, fix a semi-honest adversary $\mathcal{A}_{\mathsf{Q}}$ corrupting party $\mathsf{Q} \neq \mathsf{P}$. By symmetry, we may assume that $\mathsf{Q} = \mathsf{A}$. Let i denote the round where the malicious adversary $\mathcal{A}_3$ aborts, set to $r + 1$ if no abort occurred. Then the view of $\mathcal{A}_{\mathsf{A}}$ is exactly $\mathsf{ac}_0 = \mathsf{a}_0, \ldots, \mathsf{ac}_i = \mathsf{a}_i$. Now, consider a semi-honest adversary $\mathcal{A}_{2,\mathsf{A}}$ corrupting A in the two-party protocol π_2 that outputs $\mathsf{a}_0, \ldots, \mathsf{a}_i$. Then its simulator $\mathsf{Sim}'_{\mathsf{A}}$ (that is assumed to exist) also simulates $\mathcal{A}_{\mathsf{A}}$ in the three-party protocol π_3. □

4.2 FaF Secure Protocols for Boolean Functionalities

In this section, we consider a Boolean three-party functionality that depends only on two inputs. We provide three classes of such functions that can be computed with FaF security. Before stating the theorem, we first introduce some notations.

Notations. For a 2-ary three-party functionality $f : \mathcal{X} \times \mathcal{Y} \times \{\lambda\} \to \{0, 1\}$, we will write $f(x, y)$ instead of $f(x, y, \lambda)$ for brevity. Additionally, we associate a matrix $M_f \in \{0, 1\}^{|\mathcal{X}| \times |\mathcal{Y}|}$, whose rows are indexed by elements $x \in \mathcal{X}$, whose columns are indexed by elements $y \in \mathcal{Y}$, and is defined as $M_f(x, y) = f(x, y)$. We further define the negated matrix $\overline{M}_f$ as $\overline{M}_f(x, y) = 1 - M_f(x, y)$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.

Definition 4.2. *The* affine span *of a collection of vectors over $\mathbb{R}$ is the set of all their linear combinations where the sum of coefficients is exactly 1.*

As a corollary of Theorem 4.1, we apply the characterization from [3] of the 2-party symmetric Boolean functionalities that can be computed with full security. We obtain the following result.

Corollary 4.3. *Let $f : \mathcal{X} \times \mathcal{Y} \times \{\lambda\} \to \{0, 1\}$ be a Boolean 3-party functionality. Suppose that the all-one vector or the all-zero vector is in the affine span of either the rows or the columns of M_f. Then, assuming secure protocols for OT exist, f can be computed with $(1, 1)$-FaF security in the dealer model.*

Proof. Asharov et al. [3] proved that any function satisfying the conditions in the statement can be computed with full security when viewed as a symmetric two-party functionality. We further note that the protocol of [3] for computing such functions admits security against semi-honest adversaries. To see this, recall that their protocol follows the special round paradigm of [14], where before the special round i^* the parties receive random independent value, and after i^* they receive the output.[10] Thus, the view of the semi-honest party can be easily simulated using random values before i^*, and using the output of the trusted party starting from i^*. Therefore, by Theorem 4.1 the function f can be computed as a three-party functionality with $(1, 1)$-FaF security as claimed. □

We next state the main result of this section. We consider a collection of systems of linear equations (that depend on the function f). The theorem roughly states that if any single one of them has a solution, then there exists a FaF secure protocol computing f.

[10] In [3], at round $i^* - 1$ party B receives a constant bit whose value depends only on the function (and not the inputs).

Theorem 4.4. *Let $f : \mathcal{X} \times \mathcal{Y} \times \{\lambda\} \to \{0,1\}$ be a Boolean 3-party functionality. Suppose there exists a probability vector $\mathbf{p} \in \mathbb{R}^{|\mathcal{X}|}$ with no 0 entries, i.e., $\mathbf{p} = (p_x)_{x\in\mathcal{X}}$ satisfies $p_x > 0$ for all $x \in \mathcal{X}$ and $\sum_{x\in\mathcal{X}} p_x = 1$, such that for all $x \in \mathcal{X}$ it holds that $\mathrm{Im}(M_f^T)$ contains the vector*

$$\mathbf{v}_x = \left(M_f(x,y) \cdot \left(\mathbf{p}^T \cdot M_f(\cdot,y)\right)\right)_{y\in\mathcal{Y}}^T,$$

and such that $\mathrm{Im}(\overline{M}_f^T)$ contains the vector

$$\tilde{\mathbf{v}}_x = \left(\overline{M}_f(x,y) \cdot \left(\mathbf{p}^T \cdot \overline{M}_f(\cdot,y)\right)\right)_{y\in\mathcal{Y}}^T.$$

Then, assuming secure protocols for OT exist, f can be computed with $(1,1)$-FaF security in the dealer model.

Proof. We present a protocol that $(1,1)$-FaF securely computes f in the dealer model. By Theorem 3.2 this implies the existence of a $(1,1)$-FaF secure protocol in the real world (assuming secure protocols for OT). The protocol follows the special round paradigm of Gordon et al. [14], where until a special (random and unknown) round i^* the parties' backup values are independent, and from i^* the backup values equal to the output of f. We next present the protocol. Recall that in the dealer model, we may only describe the distribution of the backup values computed by the dealer.

First, we denote the *geometric distribution with parameter* $\alpha > 0$ as $\mathsf{Geom}(\alpha)$, and it is defined as $\Pr_{i\leftarrow\mathsf{Geom}(\alpha)}[i = n] = (1-\alpha)^{n-1} \cdot \alpha$, for all integers $n \geq 1$. We further fix $r(\kappa) = r = \omega(\log\kappa)$ to be the number of rounds. We are now ready to describe the distribution of the backup values, given inputs x and y of A and B, respectively. The dealer samples $i^* \leftarrow \mathsf{Geom}(\alpha)$, where $\alpha > 0$ is sufficiently small that will be chosen below. Then, for every $i \in \{0,\ldots,r\}$, the dealer computes backup values as follows. For every $i \in \{0,\ldots,i^*\}$ sample $\tilde{x}_i \leftarrow \mathbf{p}$ and for every $i \in \{0,\ldots,i^*+1\}$ sample $\tilde{y}_i \leftarrow \mathcal{Y}$ (i.e., $\tilde{y}_i$ is uniformly distributed over $\mathcal{Y}$), independently. Then for every $i \in \{0,\ldots,r\}$ the dealer sets $\mathsf{ab}_i = f(x,y)$ and sets

$$\mathsf{ac}_i = \begin{cases} f(x,\tilde{y}_i) & \text{if } i < i^* \\ f(x,y) & \text{otherwise} \end{cases}; \quad \mathsf{bc}_i = \begin{cases} f(\tilde{x}_i,y) & \text{if } i < i^*+1 \\ f(x,y) & \text{otherwise} \end{cases}$$

The choice of setting bc_i to equal $f(x,y)$ only from round i^*+1 is so that A and C learn the output before B and C. Since $r = \omega(\log\kappa)$ it follows that $i^*+1 \leq r$ except with negligible probability. Therefore $\mathsf{ab}_r = \mathsf{bc}_r = \mathsf{ac}_r = f(x,y)$ except with negligible probability, and thus the protocol is correct.

We now prove that the protocol is $(1,1)$-FaF secure. Fix an adversary $\mathcal{A}$ corrupting a party $\mathsf{P} \in \{\mathsf{A},\mathsf{B},\mathsf{C}\}$. Observe that a malicious C cannot attack the protocol, nor provide A and B with any information they cannot have computed by themselves. Indeed, if C aborts at round i than A and B output $\mathsf{ab}_{i-1} = f(x,y)$. Moreover, a semi-honest A receives $\mathsf{ac}_0,\ldots,\mathsf{ac}_i$, which are either random and independent bits, or equal to $f(x,y)$. A semi-honest B receives similar values. Thus we may assume $\mathsf{P} \neq \mathsf{C}$. We next separate the proof into two cases.

Case 1: $\mathsf{P} = \mathsf{A}$**.** We first construct a simulator $\mathsf{Sim}_{\mathcal{A}}$ for the adversary. Note that since A does not receive messages during the execution of the protocol, as we can let the simulator abort at the

same round as A, we only need to describe the distribution over the inputs that the simulator sends to the trusted party. The simulator we construct is different from other simulators used in similar protocols, where the advantage of the adversary was simulated by sending to the trusted party an input sampled according to a carefully chosen distribution. Instead, our malicious simulator sends an input according to the distribution used in the real world. Then the simulator for the semi-honest party simulates the advantage of the semi-honest party by sampling the backup values differently from the real world. We now describe the malicious simulator.

1. Query $\mathcal{A}$ for the input x it sends to the dealer.
2. Sample $i^* \leftarrow \mathsf{Geom}(\alpha)$.
3. Approach $\mathcal{A}$ in every round to obtain either $\mathsf{continue}$ or abort.
4. If $\mathcal{A}$ aborts at round i, where $i < i^* + 1$, then send $x^* \leftarrow \mathbf{p}$ to the trusted party T.
5. If $\mathcal{A}$ aborts at round i where $i \geq i^* + 1$, or it never aborted, then send $x^* = x$ to T.
6. If $\mathcal{A}$ does not abort, send it the output that T sent to $\mathcal{A}$. Otherwise, send $\mathcal{A}$ nothing. In either case, output whatever $\mathcal{A}$ outputs, and halt.

Next, fix a semi-honest adversary $\mathcal{A}_{\mathsf{Q}}$ corrupting a party $\mathsf{Q} \in \{\mathsf{B}, \mathsf{C}\}$. We now construct a simulator for $\mathcal{A}_{\mathsf{Q}}$. We may only consider the case where $\mathsf{Q} = \mathsf{C}$, since the simulator for a semi-honest B holds both inputs x and y and thus can compute all backup values of A and B, which are all equal to $f(x, y)$. We construct the simulator $\mathsf{Sim}_{\mathcal{A},\mathsf{C}}$ as follows.

1. Obtain the output w from the trusted party T, and receive the input x (sent to the dealer), the randomness, and auxiliary input aux of $\mathsf{Sim}_{\mathcal{A}}$.
2. Compute i^* and x^* using the inputs and randomness of $\mathsf{Sim}_{\mathcal{A}}$. Additionally, let i denote the round where $\mathcal{A}$ aborted (set to $r + 1$ if no such round exists).
3. For every $j \in \{0, \ldots, i-2\}$, sample ac_j the same as in the real world. That is, if $j < i^*$ then $\mathsf{ac}_j = f(x, \tilde{y}_j)$ where $\tilde{y}_j \leftarrow \mathcal{Y}$, and otherwise, set $\mathsf{ac}_j = w$ (note that in this case, $i \geq i^*$ and so $w = f(x, y)$).
4. Sample ac_{i-1} as follows.
   - If $i < i^* + 1$ then set $\mathsf{ac}_{i-1} = 1$ with probability $q_{x,x^*,w}$ to be determined by the analysis below, and set $\mathsf{ac}_{i-1} = 0$ with the remaining probability.
   - Otherwise, if $i \geq i^* + 1$ then set $\mathsf{ac}_{i-1} = w$.
5. Send $\mathsf{ac}_0, \ldots, \mathsf{ac}_{i-1}$ to the semi-honest adversary, output whatever it outputs, and halt.

Before proving that the simulator is successful, we introduce additional notations.

Notations. For $n \in \mathbb{N}$ we let $\mathbf{1}_n$ and $\mathbf{0}_n$ denote the all-one and the all-zero vectors, respectively, of dimension n. When n is clear from context, we will remove it for brevity. For a matrix $M \in \mathbb{R}^{n \times m}$ and a row $i \in [n]$, we let $M(i, \cdot)$ denote the row vector $(M(i,j))_{j \in [m]}$. Similarly, for a column $j \in [m]$, we let $M(\cdot, j)$ denote the column vector $(M(i,j))_{i \in [n]}$.

We now turn to analyzing the simulator. Let i denote the round where $\mathcal{A}$ aborted (set to $r+1$ if no abort occurred). Then in both worlds, the backup values given to the semi-honest adversary $\mathcal{A}_{\mathsf{C}}$ are $\mathsf{ac}_0, \ldots, \mathsf{ac}_{i-1}$. Thus, it suffices to compare $(\mathsf{ac}_0, \ldots, \mathsf{ac}_{i-1}, \text{OUT})$ in both worlds, where OUT denotes the output of the honest parties. First, note that $\mathsf{ac}_0, \ldots, \mathsf{ac}_{i-2}$ are identically distributed in both worlds. Moreover, ac_{i-1} and OUT can be computed given i^* without using $\mathsf{ac}_0, \ldots, \mathsf{ac}_{i-2}$. Thus, we may only compare the distribution of $(\mathsf{ac}_{i-1}, \text{OUT})$ in both worlds. Next, observe that if $i > i^* + 1$ then $\mathsf{ac}_{i-1} = \text{OUT} = f(x, y)$ in both worlds, hence we may condition on $i \leq i^* + 1$.

Let us first compute the probability of the event $(\mathsf{ac}_{i-1}, \text{OUT}) = (1,1)$. Observe that in the real world, it holds that $\text{OUT} = \mathsf{bc}_{i-1}$. Additionally, since we condition on the event $i \leq i^* + 1$, it holds that $\mathsf{bc}_{i-1} = f(\tilde{x}_{i-1}, y)$ where $\tilde{x}_{i-1} \leftarrow \mathcal{X}$. Moreover, if $i = i^* + 1$ then $\mathsf{ac}_{i-1} = \mathsf{ac}_{i^*} = f(x, y)$, and if $i < i^* + 1$ then $\mathsf{ac}_{i-1} = f(x, \tilde{y}_{i-1})$ where $\tilde{y}_{i-1} \leftarrow \mathcal{Y}$. Therefore, in the real world, the event $(\mathsf{ac}_{i-1}, \text{OUT}) = (1,1)$ occurs with probability

$$\begin{aligned}
&\Pr\left[(\mathsf{ac}_{i-1}, \text{OUT}) = (1,1) \mid i \leq i^* + 1\right] \\
&\quad = \Pr\left[i < i^* + 1 \mid i \leq i^* + 1\right] \cdot \Pr\left[f(x, \tilde{y}) = 1\right] \cdot \Pr\left[f(\tilde{x}, y) = 1\right] \\
&\qquad + \Pr\left[i = i^* + 1 \mid i \leq i^* + 1\right] \cdot f(x, y) \cdot \Pr\left[f(\tilde{x}, y) = 1\right] \\
&\quad = (1 - \alpha) \cdot \Pr\left[f(x, \tilde{y}) = 1\right] \cdot \Pr\left[f(\tilde{x}, y) = 1\right] + \alpha \cdot f(x, y) \cdot \Pr\left[f(\tilde{x}, y) = 1\right],
\end{aligned}$$

where $\tilde{x} \leftarrow \mathbf{p}$ and $\tilde{y} \leftarrow \mathcal{Y}$. On the other hand, in the ideal world $\text{OUT} = f(x^*, y)$ where x^* is chosen with probability p_{x^*}. Moreover, by the definition of the simulator, the probability that $\mathsf{ac}_{i-1} = 1$ given x^* such that $f(x^*, y) = 1$ is $q_{x,x^*,1}$. Thus $(\mathsf{ac}_{i-1}, \text{OUT}) = (1,1)$ occurs with probability

$$\sum_{x^* \in \mathcal{X} : f(x^*, y) = 1} p_{x^*} \cdot q_{x,x^*,1} = \sum_{x^* \in \mathcal{X}} p_{x^*} \cdot q_{x,x^*,1} \cdot f(x^*, y).$$

To show that the simulator is successful, it suffices to show that for a sufficiently small $\alpha > 0$, there exists $q_{x,x^*,1}$ such that the two expressions are equal for all $y \in \mathcal{Y}$. Let $s_x = \Pr\left[f(x, \tilde{y}) = 1\right]$ and let $\mathbf{x}_{x,1} = (p_{x^*} \cdot q_{x,x^*,1})_{x^* \in \mathcal{X}}$. Then the equality of the two expressions may be written as

$$\mathbf{x}_{x,1}^T \cdot M_f(\cdot, y) = (1 - \alpha) \cdot s_x \cdot \left(\mathbf{p}^T \cdot M_f(\cdot, y)\right) + \alpha M_f(x, y) \cdot \left(\mathbf{p}^T \cdot M_f(\cdot, y)\right). \tag{1}$$

Now, since we assume $p_{x^*} > 0$ for all $x^* \in \mathcal{X}$, we can take $q_{x,x^*,1} = \mathbf{x}_{x,1}(x^*)/p_{x^*}$. This value is between 0 and 1 if and only if $0 \leq \mathbf{x}_{x,1}(x^*) \leq p_{x^*}$. Therefore, if a solution $\mathbf{x}_{x,1}$ exists, such that $0 \leq \mathbf{x}_{x,1}(x^*) \leq p_{x^*}$ for all $x^* \in \mathcal{X}$, then this immediately implies the existence of $q_{x,x^*,1}$ (and hence the existence of a simulator). We now show that such a solution exists. First, recall that in the statement of the lemma, $\mathrm{Im}(M_f^T)$ contains

$$\mathbf{v}_x = \left(M_f(x, y) \cdot \left(\mathbf{p}^T \cdot M_f(\cdot, y)\right)\right)_{y \in \mathcal{Y}}^T.$$

Then, as Equation (1) should hold for all $y \in \mathcal{Y}$, it is equivalent to

$$\mathbf{x}_{x,1}^T \cdot M_f = (1 - \alpha) \cdot s_x \cdot \left(\mathbf{p}^T \cdot M_f\right) + \alpha \mathbf{v}_x^T, \tag{2}$$

i.e., $(\mathbf{x}_{x,1} - (1-\alpha)\cdot s_x \cdot \mathbf{p})^T \cdot M_f = \alpha \mathbf{v}_x^T$. Next, define the vector

$$\widetilde{\mathbf{x}}_{x,1} = \frac{\mathbf{x}_{x,1} - (1-\alpha)\cdot s_x \cdot \mathbf{p}}{\alpha}.$$

Then a solution to Equation (2) exists if and only if there exists a solution $\widetilde{\mathbf{x}}_{x,1}$ to the system of equations

$$\widetilde{\mathbf{x}}_{x,1}^T \cdot M_f = \mathbf{v}_x^T.$$

As we assume that $\mathbf{v}_x \in \mathrm{Im}(M_f^T)$, a solution exists. However, recall that the solution must be so that $0 \le \mathbf{x}_{x,1}(x^*) \le p_{x^*}$ for all $x^* \in \mathcal{X}$. This holds if and only if for all $x^* \in \mathcal{X}$ it holds that

$$-\frac{(1-\alpha)\cdot s_x \cdot p_{x^*}}{\alpha} \le \widetilde{\mathbf{x}}_{x,1}(x^*) \le \frac{p_{x^*} - (1-\alpha)\cdot s_x \cdot p_{x^*}}{\alpha}.$$

Since $(1-\alpha)\cdot s_x \cdot p_{x^*} \le p_{x^*}$, the above interval grows arbitrarily large in both directions, as α tends to 0. Thus, taking a sufficiently small $\alpha > 0$ satisfies the constraints.

Let us now compute the probability of the event $(\mathsf{ac}_{i-1}, \textsc{out}) = (0,1)$. In the real world, this occurs with probability

$$(1-\alpha)\cdot(1-\Pr[f(x,\tilde{y})=1])\cdot\Pr[f(\tilde{x},y)=1] + \alpha\cdot(1-f(x,y))\cdot\Pr[f(\tilde{x},y)=1], \tag{3}$$

where $\tilde{x} \leftarrow \mathbf{p}$ and $\tilde{y} \leftarrow \mathcal{Y}$. On the other hand, in the ideal world this occurs with probability

$$\sum_{x^*\in\mathcal{X}: f(x^*,y)=1} p_{x^*}\cdot(1-q_{x,x^*,1}).$$

Now, recall that in the previous case, where we considered the probability that $(\mathsf{ac}_{i-1}, \textsc{out}) = (1,1)$, we showed that

$$(1-\alpha)\cdot\Pr[f(x,\tilde{y})=1]\cdot\Pr[f(\tilde{x},y)=1] + \alpha\cdot f(x,y)\cdot\Pr[f(\tilde{x},y)=1] = \sum_{x^*\in\mathcal{X}: f(x^*,y)=1} p_{x^*}\cdot q_{x,x^*,1}.$$

Plugging this into Equation (3) we obtain

$$\begin{aligned}
&(1-\alpha)\cdot(1-\Pr[f(x,\tilde{y})=1])\cdot\Pr[f(\tilde{x},y)=1] + \alpha\cdot(1-f(x,y))\cdot\Pr[f(\tilde{x},y)=1] \\
&\quad= (1-\alpha)\Pr[f(\tilde{x},y)=1] + \alpha\Pr[f(\tilde{x},y)=1] \\
&\qquad - \Big((1-\alpha)\cdot\Pr[f(x,\tilde{y})=1]\cdot\Pr[f(\tilde{x},y)=1] + \alpha\cdot f(x,y)\cdot\Pr[f(\tilde{x},y)=1]\Big) \\
&\quad= (1-\alpha)\Pr[f(\tilde{x},y)=1] + \alpha\Pr[f(\tilde{x},y)=1] - \sum_{x^*\in\mathcal{X}: f(x^*,y)=1} p_{x^*}\cdot q_{x,x^*,1} \\
&\quad= \Pr[f(\tilde{x},y)=1] - \sum_{x^*\in\mathcal{X}: f(x^*,y)=1} p_{x^*}\cdot q_{x,x^*,1} \\
&\quad= \sum_{x^*\in\mathcal{X}: f(x^*,y)=1} (p_{x^*} - p_{x^*}\cdot q_{x,x^*,1}) \\
&\quad= \sum_{x^*\in\mathcal{X}: f(x^*,y)=1} p_{x^*}(1-q_{x,x^*,1}),
\end{aligned}$$

which is the probability of the event occurring in the ideal world.

Finally, as the probabilities of all four cases sum to 1, it suffices to compute the probability of the event $(\mathsf{ac}_{i-1}, \text{OUT}) = (1, 0)$. In the real world, this occurs with probability

$$(1-\alpha) \cdot \Pr\left[f(x, \tilde{y}) = 1\right] \cdot (1 - \Pr\left[f(\tilde{x}, y) = 1\right]) + \alpha \cdot f(x, y) \cdot (1 - \Pr\left[f(\tilde{x}, y) = 1\right]),$$

where $\tilde{x} \leftarrow \mathbf{p}$ and $\tilde{y} \leftarrow \mathcal{Y}$. On the other hand, in the ideal world this occurs with probability

$$\sum_{x^* \in \mathcal{X} : f(x^*, y) = 0} p_{x^*} \cdot q_{x, x^*, 0}.$$

Setting $\mathbf{x}_{x,0} = (p_{x^*} \cdot q_{x,x^*,0})_{x^* \in \mathcal{X}}$, yields that equality of the two expressions may be written as

$$\mathbf{x}_{x,0}^T \cdot \overline{M}_f(\cdot, y) = (1-\alpha) s_x \cdot \left(\mathbf{p}^T \cdot \overline{M}_f(\cdot, y)\right) + \alpha \overline{M}_f(x, y) \cdot \left(\mathbf{p}^T \cdot \overline{M}_f(\cdot, y)\right).$$

A similar analysis to the first case shows that the assumption $\tilde{\mathbf{v}}_x \in \text{Im}(\overline{M}_f^T)$ implies that for any sufficiently small $\alpha > 0$ there exists $q_{x,x^*,0}$ such that the two expressions are equal.

Case 2: $\mathsf{P} = \mathsf{B}$. Intuitively, this is a simpler case than the previous one, since A and C obtain the output before B and C, hence there is no advantage to simulate. Indeed, if B aborts at round $i \geq i^* + 1$ then, although C receives $\mathsf{bc}_i = f(x, y)$, it holds that A and C output $\mathsf{ac}_{i-1} = f(x, y)$. On the other hand, if B abort at round $i \leq i^*$, then both bc_i and ac_{i-1} are random and independent. We next formalize the above intuition. We construct a simulator $\mathsf{Sim}_{\mathcal{A}}$ for the adversary as follows.

1. Query $\mathcal{A}$ for the input y it sends to the dealer.
2. Sample $i^* \leftarrow \mathsf{Geom}(\alpha)$.
3. If $\mathcal{A}$ aborts at round i, where $i < i^* + 1$, then send $y^* \leftarrow \mathcal{Y}$ to the trusted party T.
4. If $\mathcal{A}$ aborts at round i where $i \geq i^* + 1$ or it never aborted, then send $y^* = y$ to T.
5. If $\mathcal{A}$ did not abort, send it the output that T sent to $\mathcal{A}$. Otherwise, send $\mathcal{A}$ nothing. In either case, output whatever $\mathcal{A}$ outputs, and halt.

Next, fix a semi-honest adversary $\mathcal{A}_{\mathsf{Q}}$ corrupting a party $\mathsf{Q} \in \{\mathsf{A}, \mathsf{C}\}$. We now construct a simulator for $\mathcal{A}_{\mathsf{Q}}$. Similarly to the previous case, we may only consider the case $\mathsf{Q} = \mathsf{C}$. We construct the simulator $\mathsf{Sim}_{\mathcal{A},\mathsf{C}}$ as follows.

1. Obtain the output w from the trusted party T, and receive the input x (sent to the dealer), the randomness, and auxiliary input aux of $\mathsf{Sim}_{\mathcal{A}}$.
2. Compute i^* and x^* using the inputs and randomness of $\mathsf{Sim}_{\mathcal{A}}$. Additionally, let i denote the round where $\mathcal{A}$ aborted (set to r if no such round exists).
3. For every $j \in \{0, \ldots, i\}$, sample bc_j the same as in the real world. That is, if $j < i^* + 1$ then $\mathsf{bc}_j = f(\tilde{x}_j, y)$ where $\tilde{x}_j \leftarrow \mathbf{p}$, and otherwise, set $\mathsf{bc}_j = w$ (note that in this case, $i \geq i^* + 1$ and so $w = f(x, y)$).
4. Send $\mathsf{bc}_0, \ldots, \mathsf{bc}_i$ to the semi-honest adversary, output whatever it outputs, and halt.

By construction, as noted earlier, in the real world the joint view of the semi-honest adversary is $\mathsf{bc}_0, \ldots, \mathsf{bc}_i$, and the output of the honest party is ac_{i-1} (set to r if no such round exists). By construction, the same holds in the ideal world. Therefore, they are identically distributed. □

Theorem 4.4 identifies a set of functionalities that can be computed with $(1,1)$-FaF security. We do not know if there are functionalities that are not captured by Theorem 4.4, and we leave their existence as an open question. Corollary 4.6 below provides two simple classes of functionalities captured by Theorem 4.4.

The following lemma, states that for certain families of functionalities, there exists a solution to one of the system of equations considered in Theorem 4.4.

Lemma 4.5. *Let $f : \mathcal{X} \times \mathcal{Y} \times \{\lambda\} \to \{0,1\}$ be a three-party 2-ary Boolean functionality. Suppose that one of the following holds.*

1. *Both M_f and $\overline{M}_f$ have a trivial kernel.*
2. *The all-one vector is a linear combination of the rows of M_f, where all coefficients are strictly positive.*

Then there exists a probability vector $\mathbf{p} \in \mathbb{R}^{|\mathcal{X}|}$ with no 0 entries, such that for all $x \in \mathcal{X}$ it holds that $\mathrm{Im}(M_f^T)$ contains the vector

$$\mathbf{v}_x = \left(M_f(x,y) \cdot \left(\mathbf{p}^T \cdot M_f(\cdot,y)\right)\right)^T_{y \in \mathcal{Y}}$$

and $\mathrm{Im}(\overline{M}_f^T)$ contains the vector

$$\widetilde{\mathbf{v}}_x = \left(\overline{M}_f(x,y) \cdot \left(\mathbf{p}^T \cdot \overline{M}_f(\cdot,y)\right)\right)^T_{y \in \mathcal{Y}}.$$

Proof. Let us first assume that both M_f and $\overline{M}_f$ have a trivial kernel. Here, any choice of $\mathbf{p}$ with no zero entries works (e.g., the uniform probability vector). Indeed, $\mathbf{v}_x \in \mathrm{Im}(M_f^T)$ if and only if it is orthogonal to the kernel of M. By assumption, $\ker(M_f) = \{\mathbf{0}\}$ hence any vector is orthogonal to it. Similarly, $\widetilde{\mathbf{v}}_x \in \mathrm{Im}(\overline{M}_f^T)$.

We now assume there exists a vector $\mathbf{u} \in \mathbb{R}^{|\mathcal{X}|}$ with strictly positive entries, such that $\mathbf{u}^T \cdot M_f = \mathbf{1}^T$. Here we take $\mathbf{p} = \mathbf{u}/||\mathbf{u}||_1$, where $||\mathbf{u}||_1 = \sum_{x \in \mathcal{X}} u_x$ is the ℓ_1 norm of $\mathbf{u}$. Let $\delta > 0$ be such that

$$\mathbf{p}^T \cdot M_f = \delta \cdot \mathbf{1}^T. \tag{4}$$

Then

$$\mathbf{v}_x = \left(M_f(x,y) \cdot \left(\mathbf{p}^T \cdot M_f(\cdot,y)\right)\right)_{y \in \mathcal{Y}} = \left(M_f(x,y) \cdot \delta\right)_{y \in \mathcal{Y}} = (\delta \cdot \mathbf{e}_x) \cdot M_f,$$

where $\mathbf{e}_x$ is the x^{th} standard basis vector. Thus, $\mathbf{v}_x \in \mathrm{Im}(M_f^T)$. It is left to show that $\widetilde{\mathbf{v}}_x \in \mathrm{Im}(\overline{M}_f^T)$. We assume that M_f is not the all-one matrix, as otherwise, the claim is trivial since $\widetilde{\mathbf{v}}_x = \mathbf{0}$ and $\overline{M}_f$ is the all-zero matrix. Let J denote the $|\mathcal{X}| \times |\mathcal{Y}|$ all-one matrix. Observe that by Equation (4) and since $\mathbf{p}$ is a probability vector

$$\mathbf{p}^T \cdot \overline{M}_f = \mathbf{p}^T \cdot (J - M_f) = \mathbf{p}^T \cdot J - \delta \cdot \mathbf{1}^T_{|\mathcal{Y}|} = (\mathbf{p}^T \cdot \mathbf{1}_{|\mathcal{X}|} - \delta) \cdot \mathbf{1}^T_{|\mathcal{Y}|} = (1 - \delta) \cdot \mathbf{1}^T.$$

Since M_f is Boolean and $\mathbf{p}$ is a probability vector, for every $y \in \mathcal{Y}$ it follows that

$$\delta = \mathbf{p}^T \cdot M(\cdot, y) \leq \mathbf{p}^T \cdot \mathbf{1} = 1,$$

with equality if and only if for every $x \in \mathcal{X}$ such that $p_x > 0$ it holds that $M(x, y) = 1$. Since $\mathbf{p}$ has no zero entries and M is not the all-one matrix, we conclude that the inequality is strict; i.e., $\delta < 1$. Therefore, a similar analysis to the previous case shows that

$$\widetilde{\mathbf{v}}_x^T = ((1 - \delta) \cdot \mathbf{e}_x) \cdot \overline{M}_f.$$

□

Note that if M_f^T satisfies the conditions in Lemma 4.5, then a secure protocol can be obtained by switching the roles of A and B. Thus, we obtain the following corollary. Although less general than Theorem 4.4, it is conceptually simpler.

Corollary 4.6. *Let $f : \mathcal{X} \times \mathcal{Y} \times \{\lambda\} \to \{0, 1\}$ be a three-party 2-ary Boolean functionality. Suppose that one of the following holds.*

1. *Both M_f and $\overline{M}_f$ have a trivial kernel, or both M_f^T and $\overline{M}_f^T$ have a trivial kernel, i.e., it contains only the all-zero vector.*
2. *The all-one vector is a linear combination of either the rows or columns of M_f, where all coefficients are strictly positive.*

Then, assuming secure protocols for OT exist, f can be computed with $(1, 1)$-FaF security in the dealer model.

Proof. If f satisfies Item 1 then either both M_f and $\overline{M}_f$ have a trivial kernel, or both M_f^T and $\overline{M}_f^T$ have a trivial kernel. In the former case, a secure protocol exists by applying both Theorem 4.4 and Lemma 4.5. In the latter case, we switch the roles of A and B. Thus, the matrix associated with the new function is M_f^T, and so we can again apply Theorem 4.4 and Lemma 4.5. The second case where f satisfies Item 2 is handled similarly to the previous case. □

As an example of Corollary 4.6, consider the equality function $\mathsf{EQ}_m : [m]^2 \times \{\lambda\} \to \{0, 1\}$, where $m \geq 1$ is an integer. It is defined as

$$\mathsf{EQ}_m(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}.$$

Then, M_{EQ_m} is the $m \times m$ identity matrix. Therefore, it satisfies Item 1 of Corollary 4.6, hence it can be computed with $(1, 1)$-FaF security. To exemplify Item 2, consider the functionality f given by the following matrix

$$M_f = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Observe that the kernel of both M_f and M_f^T contain $(1, 1, -1, -1)^T$, hence Item 1 does not hold for f. However, note that

$$M_f \cdot (1/4, 1/4, 1/4, 1/4)^T = (1/2, 1/2, 1/2, 1/2)^T.$$

Therefore f satisfies Item 2, hence it can be computed with $(1, 1)$-FaF security.

Remark 4.7. *Although only proved for deterministic functionalities, Corollary 4.6 (and the more general Theorem 4.4) can be easily generalized to randomized functionalities by defining* $M_f(x,y) = \Pr[f(x,y)=1]$ *for all* $x \in \mathcal{X}$ *and* $y \in \mathcal{Y}$.

5 Lower Bound on the Round Complexity of FaF Secure Protocols

In this section, we present a lower bound on the round complexity required for certain FaF secure computations. Specifically, we focus on deterministic three-party functionalities that depend on two inputs. Before stating the result, we first define the notion of *maximally informative input*. Roughly, an input $x \in \mathcal{X}$ for party A is said to be maximally informative if for any other input x' the input-output pair $(x', f(x',y))$ does not give to A more information about the input y of B than the input-output pair $(x, f(x,y))$. We formalize this by requiring that for any x' there exists $y_0, y_1 \in \mathcal{Y}$ such that the input x can distinguish y_0 from y_1, while x' cannot distinguish them. Formally we define it as follows.

Definition 5.1 (Maximally informative input)**.** *Let* $f : \mathcal{X} \times \mathcal{Y} \times \{\lambda\} \to \mathcal{W}$ *be a deterministic three-party functionality. We say that an input* $x \in \mathcal{X}$ *is* maximally informative *if for every* $x' \in \mathcal{X} \setminus \{x\}$ *there exists* $y_0, y_1 \in \mathcal{Y}$ *such that* $f(x,y_0) \neq f(x,y_1)$ *and* $f(x',y_0) = f(x',y_1)$. *A maximally informative input* $y \in \mathcal{Y}$ *is defined analogously.*

We are now ready to state our theorem. Roughly, it states that for any deterministic 2-ary functionalities, if all inputs do not fix the output and are maximally informative, then for any ε, the function cannot be computed with an $O(\frac{\log \varepsilon^{-1}}{\log|\mathcal{X}|+\log|\mathcal{Y}|})$-round FaF secure protocol.

Theorem 5.2. *Let* $f : \mathcal{X} \times \mathcal{Y} \times \{\lambda\} \to \mathcal{W}$ *be a deterministic three-party functionality. For every* $x \in \mathcal{X}$ *let* $p_x := \max_{w \in \mathcal{W}} \Pr[f(x,y)=w]$ *where* $y \leftarrow \mathcal{Y}$, *and let* $p_1 := \max_{x \in \mathcal{X}} p_x$. *Similarly, for every* $y \in \mathcal{Y}$ *let* $p_y := \max_{w \in \mathcal{W}} \Pr[f(x,y)=w]$ *where* $x \leftarrow \mathcal{X}$, *and let* $p_2 := \max_{y \in \mathcal{Y}} p_y$. *Finally, denote* $p = \max\{p_1, p_2\}$. *Assume that there is no input that fixes the output of* f *and that all* $x \in \mathcal{X}$ *and* $y \in \mathcal{Y}$ *are maximally informative (observe that this implies that* $p < 1$*). Then for any* $\varepsilon = \varepsilon(\kappa)$ *and any* r*-round protocol* π *computing* f *with* $(1,1)$*-FaF* ε*-security, it holds that*

$$r \geq \frac{\log\left(\frac{1}{4\varepsilon}\right) - \log\left(\frac{1}{1-p}\right)}{\log(9 \cdot |\mathcal{X}| \cdot |\mathcal{Y}|)}.$$

The proof is given below. As a corollary, we get that for any f satisfying the conditions in Theorem 5.2 there is no $O(\log \kappa)$-round protocol computing f with $(1,1)$-FaF security.

Corollary 5.3. *Let* $f : \mathcal{X} \times \mathcal{Y} \times \{\lambda\} \to \mathcal{W}$ *be a deterministic three-party functionality. Assume that there is no input that fixes the output of* f *and that all* $x \in \mathcal{X}$ *and* $y \in \mathcal{Y}$ *are maximally informative. Then there is no* $O(\log \kappa)$*-round protocol computing* f *with* $(1,1)$*-FaF security.*

Proof. Fix a constant $c \in \mathbb{N}$ and let $\varepsilon(\kappa) = \kappa^{-c'}$, where $c' = 2c \cdot \log(9 \cdot |\mathcal{X}| \cdot |\mathcal{Y}|)$. Since p, $|\mathcal{X}|$, and $|\mathcal{Y}|$ are constant, it holds that

$$c' \geq \frac{c \cdot \log\kappa \cdot \log(9 \cdot |\mathcal{X}| \cdot |\mathcal{Y}|) + \log\left(\frac{1}{1-p}\right)}{\log\left(\frac{\kappa}{4}\right)},$$

for all sufficiently large κ. By Theorem 5.2 it follows that $r \geq c \cdot \log\kappa$. □

Below we show an example of a Boolean functionality that can be computed with $(1,1)$-FaF security and satisfies the conditions of Theorem 5.2.

For Boolean functions, the result can be stated in simpler terms using the associated matrix M_f of the function. Observe that an input $x \in \mathcal{X}$ is maximally informative if and only if the row $M_f(x,\cdot)$ is either constant or the negation of the row, namely $\overline{M}_f(x,\cdot)$, does not appear in M_f. Additionally, note that duplicating rows and columns, and removing duplications does not compromise the FaF security of the protocol. Thus, we have the following corollary.

Corollary 5.4. *Let $f : \mathcal{X} \times \mathcal{Y} \times \{\lambda\} \to \{0,1\}$ be a deterministic three-party Boolean functionality. Assuming that the matrix M_f has no constant rows, no constant columns, and that no row or column has its negation appearing in M_f. Then there is no $O(\log \kappa)$-round protocol computing f with $(1,1)$-FaF security.*

As an example, for an integer $m \geq 3$, consider the equality function $\mathsf{EQ}_m : [m]^2 \times \{\lambda\} \to \{0,1\}$ defined as $\mathsf{EQ}_m(x,y) = 1$ if $x = y$, and $\mathsf{EQ}_m(x,y) = 0$ otherwise. Then M_{EQ_m} is the $m \times m$ identity matrix. It has no constant rows and columns, and since $m \geq 3$ no row or column has its negation appearing in M_{EQ_m}. Therefore, by Corollary 5.4 any protocol computing it must have round complexity of $\omega(\log \kappa)$. Note that this matches the round complexity of the protocol given by Corollary 4.6.

Towards proving Theorem 5.2, we first prove the following lemma, stating that in a FaF secure protocol, if a semi-honest C learns an output $f(x,y)$ with high probability and x and y are maximally informative, then with high probability the malicious simulator (of either a malicious A or a malicious B) must send to the trusted party the input given to the party (i.e., x or y depending on the identity of the corrupted party). In fact, we prove a more general result, which holds even for inputs that are not maximally informative. We first define a way to compare the information that two inputs can give. Intuitively, we say that x' is more informative than x if for any two inputs y_0 and y_1 of B, if x can distinguish them, then so can x'.

Definition 5.5. *Let $f : \mathcal{X} \times \mathcal{Y} \times \{\lambda\} \to \mathcal{W}$ be a deterministic three-party functionality. For two inputs $x, x' \in \mathcal{X}$, we say that x' is* more informative *than x, denoted $x \preceq x'$, if the following holds. For any $y_0, y_1 \in \mathcal{Y}$, if $f(x,y_0) \neq f(x,y_1)$ then $f(x',y_0) \neq f(x',y_1)$. We define the notion over the inputs of B analogously.*

We next state the lemma.

Lemma 5.6. *Let $f : \mathcal{X} \times \mathcal{Y} \times \{\lambda\} \to \mathcal{W}$ be a deterministic three-party functionality, and suppose that there exists a protocol π computing it with $(1,1)$-FaF ε-security, where $\varepsilon = \varepsilon(\kappa)$. Let $\mu > 0$ and fix a malicious adversary $\mathcal{A}$ corrupting A, for which there exists some algorithm M such that for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, when applied to the view of C (i.e., the backup values it receives from the dealer), it outputs $f(x,y)$ except with probability at most μ. Then for any malicious simulator $\mathsf{Sim}_{\mathcal{A}}$ for $\mathcal{A}$ corrupting A in the ideal world, it holds that the input x^* it sends to the trusted party is more informative than x, i.e., $x \preceq x^*$, except with probability at most $2|\mathcal{X}| \cdot (\mu + \varepsilon)$. An analogous statement can be made for a malicious B.*

Proof. Intuitively, if the simulator sends x' such that $x \not\preceq x'$, then there exist $y_0, y_1 \in \mathcal{Y}$ such that $f(x,y_0) \neq f(x,y_1)$ and $f(x',y_0) = f(x',y_1)$. Therefore, on the one hand, the real-world adversary can distinguish the case where B holds y_0 from the case it holds y_1. On the other hand, the simulator

will behave the same regardless of whether B has y_0 or y_1 as input, thus reaching a contradiction. We next formalize the above intuition.

Assume towards contradiction there exists such an adversary $\mathcal{A}$ and an input $x \in \mathcal{X}$, for which its simulator sends x^* such that $x \not\preceq x^*$ with probability at least $2|\mathcal{X}| \cdot (\mu + \varepsilon)$, infinitely often. We show that the simulator $\mathsf{Sim}_{\mathcal{A},\mathsf{C}}$ for the semi-honest party C cannot simulate its view with less than ε advantage. Let $P_{\kappa,x}$ denote the distribution from which x^* is sampled given that A holds input x and the security parameter is κ. To obtain a contradiction, we show that there exists $y \in \mathcal{Y}$ for which

$$\Pr_{x^* \leftarrow P_{\kappa,x}} \left[\mathsf{M}\left(\mathsf{Sim}_{\mathcal{A},\mathsf{C}}\left(1^\kappa, x, x^*, f(x^*, y)\right)\right) \neq f(x, y)\right] \geq \mu + \varepsilon, \tag{5}$$

infinitely often, where the probability is over x^* and the random coins of $\mathsf{Sim}_{\mathcal{A},\mathsf{C}}$ and M, when B holds the input y. Indeed, if Equation (5) holds then the real and ideal worlds can be distinguished with ε advantage by applying M to the view of C. For brevity, in the following, we define the (randomized) algorithm $\mathsf{S}(x, x^*, w) = \mathsf{M}\left(\mathsf{Sim}_{\mathcal{A},\mathsf{C}}\left(1^\kappa, x, x^*, w\right)\right)$.

By assumption,

$$\Pr_{x^* \leftarrow P_{\kappa,x}} \left[x \not\preceq x^*\right] \geq 2|\mathcal{X}| \cdot (\mu + \varepsilon)$$

infinitely often. Therefore, by the union bound it follows that for any such κ there exists $x' = x'_\kappa \in \mathcal{X}$, where $x \not\preceq x'$, such that

$$\Pr_{x^* \leftarrow P_{\kappa,x}} \left[x^* = x'\right] \geq 2(\mu + \varepsilon).$$

Then for every $y \in \mathcal{Y}$, it holds that

$$\begin{aligned}
&\Pr_{x^* \leftarrow P_{\kappa,x}} \left[\mathsf{S}\left(x, x^*, f(x^*, y)\right) \neq f(x, y)\right] \\
&\quad \geq \Pr_{x^* \leftarrow P_{\kappa,x}} \left[\mathsf{S}\left(x, x^*, f(x^*, y)\right) \neq f(x, y) \mid x^* = x'\right] \cdot 2\left(\mu + \varepsilon\right) \\
&\quad = \Pr\left[\mathsf{S}\left(x, x', f(x', y)\right) \neq f(x, y)\right] \cdot 2\left(\mu + \varepsilon\right)
\end{aligned}$$

infinitely often, where all probabilities are also taken over the random coins of S. To show Equation (5) and conclude the proof, it suffices to show that $\Pr\left[\mathsf{S}\left(x, x', f(x', y)\right) \neq f(x, y)\right] \geq 1/2$ for some $y \in \mathcal{Y}$.

Since $x' \not\preceq x$ there exist $y_0, y_1 \in \mathcal{Y}$ such that $f(x, y_0) \neq f(x, y_1)$ and $f(x', y_0) = f(x', y_1)$. Then $\mathsf{S}\left(x, x', f(x', y_0)\right) = \mathsf{S}\left(x, x', f(x', y_1)\right)$. However, since the only information S has on y is given by the value $f(x', y)$, this means that S cannot distinguish y_0 from y_1. Formally, let $w = f(x', y_0) = f(x', y_1)$. Then $\mathsf{S}(x, x', w)$ either outputs $f(x, y_0)$ with probability at least $1/2$, or it outputs $f(x, y_1)$ with probability at least $1/2$. Therefore it errs on one of the inputs y_0 and y_1 with probability at least $1/2$. □

Before proving Theorem 5.2, we prove the following claim. Roughly, the claim states that for a $(1,1)$-FaF secure protocol, if at a given round the backup values equal to $f(x, y)$ with somewhat high probability, then the previous backup values equal to $f(x, y)$ with a slightly smaller probability.

Claim 5.7. *Let $f : \mathcal{X} \times \mathcal{Y} \times \{\lambda\} \to \mathcal{W}$ be a deterministic three-party functionality, and suppose that there exists an r-round protocol π computing it with $(1,1)$-FaF ε-security in the dealer model. Let $i \in [r]$, let $x \in \mathcal{X}$, let $y \in \mathcal{Y}$, and let $\mu > 0$. Then,*

$$\textit{if } \Pr\left[\mathsf{ac}_i \neq f(x, y)\right] \leq \mu \textit{ then } \Pr\left[\mathsf{bc}_i \neq f(x, y)\right] \leq 2\left|\mathcal{X}\right| \cdot (\mu + \varepsilon) + \varepsilon.$$

Similarly,

$$\textit{if } \Pr[\mathsf{bc}_i \neq f(x,y)] \leq \mu \textit{ then } \Pr[\mathsf{ac}_{i-1} \neq f(x,y)] \leq 2|\mathcal{Y}| \cdot (\mu + \varepsilon) + \varepsilon.$$

In particular,

$$\textit{if } \Pr[\mathsf{ac}_i \neq f(x,y)] \leq \mu \textit{ then } \Pr[\mathsf{ac}_{i-1} \neq f(x,y)] \leq 9 \cdot |\mathcal{X}| \cdot |\mathcal{Y}| \cdot (\mu + \varepsilon).$$

Proof. The claim follows directly from Lemma 5.6 by considering a malicious adversary that sends continue to the dealer until round i or $i+1$. We prove only the first assertion, as the second is analogous. Consider the following adversary $\mathcal{A}_i$ corrupting A: Instruct A to send to the dealer x as its input, and to send continue until round $i+1$ (if $i = r$ then the A always sends continue until the termination of the protocol). In round $i+1$, instruct A to send abort to the dealer. Then C obtains ac_i, and thus can compute $f(x,y)$ except with probability at most μ. Thus, we may apply Lemma 5.6 to $\mathcal{A}_i$. Therefore, its malicious simulator must send $x^* \preceq x$, except with probability at most $2|\mathcal{X}| \cdot (\mu + \varepsilon)$. By assumption, the only option for x^* is the original input x itself. Now, observe that in the real world, the output of (the fully honest party) B when the dealer interacts with the adversary $\mathcal{A}_i$ is bc_i. Conversely, the output of B in the ideal world is $f(x,y)$ except with probability at most $2|\mathcal{X}| \cdot (\mu + \varepsilon)$. Therefore

$$\Pr[\mathsf{bc}_i \neq f(x,y)] \leq 2|\mathcal{X}| \cdot (\mu + \varepsilon) + \varepsilon,$$

as otherwise the real and ideal world can be distinguished with ε advantage by considering the output of B.

The final statement follows from applying the first two assertions. By the first part of the statement, it follows that

$$\Pr[\mathsf{bc}_i \neq f(x,y)] \leq 2|\mathcal{X}| \cdot (\mu + \varepsilon) + \varepsilon.$$

Therefore, by the second part, it follows that

$$\begin{aligned}\Pr[\mathsf{ac}_{i-1} \neq f(x,y)] &\leq 2|\mathcal{Y}| \cdot (2|\mathcal{X}| \cdot (\mu + \varepsilon) + \varepsilon + \varepsilon) + \varepsilon. \\ &\leq 4|\mathcal{X}| \cdot |\mathcal{Y}| \cdot (\mu + \varepsilon) + 4|\mathcal{Y}| \cdot \varepsilon + \varepsilon \\ &\leq 9 \cdot |\mathcal{X}| \cdot |\mathcal{Y}| \cdot (\mu + \varepsilon).\end{aligned}$$

□

We are now ready to prove Theorem 5.2.

Proof of Theorem 5.2. Fix an r-round protocol π computing f with $(1,1)$-FaF ε-security. By Theorem 3.2 we may assume without loss of generality that π is in the dealer model. Assume towards contradiction that

$$r < \frac{\log\left(\frac{1}{4\varepsilon}\right) - \log\left(\frac{1}{1-p}\right)}{\log(9 \cdot |\mathcal{X}| \cdot |\mathcal{Y}|)}.$$

The idea is to apply Lemma 5.6 inductively, starting from the last round and going backward. By Claim 5.7, at each iteration the probability the parties learn the output decreases by a factor that is roughly the domain size. We then conclude there is a noticeable probability that the parties can learn the output of the function without any interaction. We next formalize this intuition.

Consider an execution of π with $x \leftarrow \mathcal{X}$ and $y \leftarrow \mathcal{Y}$. The correctness of the protocol implies that

$$\Pr\left[\mathsf{ac}_r \neq f(x,y)\right] \leq \varepsilon$$

for all sufficiently large κ, where the probability is taken over the choice of x, y, and the random coins of the parties. By applying Claim 5.7 inductively, we obtain

$$\begin{aligned}
\Pr\left[\mathsf{ac}_0 \neq f(x,y)\right] &\leq (9 \cdot |\mathcal{X}| \cdot |\mathcal{Y}|)^r \cdot 2\varepsilon + \sum_{i=1}^{r-1} (9 \cdot |\mathcal{X}| \cdot |\mathcal{Y}|)^i \cdot \varepsilon. \\
&= (9 \cdot |\mathcal{X}| \cdot |\mathcal{Y}|)^r \cdot 2\varepsilon + \frac{9 \cdot |\mathcal{X}| \cdot |\mathcal{Y}| \cdot \left((9 \cdot |\mathcal{X}| \cdot |\mathcal{Y}|)^{r-1} - 1\right)}{9 \cdot |\mathcal{X}| \cdot |\mathcal{Y}| - 1} \cdot \varepsilon \\
&\leq (9 \cdot |\mathcal{X}| \cdot |\mathcal{Y}|)^r \cdot 2\varepsilon + 2 \cdot \left((9 \cdot |\mathcal{X}| \cdot |\mathcal{Y}|)^{r-1} - 1\right) \cdot \varepsilon \\
&\leq 4\varepsilon \cdot (9 \cdot |\mathcal{X}| \cdot |\mathcal{Y}|)^r .
\end{aligned}$$

Thus, the parties can compute $f(x,y)$ correctly with probability at least $1 - 4\varepsilon\,(9 \cdot |\mathcal{X}| \cdot |\mathcal{Y}|)^r$, without any interaction at all. By the assumption on the number of rounds r, this expression is strictly larger than p. This is impossible, and so we conclude that $r \geq \frac{\log\left(\frac{1}{4\varepsilon}\right) - \log\left(\frac{1}{1-p}\right)}{\log(9 \cdot |\mathcal{X}| \cdot |\mathcal{Y}|)}$.

□

6 Impossibility for a Two-Input Three-Party Functionality

In this section, we show that there is a function with inputs from two parties that gives the same output to 3 parties and cannot be computed with a $(1,1)$-FaF secure protocol. We prove the following.

Theorem 6.1. *Assume the existence of one-way permutations. Then there exists a three-party symmetric 2-ary functionality for which there is no protocol computing it with $(1,1)$-FaF security.*

The functionality we consider and the proof that no protocol computes it with FaF security is nearly identical to that of [1]. Let $f = \{f_\kappa : \{0,1\}^\kappa \mapsto \{0,1\}^\kappa\}_{\kappa \in \mathbb{N}}$ be a one-way permutation. Define the symmetric 3-party functionality $\mathsf{Swap} = \left\{\mathsf{Swap}_\kappa : \{0,1\}^{2\kappa} \times \{0,1\}^{2\kappa} \times \{\lambda\} \mapsto \{0,1\}^{2\kappa}\right\}_{\kappa \in \mathbb{N}}$ as follows. Parties A and B each hold two strings $(a, y_\mathsf{B}), (b, y_\mathsf{A}) \in \{0,1\}^{2\kappa}$ respectively, and party C holds no input. The output is defined as:

$$\mathsf{Swap}_\kappa\left((a, y_\mathsf{B}), (b, y_\mathsf{A}), \lambda\right) = \begin{cases} (a,b) & \text{if } f_\kappa(a) = y_\mathsf{A} \text{ and } f_\kappa(b) = y_\mathsf{B} \\ \bot & \text{otherwise} \end{cases}.$$

We first show that in the FaF secure protocol in the dealer model, there exists a round where either A and C gain an advantage in computing the correct output, or B and C gain this advantage.

Claim 6.2. *Fix a correct r-round protocol π computing Swap in the dealer model, and consider an execution of π using the following distribution over the inputs.*

- *$a, b \leftarrow \{0,1\}^\kappa$ are independent, and*
- *$y_\mathsf{A} = f_\kappa(a)$ and $y_\mathsf{B} = f_\kappa(b)$.*

Then either there exists $i \in \{0, \ldots, r\}$ *such that*

$$\Pr[\mathsf{ac}_i = (a, b)] - \Pr[\mathsf{bc}_i = (a, b)] \geq \frac{1 - \mathrm{neg}(\kappa)}{2r + 1},$$

or there exists $i \in [r]$ *such that*

$$\Pr[\mathsf{bc}_i = (a, b)] - \Pr[\mathsf{ac}_{i-1} = (a, b)] \geq \frac{1 - \mathrm{neg}(\kappa)}{2r + 1}.$$

The probabilities above are taken over the choice of inputs and of the random coins for the parties.

Proof. The proof is done using a simple averaging argument. By correctness, it holds that $\mathsf{ac}_r = (a, b)$, except with negligible probability. Additionally, since bc_0 does not depend on the input of A, i.e., on a and y_B, the fact that f_κ is one-way implies that $\mathsf{bc}_0 = (a, b)$ with negligible probability. Therefore,

$$\begin{aligned} 1 - \mathrm{neg}(\kappa) &\leq \Pr[\mathsf{ac}_r = (a, b)] - \Pr[\mathsf{bc}_0 = (a, b)] \\ &= \sum_{i=0}^{r} (\Pr[\mathsf{ac}_i = (a, b)] - \Pr[\mathsf{bc}_i = (a, b)]) + \sum_{i=1}^{r} (\Pr[\mathsf{bc}_i = (a, b)] - \Pr[\mathsf{ac}_{i-1} = (a, b)]) . \end{aligned}$$

Since there are $2r + 1$ summands, there must exist an i for which one of the differences is at least $\frac{1-\mathrm{neg}(\kappa)}{2r+1}$. □

To prove Theorem 6.1, we next show that Swap cannot be computed with $(1, 1)$-FaF security.

Lemma 6.3. *Assume that* $f = \{f_\kappa : \{0,1\}^\kappa \mapsto \{0,1\}^\kappa\}_{\kappa \in \mathbb{N}}$ *is a one-way permutation. Then there is no protocol that computes* Swap *with* $(1, 1)$*-FaF security.*

Proof. Assume for the sake of contradiction that there exists a 3-party r-round protocol that computes Swap with $(1, 1)$-FaF security. By Theorem 3.2, we may assume the protocol to be in the dealer model. We fix a security parameter κ and consider an evaluation of Swap with the output being (a, b). Formally, we consider the following distribution over the inputs.

- $a, b \leftarrow \{0,1\}^\kappa$ are independent, and
- $y_\mathsf{A} = f_\kappa(a)$ and $y_\mathsf{B} = f_\kappa(b)$.

By Claim 6.2, either there exists $i \in \{0, \ldots, r\}$ such that

$$\Pr[\mathsf{ac}_i = (a, b)] - \Pr[\mathsf{bc}_i = (a, b)] \geq \frac{1 - \mathrm{neg}(\kappa)}{2r + 1},$$

or there exists $i \in [r]$ such that

$$\Pr[\mathsf{bc}_i = (a, b)] - \Pr[\mathsf{ac}_{i-1} = (a, b)] \geq \frac{1 - \mathrm{neg}(\kappa)}{2r + 1},$$

where the probabilities above are taken over the choice of inputs and of the random coins for the parties. Assume without loss of generality that there exists an $i \in \{0, \ldots, r\}$ such that the former equality holds (the other case is done analogously). Define a malicious adversary $\mathcal{A}$ as follows. For

the security parameter κ, it corrupts A and instructs it to send (a, y_{B}) to the dealer, and until round $i+1$, send it $\mathsf{continue}$ (if $i = r$ then A sends $\mathsf{continue}$ until the termination of the protocol). When the dealer approaches A at round $i+1$, the adversary instructs A to reply with abort. This causes B to output bc_i, and C to receive ac_i. We next show that no pair of simulators $\mathsf{Sim}_{\mathcal{A}}$ and $\mathsf{Sim}_{\mathcal{A},\mathsf{C}}$ can produce a view for C in the ideal world. For that, we assume towards contradiction that such simulators do exist. Let $(a^*, y^*_{\mathsf{B}}) \in \{0,1\}^{2\kappa}$ be the input that $\mathsf{Sim}_{\mathcal{A}}$ sends to the trusted party. Additionally, denote $q = \Pr[\mathsf{bc}_i = (a,b)]$.

We next separate the proof into two cases. For the first case, let us assume that $\Pr[(a^*, y^*_{\mathsf{B}}) = (a, y_{\mathsf{B}})] \geq q + 1/p(\kappa)$ for some polynomial $p(\cdot)$ for infinitely many κ's. Let $\text{OUT}^{\mathsf{ideal}}$ be the output of (the honest party) B in the ideal world. Since f_κ is a permutation we have that

$$\Pr\left[\text{OUT}^{\mathsf{ideal}} = (a,b)\right] = \Pr[(a^*, y^*_{\mathsf{B}}) = (a, y_{\mathsf{B}})] \geq q + 1/p(\kappa).$$

On the other hand, in the real world B outputs bc_i, which satisfies

$$\Pr[\mathsf{bc}_i = (a,b)] \leq \Pr[\mathsf{ac}_i = (a,b)] - \frac{1 - \text{neg}(\kappa)}{2r+1} < q.$$

Thus, by comparing the output of the honest B to (a,b) it is possible to distinguish the real from the ideal with an advantage of at least $1/p(\kappa)$.

For the second case, we assume that $\Pr[(a^*, y^*_{\mathsf{B}}) = (a, y_{\mathsf{B}})] \leq q + \text{neg}(\kappa)$ for all sufficiently large κ. Here we show how to distinguish between the view of C in the real world from its ideal world counterpart. Recall that in the real world, the dealer sent to C the backup value ac_i. Let $\widetilde{\mathsf{ac}}_i$ denote the output of $\mathsf{Sim}_{\mathcal{A},\mathsf{C}}$ corresponding to the last backup value it simulates. First, observe that

$$\Pr[\widetilde{\mathsf{ac}}_i = (a,b) \wedge (a^*, y^*_{\mathsf{B}}) \neq (a, y_{\mathsf{B}})] = \text{neg}(\kappa).$$

Indeed, since f_κ is a permutation and B does not change the input it sends to T, the output computed by T will be $\perp$. Moreover, as f_κ is one-way, it follows that if $\widetilde{\mathsf{ac}}_i = (a,b)$ then the simulator can be used to break the security of f_κ. This can be done by sampling $a \leftarrow \{0,1\}^\kappa$, computing $f(a)$, and finally, compute $\widetilde{\mathsf{ac}}_i$ as done by $\mathsf{Sim}_{\mathcal{A},\mathsf{C}}$ (if (a^*, y^*_{B}) computed by $\mathsf{Sim}_{\mathcal{A}}$ equals to (a, y_{B}) then abort). We conclude that

$$\begin{aligned}
\Pr[\widetilde{\mathsf{ac}}_i = (a,b)] &= \Pr[\widetilde{\mathsf{ac}}_i = (a,b) \wedge a^* = a] + \Pr[\widetilde{\mathsf{ac}}_i = (a,b) \wedge (a^*, y^*_{\mathsf{B}}) \neq (a, y_{\mathsf{B}})] \\
&\leq \Pr[(a^*, y^*_{\mathsf{B}}) = (a, y_{\mathsf{B}})] + \text{neg}(\kappa) \\
&\leq q + \text{neg}(\kappa).
\end{aligned}$$

Therefore, by comparing (a,b) to the last backup value given to C, it is possible to distinguish with advantage at least $\frac{1-\text{neg}(\kappa)}{2r+1} - \text{neg}(\kappa)$. Indeed, in the real world the last backup value is ac_i which equals (a,b) with probability at least

$$\Pr[\mathsf{bc}_i = (a,b)] + \frac{1 - \text{neg}(\kappa)}{2r+1} = q + \frac{1 - \text{neg}(\kappa)}{2r+1},$$

while in the ideal world the last backup value is $\widetilde{\mathsf{ac}}_i$ which equals (a,b) with probability at most $q + \neg(\kappa)$. $\square$

Bibliography

[1] B. Alon, E. Omri, and A. Paskin-Cherniavsky. MPC with friends and foes. In *Advances in Cryptology–CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Proceedings, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 677–706. Springer, 2020.

[2] G. Asharov. Towards characterizing complete fairness in secure two-party computation. In *Proceedings of the 11th Theory of Cryptography Conference (TCC)*, volume 8349 of *Lecture Notes in Computer Science*, pages 291–316, 2014.

[3] G. Asharov, A. Beimel, N. Makriyannis, and E. Omri. Complete characterization of fairness in secure two-party computation of Boolean functions. In *Proceedings of the 12th Theory of Cryptography Conference (TCC), part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 199–228, 2015.

[4] A. Beimel, T. Malkin, and S. Micali. The all-or-nothing nature of two-party secure computation. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 80–97. Springer, 1999.

[5] A. Beimel, E. Omri, and I. Orlov. Protocols for multiparty coin toss with a dishonest majority. *Journal of Cryptology*, 28(3):551–600, 2015.

[6] A. Beimel, Y. Lindell, E. Omri, and I. Orlov. 1/p-secure multiparty computation without an honest majority and the best of both worlds. *J. Cryptol.*, 33(4):1659–1731, 2020.

[7] G. R. Blakley. Safeguarding cryptographic keys. In *Managing Requirements Knowledge, International Workshop on*, pages 313–313. IEEE Computer Society, 1979.

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

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

[10] D. Dachman-Soled. Revisiting fairness in MPC: polynomial number of parties and general adversarial structures. In *Proceedings of the 18th Theory of Cryptography Conference (TCC), part II*, volume 12551, pages 595–620. Springer, 2020.

[11] V. Daza and N. Makriyannis. Designing fully secure protocols for secure two-party computation of constant-domain functions. In *Proceedings of the 15th Theory of Cryptography Conference (TCC), part I*, pages 581–611, 2017.

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

[13] S. D. Gordon and J. Katz. Complete fairness in multi-party computation without an honest majority. In *Proceedings of the 6th Theory of Cryptography Conference (TCC)*, volume 5444 of *Lecture Notes in Computer Science*, pages 19–35, 2009.

[14] S. D. Gordon, C. Hazay, J. Katz, and Y. Lindell. Complete fairness in secure two-party computation. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 413–422, 2008.

[15] S. Halevi, Y. Ishai, E. Kushilevitz, N. Makriyannis, and T. Rabin. On fully secure MPC with solitary output. In *Proceedings of the 17th Theory of Cryptography Conference (TCC), part I*, volume 11891 of *Lecture Notes in Computer Science*, pages 312–340, 2019.

[16] R. Impagliazzo and M. Luby. One-way functions are essential for complexity based cryptography. In *30th Annual Symposium on Foundations of Computer Science*, pages 230–235. IEEE Computer Society, 1989.

[17] Y. Ishai, E. Kushilevitz, Y. Lindell, and E. Petrank. On combining privacy with guaranteed output delivery in secure multiparty computation. In C. Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pages 483–500. Springer, 2006.

[18] Y. Ishai, E. Kushilevitz, and A. Paskin. Secure multiparty computation with minimal interaction. In *Annual Cryptology Conference*, pages 577–594. Springer, 2010.

[19] Y. Ishai, J. Katz, E. Kushilevitz, Y. Lindell, and E. Petrank. On achieving the "best of both worlds" in secure multiparty computation. *SIAM journal on computing*, 40(1):122–141, 2011.

[20] J. Katz. On achieving the "best of both worlds" in secure multiparty computation. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 11–20. ACM, 2007.

[21] N. Makriyannis. On the classification of finite Boolean functions up to fairness. In *Proceedings of the 9th Conference on Security and Cryptography for Networks (SCN)*, volume 8642 of *Lecture Notes in Computer Science*, pages 135–154, 2014.

[22] N. Makriyannis. *Fairness in two-party computation: characterizing fair functions*. PhD thesis, Universitat Pompeu Fabra, 2016.

[23] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 73–85, 1989.

[24] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 387–394. ACM, 1990.

[25] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[26] A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 160–164, 1982.