MPC with Friends and Foes[*]

Bar Alon[†] Eran Omri[‡] Anat Paskin-Cherniavsky[§]

June 10, 2020

Abstract

Classical definitions for secure multiparty computation assume the existence of a single adversarial entity controlling the set of corrupted parties. Intuitively, the definition requires that the view of the adversary, corrupting t parties, in a real-world execution can be simulated by an adversary in an ideal model, where parties interact only via a trusted-party. No restrictions, however, are imposed on the view of honest parties in the protocol, thus, if honest parties obtain information about the private inputs of other honest parties – it is not counted as a violation of privacy. This is arguably undesirable in many situations that fall into the MPC framework. Nevertheless, there are secure protocols (e.g., the 2-round multiparty protocol of Ishai et al. [CRYPTO 2010] tolerating a single corrupted party) that instruct the honest parties to reveal their private inputs to all other honest parties (once the malicious party is somehow identified).

In this paper, we put forth a new security notion, which we call *FaF-security*, extending the classical notion. In essence, (t, h^*)-FaF-security requires the view of a subset of up to h^* honest parties to also be simulatable in the ideal model (in addition to the view of the malicious adversary, corrupting up to t parties). This property should still hold, even if the adversary leaks information to honest parties by sending them non-prescribed messages. We provide a thorough exploration of the new notion, investigating it in relation to a variety of existing security notions. We further investigate the feasibility of achieving FaF-security and show that every functionality can be computed with (computational) (t, h^*)-FaF full-security, if and only if $2t + h^* < m$. Interestingly, the lower-bound result actually shows that even fair FaF-security is impossible in general when $2t + h^* \geq m$ (surprisingly, the view of the malicious attacker is not used as the trigger for the attack).

We also investigate the optimal round complexity for (t, h^*)-FaF-secure protocols and give evidence that the leakage of private inputs of honest parties in the protocol of Ishai et al. [CRYPTO 2010] is inherent.

Finally, we investigate the feasibility of statistical/perfect FaF-security, employing the viewpoint used by Fitzi et al. [ASIACRYPT 1999] for *mixed-adversaries*.

[*]Research supported by ISF grant 152/17, and by the Ariel Cyber Innovation Center in conjunction with the Israel National Cyber directorate in the Prime Minister's Office. A preliminary version of this work appeared in [1].

[†]Department of Computer Science, Ariel University. E-mail: `alonbar08@gmail.com`.

[‡]Department of Computer Science, Ariel University. E-mail: `omrier@ariel.ac.il`.

[§]Department of Computer Science, Ariel University. E-mail: `anatpc@ariel.ac.il`.

Contents

1 Introduction

In the setting of secure multiparty computation (MPC), the goal is to allow a set of m mutually distrustful parties to compute some function of their private inputs in a way that preserves some security properties, even in the face of adversarial behavior by some of the parties. Classical security definitions (cf., [28]) assume the existence of a single adversarial entity controlling the set of corrupted parties. The two most common types of adversaries are *malicious* adversaries (which may instruct the corrupted parties to deviate from the prescribed protocol in any possible way), and *semi-honest* adversaries (which must follow the instructions of the protocol, but may try to infer additional information based on the joint view of the corrupted parties).

Classical security definition. Some of the most basic security properties that may be desired are correctness, privacy, independence of inputs, fairness, and guaranteed output delivery. A general paradigm for defining the desired security of protocols is known as the ideal vs. real paradigm. This paradigm avoids the need to specify a list of desired properties. Rather, security is defined by describing an ideal functionality, where parties interact via a trusted party to compute the task at hand. A real-world protocol is then deemed secure (against a class $\mathcal{C}$ of adversaries), if no adversary $\mathcal{A} \in \mathcal{C}$ can do more harm than an adversary in the ideal-world. In some more detail, the definition requires that the view of the adversary, corrupting t parties, in a real-world execution can be simulated by an adversary (corrupting the same t parties) in the ideal-world.

Classical instantiations of this paradigm, however, pose *no* restrictions on the view of honest parties in the protocol. Hence, such definitions do not count it as a violation of privacy if honest parties learn private information about other honest parties. 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). Indeed, there is no guarantee that an honest party would not get corrupted at some later point in the future. If that honest party has learned some sensitive information about another party's input during the protocol's execution (say, the password to its bank account), then this information may still be used in a malicious manner. Furthermore, as most of us are reluctant to reveal the password to our bank account even to our own friends, it is natural to consider a model, where every uncorrupted party is honest-but-curious by itself, operating simultaneously to the malicious adversary.[1]

There are two manners in which honest parties may come to learn some private information about other parties (in a secure protocol). The first is if the protocol itself instructs the honest parties to reveal some information about their private inputs (which is not implied by the output) to all other honest parties (once all malicious parties are somehow identified). An example of such a protocol is the 2-round m-party protocol (with $m \geq 5$) of Ishai et al. [35], tolerating a single malicious party.

Alternatively, honest parties may also be exposed to the private information of other parties if the adversary sends them parts of its view during the execution (although, not instructed to do so by the protocol). We stress that such an attack is applicable to many classical results in MPC that assume an upper bound of t malicious parties and rely on $(t+1)$-out-of-m secret sharing. Consider, for example, the BGW protocol [12], which is secure against $t < m/3$ corruptions. In the first round of the protocol, the parties share their inputs in a $(t+1)$-out-of-m Shamir's secret

[1]This is indeed the origin of the term FaF-security (protecting one's privacy from friends and foes alike).

sharing scheme [46]. If an adversary, controlling t parties, sends all its t shares to an honest party, then this honest party can reconstruct the inputs of all other parties.

It may be natural to try to overcome the second type of information leakage by simply instructing honest parties to disregard and erase unsolicited messages sent to them by the adversary. However, in many settings assuming that the parties are able to reliably erase parts of their state might be unrealistic, due to e.g., physical limitations on erasures. Moreover, it is not even always clear how to define what should be erased in the first place. Consider, for example, the case that the adversary has some room for action or some redundancy in the messages it is instructed to send by the protocol. In such a case, the adversary can implant additional non-prescribed information about other parties into these messages. Thus, the honest parties receiving these messages are not able to detect the leakage of information. If, say, the adversary implanted a sharing of some private information among a subset of honest parties, then, a 'semi-honest' entity can reconstruct this information by taking control over the parties in this subset and seeing their internal states.

In this paper, we investigate the following question that arises from the above discussion.

Can the classical notion of security for malicious adversaries be extended to also prevent leakage of private information to (possibly colluding) subsets of (semi)-honest parties?

The issue of honest parties being able to obtain information (not available to them from their inputs and from the output of the functionality) was already shortly mentioned in [45]. They showed how to construct verifiable secret sharing (and thus compute any functionality) with unconditional security, assuming broadcast and an honest majority. Their solution for preventing from honest parties learning additional information was to increase the threshold for the secret sharing used in the protocol. However, this came at the expense of the bound on the number of corrupted parties.

The solution of [45] may seem as a natural answer to the above question, and it may further seem that any secure protocol could be turned into one that prevents leakage to honest parties by increasing the bound on the number of corrupt parties. Say, for example that the protocol should withstand t malicious parties and we wish to avoid leakage to sets of size h^* semi-honest parties. In this case, taking a protocol that is secure (by classical definition) against $t+h^*$ malicious parties may seem to suffice for the desired security. However, now one must consider the efficiency toll incurred by increasing the security threshold. Furthermore, it could be the case that increasing the threshold would render the protocol altogether insecure. Indeed, in Section 4.1, we give such an example of a functionality that cannot be computed in the face of $t+h^*$ malicious parties, but can be computed with full security in the face of t malicious, while avoiding leakage to any subset of h^* honest parties.

Quite surprisingly, we further show that the approach of increasing the threshold simply *does not work* in general. In particular, there exist protocols with standard full security against $t+1$ malicious parties, yet t malicious parties could leak information to an honest party.[2]

1.1 Our Contribution

In this paper, we address the above question by putting forth a new security notion, which we call *FaF-security*, extending standard (static, malicious) MPC security (in the stand-alone model). We give a *full-security* variant as well as a *security-with-abort* variant for the new notion. In essence, (t, h^*)-FaF-security requires that for every malicious adversary $\mathcal{A}$ corrupting t parties, and for any

[2]It remains open whether preventing leakage using a *different* protocol is possible.

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. A more elaborate summary of the various definitions is given in Section 1.1.1. A comprehensive discussion appears in Section 3.

We accompany the new security notion with a thorough investigation of its feasibility and limitations in the various models of interest. Most notably, we discuss the feasibility of achieving full FaF-security against computational adversaries, and show that it is achievable for any functionality if and only if $2t + h^* < m$. Interestingly, the lower-bound result actually shows that any protocol admits a round in which the adversary can leak the output to some parties without learning it, however, not allowing other honest parties to learn it. Hence, even fair FaF-security is impossible in general when $2t + h^* \geq m$. In Section 1.1.2 we elaborate on these results. We also investigated the optimal round-complexity of FaF-secure protocols, and the feasibility of obtaining statistical/perfect FaF-security. A summary of these also appears in Section 1.1.2.

Finally, we provide an thorough exploration of how the new notion relates to a variety of existing security notions. Specifically, we show some counter intuitive facts on how FaF-security relates to standard malicious security and mixed-adversaries security. See Section 1.1.3 for more on that.

1.1.1 FaF-Security – A Generalization of Classical Security

Before moving on to describe our new security notion in more detail, we first recall the notion of static, malicious, stand-alone security. We stress that while there are stronger security notions, some of which we mention below, this is arguably the most standard notion, serving much of the works on secure multiparty computation. Security is defined via the real vs. ideal paradigm. Here, the security is described as an ideal functionality, where all parties (including the adversary) interact with a trusted entity. A malicious adversary is, thus, limited to selecting the inputs of the subset of corrupted parties.

A real-world protocol (for the functionality at hand) is deemed secure if it emulates the ideal setting. In a bit more detail, the protocol is t-secure, for a class $\mathcal{C}$ of adversaries, if for every adversary $\mathcal{A} \in \mathcal{C}$, corrupting at most t parties and interacting with the remaining parties, there exists an ideal-world adversary (called simulator) that outputs a view for the real-world adversary that is distributed closely to its view in an actual random execution of the real protocol. A static adversary is one that chooses which parties to corrupt before the execution of the protocol begins. A formal definition of security is given in Section 3 as a special case of FaF-security.

The notion of FaF-security. We now give a more detailed overview of the new notion of security. As above, we follow the real vs. ideal paradigm, and strengthen the requirements of standard security. We say that a protocol Π computes a functionality f with (t, h^*)-FaF security (with respect to a class $\mathcal{C}$ of adversaries), if for any adversary $\mathcal{A} \in \mathcal{C}$ (statically) corrupting at most t parties, the following holds: (i) there exists a simulator S that can simulate (in the ideal-world) $\mathcal{A}$'s view in the real-world (so far, this is standard security), and (ii) for any subset $\mathcal{H}$ (of size at most h^*) of the uncorrupted parties, there exists a "semi-honest" simulator $\mathsf{S}_{\mathcal{H}}$, such that, given the parties' inputs and S's ideal-world view (i.e., its randomness, inputs, auxiliary input, and output received from the trusted party), $\mathsf{S}_{\mathcal{H}}$ generates a view that is indistinguishable form the real-world view of the parties in $\mathcal{H}$, i.e., $\left(\textsc{view}_{\mathcal{H}}^{\text{REAL}}, \text{Out}^{\text{REAL}}\right)$ is indistinguishable from $\left(\textsc{view}_{\mathsf{S}_{\mathcal{H}}}^{\text{IDEAL}}, \text{Out}^{\text{IDEAL}}\right)$.

The reason for giving $\mathsf{S}_{\mathcal{H}}$ the ideal-world view of S is that in the real-world, nothing prevents the adversary from sending its view to honest parties. Observe that since the definition requires that the adversary is simulatable according to the standard definition, it also protects the parties in $\mathcal{H}$ from the adversary. This condition is in agreement with our motivation, where the parties in $\mathcal{H}$ are honest but might later collude in a different protocol. The universal quantifier on $\mathcal{H}$ yields, for example, that the definition also captures the model where every uncorrupted party is honest-but-curious by itself. The formal definition appears in Section 3.

FaF full-security and FaF security-with-abort. So far, we left vague the way that outputs are being distributed to parties by the trusted party in the ideal-world. The first option is that the trusted party sends the appropriate output to each of the parties. This captures the notion of full-security, as it guarantees that the honest parties always receive the output of the computation (in addition to other properties, such as correctness and privacy). Cleve [18] showed that (standard) full-security is not generally achievable. This led to a relaxed notion of security, called security-with-abort. This notion is captured very similarly to the above full-security, with the difference being that in the ideal-world, the trusted party first gives the output to the adversary, which in turn decides whether the honest parties see the output or not. This notion is naturally augmented with identifiability, by requiring the adversary to identify at least one malicious party in case the output is not given to all honest parties.

In this work, we appropriately define and consider a full-security variant and a security with (identifiable) abort variant of FaF-security. To define FaF security-with-identifiable-abort, we need to account for scenarios, where some of the uncorrupted parties learn their output in the real-world while others do not. Therefore, in the ideal execution, we explicitly allow the "semi-honest" simulator $\mathsf{S}_{\mathcal{H}}$ to receive the output from the trusted party. The formal definition appears in Section 3.1.

It is also natural to consider a stronger security notion, where the joint view of the malicious adversary is simulatable *together* with the view of parties in $\mathcal{H}$. In Section 5.3, we show that this variant is strictly stronger than the variant defined above. In fact, we show that the GMW protocol [29] satisfies the weaker notion of FaF-security, but not the stronger notion. In the following, we will sometimes refer to the weaker notion as *weak FaF-security*, and refer to the stronger notion as *strong FaF-security.*

A natural property that is highly desirable from any definition is to allow (sequential) composition. We show that both the weak variant and the strong variant of FaF-security satisfy this property. In Section 9, we prove the appropriate composition theorem, similar to that of Canetti [14] for standard security.

1.1.2 Feasibility and Limitations of FaF-Secure Computation

Our main theorem provides a characterization of the types of adversaries, for which we can compute any multiparty functionality with computational FaF full-security.

Theorem 1.1 (informal)**.** *Let $t, h^*, m \in \mathbb{N}$. Assuming OT and OWP exist, any m-party functionality f can be computed with (weak) computational (t, h^*)-FaF full-security, if and only if $2t+h^* < m$.*

For the positive direction, we first show that the GMW protocol admits FaF security-with-identifiable-abort. Then, we reduce the computation to FaF security-with-identifiable-abort, using a player elimination technique. That is, the parties compute a functionality whose output is an $(m-t)$-out-of-m secret sharing of f. Since the joint view of the malicious and semi-honest parties contain $t + h^* < m - t$ shares, they learn nothing from the output. We stress that the adversary itself cannot see the output unless all honest parties see it, and hence, cannot bias the output.

We now turn to the negative direction. Interestingly, we essentially show that for $m \leq 2t + h^*$, any m-party protocol admits a round in which an adversary (corrupting t parties) can leak the output to some h^* uncorrupted parties, while, not allowing other honest parties to learn the output.

Somewhat surprisingly, for the case where $t < m/2$, there are protocols where the adversary's view consists of only random values throughout the execution. Indeed, in our attack, the adversary learns nothing about the output, and furthermore, the view of the adversary is not used as a trigger for the attack.

We next give an overview of the proof. First, by a simple player partitioning argument, we reduce the general m-party case to the 3-party case, where $t = h^* = 1$. Let A, B, and C be three parties. Let f be a one-way permutation. We consider the following functionality. Party A holds a string a, party C holds a string c, and party B holds $y_{\mathrm{A}}, y_{\mathrm{C}}$. The output of all parties is (a, c) if $f(a) = y_{\mathrm{A}}$ and $f(c) = y_{\mathrm{C}}$, and $\perp$ otherwise. We assume the strings a and c are sampled uniformly, and that $y_{\mathrm{A}} = f(a), y_{\mathrm{C}} = f(c)$.

An averaging argument yields that there must exists a round i, where two parties, say A together with B, can recover (a, c) with significantly higher probability than C together with B. Our attacker corrupts A, acts honestly (using its original input a) until round i and then aborts (regardless of its view so far). Finally, as the protocol terminates, A will send its entire view to B. This allows B it to recover (a, c) with significantly higher probability than C.

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

We stress that since A aborts at round i, independently of its view, our attack works even if the parties have a simultaneous broadcast channel. The detailed proof appears in Section 4.2.

Low round complexity. Optimal round complexity of protocols is a well studied question for classical MPC (see, e.g., [7, 8, 27, 35]). Here, we explore the optimal number of rounds required for general computation with $(1, 1)$-FaF full-security. Our motivation for investigating this question comes from the two-round protocol of Ishai et al. [35], tolerating a single malicious party. In the second round, the honest parties can either complete the computation or are able to detect the malicious party. If a party was detected cheating, then the honest parties *reveal their inputs* to some of the other honest parties.

Clearly, this is not considered secure according to FaF-security. Indeed, we prove that there are functionalities that cannot be computed with $(1, 1)$-FaF security in less than three round. We interpret this as evidence that some kind of leakage on the inputs of honest parties is necessary in order to achieve a two-round protocol.[3] The next theorem completes the picture, asserting that for $m \geq 9$ parties, the optimal round for $(1, 1)$-FaF full-security is three.

Theorem 1.2 (informal). *Let $m \geq 9$. There exists an m-party functionality that has no 2-round protocol that computes it with (weak) $(1, 1)$-FaF full-security. On the other hand, assuming that*

[3]Naturally, we do not claim that the protocol must instruct honest parties to leak information. Rather, we prove that a malicious adversary can leak the private information of honest parties.

pseudorandom generators exist, for any m-party functionality, there exists a 3-round protocol that computes it with strong $(1, 1)$-FaF full-security.

We now present an overview of the proof. For the negative direction, we rely on the proof by Gennaro et al. [27] of the impossibility to compute $(x_1, x_2, \perp, \ldots, \perp) \mapsto x_1 \wedge x_2$ in two rounds against two corrupted parties. We observe that the adversary they proposed corrupts one party maliciously and another semi-honestly. Moreover, the semi-honest corrupted party has no input, hence the actions of the adversary can be adopted into our setting. More concretely, we show that an adversary corrupting P_2, can force all of the parties to gain specific information on x_1, yet by sending its view (at the end of the interaction) to a carefully chosen honest party, it can "teach" that party some information about x_1 that no other party has (not even the adversary itself). This proof, in fact, works for any $m \geq 3$.

For the positive direction, we consider the protocol of Damgård and Ishai [20]. Using share conversion techniques ([19]) and the 2-round verifiable secret sharing (VSS) protocol of [26], they were able to construct a 3-round protocol that tolerates $t < m/5$ corruptions. We follow similar lines as [20]. First we show how to slightly modify the VSS protocol so it will admit FaF-security. Then, by making the observation that the parties in the protocol of [20] hold only shares of the other parties' input, we are able to show that by increasing the threshold of the sharing scheme, the protocol admits FaF-security. The construction of the VSS protocol follows similar lines as in [26]. We further show that the protocol can be generalized to admit (t, h^*)-FaF full-security, whenever $5t + 3h^* < m$.

Information theoretic FaF-security. Information theoretic security have been studied extensively in the MPC literature, see e.g., [12, 45, 24] . We further generalize the corruption model to allow non-threshold adversaries (for both the malicious and the semi-honest adversaries). We consider the same adversarial structure as Fitzi et al. [24], called *monotone mixed adversarial structure.* Roughly, it states that turning a malicious party to being semi-honest does not compromise the security of the protocol. As discussed previously, this is not generally the case.

We prove the following theorem, characterizing the types of adversaries, for which we can compute any multiparty functionality with information theoretic security.

Theorem 1.3 (informal). *Let $\mathcal{R} \subseteq \{(\mathcal{I}, \mathcal{H}) : \mathcal{I} \cap \mathcal{H} = \emptyset\}$ be a monotone mixed adversarial structure over a set of parties $\mathcal{P}$. Then:*

1. *Any m-party functionality f can be computed with $\mathcal{R}$-FaF full-security, assuming an available broadcast channel, if and only if*

$$\mathcal{I}_1 \cup \mathcal{H}_1 \cup \mathcal{I}_2 \cup \mathcal{H}_2 \neq \mathcal{P},$$

for every $(\mathcal{I}_1, \mathcal{H}_1), (\mathcal{I}_2, \mathcal{H}_2) \in \mathcal{R}$.

2. *Any m-party functionality f can be computed with $\mathcal{R}$-FaF full-security (without broadcast), if and only if*

$$\mathcal{I}_1 \cup \mathcal{H}_1 \cup \mathcal{I}_2 \cup \mathcal{H}_2 \neq \mathcal{P} \text{ and } \mathcal{I}_1 \cup \mathcal{I}_2 \cup \mathcal{I}_3 \neq \mathcal{P},$$

for every $(\mathcal{I}_1, \mathcal{H}_1), (\mathcal{I}_2, \mathcal{H}_2), (\mathcal{I}_3, \mathcal{H}_3) \in \mathcal{R}$.

3. *Any m-party functionality f can be computed with $\mathcal{R}$-FaF full-security, if and only if*

$$\mathcal{I}_1 \cup \mathcal{H}_1 \cup \mathcal{I}_2 \cup \mathcal{H}_2 \cup \mathcal{I}_3 \neq \mathcal{P},$$

for every $(\mathcal{I}_1, \mathcal{H}_1), (\mathcal{I}_2, \mathcal{H}_2), (\mathcal{I}_3, \mathcal{H}_3) \in \mathcal{R}$.

Interestingly, the positive direction holds with respect to strong FaF-security, and the negative holds with respect to weak FaF-security. Additionally, as Fitzi et al. [24] showed that the same conditions hold with respect to mixed adversaries, this yields an equivalence between all three notions of security, as far as general MPC goes for monotone adversarial structures.

The proof follows similar lines as [24]. For the positive direction we show how the parties can securely emulate a 4-party BGW protocol tolerating a single malicious party. The negative direction is done by reducing the computation to a functionality known to be impossible to compute securely (according to the standard definition), using a player partitioning argument. The formal definition of monotone structures and proofs appear in Section 7.

1.1.3 The Relation Between FaF Security and Other Definitions

The relation between FaF-security and standard full-security. It is natural to explore how the new definition relates to classical definitions both in the computational and in the information-theoretic settings.

We start by comparing FaF-security to the standard definition (for static adversaries). It is easy to see that standard t-security does not imply in general (t, h^*)-FaF full-security, even for functionalities with no inputs (see Section 5.1 for a simple example showing this). Obviously, (t, h^*)-FaF-security readily implies its classical t-security counterpart. One might expect that classical $(t + h^*)$-security must imply (t, h^*)-FaF-security. We show that this is not the case in general. Specifically, in Example 5.1, we present a protocol that admits traditional (static) malicious security against t corruptions, however, it does not admit $(t - 1, 1)$-FaF-security.

In contrast to the above, we claim that adaptive $(t + h^*)$-security implies strong (t, h^*)-FaF full-security. Recall that an adaptive adversary is one that can choose which parties to corrupt during the execution and after the termination of the protocol and depending on its view. Indeed, strong FaF-security can be seen as a special case of adaptive security. We do believe, however, that the FaF model is of special interest, and specifically, that in many scenarios, the full power of adaptive security is an overkill.

The relation between FaF-security and mixed-adversaries security. The notion of "mixed adversaries" was introduced in [24]. It considers a *single* entity that corrupts a subset $\mathcal{I}$ maliciously, and another subset $\mathcal{H}$ semi-honestly. Similarly, the simulator for a mixed adversary is a single simulator controlling the parties in $\mathcal{I} \cup \mathcal{H}$, with the restriction of only being able to change the inputs of the parties in $\mathcal{I}$.

It is instructive to compare the mixed-adversary notion to that of FaF-security, which in turn, can be viewed as if there are two *distinct* adversaries (which do not collude) – one malicious and one semi-honest. One might expect that (t, h^*)-mixed full-security would imply (t, h^*)-FaF full-security. However, similarly to the case with standard security, we show the that this is not generally the case in the *computational* setting (cf., Example 5.4).

1.2 Related Works

Definitions of standard MPC where the subject of much investigation in the area of MPC. Notable works introducing various definitions are [43, 5, 6, 14, 28]. The question of achieving (standard) full-security was given quite some attention. See, e.g., [4, 18, 21, 31, 42] for two parties, [4, 12, 30] in the multiparty setting.

The definition we propose can also be viewed as if there where two different adversaries, one is corrupting *actively* and the second is corrupting *passively*, while the adversaries cannot exchange messages outside of the environment. Some forms of "decentralized" adversaries were considered in [2, 3, 15, 38], with the motivation of achieving *collusion-free* protocols. However, unlike our definition, the definitions they proposed were both complicated, and did not allow an external entity to corrupt more than a single party.

Fitzi et al. [24] where the first to consider the notion of mixed adversaries. In their model, an adversary can corrupt a subset of the parties actively, and another subset passively. Moreover, their work considered general non-threshold adversary structures. They gave a complete characterization of the adversary structures for which general unconditional MPC is possible, for four different models: Perfect security with and without broadcast, and statistical security (with negligible error probability) with and without broadcast. Beerliová-Trubíniová et al. [9], Hirt et al. [34] further studied adversaries that can additionally fail-corrupt another subset of parties. They give the exact conditions for general *secure function evaluation* (SFE) and general MPC to be possible for perfect security, statistical security, and for computational security, assuming a broadcast channel. In all these settings they confirmed the strict separation between SFE and MPC. Koo [39] considered adversaries that can maliciously corrupt certain parties, and in addition omission corrupt others. Omission corruptions allow the adversary to either block incoming and outgoing messages. Zikas et al. [49] further refined this model by introducing the notions of send-omission corruptions, where the adversary can selectively block outgoing messages, and receive-omission corruption, where the adversary can selectively block incoming messages. For a full survey of works on those notions of mixed adversaries see Zikas [48].

1.3 Organization

In Section 2 we present the required preliminaries. In Section 3 we formally define our new notion of FaF-security. Then, in Section 4 we characterize computational FaF full-security. In Section 5 we compare the new definition to other existing notions of security. Section 6 is dedicated to the low-round complexity theorems. In Section 7 we explore the feasibility of general FaF-secure MPC in the information theoretic settings. In Section 8, we review some of the subtleties in defining the proposed notion. Finally, the composition theorem is given in Section 9.

2 Preliminaries

We use calligraphic letters to denote sets, uppercase for random variables, lowercase for values, and we use bold characters to denote vectors. For $n \in \mathbb{N}$, let $[n] = \{1, 2 \ldots n\}$. 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. A PPTM is a polynomial time Turing machine.

A function $\mu(\cdot)$ is called negligible, if for every polynomial $p(\cdot)$ and all sufficiently large n, it holds that $\mu(n) < 1/p(n)$. For a vector $\mathbf{v}$ of dimension n, we write v_i for its i-th component, and for $\mathcal{S} \subseteq [n]$ we write $\mathbf{v}_{\mathcal{S}} = (v_i)_{i \in \mathcal{S}}$. 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 $f(x; r)$ to denote the value when the randomness of f is fixed to r.

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. The statistical distance between two finite distributions is defined as follows.

Definition 2.1. *The statistical distance between two finite random variables X and Y is*

$$\mathrm{SD}(X, Y) = \frac{1}{2} \sum_a |\Pr[X = a] - \Pr[Y = a]| .$$

Two ensembles $X = \{X_{a,n}\}_{a \in \mathcal{D}_n, n \in \mathbb{N}}$ and $Y = \{Y_{a,n}\}_{a \in \mathcal{D}_n, n \in \mathbb{N}}$ are said to be statistically close, denoted $X \overset{S}{\equiv} Y$, if there exists a negligible function $\mu(\cdot)$, such that for all n and $a \in \mathcal{D}_n$, it holds that

$$\mathrm{SD}(X_{a,n}, Y_{a,n}) \leq \mu(n).$$

Computational indistinguishability is defined as follows.

Definition 2.2. *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. We say that X and Y are computationally indistinguishable, denoted $X \overset{c}{\equiv} Y$, if for every non-uniform PPTM D, there exists a negligible function $\mu(\cdot)$, such that for all n and $a \in \mathcal{D}_n$, it holds that*

$$|\Pr[\mathsf{D}(X_{a,n}) = 1] - \Pr[\mathsf{D}(Y_{a,n}) = 1]| \leq \mu(n).$$

Definition 2.3. *Let $X = \{X_n\}_{n \in \mathbb{N}}$ be an ensemble, such that each X_n takes values in $\{0,1\}^m$, for $m = m(n)$. Denote $U = \{U_m\}_{m \in \mathbb{N}}$ where U_m is the uniform distribution over $\{0,1\}^m$. We say that X is* pseudorandom *if $X \overset{c}{\equiv} U$.*

A a polynomial time computable function $\mathsf{G} : \{0,1\}^n \mapsto \{0,1\}^m$ with $m = m(n) > n$, is called pseudorandom generator *if $\{\mathsf{G}(U_n)\}_{n \in \mathbb{N}}$ is pseudorandom.*

2.1 Cryptographic Tools

Oblivious Transfer. Oblivious Transfer (OT) [44, 23] is a 2-party functionality, fundamental to secure multiparty computation. In the setting 1-out-of-2 OT, we have a receiver holding a bit $b \in \{0,1\}$, and a sender holding two messages m_0, m_1. At the end of the interaction, the receiver learns m_b and nothing else, and the sender learns nothing.

Secret Sharing Schemes. A $(t+1)$-out-of-m secret-sharing scheme (also called t-private) is a mechanism for sharing data among a set of parties $\mathcal{P}$ of size m, such that every set of size $t+1$ can reconstruct the secret, while any smaller set knows nothing about the secret. As a convention, for a secret s and a party $\mathrm{P}_i \in \mathcal{P}$, we let $s[i]$ be the share received by P_i. For a subset $\mathcal{S} \subseteq \mathcal{P}$ we denote $s[\mathcal{S}] = (s[i])_{i \in \mathcal{S}}$. In this work, we rely on several variants of secret sharing scheme. We next briefly describe the sharing schemes.

In a $(t+1)$-out-of-m *Shamir's secret sharing scheme* over a field $\mathbb{F}$, where $|\mathbb{F}| > m$, a secret $s \in \mathbb{F}$ is shared as follows: A polynomial $p(\cdot)$ of degree at most $t+1$ over $\mathbb{F}$ is picked uniformly at random, conditioned on $p(0) = s$. Each party P_i, for $1 \le i \le m$, receives a share $s[i] := p(i)$ (we abuse notation and let i be the element in $\mathbb{F}$ associated with P_i).

We will make use of the following homomorphic property of Shamir secret sharing.

Fact 2.4. *Let $Q(x_1, \ldots, x_m; r_1, \ldots, r_k)$ be a multivariate polynomial of degree d over the field $\mathrm{GF}(2^n)$. Here, we assume that each x_i is being held by party P_i and that each $r_i \in \{0,1\}^n$ is random. In addition, the x_i's and r_j's are all shared among all parties in a $(t+1)$-out-of-m Shamir secret sharing scheme, and the value $z = 0$ is shared among all parties in a $(dt+1)$-out-of-m Shamir secret sharing scheme. Then $Q\left(x_1[i], \ldots, x_m[i]; r_1[i], \ldots, r_k[i]\right) + z[i]$ is a dt-private Shamir share of $Q(x_1, \ldots, x_m; r_1, \ldots, r_k)$.*

Beimel et al. [11] presented a way to construct a secret sharing scheme *with respect to a certain party.* We use that in our construction as well, and refer to it as $(t + 1)$-out-of-m Shamir secret sharing scheme with respect to (the designated) party P_j.[4]

Construction 2.5. *Let s be some secret taken from some finite field $\mathbb{F}$. We share s among m parties* with respect to *a special party P_j in a $(t + 1)$-out-of-m secret-sharing scheme as follows:*

1. *Choose shares $(s[1], s[2])$ of the secret s in a two-out-of-two secret-sharing scheme, that is, select $s[1] \in \mathbb{F}$ uniformly at random and compute $s[2] = s - s[1]$. Denote these shares by* $\mathrm{mask}_j(s)$ *and* $\mathrm{comp}(s)$*, respectively.*

2. *Generate shares $(s[2,1], \ldots, s[2,j-1], s[2,j+1], \ldots, s[2,m])$ of the secret* $\mathrm{comp}(s)$ *in a t-out-of-$(m-1)$ Shamir's secret-sharing scheme. For each $\ell \neq j$, denote* $\mathrm{comp}_\ell(s) = s[2,\ell]$.

Output:

- *The share of party P_j is* $\mathrm{mask}_j(s)$*. We call this share, P_j's* masking *share.*

- *The share of each party P_ℓ, where $\ell \neq j$, is* $\mathrm{comp}_\ell(s)$*. We call this share, P_ℓ's* complement *share.*

In the above, the secret s is shared among the parties in $\mathcal{P}$ in a secret-sharing scheme such that any set of size at least $t + 1$ that contains P_j can reconstruct the secret. In addition, similarly to Shamir's secret-sharing scheme, the following property holds: for any set of $t' \le t$ parties (regardless if the set contains P_j), the shares of these parties are uniformly distributed and independent of the secret. Furthermore, given such $t' \le t$ shares and a secret s, one can *efficiently* complete them to m shares of the secret s and *efficiently* select uniformly at random one vector of shares completing the t' shares to m shares of the secret s.

In a $(t + 1)$-out-of-m *CNF scheme* over an Abelian group G, a secret $s \in \mathbb{F}$ is shared as follows: For every subset $\mathcal{S} \subseteq [m]$ of size t, pick $s[\mathcal{S}]$ from G uniformly at random, conditioned on $s = \sum_{\mathcal{S}\subseteq[m]:|\mathcal{S}|=t} s[\mathcal{S}]$. Each party P_i, for $1 \le i \le m$, receives the $\binom{m-1}{t}$ shares $\mathbf{s}[i] := (s[\mathcal{S}])_{\mathcal{S}\subseteq[m]\setminus\{i\}:|\mathcal{S}|=t}$.

[4] In standard secret sharing terminology, the scheme is a secret sharing scheme for an access structure with minterm set $\{\mathrm{P}_j \cup \mathcal{S} : \mathcal{S} \subseteq \mathcal{P}, |\mathcal{S}| = t\}$.

Verifiable Secret Sharing Schemes. Verifiable Secret Sharing (VSS) [17, 28] is a secure multiparty protocol for computing the randomized functionality corresponding to some (non-verifiable) secret sharing scheme. It is a special type of secret sharing scheme, that allows a dealer to share a secret among several players in a way that would later allow a unique reconstruction of the secret. Specifically, when a dealer D shares a secret s the following should hold:

Privacy: If D is honest, then the adversary's view reveals no information on s.

Correctness: If D is honest, then the honest parties will *always* be able to reconstruct the secret s, regardless of the views provided by the adversary.

Commitment: If D is malicious, the honest parties will always reconstruct a unique value s^*, regardless of the views provided by additional malicious parties.

Error Correcting Secret Sharing Schemes. A $(t+1)$-out-of-m secret-sharing scheme is also *error correcting*,[5] if the reconstruction algorithm outputs the correct secret even when up to t shares are arbitrarily modified. It is a known fact that a $(t+1)$-out-of-m Shamir's secret sharing scheme is error correcting if $d + 2t < m$, where $d \geq t$ is the degree of the polynomial used in the sharing phase.

Secure Multicast. Secure multicast with sender D and multicast set $\mathcal{M} \subseteq \mathcal{P}$, where $\mathrm{D} \in \mathcal{M}$, may be formally defined as a secure multiparty evaluation of the following function: D holds a message M as input, and all other parties have no input. All parties in $\mathcal{M}$ receive M as their output, and all other parties receive no output. Specifically, the computation should satisfy the following conditions:

Privacy: If all parties in $\mathcal{M}$ are honest, then the adversary learns no information on M.

Correctness: If D is honest, then all honest parties in $\mathcal{M}$ output x.

Agreement: All honest parties in $\mathcal{M}$ output the same value, even if D is corrupted.

One-Way Permutations. A one-way permutation is an efficiently computable permutation, that no polynomial time algorithm can invert. Formally, it is defined as follows.

Definition 2.6. *Let $f = \{f_n : \{0,1\}^n \mapsto \{0,1\}^n\}$ be efficiently computable permutations. We say that f is a one-way permutation if for any PPTM A there exists a negligible function $\mu(\cdot)$, such that for all sufficiently large n, we have*

$$\Pr_{x \leftarrow \{0,1\}^n}\left[\mathsf{A}\left(1^n, f_n(x)\right) = x\right] \leq \mu(n).$$

3 The New Definition – FaF Full-Security

In this section, we present our new security notion, aiming to strengthen the classical definition of security in order to impose privacy restrictions on (subsets of) honest parties, even in the presence of malicious behavior by other parties. Crucially, we wish to prevent the adversary from leaking

[5]Such schemes are also known as robust secret sharing.

private information of one subset of parties to another subset of parties, even though neither subset is under its control. The definition is written alongside the classical definition.

We follow the standard *ideal vs. real* paradigm for defining security. Intuitively, the security notion 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 deemed 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 adversary. For FaF security, we further require that the view of any subset of the uncorrupted parties can be simulated in the ideal-world (including the interaction with the adversary).

To shed some light on some of the subtleties in defining the proposed notion, inSection 8we review several possible approaches for capturing the desired security notion (avoiding leakage to honest parties), and demonstrate why they fall short in doing so. In Section 5, we compare the actual definition we put forth with the standard full-security definition, and with the mixed-adversaries definition.

To make the above intuition more formal, fix a (possibly randomized) m-ary function $f = \{f_n \colon \mathcal{X}_1^n \times \cdots \times \mathcal{X}_m^n \mapsto \mathcal{Y}_1^n \times \cdots \times \mathcal{Y}_m^n\}_{n\in\mathbb{N}}$ and let Π be a protocol for computing f. We further let $\mathcal{X}^n = \mathcal{X}_1^n \times \cdots \times \mathcal{X}_m^n$.

The FaF Real Model

An m-party protocol Π for computing the function f is defined by a set of m interactive probabilistic polynomial-time Turing machines $\mathcal{P} = \{\mathrm{P}_1, \ldots, \mathrm{P}_m\}$. Each Turing machine (party) holds the security parameter 1^n as a joint input and a private input $x_i \in \mathcal{X}_i^n$. The computation proceeds in rounds. In each round, the parties either broadcast and receive messages over a common broadcast channel, or send messages to an individual party over a secure channel. The number of rounds in the protocol is expressed as some function $r(n)$ in the security parameter (where $r(n)$ is bounded by some polynomial). At the end of the protocol, the (honest) parties output a value according to the specifications of the protocol. When the security parameter is clear from the context, we remove it from the notations. The view of a party consists of the party's input, randomness, and the messages received throughout the interaction.

We consider two adversaries. The first is a malicious adversary $\mathcal{A}$ that controls a subset $\mathcal{I} \subset \mathcal{P}$. The adversary has access to the full view of all corrupted parties. Additionally, the adversary may instruct the corrupted parties to deviate from the protocol in any way it chooses. We make explicit the fact that 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 (see Remark 3.1 for more on this). The adversary is non-uniform, and is given an auxiliary input $z_{\mathcal{A}}$. The second adversary is a semi-honest adversary $\mathcal{A}_{\mathcal{H}}$ that controls a subset $\mathcal{H} \subset \mathcal{P} \setminus \mathcal{I}$ of the remaining parties (for the sake of clarity, we will only refer to the parties in $\mathcal{I}$ as corrupted). Similarly to $\mathcal{A}$, this adversary also has access to the full view of its parties. However, $\mathcal{A}_{\mathcal{H}}$ *cannot* instruct the parties to deviate from the prescribed protocol in any way, but may try to infer information about non-corrupted parties, given its view in the protocol (which includes the joint view of parties in $\mathcal{H}$). This adversary is also non-uniform, and is given an auxiliary input $z_{\mathcal{H}}$. When we say that the adversary is computationally bounded, we mean it is a PPTM. Both adversaries are static, that is, they choose the subset to corrupt prior to the execution of the protocol. For a subset of parties $\mathcal{S} \subseteq \mathcal{P}$, we let $\mathbf{x}_{\mathcal{S}}$ be the vector of inputs of the parties in $\mathcal{S}$, specifically, $\mathbf{x}_{\mathcal{I}}$ and $\mathbf{x}_{\mathcal{H}}$ denote the vector of inputs of the parties controlled by $\mathcal{A}$ and $\mathcal{A}_{\mathcal{H}}$ respectively.

We next define the real-world global view for security parameter $n \in \mathbb{N}$, an input sequence $\mathbf{x} = (x_1, \ldots, x_m)$, and auxiliary inputs $z_{\mathcal{A}}, z_{\mathcal{H}} \in \{0,1\}^*$ with respect to adversaries $\mathcal{A}$ and $\mathcal{A}_{\mathcal{H}}$ controlling the parties in $\mathcal{I} \subset \mathcal{P}$ and $\mathcal{H} \subset \mathcal{P} \setminus \mathcal{I}$ respectively. Let $\text{OUT}^{\text{REAL}}_{\mathcal{A},\Pi}(1^n, \mathbf{x})$ denote the outputs of the uncorrupted parties (those in $\mathcal{P} \setminus \mathcal{I}$) in a random execution of Π, with $\mathcal{A}$ corrupting the parties in $\mathcal{I}$. Further let $\text{VIEW}^{\text{REAL}}_{\mathcal{A},\Pi}(1^n, \mathbf{x})$ be the adversary's view during an execution of Π, which contains its auxiliary input, its random coins, the inputs of the parties in $\mathcal{I}$, and the messages they see during the execution of the protocol. In addition, we let $\text{VIEW}^{\text{REAL}}_{\mathcal{A},\mathcal{A}_{\mathcal{H}},\Pi}(1^n, \mathbf{x})$ be the view of $\mathcal{A}_{\mathcal{H}}$ during an execution of Π when running alongside $\mathcal{A}$ (this view consists of their random coins, their inputs, and the messages they see during the execution of the protocol, and specifically, those non-prescribed messages sent to them by the adversary).

We denote the global view in the real model by

$$\text{REAL}^{\Pi,\mathcal{A},\mathcal{A}_{\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}} = \left(\text{VIEW}^{\text{REAL}}_{\mathcal{A},\Pi}(1^n, \mathbf{x}),\ \text{VIEW}^{\text{REAL}}_{\mathcal{A},\mathcal{A}_{\mathcal{H}},\Pi}(1^n, \mathbf{x}),\ \text{OUT}^{\text{REAL}}_{\mathcal{A},\Pi}(1^n, \mathbf{x})\right).$$

It will be convenient to denote

$$\text{REAL}^{\Pi,\mathcal{A},\mathcal{A}_{\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}}(\mathcal{A}) = \left(\text{VIEW}^{\text{REAL}}_{\mathcal{A},\Pi}(1^n, \mathbf{x}),\ \text{OUT}^{\text{REAL}}_{\mathcal{A},\Pi}(1^n, \mathbf{x})\right),$$

i.e., the projection of $\text{REAL}^{\Pi,\mathcal{A},\mathcal{A}_{\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}}$ to their view of the adversary and the uncorrupted parties' output (those in $\mathcal{P} \setminus \mathcal{I}$), and denote

$$\text{REAL}^{\Pi,\mathcal{A},\mathcal{A}_{\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}}(\mathcal{A}_{\mathcal{H}}) = \left(\text{VIEW}^{\text{REAL}}_{\mathcal{A},\mathcal{A}_{\mathcal{H}},\Pi}(1^n, \mathbf{x}),\ \text{OUT}^{\text{REAL}}_{\mathcal{A},\Pi}(1^n, \mathbf{x})\right).$$

When Π is clear from the context, we will remove it for brevity.

Remark 3.1. *A subtlety in the proposed model is how to deal with messages sent by the adversary at a later point in time, after the protocol execution terminated. Specifically, if honest parties need to react to such messages, then the protocol has no predefined termination point. It is possible to incorporate a parameter τ of time to the security definition, asserting that the protocol is secure until time τ. To keep the definition clean and simple, we overcome this subtlety by only allowing the real-world adversary to communicate with other (non-corrupted) parties until the last round of the protocol.*

The FaF Ideal Model

We next describe the interaction in the *FaF full-security ideal model*, which specifies the requirements for fully secure FaF computation of the function f with security parameter n. Let $\mathcal{A}$ be an adversary in the ideal-world, which is given an auxiliary input $z_{\mathcal{A}}$ and corrupts the subset $\mathcal{I}$ of the parties called *corrupted*. Further let $\mathcal{A}_{\mathcal{H}}$ be a semi-honest adversary, which is controlling a set of parties denoted $\mathcal{H}$ and is given an auxiliary input $z_{\mathcal{H}}$. We stress that the classical formulation of the ideal model assumes $\mathcal{H} = \emptyset$.

The FaF ideal model – full-security.

Inputs: Each party P_i holds 1^n and $x_i \in \mathcal{X}_i^n$. The adversaries $\mathcal{A}$ and $\mathcal{A}_{\mathcal{H}}$ are given each an auxiliary input $z_{\mathcal{A}}, z_{\mathcal{H}} \in \{0,1\}^*$ respectively, and x_i for every P_i controlled by them. The trusted party T holds 1^n.

Parties send inputs: Each uncorrupted party $\mathrm{P}_j \in \mathcal{P} \backslash \mathcal{I}$ sends x_j as its input to T. The malicious adversary $\mathcal{A}$ sends a value $x_i' \in \mathcal{X}_i^n$ as the input for party $\mathrm{P}_i \in \mathcal{I}$. Write $(x_1', \ldots, x_m')$ for the tuple of inputs received by the trusted party.

The trusted party performs computation: The trusted party T selects a random string r and computes $\mathbf{y} = (y_1, \ldots, y_m) = f\left(x_1' \ldots, x_m'; r\right)$ and sends y_i to each party P_i.

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

Outputs: Each uncorrupted party (i.e., not in $\mathcal{I}$) outputs whatever output it received from T, the parties in $\mathcal{I}$ output nothing. $\mathcal{A}$ and $\mathcal{A}_{\mathcal{H}}$ output some function of their respective view.

Note that we gave $\mathcal{A}_{\mathcal{H}}$ the ideal-world view of $\mathcal{A}$. 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. Consider the following example. Suppose three parties computed a functionality $(\perp, \perp, \perp) \mapsto (r, \perp, r)$, where r is some random string. A corrupted P_1 can send r to P_2 at the end of the interaction, thereby teaching it the output of an honest party. In the ideal-world described above, $\mathcal{A}_{\mathrm{P}_2}$ will receive r as well, allowing us to simulate this interaction.

We next define the ideal-world global view for security parameter $n \in \mathbb{N}$, an input sequence $\mathbf{x} = (x_1, \ldots, x_m)$, and auxiliary inputs $z_{\mathcal{A}}, z_{\mathcal{H}} \in \{0,1\}^*$ with respect to adversaries $\mathcal{A}$ and $\mathcal{A}_{\mathcal{H}}$ controlling the parties in $\mathcal{I} \subset \mathcal{P}$ and $\mathcal{H} \subset \mathcal{P} \backslash \mathcal{I}$ respectively. Let $\textsc{out}^{\mathrm{IDEAL}}_{\mathcal{A},f}\left(1^n, \mathbf{x}\right)$ denote the outputs of the uncorrupted parties (those in $\mathcal{P} \setminus \mathcal{I}$) in a random execution of the above ideal-world process, with $\mathcal{A}$ corrupting the parties in $\mathcal{I}$. Further let $\textsc{view}^{\mathrm{IDEAL}}_{\mathcal{A},f}\left(1^n, \mathbf{x}\right)$ be the (simulated, real-world) view description being the *output* of $\mathcal{A}$ in such a process. In addition, we let $\textsc{view}^{\mathrm{IDEAL}}_{\mathcal{A},\mathcal{A}_{\mathcal{H}},f}\left(1^n, \mathbf{x}\right)$ be the view description being the *output* of $\mathcal{A}_{\mathcal{H}}$ in such a process, when running alongside $\mathcal{A}$. We denote the global view in the ideal model by

$$\textsc{ideal}^{f,\mathcal{A},\mathcal{A}_{\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}} = \left(\textsc{view}^{\mathrm{IDEAL}}_{\mathcal{A},f}\left(1^n, \mathbf{x}\right),\ \textsc{view}^{\mathrm{IDEAL}}_{\mathcal{A},\mathcal{A}_{\mathcal{H}},f}\left(1^n, \mathbf{x}\right),\ \textsc{out}^{\mathrm{IDEAL}}_{\mathcal{A},f}\left(1^n, \mathbf{x}\right)\right).$$

As in the real model, it will be convenient to denote

$$\textsc{ideal}^{f,\mathcal{A},\mathcal{A}_{\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}}\left(\mathcal{A}\right) = \left(\textsc{view}^{\mathrm{IDEAL}}_{\mathcal{A},f}\left(1^n, \mathbf{x}\right),\ \textsc{out}^{\mathrm{IDEAL}}_{\mathcal{A},f}\left(1^n, \mathbf{x}\right)\right)$$

and

$$\textsc{ideal}^{f,\mathcal{A},\mathcal{A}_{\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}}\left(\mathcal{A}_{\mathcal{H}}\right) = \left(\textsc{view}^{\mathrm{IDEAL}}_{\mathcal{A},\mathcal{A}_{\mathcal{H}},f}\left(1^n, \mathbf{x}\right),\ \textsc{out}^{\mathrm{IDEAL}}_{\mathcal{A},f}\left(1^n, \mathbf{x}\right)\right).$$

When f is clear from the context, we will remove it for brevity. We first define correctness.

Definition 3.2 (correctness)**.** *We say that a protocol* Π *computes a function* f *if for all* $n \in \mathbb{N}$ *and for all* $\mathbf{x} \in \mathcal{X}^n$*, in an honest execution, the joint output of all parties is identically distributed to a sample of* $f(\mathbf{x})$*.*

We next give the definition for the classical definition of computational security alongside FaF-security.

Definition 3.3 (classical malicious and FaF security)**.** *Let* Π *be a protocol for computing* f*. We say that* Π *computes* f *with* computational (t, h^*)-FaF full-security*, if the following holds. For every non-uniform PPTM adversary* $\mathcal{A}$*, controlling a set* $\mathcal{I} \subset \mathcal{P}$ *of size at most* t *in the real-world, there exists a non-uniform PPTM adversary* $\mathsf{S}_{\mathcal{A}}$*, controlling* $\mathcal{I}$ *in the ideal model; and for every subset of the remaining parties* $\mathcal{H} \subset \mathcal{P} \backslash \mathcal{I}$ *of size at most* h^**, controlled by a non-uniform* semi-honest *PPTM adversary* $\mathcal{A}_{\mathcal{H}}$*, there exists a non-uniform PPTM adversary* $\mathsf{S}_{\mathcal{A},\mathcal{H}}$*, controlling* $\mathcal{H}$ *in the ideal-world, such that*

$$\left\{\mathrm{IDEAL}^{\mathsf{S}_{\mathcal{A}},\mathsf{S}_{\mathcal{A},\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}}(\mathsf{S}_{\mathcal{A}})\right\}_{\mathbf{x}\in\mathcal{X},z_{\mathcal{A}},z_{\mathcal{H}}\in\{0,1\}^*,n\in\mathbb{N}} \overset{c}{\equiv} \left\{\mathrm{REAL}^{\mathcal{A},\mathcal{A}_{\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}}(\mathcal{A})\right\}_{\mathbf{x}\in\mathcal{X},z_{\mathcal{A}},z_{\mathcal{H}}\in\{0,1\}^*,n\in\mathbb{N}}, \quad (1)$$

and

$$\left\{\mathrm{IDEAL}^{\mathsf{S}_{\mathcal{A}},\mathsf{S}_{\mathcal{A},\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}}(\mathsf{S}_{\mathcal{A},\mathcal{H}})\right\}_{\mathbf{x}\in\mathcal{X},z_{\mathcal{A}},z_{\mathcal{H}}\in\{0,1\}^*,n\in\mathbb{N}} \overset{c}{\equiv} \left\{\mathrm{REAL}^{\mathcal{A},\mathcal{A}_{\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}}(\mathcal{A}_{\mathcal{H}})\right\}_{\mathbf{x}\in\mathcal{X},z_{\mathcal{A}},z_{\mathcal{H}}\in\{0,1\}^*,n\in\mathbb{N}}. \quad (2)$$

We say that Π *computes* f *with computational* t*-security if it computes it with computational* $(t, 0)$*-FaF full-security.*

Finally, we say that Π *computes* f *with* strong *computational* (t, h^*)*-FaF full-security if*

$$\left\{\mathrm{IDEAL}^{\mathsf{S}_{\mathcal{A}},\mathsf{S}_{\mathcal{A},\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}}\right\}_{\mathbf{x}\in\mathcal{X},z_{\mathcal{A}},z_{\mathcal{H}}\in\{0,1\}^*,n\in\mathbb{N}} \overset{c}{\equiv} \left\{\mathrm{REAL}^{\mathcal{A},\mathcal{A}_{\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}}\right\}_{\mathbf{x}\in\mathcal{X},z_{\mathcal{A}},z_{\mathcal{H}}\in\{0,1\}^*,n\in\mathbb{N}}. \quad (3)$$

To abbreviate notations, whenever $\mathcal{H} = \{\mathrm{P}\}$ we denote its simulator by $\mathsf{S}_{\mathcal{A},\mathrm{P}}$.

In Section 7, we give the statistical/perfect security variants of the above definitions. These variants are obtained naturally from the above definition by replacing computational indistinguishability with statistical distance.

Remark 3.4. *Observe that for the two-party case, since we also protect* $\mathcal{H}$ *from* $\mathcal{A}$*, (weak)* $(1, 1)$*-FaF-security is equivalent to the security considered by Beimel et al. [10]. There, security holds if and only if no malicious adversary and no semi-honest adversary can attack the protocol.*

Remark 3.5. *Observe that according to the definition, we first need to describe a malicious simulator before fixing the semi-honest parties in* $\mathcal{H}$*. This should be considered in regard to the definition of the ideal-model, where the malicious simulator* $\mathsf{S}_{\mathcal{A}}$ *sends to the semi-honest simulator* $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ *its ideal-world view, implying that* $\mathsf{S}_{\mathcal{A}}$ *should know the identities of* $\mathcal{H}$*. Formally, we let the malicious simulator have an additional tape, where it writes its ideal-world view on it, and then the semi-honest simulator reads from it.*

f-Hybrid Model. Let f be a m-ary functionality. The f-hybrid model is identical to the real model of computation discussed above, but in addition, each m-size subset of the parties involved, has access to a trusted party realizing f.

3.1 FaF Security-With-Identifiable-Abort

We also make use of protocols admitting *security-with-identifiable-abort.* In terms of the definition, the only requirement that is changed, is to have the ideal-world simulator operate 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 n. Unlike the full-security ideal model, here the malicious adversary can instruct the trusted party to not 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 won't learn the output, we always let the semi-honest parties to receive their output in the ideal execution.

Let $\mathcal{A}$ be a malicious adversary in the ideal-world, which is given an auxiliary input $z_{\mathcal{A}}$ and corrupts the subset $\mathcal{I}$ of the parties. Further let $\mathcal{A}_{\mathcal{H}}$ be a semi-honest adversary, which is controlling a set of parties denoted $\mathcal{H}$ and is given an auxiliary input $z_{\mathcal{H}}$. Just like in the full-security ideal-world, the standard formulation of security-with-identifiable-abort assumes $\mathcal{H} = \emptyset$.

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

Inputs: Each party P_i holds 1^n and $x_i \in \mathcal{X}_i^n$. The adversaries $\mathcal{A}$ and $\mathcal{A}_{\mathcal{H}}$ are given each an auxiliary input $z_{\mathcal{A}}, z_{\mathcal{H}} \in \{0,1\}^*$ respectively, and x_i for every P_i controlled by them. The trusted party T holds 1^n.

Parties send inputs: Each uncorrupted party $\mathrm{P}_j \in \mathcal{P} \setminus \mathcal{I}$ sends x_j as its input to T. The malicious adversary sends a value $x_i' \in \mathcal{X}_i^n$ as the input for party $\mathrm{P}_i \in \mathcal{I}$. Write $(x_1', \ldots, x_m')$ for the tuple of inputs received by the trusted party.

The trusted party performs computation: The trusted party T selects a random string r and computes $\mathbf{y} = (y_1, \ldots, y_m) = f\left(x_1' \ldots, x_m'; r\right)$ and sends $\mathbf{y}_{\mathcal{I}}$ to $\mathcal{A}$ and $\mathbf{y}_{\mathcal{H}}$ to $\mathcal{A}_{\mathcal{H}}$.

The malicious adversary sends its (ideal-world) view: $\mathcal{A}$ sends to $\mathcal{A}_{\mathcal{H}}$ 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 `continue` or $(\texttt{abort}, \mathrm{P}_i)$ for some $\mathrm{P}_i \in \mathcal{I}$ to T. If it sent `continue`, then for every honest party P_j the trusted party sends y_j. Otherwise, if $\mathcal{A}$ sent $(\texttt{abort}, \mathrm{P}_i)$, then T sends $(\texttt{abort}, \mathrm{P}_i)$ to the each honest party P_j.

Outputs: Each uncorrupted party outputs whatever output it received from T (the parties in $\mathcal{H}$ output $(\texttt{abort}, \mathrm{P}_i)$ if they received it in the last step), the parties in $\mathcal{I}$ output nothing. The adversaries output some function of their respective view.

4 Characterizing Computational FaF-Security

In this section we prove our main theorem regarding FaF-security. We give a complete characterization the types of adversaries, for which we can compute any multiparty functionality with computational FaF full-security. We prove the following result.

Theorem 4.1. *Let $t, h^*, m \in \mathbb{N}$. Then under the assumption that OT and OWP exist, any m-party functionality f can be computed with (weak) computational (t, h^*)-FaF full-security, if and only if $2t + h^* < m$. Moreover, the negative direction holds even assuming the availability of simultaneous broadcast.*

In Section 4.1 we show the positive direction, while in Section 4.2 we prove the negative direction.

4.1 Feasibility of FaF-Security

In this section, we prove the positive direction of Theorem 4.1. In fact, we show how to reduce FaF full-security to FaF security-with-identifiable-abort whenever $2t + h^* < m$. In addition, we explore the feasibility of both FaF full-security and FaF security-with-identifiable-abort, and provide interesting consequences of these results. We first show that the GMW protocol [29] admits FaF security-with-identifiable-abort, for all possible threshold values of t and h^*, and admits FaF full-security assuming $t + h^* < m/2$. In Section 4.1.2 we show that, assuming an *uncorrupted majority* (i.e., $t < m/2$), residual FaF full-security is (perfectly) reducible to FaF-security-with-identifiable-abort. The notion of *residual security* [33], intuitively allows an adversary to learn the output of the function on many choices of inputs for corrupted parties. A formal definition and some motivation for using residual security variant appear in Section 4.1.2.

4.1.1 Feasibility of FaF Security-With-Identifiable-Abort

We next show that the GMW protocol admits FaF security-with-identifiable-abort, and admits FaF full-security in the presence of an honest majority (i.e., $t + h^* < m/2$).

Theorem 4.2. *Let $m, t, h^* \in \mathbb{N}$ be such that $t + h^* \leq m$ and Let f be an m-party functionality. Then, assuming OT exists, there exists a protocol for computing f with (weak) computational (t, h^*)-FaF security-with-identifiable-abort. Moreover, if $t + h^* < m/2$ then the protocol admits computational (t, h^*)-FaF full-security.*

Proof Sketch. We will show that a slight variation on the GMW protocol [29], setting the secret sharing (for sharing the inputs) to a $(t + h^* + 1)$-out-of-m scheme, admits FaF-security.

Fix an adversary $\mathcal{A}$ corrupting $\mathcal{I}$ of size at most t, and let $\mathcal{H} \subseteq \mathcal{P} \setminus \mathcal{I}$ be of size at most h^*. The semi-honest simulator $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ will work very similarly to the malicious simulator $\mathsf{S}_{\mathcal{A}}$. The only difference is that the messages it sends to the adversary on behalf of the parties in $\mathcal{H}$, are the real message that the protocol instruct them to send (e.g., in the input commitment phase it will commit to the real input unlike $\mathsf{S}_{\mathcal{A}}$, which commits to 0). Additionally, if the adversary did not abort, for every output wire held by a party in $\mathcal{I} \cup \mathcal{H}$, set the message received from the honest parties (i.e., from $\mathcal{P} \setminus (\mathcal{I} \cup \mathcal{H})$) as the XOR of the output of that wire and the shares of the wire held by the corrupted and semi-honest parties.

Security follows from the fact that the messages $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ sends to $\mathcal{A}$ are consistent with the inputs of the malicious and the semi-honest parties.

□

4.1.2 Reducing Residual FaF Full-Security to FaF Security-With-Identifiable-Abort

In this section, we present a reduction from residual FaF full-security to FaF security-with-identifiable-abort, in the uncorrupted majority setting. This reduction further has the property that if $2t + h^* < m$ then FaF full-security is obtained (i.e., not residual). We first formally define the residual function. Intuitively, the residual of an m-ary function with respect to a subset $\mathcal{S}$ of the indexes, fixes the inputs on the indexes $[m] \setminus \mathcal{S}$. More formally, it is defined as follows.

Definition 4.3 (Residual Function [33, 32]). *Let $f : \mathcal{X} \mapsto \mathcal{Y}$ be an m-ary functionality, let $\mathbf{x} = (x_1, \ldots, x_m)$ be an input to f, and let $\mathcal{S} = \{i_1, ..., i_{m'}\} \subseteq [m]$ be a subset of size m'. The residual function of f for $\mathcal{S}$ and $\mathbf{x}$ is an m'-ary function $f_{\mathcal{S},\mathbf{x}} : \mathcal{X}_{i_1} \times \ldots \times \mathcal{X}_{i_{m'}} \mapsto \mathcal{Y}_{i_1} \times \ldots \times \mathcal{Y}_{i_{m'}}$, obtained from f by restricting the input variables indexed by $[m] \setminus \mathcal{S}$ to their values in $\mathbf{x}$. That is, $f_{\mathcal{S},\mathbf{x}}(x'_1, \ldots, x'_{m'}) = f(x_1, \ldots, x_m)$, where for $k \notin \mathcal{S}$ we have $x'_k = x_k$, while for $k = i_j \in \mathcal{S}$ we have $x'_k = x_j$.*

Residual FaF full-security is defined similarly to FaF full-security, with the only exception being in the ideal-world, the two adversaries receive the residual function $f_{\mathcal{I},\mathbf{x}}$ instead of a single output (all the uncorrupted parties still receive an output from T, which they output).

Before stating the result, we first define the functionalities to which we reduce the computation. For an m-party functionality f, and for $m' \in \{m - t, \ldots, m\}$, we define the m'-party functionality $f'_{m'}(\mathbf{x})$ in the security-with-identifiable-abort model as follows. Let $\mathbf{y} = (y_1, \ldots, y_m)$ be the output of $f(\mathbf{x})$. Share each y_i in an $(m - t)$-out-of-m' Shamir's secret sharing with respect to party P_i (See Construction 2.5). The output of party P_j is its respective shares of each y_i, i.e., P_j receives $(y_i[j])_{i=1}^{m}$. We next present the statement. The proof is given in Section 4.1.3.

Lemma 4.4. *Let $m, t, h^* \in \mathbb{N}$ be such that $t + h^* \leq m(n)$ and that $t < m/2$, and let f be an m-party functionality. Then there exists a protocol Π that computes f with strong perfect (t, h^*)-residual FaF full-security in the $(f'_{m-t}, \ldots, f'_m)$-hybrid model. Moreover, the protocol satisfies the following.*

1. *Standard malicious security achieved is standard security (i.e., not residual)*
2. *If $2t + h^* < m$ then Π admits (t, h^*)-FaF full-security in the $(f'_{m-t}, \ldots, f'_m)$-hybrid model.*

Remark 4.5. *Note that in general, classical generic protocols, such as the GMW protocol, will not achieve FaF full-security, even if we increase the threshold for the secret sharing scheme to $t + h^* + 1$. As an example, consider the 3-party functionality $(a, \perp, c) \mapsto a \oplus b \oplus c$, where $b \leftarrow \{0, 1\}$, and let $t, h^* = 1$. Using a 2-out-of-3 secret sharing scheme, would allow a corrupted P_1 to help P_2 to learn c. Using a 3-out-of-3 secret sharing scheme, would allow the adversary to withhold information on the output.*

We stress that even standard techniques, such as having the parties compute a functionality whose output is a secret sharing of the original output, fail to achieve security. This is due to the fact that an adversary can abort the execution forcing the parties in $\mathcal{H}$ to (possibly) learn an output. Then, after executing the same protocol with one party labeled inactive, the parties in $\mathcal{H}$ will learn an additional output, which cannot be simulated. In Section 4.1.3 we show that such protocol can achieve residual security*, namely the parties in $\mathcal{H}$ will not learn more than the function on many choices of inputs for corrupted parties.*

Assuming that OT exists, we can apply the composition theorem to combine Lemma 4.4 with Theorem 4.2 and get as a corollary that whenever an uncorrupted majority is present (i.e., $t < m/2$), any functionality can be computed with (weak) computational residual FaF full-security.

Corollary 4.6. *Let $m, t, h^* \in \mathbb{N}$ be such that $t + h^* \leq m$ and that $t < m/2$, and let f be an m-party functionality. Then, assuming OT exists, there exists a protocol Π that computes f with (weak) computational (t, h^*)-residual FaF full-security.*

1. *Standard malicious security achieved is standard security (i.e., not residual)*

2. *If $2t + h^* < m$ then Π admits (t, h^*)-FaF full-security.*

Item 2 of the above corollary concludes the positive direction of Theorem 4.1. The proof of Lemma 4.4 is given in Section 4.1.3. Before providing a proof, we first discuss some interesting consequences. One interesting family of functionalities to consider in the corollary, is the family of *no-input functionalities* (e.g., coin-tossing). Since there are no inputs, it follows that such functionalities can be computed with FaF full-security (i.e., not residual).

Corollary 4.7. *Let $m, t, h^* \in \mathbb{N}$ be such that $t + h^* \leq m$ and that $t < m/2$, and let f be an m-party no-input functionality. Then, assuming OT exists, there exists a protocol Π that computes f with (weak) computational (t, h^*)-FaF full-security.*

As a result, in the computational setting, we claim that we have separation between (weak) FaF-security and mixed-security. Recall that a mixed adversary is one that controls a subset of the parties maliciously and another subset semi-honestly. Consider the 3-party functionality $f(\perp, \perp, \perp) = (b, \perp, b)$, where $b \leftarrow \{0, 1\}$. As we proved in Corollary 4.7, this functionality can be computed with computational $(1, 1)$-FaF full-security. However, we claim that f cannot be computed with computational $(1, 1)$-mixed security.

Theorem 4.8. *No protocol computes f with $(1, 1)$-mixed full-security.*

The proof follows from a simple observation on a result by Ishai et al. [36]. They showed that for any protocol computing the functionality $g(a, \perp, c) = (a \oplus c, \perp, a \oplus c)$, where a and c are chosen uniformly at random, there exists a mixed adversary successfully attacking the protocol. Consequently, the same attack would work on any protocol computing f. As a result, we conclude that for no-input functionalities, the definition of security against mixed adversaries is *strictly stronger* than FaF security.

Even for various functionalities with inputs, Lemma 4.4 implies FaF full-security for interesting choices of parameters. For example, consider the 3-party XOR functionality. Then it can be computed with $(1, 1)$-FaF full-security since the input of the honest party can be computed by the semi-honest party's simulator.

4.1.3 Proof of Lemma 4.4

We next provide the proof of Lemma 4.4. Recall that for an m-party functionality f and for $m' \in \{m - t, \ldots, m\}$, we define the m'-party functionality $f'_{m'}(\mathbf{x})$ in the security-with-identifiable-abort model as follows. Let $\mathbf{y} = (y_1, \ldots, y_m)$ be the output of $f(\mathbf{x})$. Share each y_i in an $(m - t)$-out-of-m' Shamir's secret sharing with respect to party P_i (See Construction 2.5). The output of party P_j is its respective shares of each y_i, i.e., P_j receives $(y_i[j])_{i=1}^{m}$.

Proof of Lemma 4.4. The protocol Π in the real world is described as follows:

Protocol 4.9.
Input: Party P_i holds an input $x_i \in \mathcal{X}_i$.
Common input: Security parameter 1^n.

1. *The parties call the functionality $f'_{m'}$, where m' is the number of active parties, and the inputs of the inactive parties is set to a default value.*

2. *If the computation followed through, then the parties broadcast their shares, reconstruct the output, and halt.*[6]

3. *Otherwise, they have the identity of a corrupted party. The parties then go back to Step 1 without said party (updating m' in the process and setting its input to a default value).*

Intuitively, the protocol works since there is an honest majority, so the parties can always reconstruct the output in case the computation in Step 1 followed through. Moreover, the only information the parties receive in case of an abort during Step 1, is an output of f that is consistent with their inputs. In particular the adversary cannot add additional information to any subset of the honest parties. We next present the formal argument.

Fix an adversary $\mathcal{A}$ corrupting a set of parties $\mathcal{I} \subset \mathcal{P}$ of size at most t, and let $\mathcal{H} \subset \mathcal{P} \setminus \mathcal{I}$ be a subset of the uncorrupted parties of size at most h^*. We first construct the simulator $\mathsf{S}_{\mathcal{A}}$ for the adversary. To prove Item 1 of the "moreover" part, we will construct the simulator $\mathsf{S}_{\mathcal{A}}$ assuming that receives a single output from the trusted party. This is indeed a stronger result, since a simulator with the residual function can always simulate the simulator that received a single output. With an auxiliary input $z_{\mathcal{A}}$, the simulator $\mathsf{S}_{\mathcal{A}}$ does the following:

1. Let m' be the number of active parties. Share some garbage value m times independently as follows. Denote $y'_j = (\hat{y}_i[j])_{i=1}^{m}$ the shares held by P_j, where $\hat{y}_i$ is a garbage value, shared in a $(m-t)$-out-of-m' Shamir's secret sharing scheme with respect to party P_i.

2. Send $\mathbf{y}'_{\mathcal{I}}$ to $\mathcal{A}$ to receive the message it sends to $f'_{m'}$.

3. If $\mathcal{A}$ replied with $(\mathtt{abort}, \mathrm{P}_i)$, then go back to Step 1 with P_i labeled inactive.

4. Otherwise, $\mathcal{A}$ sent some vector of inputs $\hat{\mathbf{x}}_{\mathcal{I}}$. Pass $\hat{\mathbf{x}}_{\mathcal{I}}$ to the trusted party to receive an output $\mathbf{y}_{\mathcal{I}}$. Complete the t shares held by $\mathcal{A}$ to a sharing of the real output $\mathbf{y}_{\mathcal{I}}$ (recall that $t < m/2$ so this is possible by the properties of the secret sharing scheme).

5. Output all of the $\mathbf{y}'_{\mathcal{I}}$'s generated and the completed shares, and halt.

We next describe the simulator $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ for the adversary $\mathcal{A}_{\mathcal{H}}$ controlling the parties in $\mathcal{H}$ interacting with $\mathcal{A}$. The idea is to have the simulator use the shares generated by $\mathsf{S}_{\mathcal{A}}$ to ensure consistencies between their views. Additionally, for the last iteration, where the shares should be reconstructed to the output, we modify the shares not held by $\mathcal{A}$ so the output will also be consistent with generated view. In addition, for every abort occurred, the simulator will use the residual function to hand over to $\mathcal{H}$ the output of that iteration. Formally, given an auxiliary input $z_{\mathcal{H}}$, $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ operates as follows.

1. Receive the residual function $f_{\mathcal{I},\mathbf{x}}$ from the trusted party, and receive $(\mathbf{x}_{\mathcal{I}}, r, z_{\mathcal{A}})$ – $\mathsf{S}_{\mathcal{A}}$'s input, randomness, and the auxiliary input, respectively.

[6]For this step to work, we need to assume that the adversary does not change its shares. We can force it to send the correct shares using standard techniques. One way to do so is to sign each output of each $f'_{m'}$ using a MAC and give the other parties the key for verification. For the sake of clarity of presentation, however, we decide to skip this and assume that the corrupted parties are using correct shares.

2. Apply $\mathsf{S}_{\mathcal{A}}$ to receive its view, which consists of $\mathbf{y}'_{\mathcal{I}}$ – shares of some values, held by the adversary.

3. Query $\mathcal{A}$ on each $\mathbf{y}'_{\mathcal{I}}$ to receive the messages it sends to $\mathcal{H}$, and in case of an abort, get the identity of a corrupted party.

4. Complete each $\mathbf{y}'_{\mathcal{I}}$ to shares of an output $\hat{\mathbf{y}}$ computed using the residual function $f_{\mathcal{I},\mathbf{x}}$ (fixing the input of the inactive parties to be a default value, and input of the active corrupted parties according to the choice of $\mathcal{A}$), so that the last $\mathbf{y}'_{\mathcal{I}}$ is completed to shares of the real output. Note that by the properties of the secret sharing scheme, this can be done efficiently.

5. Output all of the completed shares and the messages sent by $\mathcal{A}$, and halt.

In every iteration, the view generated by $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ is consistent with the view generated by the malicious simulator $\mathsf{S}_{\mathcal{A}}$. Moreover, they send to $\mathcal{A}$ the exactly the same messages, hence they will receive the same identities of the aborting parties, and inputs given to the functionalities $f'_{m'}$. Since this is generated with the same distribution as in the real-world, we conclude that joint view of the two adversaries with the output of the honest parties, is identically distributed in both worlds.

Finally, in order to see why Item 2 of the "moreover" part is true, observe that if $2t + h^* < m$ then $t + h^* < m - t$, implying that the number of shares that can be held by the $\mathcal{A}$ and $\mathcal{H}$ is smaller than the secret sharing threshold. Thus, $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ can use random shares for each iteration (except the last iteration), without using the output. □

4.2 Impossibility Result

In this section, we prove the negative direction of Theorem 4.1. Specifically, we prove the following lemma.

Lemma 4.10. *Let $m, t, h^* \in \mathbb{N}$ be such that $2t + h^* = m$. Then there exists an m-party functionality that no protocol computes it with (weak) computational (t, h^*)-FaF full-security. Moreover, the claim holds even assuming the availability of simultaneous broadcast.*

For the proof, we first show that it holds for the 3-party case where $t, h^* = 1$. Then, using a player-partitioning argument, we generalize the result to more than three parties. The following lemma states the result for the 3-party case. Throughout the remainder of the section, we denote the parties by A, B, and C.

Lemma 4.11. *Assume that one-way permutation exists. Then there exists a 3-party functionality that no protocol computes it with (weak) computational $(1, 1)$-FaF full-security. Moreover, the following hold*

1. *The malicious adversary we construct corrupts either* A *or* C*, while the remaining third party* B *will be in $\mathcal{H}$.*

2. *The claim holds even assuming the availability of simultaneous broadcast.*

The proof of Lemma 4.10 is deferred to Appendix A. We next give an overview of the proof of Lemma 4.11. We assume that each round is composed of 3 broadcast messages, the first sent by A, the second sent by B, and the third by C (this is without loss of generality, as we allow the adversary to be rushing). Intuitively, the proof is done as follows. By an averaging argument there must exists a round where two parties, say A and B, together can reconstruct the output with significantly higher probability than C and B. We then have A act honestly (using the original input it held) and abort at that round. As a result, with high probability the output of C will change. Finally, A will send its entire view to B, allowing it to recover the correct entry with significantly higher probability than C. We show that for an appropriate functionality, the advantage of the pair (A, B) over (C, B) cannot be simulated.

Proof of Lemma 4.11. Let $f = \{f_n : \{0,1\}^n \mapsto \{0,1\}^n\}_{n\in\mathbb{N}}$ be a one-way permutation. Define the symmetric 3-party functionality $\mathsf{Swap} = \{\mathsf{Swap}_n : \{0,1\}^n \times \{0,1\}^{2n} \times \{0,1\}^n \mapsto \{0,1\}^{2n}\}_{n\in\mathbb{N}}$ as follows. Parties A and C each hold a string $a, c \in \{0,1\}^n$ respectively. Party B holds two strings $y_{\mathrm{A}}, y_{\mathrm{C}} \in \{0,1\}^n$. The output is then defined to be

$$\mathsf{Swap}_n\left(a, \left(y_{\mathrm{A}}, y_{\mathrm{C}}\right), c\right) = \begin{cases} (a, c) & \text{if } f_n(a) = y_{\mathrm{A}} \text{ and } f_n(c) = y_{\mathrm{C}} \\ \perp & \text{otherwise} \end{cases}$$

Assume for the sake of contradiction that there exists a 3-party protocol Π that computes Swap with computational $(1,1)$-FaF full-security. We fix a security parameter n, we let r denote the number of rounds in Π, and consider an evaluation of Swap with the output being (a, c). Formally, we consider the following distribution over the inputs.

- a, c are each selected from $\{0,1\}^n$ uniformly at random and independently.
- $y_{\mathrm{A}} = f_n(a)$ and $y_{\mathrm{C}} = f_n(c)$.

For $i \in \{0, \ldots, r\}$ let a_i be the final output of A assuming that C aborted after sending i messages. Similarly, for $i \in \{0, \ldots, r\}$ we define c_i to be the final output of C assuming that A aborted after sending i messages. Observe that a_r and c_r are the outputs of A and C respectively. We first claim that there exists a round where either A and B gain an advantage in computing the correct output, or C and B gain this advantage.

Claim 4.12. *Either there exists $i \in \{0, \ldots, r\}$ such that*

$$\Pr\left[a_i = (a, c)\right] - \Pr\left[c_i = (a, c)\right] \geq \frac{1 - \operatorname{neg}(n)}{2r + 1},$$

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

$$\Pr\left[c_i = (a, c)\right] - \Pr\left[a_{i-1} = (a, c)\right] \geq \frac{1 - \operatorname{neg}(n)}{2r + 1}.$$

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

The proof is done using a simple averaging argument, and is proven below. We first use this fact to show an attack.

Assume without loss of generality that there exists an $i \in [r]$ such that the former equality holds (the other case is done analogously). Define a malicious adversary $\mathcal{A}$ as follows. For the security parameter n, it receives as auxiliary input the round i. Now, $\mathcal{A}$ corrupts A and have it act honestly (using the party's original input a) up to and including round i. After receiving the i-th message, the adversary instructs A to abort. Finally, the adversary sends its entire view to B. We next show that no pair of simulators $\mathsf{S}_{\mathcal{A}}$ and $\mathsf{S}_{\mathcal{A},\mathrm{B}}$ can produce views for $\mathcal{A}$ and B so that

Equations (1) and (2) would hold. For that, we assume towards contradiction that such simulators do exist. Let $a^* \in \{0,1\}^n$ be the input that $\mathsf{S}_{\mathcal{A}}$ sent to the trusted party. Additionally, denote $q = \Pr[c_i = (a,c)]$.

We next separate into two cases. For the first case, let us assume that $\Pr[a^* = a] \geq q + 1/p(n)$ for some polynomial $p(\cdot)$ for infinitely many n's. Let $\textsc{out}_{\mathrm{C}}^{\mathrm{IDEAL}}$ be the output of C in the ideal world. Since f_n is a permutation we have that

$$\Pr\left[\textsc{out}_{\mathrm{C}}^{\mathrm{IDEAL}} = (a,c)\right] = \Pr[a^* = a] \geq q + 1/p(n).$$

Thus, by comparing the output of C to (a,c) it is possible to distinguish the real from the ideal with advantage at least $1/p(n)$.

For the second case, we assume that $\Pr[a^* = a] \leq q + \mathrm{neg}(n)$. Here we show how to distinguish between the view of B in the real world from its ideal world counterpart. Recall that in the real world $\mathcal{A}$ sent its view to B. Let M be the algorithm specified by the protocol, that A and B use to compute their output assuming C has aborted. Namely, M outputs a_i in the real world. By Claim 4.12 it holds that $\Pr[a_i = (a,c)] \geq q + \frac{1-\mathrm{neg}(n)}{2r+1}$. We next consider the ideal world. Let V be the view generated by $\mathsf{S}_{\mathcal{A},\mathrm{B}}$. We claim that

$$\Pr[\mathrm{M}(V) = (a,c) \wedge a^* \neq a] \leq \mathrm{neg}(n).$$

Indeed, since f_n 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_n is one-way it follows that if $\mathrm{M}(V)$ did output (a,c), then it can be used to break the security of f_n. This can be done by sampling $a \leftarrow \{0,1\}^n$, computing $f(a)$, and finally, compute a view V using the simulators and apply M to it (if a^* computed by $\mathsf{S}_{\mathcal{A}}$ equals to a then abort). We conclude that

$$\begin{aligned}\Pr[\mathrm{M}(V) = (a,c)] &= \Pr[\mathrm{M}(V) = (a,c) \wedge a^* = a] + \Pr[\mathrm{M}(V) = (a,c) \wedge a^* \neq a]\\ &\leq \Pr[a^* = a] + \mathrm{neg}(n)\\ &\leq q + \mathrm{neg}(n).\end{aligned}$$

Therefore, by applying M to the view it is possible to distinguish with advantage at least $\frac{1-\mathrm{neg}(n)}{2r+1} - \mathrm{neg}(n)$. To conclude the proof we next prove Claim 4.12.

Proof of Claim 4.12. The proof follows by the following averaging argument. By correctness and the fact that f_n is one-way, it follows that

$$\begin{aligned}1 - \mathrm{neg}(n) &\leq \Pr[a_r = (a,c)] - \Pr[c_0 = (a,c)]\\ &= \sum_{i=0}^{r}(\Pr[a_i = (a,c)] - \Pr[c_i = (a,c)]) + \sum_{i=1}^{r}(\Pr[c_i = (a,c)] - \Pr[a_{i-1} = (a,c)])\end{aligned}$$

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

Finally, in order to see why Item 2 is true, observe that the attack is not based on the view of $\mathcal{A}$, hence the same attack works assuming simultaneous broadcast.

$\square$

Remark 4.13. *Intuitively, we showed that in the real world the parties* A *and* B *hold more information on the output, than what* B *and* C *hold. To make this statement formal, observe that the proof in fact shows that* Swap *cannot be computed with* fairness*. Roughly, for fairness to hold we require that either all parties receive an output, or none of them do. To see this, observe that for the functionality at hand, aborting in the ideal world is the same as sending a different input a. Therefore the attack cannot be simulated. We present the formal definition of fairness in Appendix B.1.*

5 Comparison Between FaF-Security and Other Definitions

In this section, we compare the notion of FaF-security to other existing notions. In Section 5.1, we investigate how FaF-security relates to classical full-security. In Section 5.2, we review the differences between our notion and the notion of mixed adversaries. In the mixed-adversary scenario, a single adversary controls a set $\mathcal{I}$ of parties, however, within $\mathcal{I}$ different limitations are imposed on the behavior (deviation) of different parties. In Section 5.3, we show that strong FaF-security is a strictly stronger notion than (weak) FaF-security.

5.1 The Relation Between FaF-Security and Standard Full-Security

We start with comparing FaF-security to the standard definition. It is easy to see that standard t-security does not imply in general (t, h^*)-FaF full-security, even for functionalities with no inputs. Consider the following example. Let f be a 3-party *no-input* functionality defined as $(\bot, \bot, \bot) \mapsto (\bot, \bot, r)$ where $r \leftarrow \{0,1\}^n$, and let $t, h^* = 1$. Consider the following protocol: P_1 and P_2 sample $r_1, r_2 \leftarrow \{0,1\}^n$, respectively and send the random strings to P_3. The output of P_3 is then $r_1 \oplus r_2$.

It is easy to see that the protocol computes f with perfect full-security tolerating a single corruption. However, a malicious P_1 can send r_1 to P_2 as well, thereby allowing P_2 to learn P_3's output. Indeed, this protocol is insecure according to Definition 3.3. Obviously, (t, h^*)-FaF-security readily implies the classical t-security counterpart. Conversely, one might expect that classical $(t + h^*)$-security must imply (t, h^*)-FaF-security. We next show that this is not the case in general. We present an example of a protocol that admits traditional malicious security against t corruptions, however, it does not admit $(t-1, 1)$-FaF-security. Intuitively, this somewhat surprising state of affairs is made possible by the fact that in $(t-1, 1)$-FaF-security *both* the attacker *and* the two simulators are weaker.

The following example is a simple extension of the known example (cf., [10]), showing that for standard security, there exists a maliciously secure protocol (for computing the two-party, one-sided OR function), but none semi-honest secure.

Example 5.1. *Let* A, B, *and* C *be three parties with inputs $a, b, c \in \{0,1\}$ respectively. Consider the 3-party functionality* $\mathsf{3OR} : \{0,1\}^3 \mapsto \{0,1\}^3$ *defined as* $\mathsf{3OR}(a, b, c) = (\bot, \bot, (a \oplus b) \vee c)$*, with the following protocol for computing it. In the first round, parties* A *and* B *both select shares for their respective inputs with each other. That is,* A *selects* $a_1 \leftarrow \{0,1\}$ *and sends* $a_2 = a \oplus a_1$ *to* B*, and* B *selects* $b_2 \leftarrow \{0,1\}$ *and sends* $b_1 = b \oplus b_2$ *to* A*. In the second round,* A *sends* $a_1 \oplus b_1$ *to* C *and* B *sends* $a_2 \oplus b_2$ *to* C*. Party* C *outputs* $(a_1 \oplus b_1 \oplus a_2 \oplus b_2) \vee c$*.*

We first claim that the protocol computes $\mathsf{3OR}$ *with* perfect *full-security tolerating coalitions of size at most 2. Indeed, an adversary that maliciously corrupts* A, B, *or both, learns nothing and can be simulated by selecting the inputs defined by the shared values. An adversary that maliciously* *corrupts* C *can be simulated by sending* $c = 0$ *to the trusted party, and as a result, learning the same information as in the protocol. For example, corrupting* A *and* C *and sending* $a, 0$ *(resp.) to the trusted party, the adversary learns* b*.*

We argue that although the protocol is 2-secure in the standard definition, it does not compute 3OR *with* $(1,1)$*-FaF full-security. Specifically, a semi-honest* C *cannot be simulated. Take for example, an adversary* $\mathcal{A}$ *that corrupts* A *maliciously and let* $\mathcal{H} = \{\mathrm{C}\}$*. In the real-world,* $\mathcal{A}$ *can reveal* b *to* C*. However, in the ideal-world, this cannot be simulated (when* $c = 1$*).*

Remark 5.2. *Example 5.1 shows that "moving" a party from being malicious to being semi-honest (i.e., taking a party from* $\mathcal{I}$ *and moving it to* $\mathcal{H}$*) could potentially break the security of the protocol. Similarly to [10], it is arguably natural to consider a definition that requires the protocol to be* (t, h^*)*-FaF-security if and only if it is* $(t-1, h^*+1)$*-FaF-security. Our definition does not impose this extra requirement, however, all of our protocols satisfy it. In Section 7 we FaF-security under this restriction in the information-theoretic setting.*

In contrast to the above example, we claim that adaptive $(t + h^*)$-security does imply strong (t, h^*)-FaF full-security. Intuitively, this follows from the fact that an adaptive adversary is allowed to corrupt some of the parties after the execution of the protocol terminated. We formulate the theorem for the full-security setting, however, we stress that it also holds in the security with (identifiable) abort setting. The definition of adaptive security is given in Appendix B.2.

Theorem 5.3. *Let* `type` $\in$ {computational, statistical, perfect} *and let* Π *be an* m*-party protocol computing some* m*-party functionality* f *with* `type` *adaptive* $(t + h^*)$*-security. Then* Π *computes* f *with* `type` (t, h^*)*-FaF full-security.*

Proof Sketch.[Proofsketch of Theorem 5.3] Let $\mathcal{A}$ be a non-adaptive malicious adversary corrupting a set $\mathcal{I} \subseteq \mathcal{P}$ of the parties, and let $\mathcal{H} \subseteq \mathcal{P} \setminus \mathcal{I}$ be a subset of the remaining parties. Define the following adaptive adversary $\mathcal{A}'$. $\mathcal{A}'$ runs $\mathcal{A}$ during the execution of the protocol. After the protocol had terminated, $\mathcal{A}'$ interacts with the environment $\mathcal{Z}$ that always request to corrupt the parties in $\mathcal{H}$. Here, $\mathcal{A}'$ answers with their view, which $\mathcal{Z}$ then outputs. By the security assumption, there exists an *adaptive* simulator $\mathsf{S}_{\mathcal{A}',\mathcal{Z}}$ for $\mathcal{A}'$ that interacts with $\mathcal{Z}$. Observe that as the identities of the *corrupted* parties (i.e., those in $\mathcal{I}$) appear in the global output, $\mathsf{S}_{\mathcal{A}',\mathcal{Z}}$ must eventually corrupt exactly the parties in $\mathcal{I}$.

We next construct the two simulators $\mathsf{S}_{\mathcal{A}}$ and $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ in order to prove FaF-security. The malicious simulator $\mathsf{S}_{\mathcal{A}}$ will run $\mathsf{S}_{\mathcal{A}',\mathcal{Z}}$, and when $\mathsf{S}_{\mathcal{A}'}$ corrupts a new party P, $\mathsf{S}_{\mathcal{A}}$ provides it with the input of P. Finally, $\mathsf{S}_{\mathcal{A}}$ outputs whatever $\mathsf{S}_{\mathcal{A}',\mathcal{Z}}$ outputs. The semi-honest simulator $\mathsf{S}_{\mathcal{A},\mathcal{H}}$, will interact with $\mathsf{S}_{\mathcal{A}',\mathcal{Z}}$ in the post-protocol corruption phase, acting as the environment $\mathcal{Z}$, while using the randomness of $\mathsf{S}_{\mathcal{A}}$ to ensure consistency. Since $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ has the inputs and randomness of $\mathsf{S}_{\mathcal{A}}$, the joint output of the two simulators will be distributed the same as the joint output of $\mathsf{S}_{\mathcal{A}',\mathcal{Z}}$ and $\mathcal{Z}$. □

By applying recent results on adaptive security, we get that there exist constant-round protocol that are FaF secure-with-abort [16, 13, 25].

5.2 The Relation Between FaF-Security and Mixed-Security

The notion of "mixed adversaries" [24, 48] considers a single entity that corrupts a subset $\mathcal{I}$ maliciously, and another subset $\mathcal{H}$ semi-honestly.[7] A simulator for a mixed adversary, is a single simulator controlling the parties in $\mathcal{I} \cup \mathcal{H}$. This simulator is restricted so to only be allowed to change inputs for the parties in $\mathcal{I}$ (i.e., the simulator is not allowed to change the inputs for the parties in $\mathcal{H}$). We say that a protocol has computational (t, h^*)-mixed full-security, if Equation (2) is written with respect to a mixed adversary and its simulator.

In comparison, FaF-security can be viewed as if there are two *distinct* adversaries – one malicious and one semi-honest, making it a natural question to compare the two definitions. One might expect that (t, h^*)-mixed full-security would imply (t, h^*)-FaF full-security. However, similarly to the case with standard security, we show the that this is not generally the case in the *computational* setting (note that the protocol from Example 5.1 is not $(1, 1)$-mixed secure).

Example 5.4. *Consider the 5-party functionality* $f : (\{0,1\}^n)^3 \times \emptyset^2 \mapsto (\{0,1\}^n)^2 \times \emptyset^3$ *whose output on input* $(x_1, x_2, x_3, \perp, \perp)$*, is defined as follows. If* $x_1 = x_2$*, then* P_1 *and* P_2 *will each receive a share of a 2-out-of-2 secret sharing of* x_3*, i.e.,* P_1 *will receive* $x_3[1]$ *and* P_2 *will receive* $x_3[2]$*. If* $x_1 \neq x_2$ *then* P_1 *and* P_2 *will each receive a string of length* n *chosen uniformly at random and independently. In both cases, all other 3 parties will receive no output. We next show a protocol that is secure against any adversary corrupting at most 2 parties (including mixed adversaries), yet it does not admits* $(1, 1)$*-FaF full-security. In the following we let* $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ *be a non-malleable and semantically secure public-key encryption scheme [22].*

Protocol 5.5.

1. *The parties will compute a functionality whose output to* P_i *for* $i \in \{1, 2, 3\}$ *is* pk*, and for party* P_i*, for* $i \in \{4, 5\}$ *is* $(\mathsf{pk}, \mathsf{sk}[i])$*, where the* $\mathsf{sk}[i]$*s are shares of* sk *in a 2-out-of-2 secret sharing, and where* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathrm{Gen}(1^n)$*. This can be done using, say the GMW protocol [29].*

2. P_2 *sends* $c_2 \leftarrow \mathrm{Enc}_{\mathsf{pk}}(x_2, 2)$ *to* P_1*.*

3. *The parties compute the following 5-party functionality* g*. The input of* P_1 *is* $c_1 \leftarrow \mathrm{Enc}_{\mathsf{pk}}(x_1, 1)$*, the input of* P_2 *is* c_2*, and the input of* P_3 *is* x_3*. The input of* P_i*, for* $i \in \{4, 5\}$*, is the pair* $(\mathsf{pk}, \mathsf{sk}[i])$*.*

   The output is defined as follows. P_3*,* P_4*, and* P_5 *receive no output.*

   - *If* $\mathrm{Dec}_{\mathsf{sk}}(c_i) = (x_i, i)$*, for every* $i \in \{1, 2, 3\}$ *and* $x_1 = x_2$*, then* P_1 *will receive* $x_3[1]$ *and* P_2 *will receive* $x_3[2]$*.*
   - *Else, if* $\mathrm{Dec}_{\mathsf{sk}}(c_1) = (x_1, 2)$*,* $\mathrm{Dec}_{\mathsf{sk}}(c_2) = (x_2, 2)$*, and* $x_1 = x_2$*, then* P_1 *will receive a random string* $r \in \{0,1\}^n$ *and* P_2 *will receive* $(x_3[1], x_3[2])$*.*
   - *Otherwise, both* P_1 *and* P_2 *will receive random strings* $r_1, r_2 \in \{0,1\}^n$ *respectively, chosen independently and uniformly.*

   As in Step 1, this can be done using the GMW protocol [29].

[7]There are various types of mixed adversaries one can consider. For example, [34] also gave the adversary the ability to fail-corrupt parties, based on its adversarial structure. Here, we only consider the notion considered by [24].

4. P_1 *output what it received from* g*. If* P_2 *received one random string* r_2 *from* g *then output* r_2*, and if* P_2 *received two random strings from* g*, then output the second one.*

Claim 5.6. *Protocol 5.5 computes* f *with computational 2-security and with computational* $(1,1)$*-mixed security, yet it does not compute* f *with computational* $(1,1)$*-FaF full-security.*

Proof. We first prove the security properties of the protocol. For all of the adversaries, the corresponding simulator we construct will not change the inputs they send to T, unless the adversary changes its input. As a result, the 2-security of the protocol implies that it is also $(1,1)$-mixed fully secure.

Any adversary corrupting 2 parties from $\{\mathrm{P}_3, \mathrm{P}_4, \mathrm{P}_5\}$, or corrupting P_2 and one party from $\{\mathrm{P}_3, \mathrm{P}_4, \mathrm{P}_5\}$, can be simulated using the composition theorem of Canetti [14] for both Steps 1 and 3. That fact that there is an honest majority implies the existence of the simulator.

Let $\mathcal{A}$ be an adversary corrupting P_1 and P_2. The simulator $\mathsf{S}_{\mathcal{A}}$ works as follows. Compute a pair of keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathrm{Gen}(1^n)$ and send pk to $\mathcal{A}$. Let c_1 and c_2 be the inputs of P_1 and P_2 respectively, that $\mathcal{A}$ chose for the computation of g. If $c_1 = \mathrm{Enc}_{\mathsf{pk}}(x_1, 1)$, $c_2 = \mathrm{Enc}_{\mathsf{pk}}(x_2, 2)$, and $x_1 = x_2$, then send (x_1, x_2) to the trusted party T to receive two shares of x_3, hand them over to $\mathcal{A}$, output the view and halt. Else, if $c_1 = c_2 = \mathrm{Enc}_{\mathsf{pk}}(x_2, 2)$, then send (x_2, x_2) to the trusted party T to receive two shares of x_3. Sample a random string r, hand and the two shares to $\mathcal{A}$, output the view and halt. Otherwise, give to $\mathcal{A}$ two random and independent strings $r_1, r_2 \in \{0,1\}^n$, output the view and halt.

Next, consider an adversary $\mathcal{A}'$ corrupting P_1 and a party from $\{\mathrm{P}_3, \mathrm{P}_4, \mathrm{P}_5\}$. We construct its simulator $\mathsf{S}_{\mathcal{A}'}$ as follows. Compute a pair of keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathrm{Gen}(1^n)$, and use [14] to simulate Step 1. Compute an encryption $c \leftarrow \mathrm{Enc}_{\mathsf{pk}}(0^n, 2)$ and send both pk and c to $\mathcal{A}'$. Let c' be the ciphertext chosen by $\mathcal{A}$ as the input for P_1 in Step 3. Decrypt c' to recover an input x_1', which the simulator then sends to T. Finally, simulate Step 3 using [14] with input c' for P_1. Since the encryption scheme is semantically secure and $\mathcal{A}$ does not hold sk, it follows that the value of c' in the real-world is computationally indistinguishable from its ideal-world value. Moreover, as the encryption scheme is non-malleable, if $x_1 \neq x_2$ then $c' = \mathrm{Enc}_{\mathsf{pk}}(x_2, 1)$ holds with negligible probability. Therefore, in this case the output of P_1 and P_2 will be random strings., as given by T. In the other case, where $x_1 = x_2$, the output given by T to P_1 and P_2 are sharing of x_3. In addition, the output of g will be a sharing of x_3 as well. Security follows.

We next show that Protocol 5.5 is not $(1,1)$-FaF fully-secure. This follows from the observation that a corrupt P_1 can use c_2 as its input for the computation of g. Thus, party P_2 receives two shares of x_3 in a 2-out-of-2 secret sharing, thereby learning x_3. In the ideal-world, any simulator for P_2 will receive from T a share of x_3 only if $x_1 = x_2$. Since for the case where $x_1 \neq x_2$ this happens only with negligible probability (with the probability being over the choices of the simulator $\mathsf{S}_{\mathcal{A}}$ for the *adversary's input*), we conclude that no such simulator exists. □

In Section 7 we further compare the two definitions in the information theoretic setting, with respect to non-threshold adversarial structures.

5.3 Comparison Between (Weak) FaF-Security and Strong FaF-Security

In this section, we separate the notion of (weak) FaF-security from strong FaF-security in the computational setting. Specifically, we show a protocol that admits (weak) FaF-security, yet it does not admit strong FaF-security. We assume we have available a commitment scheme. Consider the 3-party functionality f mapping $(\perp, b, \perp) \mapsto (\perp, \perp, b)$, where $b \in \{0,1\}$, and let $t, h^* = 1$. Consider the following protocol: P_2 broadcasts a commitment to b, and then sends the decommitment only to P_3.

Claim 5.7. *The above protocol computes f with (weak) computational $(1,1)$-FaF full-security, yet does not provide strong computational $(1,1)$-FaF full-security.*

Proof. We first show that the protocol computes f with weak computational $(1,1)$-FaF full-security. We go over all 6 possible (malicious and semi-honest) corruptions and construct two simulators for each case.

First, suppose $\mathcal{A}$ corrupts P_1. Its simulator $\mathsf{S}_{\mathcal{A}}$ will output a commitment to 0 and halt. By the hiding property of the commitment scheme, the simulator's output will be indistinguishable from the adversary's real-world view. The semi-honest simulator $\mathsf{S}_{\mathcal{A},\mathrm{P}_2}$ will query the adversary on the commitment of its input b to receive the messages it sends to P_2, output them and halt. The other semi-honest simulator $\mathsf{S}_{\mathcal{A},\mathrm{P}_3}$ will receive b from the trusted party T, and do the same as $\mathsf{S}_{\mathcal{A},\mathrm{P}_2}$.

Next, suppose $\mathcal{A}$ corrupts P_2. Its simulator $\mathsf{S}_{\mathcal{A}}$ will ask $\mathcal{A}$ for a commitment and a decommitment, supposedly sent to P_3, and recover the committed value $\hat{b}$ (if the decommitment is invalid or the adversary did not send a message, then choose $\hat{b}$ to be a default value). It then sends $\hat{b}$ to T and halt. By the binding property of the commitment scheme, the adversary cannot decommit to a different value, hence the output distribution of P_3 in both worlds are indistinguishable. The semi-honest simulators $\mathsf{S}_{\mathcal{A},\mathrm{P}_1}$ and $\mathsf{S}_{\mathcal{A},\mathrm{P}_3}$, which hold $\mathsf{S}_{\mathcal{A}}$'s input, randomness, and auxiliary input, query $\mathcal{A}$ to receive the messages it sends to P_1 and P_3, respectively, output them, and halt.

Finally, suppose $\mathcal{A}$ corrupts P_3. Its simulator $\mathsf{S}_{\mathcal{A}}$ will output a commitment and the corresponding decommitment to the output b received from T and halt. The semi-honest simulator $\mathsf{S}_{\mathcal{A},\mathrm{P}_2}$, which holds b as well, will query $\mathcal{A}$ and output the messages received from it. The other simulator $\mathsf{S}_{\mathcal{A},\mathrm{P}_1}$ will do the same with the addition of outputting a commitment to b.

We now show that the protocol is not strong $(1,1)$-FaF fully-secure. Consider an adversary $\mathcal{A}$, which corrupts P_1 and *does nothing*, and let $\mathcal{H} = \{\mathrm{P}_3\}$. Assume towards contradiction that the pair of simulators $\mathsf{S}_{\mathcal{A}}$ and $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ exists so that $\left\{\mathrm{IDEAL}_b^{\mathsf{S}_{\mathcal{A}},\mathsf{S}_{\mathcal{A},\mathcal{H}}}\right\}_{b\in\{0,1\}}$ and $\left\{\mathrm{REAL}_b^{\mathcal{A},\mathcal{A}_{\mathcal{H}}}\right\}_{b\in\{0,1\}}$ are indistinguishable. We show how to construct a sender that breaks the binding property of the commitment scheme. The sender sends to the receiver the commitment c' generated by $\mathsf{S}_{\mathcal{A}}$. The decommitment will be the commitment c and its decommitment d, generated by $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ on input $b = 0$ (and $\mathsf{S}_{\mathcal{A}}$'s randomness). By assumed strong security of the protocol, the view $(c', (c, d))$ of the receiver is indistinguishable from $(c, (c, d))$, and therefore it will output 0. However, the same holds when applying $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ on input $b = 1$ in the decommitment phase. Thus, the sender can force the output of the receiver in the decommitment phase to be any value it chooses with probability $1 - \mu(n)$, for some negligible function $\mu(\cdot)$. $\square$

One consequence of the above claim, is that protocols where the parties commit to their inputs, e.g., the GMW protocol, will not satisfy strong FaF-security in general.

6 Low Round Complexity

In this section, we study the round complexity of general functionalities with computational FaF full-security. The combination of the two theorems we give below yields that the optimal round complexity of a protocol admitting $(1,1)$-FaF full-security is three rounds (assuming at least 9 parties are present). Roughly, this is done by showing that in a two-round protocol, an adversary can "help" an honest party to gain information about another honest party's input, where in 3-rounds we can overcome that.

Impossibility of two-round FaF-secure computation. The following theorem states that it is impossible to construct a generic 2-rounds protocol, that is $(1,1)$-FaF secure – one malicious and one semi-honest that are *not* colluding. This result is a simple observation regarding the impossibility of computing the AND functionality in two rounds against 2 corrupted parties, proved by Gennaro et al. [27]. For the sake of completeness, we provide the proof. Just as in [27], the adversary we construct is not rushing, hence the theorem holds even in the fully synchronous model, where all messages are received simultaneously. One can arguably view this theorem as an evident for the necessity of leakage from honest parties private inputs in the two-round protocol of Ishai et al. [35].

Theorem 6.1. *Let $m \geq 3$ and let $\mathsf{AND}(x_1, x_2, \perp, \ldots, \perp) = x_1 \wedge x_2$ with $x_1, x_2 \in \{0,1\}$ be an m-party functionality. There is no 2-round protocol that computes* AND *with computational $(1,1)$-FaF full-security, for an arbitrarily large number of players $m \geq 3$, even in the fully synchronous model.*

The proof of Theorem 6.1 is given in Section 6.1. We next present the complementary possibility result for three-round computation. In fact, we show that we can achieve strong FaF-security.

Theorem 6.2. *Let $m, t, h^* \in \mathbb{N}$ be three constants satisfying $5t + 3h^* < m$. Let f be a polynomial time computable m-party functionality. Then, under the assumption that pseudorandom generators exist, there exists a 3-round protocol that computes f with* strong computational *(t, h^*)-FaF full-security. Moreover, the construction is fully black-box.*

The protocol, alongside its proof of security, is given in Section 6.2.

6.1 Proof of Theorem 6.1

We next give the proof of Theorem 6.1. For the sake of presentation, in the following we omit the security parameter from the notation. Throughout the section, we fix $m \geq 3$ and the m-party functionality $\mathsf{AND}(x_1, x_2, \perp, \ldots, \perp) = x_1 \wedge x_2$. For the proof we require the following simple lemma proved in [27].

Lemma 6.3 ([27, Lemma 2])**.** *Fix a 2-round protocol for computing the* AND *functionality with computational 1-security. For $b \in \{0,1\}$ let B^b be the distribution of the messages broadcast by an honest* P_2*, on input $x_2 = b$ in round 1. Further, let M_1^b be the distribution over the private messages an honest* P_2 *sends* to P_1 only, *on input $x_2 = b$ in round 1. Then, (B^0, M_1^0) and (B^1, M_1^1) are computationally indistinguishable.*

Proof of Theorem 6.1. Assume for the sake of contradiction that there exists a 2-rounds protocol that computes AND with $(1,1)$-FaF full-security in the computational setting. For $b \in \{0,1\}$, let B^b be the distribution of the broadcast messages sent by an honest P_2 on input $x_2 = b$ in round 1, and let M^b be the distribution of the private messages, sent by an honest P_2 on input $x_2 = b$ in round 1. We further denote by M_i^b the distribution of the messages private messages sent by an honest P_2 *to* P_i *only*, on input $x_2 = b$ in round 1. In the following, we consider a scenario where P_1 chooses its input x_1 uniformly at random from $\{0,1\}$.

We next show that in the ideal world, the best prediction each party has for x_1 is the output Out that it receives from the trusted party. In the real world, however, an adversary (that corrupts P_2) can help some honest party to have a better prediction for x_1, in the sense that the honest party's view will have a better correlation with x_1 then the correlation of Out and x_1. Formally, for a distribution (B, M) we let

$$\begin{aligned}\mathrm{Cor}(B, M) = \Big| &\Pr\left[\mathrm{Out} = 1 \mid x_1 = 1, (B^{x_2}, M^{x_2}) = (B, M)\right] \\ &- \Pr\left[\mathrm{Out} = 1 \mid x_1 = 0, (B^{x_2}, M^{x_2}) = (B, M)\right] \Big|.\end{aligned}$$

That is, $\mathrm{Cor}(B, M)$ denotes the correlation of the output Out and x_1, conditioned on the distribution of the messages sent by P_2 are chosen according to the distribution (B, M). The idea is to have a corrupt P_2 send messages so as to lower the correlation. Then, by sending its view to some honest party P, P can compute a different output that have a higher correlation with x_1. We next formalize this intuition. Consider the following four quantities:

$$\begin{aligned}q_1 &= \mathrm{Cor}\left(B^0, M_1^0, M_2^0, \ldots, M_m^0\right), \\ q_2 &= \mathrm{Cor}\left(B^0, M_1^0, 0, \ldots, 0\right), \\ q_3 &= \mathrm{Cor}\left(B^1, M_1^1, 0, \ldots, 0\right), \\ q_4 &= \mathrm{Cor}\left(B^1, M_1^1, M_2^1, \ldots, M_m^1\right).\end{aligned}$$

Namely, q_1 and q_4 denote the correlation in an honest execution of P_2 on input 0 and 1 respectively, while q_2 and q_3 denote the correlation when P_2's private messages to $\mathrm{P}_3, \ldots, \mathrm{P}_m$ are all 0. By the correctness of the protocol, it follows that there exists a negligible function $\mu(\cdot)$ such that $q_1 = \mu$ and $q_4 = 1 - \mu$. Additionally, by Lemma 6.3 it follows that the difference between q_2 and q_3 is μ. Therefore, either the difference between q_1 and q_2 is at least $1/2 - \mu$, or the difference between q_3 and q_4 is at least $1/2 - \mu$. Assume without loss of generality that the former holds. By a hybrid-argument, there exists $2 \leq i \leq m - 1$, for which

$$\mathrm{Cor}\left(B^0, M_1^0, \ldots, M_i^0, 0, \ldots, 0\right) - \mathrm{Cor}\left(B^0, M_1^0, \ldots, M_{i+1}^0, 0, \ldots, 0\right) \geq \frac{1/2 - \mu}{m - 2}.$$

We are now ready to define the adversary $\mathcal{A}$: Corrupt P_2 and instruct the party to send his messages in the first round according to $(B^0, M_1^0, \ldots, M_{i+1}^0, 0, \ldots, 0)$. In the second round, P_2 will send the honest party P_{i+1} its view. Intuitively, since P_{i+1} is honest, in the second round it will act as if it received M_{i+1}^0. This guarantees a lower correlation between the output Out of the honest parties other than P_{i+1} and x_1. However, as P_{i+1} received the view of P_2 privately, it allows it to compute a (possibly) new output Out′ when P_2 plays according to $(B^0, M_1^0, \ldots, M_i^0, 0, \ldots, 0)$, which is better correlated with x_1, giving him an advantage in guessing x_1.

Formally, we prove this as follows. Observe that for any distribution (B, M), if

$$\Pr\left[\mathrm{Out} = 1 \mid x_1 = 1, (B^{x_2}, M^{x_2}) = (B, M)\right] > \Pr\left[\mathrm{Out} = 1 \mid x_1 = 0, (B^{x_2}, M^{x_2}) = (B, M)\right]$$

then

$$\begin{aligned}\Pr\left[\mathrm{Out} = x_1 \mid (B^{x_2}, M^{x_2}) = (B, M)\right] &= \tfrac{1}{2}\Pr\left[\mathrm{Out} = 1 \mid x_1 = 1, (B^{x_2}, M^{x_2}) = (B, M)\right] \\ &+ \tfrac{1}{2}\Pr\left[\mathrm{Out} = 0 \mid x_1 = 0, (B^{x_2}, M^{x_2}) = (B, M)\right] \\ &= \tfrac{1}{2}\mathrm{Cor}(B, M) + \tfrac{1}{2},\end{aligned}$$

and otherwise it holds that

$$\Pr\left[\text{Out} = x_1 \mid (B^{x_2}, M^{x_2}) = (B, M)\right] = -\tfrac{1}{2}\operatorname{Cor}(B, M) + \tfrac{1}{2}.$$

The former case implies that the event $\text{Out}' = x_1$ occurs with a significantly higher probability then $\text{Out} = x_1$, while the latter implies that $1 - \text{Out}' = x_1$ occurs with a significantly higher probability. As P_{i+1} has view according to $(B, M) = (B^0, M_1^0, \ldots, M_i^0, 0, \ldots, 0)$, its probability of guessing x_1 is significantly higher then the other parties, who have views according to $(B, M) = (B^0, M_1^0, \ldots, M_{i+1}^0, 0, \ldots, 0)$.

We next show that in the ideal world, the best a simulator for P_{i+1} can do in guessing x_1, is to output Out. Consider all the 4 potential views (x_2, Out) that it may see. The view $(0, 1)$ is impossible; for both views $(1, 0)$ and $(1, 1)$ we have $x_1 = \text{Out}$, and if the view is $(0, 0)$ then the adversary has no information about x_1, so guessing the value $x_1 = \text{Out}$ is correct with probability $1/2$ and is as good as any other way of guessing. Therefore, in the ideal world, Out predicts x_1 with the highest probability.

To conclude the proof, observe that the security guarantee of the simulator for $\mathcal{A}$ implies that in the ideal-world, the probability that $\text{Out} = x_1$ equals to $\pm\frac{1}{2}\operatorname{Cor}\left(B^0, M_1^0, \ldots, M_{i+1}^0, 0, \ldots, 0\right) + \frac{1}{2}$ (up to some negligible difference). In particular, the best guess for x_1 of the simulator for P_{i+1} will be correct with this probability. However, as we showed in the real-world P_{i+1} can compute a value Out' that will be equal to x_1 with significantly higher probability, contradicting security. $\square$

6.2 A Three-Round Protocol

In this section, we show that the 3-round protocol of Damgård and Ishai [20] admits FaF full-security, under the assumption that a PRG exists. We next present an overview of the construction of [20]. The protocol is a generalization of Yao's 2-party protocol [47] to the multiparty case. It is similar to the generalization proposed by Beaver, Micali, and Rogaway [8]. The novelty of [20] is in the usage of a *"distributive encryption scheme"* and the use of error-correction codes. The former allows the parties to somehow locally garble the circuit, while the latter replaces zero-knowledge proofs, resulting in a black-box use of a PRG. Next, we introduce several notations.

Notations. In the following, fix an m-ary function f. For simplicity, assume that the function is *deterministic and symmetric*. This assumption is without loss of generality as computing general functionalities can be reduced to deterministic and symmetric functionalities using standard techniques.

Let n be the security parameter, let t be a bound on the number of maliciously corrupted parties, and let h^* be a bound on the number of semi-honest parties. In addition, let C be a Boolean circuit that computes f, let W denote the number of wires in C, and denote its size by $|C|$ (number of gates). It is without loss of generality to assume that C consists of only NAND gates with fan-in degree 2. Let the wires of C be indexed from 0 to $W - 1$. For simplicity, assume that the circuit has just a single output bit, corresponding to the last wire, indexed $W - 1$. Each input wire w, denote by b_w the input bit assigned to it. Each party P_i will assign to each wire w two keys s_{2w}^i, s_{2w+1}^i. For a gate g and one of its input wires α, let $\mathsf{prev}(\alpha)$ be the index of the *output wire* of the previous gate, that goes into g as α.

Recall that a symmetric encryption is used to encrypt a key for the output wire of a gate g given two keys for the input wires of g. The protocol makes use of the *distributive encryption scheme* proposed by [20], defined as follows. Let $\mathsf{G} : \{0,1\}^n \mapsto \left(\{0,1\}^k\right)^\ell$ be a pseudo-random generator, where ℓ is the maximum fan-out in C, and $k = k(n)$ is a polynomial in the security parameter, with high enough degree (to be determined by the analysis). For $j \in [\ell]$, denote by $\mathsf{G}(s)|_j$ the j-th block of $\mathsf{G}(s)$. Suppose M is some message that is shared among the m parties, using a Shamir secret sharing scheme, i.e., using a random degree $d \geq t$ polynomial. Denote by $M[i]$ the i-th share of M, held by P_i. There are two keys $S_1 = (s_1^1, \ldots, s_1^m)$ and $S_2 = (s_2^1, \ldots, s_2^m)$, used to encrypt M, where initially the subkeys $s_1^i, s_2^i \in \{0,1\}^n$ are known only to party P_i. For $j_1, j_2 \in [\ell]$ define the encryption scheme by

$$\mathrm{Enc}_{S^1,S^2}^{j_1,j_2}(M) = \left(\mathsf{G}\left(s_1^i\right)|_{j_1} \oplus \mathsf{G}\left(s_2^i\right)|_{j_2} \oplus M[i]\right)_{i=1}^m.$$

Here, j_1 and j_2 represent an index of an *output wire* from a previous gate. Having received the parts of the ciphertext $\mathrm{Enc}_{S^1,S^2}^{j_1,j_2}(M)$ and the keys S^1 and S^2, one can easily decrypt each share and reconstruct M. Observe that if $d+2t < m$ then decryption is possible, regardless of the adversary's actions. This follows from the fact that Shamir secret sharing scheme is also error correcting.

Finally, in order to reduce the number of rounds, we will be using the share conversion techniques from [19]. They showed how, in two rounds, the parties can generate and distribute among themselves shares in a Shamir secret sharing scheme for different correlated random values (the actual functionalities according to which these values are selected – are listed below). Roughly, this is done as follows. First, the parties compute a CNF secret sharing scheme of a random secret. This phase can be implemented efficiently in two rounds using the multicast protocol of [26]. Below we give an overview of the protocol, and explain why it is FaF-secure.The parties then *locally* convert the CNF shares into Shamir shares with various properties, using the share conversion techniques from [19]. Furthermore, using a common PRF, the parties can produce polynomially many such Shamir shares that are pseudorandom. Let $t' = t + h^*$. We will use the following functionalities.

- $\mathsf{SSS}_0(t')$: Each party P_i obtains a $(t'+1)$-out-of-m Shamir share of a the secret $s = 0$, over the field $\mathrm{GF}(2^n)$
- $\mathsf{SSS}_{\mathrm{bin}}(t')$: Each party P_i obtains a $(t'+1)$-out-of-m Shamir share of a random secret $s \in \{0,1\}$, over the field $\mathrm{GF}(2^n)$ (i.e., each share is in $\mathrm{GF}(2^n)$, however, the secret is uniform over $\{0,1\}$).
- $\mathsf{SSS}^{\mathrm{P}}(t')$: Each party P_i obtains a $(t'+1)$-out-of-m Shamir share of a random secret s, over the field $\mathrm{GF}(2^n)$. In addition, party P receives the polynomial underlying the sharing.[8]
- $\mathsf{SSS}_{\mathrm{bin}}^{\mathrm{P}}(t')$: Same as $\mathsf{SSS}_{\mathrm{bin}}(t')$, except that party P additionally receives the polynomial underlying the sharing.

In the protocol, the parties are required to compute a VSS for each of their input wires. For $b \in \{0,1\}$ we let $\mathsf{VSS}_{\mathrm{bin}}^{\mathrm{P}}(t', b)$ be the following subroutine. The parties compute $\mathsf{SSS}_{\mathrm{bin}}^{\mathrm{P}}(t')$, so that P also receive the polynomial p from the sharing. Party P then broadcasts $r = b - p(0)$, and each party P_i outputs $r + p(i)$. Observe that the computation insures that the parties hold t-private Shamir shares of b. Furthermore, when implementing $\mathsf{SSS}_{\mathrm{bin}}^{\mathrm{P}}(t')$ with the multicast protocol from

[8]Implementing this functionality using the multicast protocol can be done by having P be in every subset of parties receiving a CNF share (even if P does not belong to that set). This will ensure that P will hold all shares of the random secret.

[26], the broadcast can be implemented in parallel to computation of $\mathsf{SSS}^{\mathrm{P}}_{\mathrm{bin}}(t')$. Additionally, by the robustness property of Shamir secret sharing, if $t' + 2t < m$ then the computation is secure (according to the standard definition of VSS). Therefore, the procedure admits (t, h^*)-FaF full-security whenever $3t + h^* < m$.

The Multicast Protocol. We now present an overview of the two-rounds multicast protocol of [26]. In the following, the set $\mathcal{M} \subseteq \mathcal{P}$ is the set of parties that should receive the message of the dealer D. The protocol makes use of a modification of the 2-round VSS protocol of [26].

Protocol 6.4.
Private Input: The dealer D *holds a message* $M \in \mathrm{GF}(2^n)$*, and all other parties hold no input.*

First round:

- *The dealer samples a random bivariate polynomial* $F(x, y)$ *of degree at most* $t+h^*$*, over the field* $\mathrm{GF}(2^n)$*, such that* $F(0, 0) = M$*.*
- *For every* $i \in [m]$*, the dealer hands the (univariate) polynomials* $f_i(x) = F(x, i)$ *and* $g_i(y) = F(i, y)$ *to party* P_i*.*
- *For every* $i, j \in [m]$*, party* P_i *samples a random pad* $r_{i,j} \in \mathrm{GF}(2^n)$ *and send it to* P_j*.*

Second Round:

- *Each party* P_i *broadcasts* $a_{i,j} = f_i(j) + r_{i,j}$ *and* $b_{i,j} = g_i(j) + r_{j,i}$ *(*$r_{i,j}$ *was sampled by* P_i *and* $r_{j,i}$ *was sample by* P_j*)*
- *Every party* P_i *sends privately* $f_i(0)$ *to every party in* $\mathcal{M}$*.*

Reconstruction: *Each party in* $\mathcal{M}$ *have enough information to locally compute the VSS reconstruction algorithm, and output its result. This is done as follows.*

1. *Define a consistency graph* G*, on* m *vertices, where vertex* i *corresponds to party* P_i*, and* $\{i, j\}$ *is an edge if and only if* $a_{i,j} = b_{j,i}$ *(i.e.,* P_i *and* P_j *broadcast consistent messages).*
2. *Find a maximal matching in the complement graph* $\bar{G}$*.*
3. *Let* $\mathcal{C}$ *be the set of vertices not in the matching (note that* $\mathcal{C}$ *is a clique in* G*).*
4. *Let* $\mathcal{C}'$ *be the set of vertices not in* $\mathcal{C}$ *that have at least* $2t + h^* + 1$ *neighbors in* $\mathcal{C}$*.*
5. *If* $|\mathcal{C}| + |\mathcal{C}'| \leq 3t + h^*$*, disqualify the dealer and output a default value.*
6. *Otherwise, use error-correction on the values* $\{f_i(0)\}_{i \in \mathcal{C} \cup \mathcal{C}'}$ *to reconstruct the polynomial* $g_0(y)$*, and output* $g_0(0)$*.*

We show that the above protocol admits FaF-security.

Theorem 6.5. *For every* m*,* t*, and* h^* *satisfying* $4t + h^* < m$*, Protocol 6.4 computes the multicast functionality with strong perfect* (t, h^*)*-FaF full-security.*

In the proof of Theorem 6.5 we make use of the following known fact.

Fact 6.6. *Let $\mathcal{S}$ denote a set of parties of size at least $d+1$, where each party P_i is holding two polynomials $f_i^*(x), g_i^*(y)$ each of degree d. Assume that for every pair of parties $\mathrm{P}_i, \mathrm{P}_j \in \mathcal{S}$ their corresponding polynomials agree, that is, it holds that $f_i^*(j) = g_j^*(i)$ and that $g_i^*(j) = f_j^*(i)$. Then there exists a unique bivariate polynomial $F^*(x,y)$ of degree d which is consistent with the values held by the parties in $\mathcal{S}$, i.e., $f_i^*(x) = F(x,i)$ and $g_i^*(y) = F(i,y)$.*

In the proof of Theorem 6.5 we make use of the following two claims.

Claim 6.7. *If $\mathrm{D} \notin \mathcal{I}$ is uncorrupted, the uncorrupted parties output M.*

Claim 6.8. *If $\mathrm{D} \in \mathcal{I}$ is corrupted, all uncorrupted parties output the same value.*

We first prove Theorem 6.5 using the above two claims. The proofs of the claims are given below.

Proof of Theorem 6.5. In the following, we denote $d = t + h^*$. Let $\mathcal{A}$ be an adversary corrupting a subset $\mathcal{I}$ of size at most t, and let $\mathcal{H} \subset \mathcal{P} \setminus \mathcal{I}$ be of size at most h^*. We separate into two cases. For the first case, let us assume that $\mathrm{D} \notin \mathcal{I}$. With an auxiliary input $z_{\mathcal{A}}$, the malicious simulator $\mathsf{S}_{\mathcal{A}}$ does the following.

1. If $\mathcal{A}$ corrupted a party in $\mathcal{M}$, receive the output M from the trusted party.
2. Sample uniformly at random a bivariate polynomial $F(x,y)$ of degree at most d, over the field $\mathrm{GF}(2^n)$. If $\mathcal{A}$ corrupted a party in $\mathcal{M}$, then the sample of F is conditioned on $F(0,0) = M$.
3. For every $\mathrm{P}_i \in \mathcal{I}$, hand over to $\mathcal{A}$ the univariate polynomials $f_i(x) = F(x,i)$ and $g_i(y) = F(i,y)$, and random pads $r_{j,i} \in \mathrm{GF}(2^n)$ for every $\mathrm{P}_j \notin \mathcal{I}$, sent on behalf of P_j.
4. The adversary sends back its choice for $r_{i,j} \in \mathrm{GF}(2^n)$ for every $\mathrm{P}_i \in \mathcal{I}$ and $\mathrm{P}_j \notin \mathcal{I}$.
5. For every $\mathrm{P}_i \notin \mathcal{I}$ and $\mathrm{P}_j \in \mathcal{P}$, compute $a_{i,j} = f_i(j) + r_{i,j}$ and $b_{i,j} = g_i(j) + r_{j,i}$ (note that $r_{j,i}$ may have been chosen by $\mathcal{A}$).
6. Output the univariate polynomials $f_i(x)$ and $g_i(y)$ held by $\mathcal{A}$, and the pads $r_{j,i} \in \mathrm{GF}(2^n)$ given to it, output the values $a_{i,j}$ and $b_{i,j}$ computed in the previous step, and if $\mathcal{A}$ corrupted a party in $\mathcal{M}$ output in addition $\{f_i(0)\}_{\mathrm{P}_i \notin \mathcal{I}}$.

We next construct the semi-honest simulator $\mathsf{S}_{\mathcal{A},\mathcal{H}}$, with an auxiliary input $z_{\mathcal{H}}$.

1. Apply $\mathsf{S}_{\mathcal{A}}$ with its randomness and $z_{\mathcal{A}}$ to compute the following values.
   - If $\mathcal{I} \cap \mathcal{M} \neq \emptyset$ compute the bivariate polynomial $F(x,y)$ generated by $\mathsf{S}_{\mathcal{A}}$.
   - Otherwise, compute only the univariate polynomials $\{f_i(x), g_i(y)\}_{\mathrm{P}_i \in \mathcal{I}}$ given to $\mathcal{A}$, and complete them to a random bivariate polynomial $F(x,y)$, so that if $\mathcal{H} \cap \mathcal{M} \neq \emptyset$ then $F(0,0) = M$ (note that this can be done since the bivariate polynomial is of degree $d \geq t$).
2. Compute the random pads $r_{j,i}$ for every $\mathrm{P}_i \in \mathcal{I}$ and $\mathrm{P}_j \in \mathcal{H}$ generated by $\mathsf{S}_{\mathcal{A}}$.
3. Compute the univariate polynomials $f_i(x) = F(x,i)$ and $g_i(y) = F(i,y)$ for every $\mathrm{P}_i \in \mathcal{H}$.

4. Send to $\mathcal{A}$ the univariate polynomials $f_i(x)$ and $g_i(y)$ for every $\mathrm{P}_i \in \mathcal{I}$, and the random pads $r_{j,i}$ sampled in Step 1.

5. The adversary sends back its choice for $r_{i,j} \in \mathrm{GF}(2^n)$ for every $\mathrm{P}_i \in \mathcal{I}$ and $\mathrm{P}_j \in \mathcal{H}$, and some addition non-prescribed messages.

6. For every $\mathrm{P}_i \notin \mathcal{I} \cup \mathcal{H}$ and $\mathrm{P}_j \in \mathcal{P}$ compute $a_{i,j} = f_i(j) + r_{i,j}$ and $b_{i,j} = g_i(j) + r_{j,i}$.

7. For every $\mathrm{P}_i \notin \mathcal{I}$ and $\mathrm{P}_j \in \mathcal{P}$, give the adversary $a_{i,j} = f_i(j) + r_{i,j}$ and $b_{i,j} = g_i(j) + r_{j,i}$. In addition, if $\mathcal{A}$ corrupted a party in $\mathcal{M}$, give it $\{f_i(0)\}_{\mathrm{P}_i \notin \mathcal{I}}$.

8. The adversary sends back values $a^*_{i,j}$ and $b^*_{i,j}$ for every $\mathrm{P}_i \in \mathcal{I}$ and every $\mathrm{P}_j \in \mathcal{P}$, and its choice for $\{h^*_i(0)\}_{\mathrm{P}_i \in \mathcal{I}}$ if $\mathcal{H} \cap \mathcal{M} \neq \emptyset$. In addition it sends back non-prescribed messages (which might include the $h^*_i(0)$'s regardless of whether or not $\mathcal{H} \cap \mathcal{M} \neq \emptyset$).

9. Output the univariate polynomials $f_i(x)$ and $g_i(y)$ held by the parties in $\mathcal{H}$, the adversary's choice for the $r_{i,j}$'s sent to the parties in $\mathcal{H}$, the computed values $a_{i,j}$ and $b_{i,j}$ for every $\mathrm{P}_i \notin \mathcal{I} \cup \mathcal{H}$ and $\mathrm{P}_j \in \mathcal{P}$, the adversary's choice for the values $a^*_{i,j}$ and $b^*_{i,j}$, and the non-prescribed messages sent by $\mathcal{A}$. In addition, if $\mathcal{H} \cap \mathcal{M} \neq \emptyset$ output $\{f_i(0)\}_{\mathrm{P}_i \notin \mathcal{I} \cup \mathcal{H}}$ and $\{h^*_i(0)\}_{\mathrm{P}_i \in \mathcal{I}}$.

Observe that the messages sent to $\mathcal{A}$ by the simulators are exactly the same, hence the answers they receive are the same as well. Furthermore, the univariate polynomials sampled and the random pads are distributed exactly the same in both the real-world and the ideal-world. Therefore, we may condition on them being the same. Since F is of degree $d = t + h^*$, if $\mathcal{I} \cap \mathcal{M} = \emptyset$ in both worlds the adversary's view is independent of $F(0,0) = M$, and if $(\mathcal{I} \cup \mathcal{H}) \cap \mathcal{M} = \emptyset$ the same holds with respect to $\mathcal{H}$. By Claim 6.7 the output of the uncorrupted parties in the real-world is M. Thus, we conclude that

$$\mathrm{IDEAL}^{\mathsf{S}_{\mathcal{A}}, \mathsf{S}_{\mathcal{A},\mathcal{H}}}_{1^n, M, z_{\mathcal{A}}, z_{\mathcal{H}}} \equiv \mathrm{REAL}^{\mathcal{A}, \mathcal{A}_{\mathcal{H}}}_{1^n, M, z_{\mathcal{A}}, z_{\mathcal{H}}}.$$

For the second case, we assume that $\mathrm{D} \in \mathcal{I}$. With an auxiliary input $z_{\mathcal{A}}$, the malicious simulator $\mathsf{S}_{\mathcal{A}}$ will do the following.

1. For every $\mathrm{P}_i \in \mathcal{I}$, hand over to $\mathcal{A}$ random pads $r_{j,i} \in \mathrm{GF}(2^n)$ for every $\mathrm{P}_j \notin \mathcal{I}$.

2. The adversary sends back its choice for $r_{i,j} \in \mathrm{GF}(2^n)$ for every $\mathrm{P}_i \in \mathcal{I}$ and $\mathrm{P}_j \notin \mathcal{I}$, and in addition its choice for the univariate polynomials $f^*_i(x)$ and $g^*_i(y)$ for every $\mathrm{P}_i \notin \mathcal{I}$.

3. For every $\mathrm{P}_i \notin \mathcal{I}$ and $\mathrm{P}_j \in \mathcal{P}$, compute $a_{i,j} = f^*_i(j) + r_{i,j}$ and $b_{i,j} = g^*_i(j) + r_{j,i}$ (note that $r_{j,i}$ could have been chosen by $\mathcal{A}$).

4. Send these values to the adversary and receive its choose for the $a^*_{i,j}$'s and $b^*_{i,j}$'s for every $\mathrm{P}_i \in \mathcal{I}$ and $\mathrm{P}_j \in \mathcal{P}$.

5. Apply the reconstruction algorithm to receive a value $M' \in \mathrm{GF}(2^n)$, and send it to the trusted party on behalf of the dealer.

6. Output the pads $r_{j,i} \in \mathrm{GF}(2^n)$ given to $\mathcal{A}$ and the values $a_{i,j}$ and $b_{i,j}$ computed in Step 3.

We next construct the semi-honest simulator $\mathsf{S}_{\mathcal{A},\mathcal{H}}$, with an auxiliary input $z_{\mathcal{H}}$.

1. Apply $\mathsf{S}_{\mathcal{A}}$ with its randomness and $z_{\mathcal{A}}$ to compute the random pads $r_{j,i}$ for every $\mathrm{P}_i \in \mathcal{I}$ and $\mathrm{P}_j \in \mathcal{H}$ generated by $\mathsf{S}_{\mathcal{A}}$, and send them to $\mathcal{A}$.

2. The adversary sends back its choice for $r_{i,j} \in \mathrm{GF}(2^n)$ for every $\mathrm{P}_i \in \mathcal{I}$ and $\mathrm{P}_j \in \mathcal{H}$, its choice for the univariate polynomials $f_i^*(x)$ and $g_i^*(y)$ for every $\mathrm{P}_i \notin \mathcal{I}$, and some additional non-prescribed messages.

3. For every $\mathrm{P}_i \notin \mathcal{I} \cup \mathcal{H}$ and $\mathrm{P}_j \in \mathcal{P}$ compute $a_{i,j} = f_i^*(j) + r_{i,j}$ and $b_{i,j} = g_i^*(j) + r_{j,i}$ (i.e., the values that P_j broadcast).

4. For every $\mathrm{P}_i \notin \mathcal{I}$ and $\mathrm{P}_j \in \mathcal{P}$, give the adversary $a_{i,j} = f_i^*(j) + r_{i,j}$ and $b_{i,j} = g_i^*(j) + r_{j,i}$.

5. The adversary sends back values $a_{i,j}^*$ and $b_{i,j}^*$ for every $\mathrm{P}_i \in \mathcal{I}$ and every $\mathrm{P}_j \in \mathcal{P}$, and its choice for $\{h_i^*(0)\}_{\mathrm{P}_i \in \mathcal{I}}$ if $\mathcal{H} \cap \mathcal{M} \neq \emptyset$. In addition it sends back non-prescribed messages (which might include the $h_i^*(0)$'s regardless of whether or not $\mathcal{H} \cap \mathcal{M} \neq \emptyset$).

6. Output the univariate polynomials $f_i^*(x)$ and $g_i^*(y)$ held the parties in $\mathcal{H}$, the adversary's choice for the $r_{i,j}$'s for every $\mathrm{P}_i \in \mathcal{I}$ and $\mathrm{P}_j \in \mathcal{H}$, the computed values $a_{i,j}$ and $b_{i,j}$ for every $\mathrm{P}_i \notin \mathcal{I} \cup \mathcal{H}$ and $\mathrm{P}_j \in \mathcal{P}$, the adversary's choice for the values $a_{i,j}^*$ and $b_{i,j}^*$ for every $\mathrm{P}_i \in \mathcal{I}$ and every $\mathrm{P}_j \in \mathcal{P}$, and the non-prescribed messages sent by $\mathcal{A}$. In addition, if $\mathcal{H} \cap \mathcal{M} \neq \emptyset$ output $\{f_i^*(0)\}_{\mathrm{P}_i \notin \mathcal{I} \cup \mathcal{H}}$ and $\{h_i^*(0)\}_{\mathrm{P}_i \in \mathcal{I}}$.

Similarly to the case where $\mathrm{D} \notin \mathcal{I}$, the view of $\mathcal{A}$ and $\mathcal{H}$ are identically distributed in both worlds. Furthermore, the malicious simulator $\mathsf{S}_{\mathcal{A}}$ computed the value M' it sends to the trusted party using the reconstruction algorithm used by the uncorrupted parties in the real-world, implying that the output of the uncorrupted parties is distributed exactly the same. Therefore, to conclude that

$$\text{IDEAL}^{\mathsf{S}_{\mathcal{A}}, \mathsf{S}_{\mathcal{A},\mathcal{H}}}_{1^n, M, z_{\mathcal{A}}, z_{\mathcal{H}}} \equiv \text{REAL}^{\mathcal{A}, \mathcal{A}_{\mathcal{H}}}_{1^n, M, z_{\mathcal{A}}, z_{\mathcal{H}}}$$

we only need to show that the parties in the real-world output the same value. This follows from Claim 6.8. □

Proof of Claim 6.7. Assume the dealer D is uncorrupted. Then, all edges in the graph G have an endpoint that corresponds to a malicious party. Thus, any maximal matching, in particular the one found, is of size $\hat{t} \leq t$. Therefore, $|\mathcal{C}| = m - 2\hat{t}$. Next, as any edge in the matching has an endpoint that corresponds to an uncorrupted party, it follows that $|\mathcal{C}'| \geq \hat{t}$. Thus, we have $|\mathcal{C} \cup \mathcal{C}'| \geq m - \hat{t} > 4t + h^* - \hat{t} = 3t + h^*$. Therefore, the dealer is not disqualified. Since the uncorrupted parties report $f_i(0)$ that are consistent with $F(0, i)$, and up to t parties report an inconsistent value, error-correctness succeeds. □

Proof of Claim 6.8. Here we prove that even if D is dishonest, the parties in $\mathcal{M}$ agree on the same value. There are two cases:

1. $|\mathcal{C}| + |\mathcal{C}'| \leq 3t + h^*$**:** In this case, all parties agree on this fact, as this is derived only from the publicly known graph G, so all parties in $\mathcal{M}$ output a default value.

2. $|\mathcal{C}| + |\mathcal{C}'| > 3t + h^*$**:** Here, the dealer is not disqualified and error-correction is to be applied by the uncorrupted parties. By the definition of $\mathcal{C}'$, this implies that $|\mathcal{C}| \geq 2t + h^* + 1$. Consider the subset $\mathsf{Good} \subseteq \mathcal{C}$ of parties who are uncorrupted (i.e., in $\mathcal{P} \setminus (\mathcal{I} \cup \mathcal{H})$). Observe that

$|\mathsf{Good}| \geq |\mathcal{C}| - t \geq 2t + h^* + 1 - t = d + 1$, where $d = t + h^*$. By Fact 6.6, the values in Good determine a unique bivariate $F^*(x, y)$ polynomial of degree d. Now, as $|\mathcal{C}| + |\mathcal{C}'| > 3t + h^*$, at least $2t + h^* + 1 = d + t + 1$ of the parties in $\mathcal{C} \cup \mathcal{C}'$ are uncorrupted. Every such party agrees with at least $d + t + 1$ of the parties in $\mathcal{C}$, and thus with at least $d + 1$ of the parties in Good. That is, for every uncorrupted party $\mathrm{P}_i \in \mathcal{C}'$, the polynomial $f_i^*(x)$ given to P_i agrees with $F^*(x, y)$ on at least $|\mathsf{Good}| \geq d + 1$ points. In particular, $f_i^*(0) = F^*(0, i)$. Finally, upon reconstruction, all uncorrupted parties apply error-correction on the values $\{f_i(0)\}_{i \in \mathcal{C} \cup \mathcal{C}'}$. As at least $|\mathcal{C} \cup \mathcal{C}'| - t$ parties in $\mathcal{C} \cup \mathcal{C}'$ are uncorrupted, the polynomial $g_0^*(y)$ (before error correction) disagrees with $F^*(0, y)$ on at most t points. As the Reed-Solomon code formed by the degree-d polynomials specified on $\ell = |\mathcal{C} \cup \mathcal{C}'|$ points has distance $\ell - d \geq 2t + 1$, the t errors are successfully corrected.

□

When constructing the simulators, we note that the resulting simulations are straight-line, black-box and achieve perfect security. Therefore, by [40] Protocol 6.4 remain secure under parallel composition.[9] Thus, we have the following claim.

Claim 6.9. *Protocol 6.4 computes the m-party multicast functionality with strong perfect (t, h^*)-FaF full-security for every t and h^* satisfying $4t + h^* < m$. Furthermore, security is preserved under parallel composition.*

The Three-Round Protocol. We next describe the protocol of Damgård and Ishai [20]. By Claim 6.9, we may assume that the parties have access to idealizations of the secret sharing functionalities described above. In this hybrid model, the protocol consists of two rounds.

Protocol 6.10 (Π_{DI})**.**
Private Input: Each party P_i holds an input $x_i \in \mathcal{X}_i$.
Common Input: The parties hold the security parameter 1^n.

1. *In the first round, the parties call the following functionalities.*
   - *For each wire $w \in \{0, \ldots, W-1\}$, the parties call $\mathsf{SSS}_{\mathrm{bin}}(t + h^*)$ to create shares of the internal values $\lambda_w \in \{0, 1\}$.*
   - *For each $i \in [m]$ and each wire $w \in \{0, \ldots, W-1\}$, the parties call $\mathsf{SSS}^{\mathrm{P}_i}(t + h^*)$ to create shares of the subkeys s_{2w}^i and s_{2w+1}^i, such that both subkeys are known only to P_i.*
   - *For each input bit b_w held by party P, the parties apply the procedure $\mathsf{VSS}_{\mathrm{bin}}^{\mathrm{P}}(t + h^*, b_w)$.*
   - *The parties prepare shares of 0 as follows.*
     - *Call $\mathsf{SSS}_0(2(t + h^*))$ for each input wire, and four times for each gate.*
     - *Call $\mathsf{SSS}_0(3(t + h^*))$ four times for each gate.*
2. *Each party P_i locally computes the following values.*
   - *For each input wire w and each $j \in [m]$, P_i computes a random share of $s_{2w+(b_w \oplus \lambda_w)}^j$. Note that since we work over a field of characteristic 2, this value can be written as a degree 2 polynomial $(1 + b_w + \lambda_w) \cdot s_{2w}^j + (b_w + \lambda_w) \cdot s_{2w+1}^j$. Therefore, the computation can be carried out using Fact 2.4.*

[9][40] proved the theorem for standard security, however, their result apply to FaF-security as well.

- *For each input wire w, compute shares of the value $b_w \oplus \lambda_w$.*
- *For each gate g in the circuit do the following. Let α and β be the two input wires of g, and let γ be one of its output wires. For each $j \in [m]$ compute random shares of the following values*

$$\begin{aligned}
\delta_{g,\gamma}^{00} &= (\lambda_\alpha \text{ NAND } \lambda_\beta) \oplus \lambda_\gamma; \quad & a_{g,\gamma}^{00,j} &= s_{2\gamma+\delta_{g,\gamma}^{00}}^{j}\\
\delta_{g,\gamma}^{01} &= (\lambda_\alpha \text{ NAND } \bar{\lambda}_\beta) \oplus \lambda_\gamma; \quad & a_{g,\gamma}^{01,j} &= s_{2\gamma+\delta_{g,\gamma}^{01}}^{j}\\
\delta_{g,\gamma}^{10} &= (\bar{\lambda}_\alpha \text{ NAND } \lambda_\beta) \oplus \lambda_\gamma; \quad & a_{g,\gamma}^{10,j} &= s_{2\gamma+\delta_{g,\gamma}^{10}}^{j}\\
\delta_{g,\gamma}^{11} &= (\bar{\lambda}_\alpha \text{ NAND } \bar{\lambda}_\beta) \oplus \lambda_\gamma; \quad & a_{g,\gamma}^{11,j} &= s_{2\gamma+\delta_{g,\gamma}^{11}}^{j}
\end{aligned}$$

  Note that these values can be written as polynomials of degree 2 and 3 in the already shared values. For instance, $a_{g,\gamma}^{00,j} = (\lambda_\alpha\lambda_\beta + \lambda_\gamma)s_{2\gamma}^{j} + (1 + \lambda_\alpha\lambda_\beta + \lambda_\gamma)s_{2\gamma+1}^{j}$. Thus, these values can be computed using Fact 2.4 as well.
- *For each $c, d \in \{0,1\}$ and each gate g and an output wire γ, denote $A_{g,\gamma}^{cd} = \left(\left(a_{g,\gamma}^{cd,j}\right)_{j=1}^{m}, \delta_{g,\gamma}^{cd}\right)$. Party P_i then computes the i-th component of $\mathrm{Enc}_{S_{2\alpha+c},S_{2\beta+d}}^{\mathsf{prev}(\alpha),\mathsf{prev}(\beta)}(A_{g,\gamma}^{cd})$ (recall that $\mathsf{prev}(\alpha)$ and $\mathsf{prev}(\beta)$ are the indexes of the output wire of the previous gate, that goes into g as α and β, respectively).*

3. *Each party P_i then broadcast the shares of $s_{2w+(b_w\oplus\lambda_w)}^{j}$, of $b_w \oplus \lambda_w$, of λ_{W-1}, and the encryptions.*

4. *Each party can now locally evaluate the output.*

Theorem 6.2 clearly follows from the following theorem asserting that Π_{DI} admits FaF full-security.

Theorem 6.11. *Let $m, t, h^* \in \mathbb{N}$ be three constants satisfying $5t + 3h^* < m$. Let f be a polynomial time computable m-party functionality. Then, under the assumption that pseudorandom generators exist, Π_{DI} is a 3-round protocol that computes f with strong computational (t, h^*)-FaF full-security. Moreover, the construction is fully black-box.*

Proof. Fix an adversary $\mathcal{A}$ corrupting a set $\mathcal{I} \subset \mathcal{P}$ of size at most t, and fix an adversary $\mathcal{A}_{\mathcal{H}}$ controlling a subset $\mathcal{H} \subset \mathcal{P} \setminus \mathcal{I}$ of size at most h^*. Observe that the maximum degree of taking over the polynomials used in the Shamir's secret sharings, is $3(t + h^*)$. Thus, by the robustness property of the sharing scheme, reconstruction of the secrets would be possible if $3(t + h^*) + 2t < m$, which holds by assumption.

Although [20] provided a simulator $\mathsf{S}_{\mathcal{A}}$ for $\mathcal{A}$, we will fully describe it, as its description will help us to construct the second simulator $\mathsf{S}_{\mathcal{A},\mathcal{H}}$. The idea in constructing both simulators is to have them execute the protocol to construct the shares and a "fake" garbled circuit. When constructing the "fake" garbled circuit, we have to make sure that the output is the same as the output received from T. To force this, the simulators will put the bit $\lambda_{W-1} \oplus y$ as the plaintext inside the encryptions of the last gate, where y is the output received from T. This is done as follows. The degree $3t$ polynomial that defines this bit is of the form $Q^{cd}(\cdot) + Z^{cd}(\cdot)$, where $Z^{cd}(0) = 0$ and $Q^{cd}(0) =$

$((c \oplus \lambda_{W-2})$ NAND $(d \oplus \lambda_{W-3})) \oplus \lambda_{W-1}$. Therefore, $\mathsf{S}_{\mathcal{A}}$ and $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ will change Z^{cd} with another degree $3t$ polynomial $\hat{Z}^{cd}$, whose value on 0 will be consistent with the output. These values are defined as $\hat{Z}^{cd}(0) = ((c \oplus \lambda_{W-2})$ NAND $(d \oplus \lambda_{W-3})) \oplus y$. Additionally, the adversary's simulator $\mathsf{S}_{\mathcal{A}}$ will have to make sure it sends to T correct inputs. This can be easily implemented as the VSS's of the first round fixes the corrupted parties' inputs. We next formally describe $\mathsf{S}_{\mathcal{A}}$ on input $\mathbf{x}_{\mathcal{I}}$ and an auxiliary input $z_{\mathcal{A}}$.

1. Generate the shares of the first round as follows.
   - For every wire $w \in \{0, \ldots, W-1\}$ choose $\lambda_w \in \{0,1\}$ uniformly at random, and share it in a $(t+h^*+1)$-out-of-m Shamir's secret sharing scheme.
   - For every P_i and every wire $w \in \{0, \ldots, W-1\}$ choose $s^i_{2w}, s^i_{2w+1} \in \{0,1\}^n$ uniformly at random and share them in a $(t+h^*+1)$-out-of-m Shamir's secret sharing scheme.
   - Generate shares for 0 as follows.
     - Share 0 in a $(2(t+h^*)+1)$-out-of-m Shamir's secret sharing scheme, for each input wire and four times for every gate but the last one.
     - Share 0 in a $(3(t+h^*)+1)$-out-of-m Shamir's secret sharing scheme, four times for every gate.
   - For every $c, d \in \{0,1\}$, generate shares of some value $v^{cd} \in \mathrm{GF}(2^n)$ in a $(2(t+h^*)+1)$-out-of-m Shamir's secret sharing scheme.
   - For each input wire w held by an uncorrupted party $\mathrm{P} \notin \mathcal{I}$, simulate the execution of $\mathsf{VSS}^{\mathrm{P}}_{\mathrm{bin}}(t+h^*, 0)$ by giving $\mathcal{A}$ a random element from $\mathrm{GF}(2^n)$.
2. Send to $\mathcal{A}$ its respective shares, and receive VSS's for each input wire held by the adversary.
3. Apply the VSS reconstruction function to reconstruct an adversary's input. Send the input to the trusted party to receive an output y.
4. Complete the shares of each v^{cd} to shares of $((c \oplus \lambda_{W-2})$ NAND $(d \oplus \lambda_{W-3})) \oplus y$, for every $c, d \in \{0,1\}$.
5. For every uncorrupted party, compute its faked garbled circuit using the shares generated in the previous steps.
6. Output the adversary's shares generated in Step 1 and the fake garbled circuits, and halt.

In order to show strong FaF-security, we have to make sure the messages that the simulators send to the adversary are exactly the same. Since the semi-honest simulator $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ holds the randomness of $\mathsf{S}_{\mathcal{A}}$, it can generate the same random shares, and complete the shares of the input bits held by the parties in $\mathcal{H}$, so that the secret is the real input. This forces the shares sent by the two simulators to $\mathcal{A}$ to be the same. Since the shares are $(t+h^*)$-private, the semi-honest simulator $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ can do so using the same shares $\mathsf{S}_{\mathcal{A}}$ sampled for it. This implies that the faked garbled circuits sent by the simulators are also the same. We now describe $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ on input $\mathbf{x}_{\mathcal{H}}$ and an auxiliary input $z_{\mathcal{H}}$.

1. Receive the output y from the trusted party, and receive r, $\mathbf{x}_{\mathcal{I}}$, and $z_{\mathcal{A}}$ – $\mathsf{S}_{\mathcal{A}}$'s randomness, input, and auxiliary input, respectively.

2. Execute $\mathsf{S}_{\mathcal{A}}\left(\mathbf{x}_{\mathcal{I}}, z_{\mathcal{A}}; r\right)$ to receive the shares generated for the first round of the protocol.

3. For every input wire w associated with a party $\mathrm{P} \in \mathcal{H}$, complete the shares of 0 generated for the adversary by $\mathsf{S}_{\mathcal{A}}$ so that the secret is b_w. Note that by the properties of the secret sharing scheme it is possible to do so by changing the shares not held by the parties in $\mathcal{I} \cup \mathcal{H}$.

4. Send to $\mathcal{A}$ its respective shares, and receive VSS's for each input wire held by the adversary, and the messages it sends to the parties in $\mathcal{H}$.

5. Use $\mathsf{S}_{\mathcal{A}}$ to complete the shares of each v^{cd} to shares of $((c \oplus \lambda_{W-2})\ \mathsf{NAND}\ (d \oplus \lambda_{W-3})) \oplus y$, for every $c, d \in \{0,1\}$.

6. For every uncorrupted party compute its faked garbled circuit using the shares generated in the previous steps. Send the fake garbled circuits to $\mathcal{A}$ to receive the messages it sends to the parties in $\mathcal{H}$.

7. Output the shares of the parties in $\mathcal{H}$, the fake garbled circuits sent to them, and the messages sent by $\mathcal{A}$, and halt.

To conclude the proof, we next show that

$$\left\{\mathrm{IDEAL}^{\mathsf{S}_{\mathcal{A}},\mathsf{S}_{\mathcal{A},\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}}\right\}_{\mathbf{x}\in\mathcal{X},z_{\mathcal{A}},z_{\mathcal{H}}\in\{0,1\}^*,n\in\mathbb{N}} \overset{\mathrm{c}}{\equiv} \left\{\mathrm{REAL}^{\mathcal{A},\mathcal{A}_{\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}}\right\}_{\mathbf{x}\in\mathcal{X},z_{\mathcal{A}},z_{\mathcal{H}}\in\{0,1\}^*,n\in\mathbb{N}}.$$

Observe that both simulators sends to $\mathcal{A}$ exactly the same messages. As a result, they will receive the same shares. Next, sort the gates in C in a topological order. Consider a series of hybrids $H_i(\mathbf{x})$, in which each one gate at a time is replaced in the real garbled circuit. The hybrid $H_0(\mathbf{x})$ contains the real garbled circuit. In contrast, $H_{|C|}(\mathbf{x})$ contains the fake garbled circuit constructed by the simulator. As both simulators generated exactly the same fake garbled circuits, it is enough to show that

$$\{H_0(\mathbf{x})\} \overset{\mathrm{c}}{\equiv} \left\{H_{|C|}(\mathbf{x})\right\}.$$

By a simple hybrid argument, if a distinguisher exists, then one can distinguish $H_i(\mathbf{x})$ from $H_{i-1}(\mathbf{x})$ for some $i \in [|C|]$. However, as everything but the values of the encryptions are identically distributed, this contradicts the semantic security of the encryption scheme. $\square$

7 General Statistically FaF-Secure MPC

In previous section we discussed the feasibility of FaF-security in the computational setting. In this section we study FaF-security under the notions of statistical and perfect security. Additionally, we further compare the notion of FaF-security with the notion of mixed-security – where a single adversary corrupts a subset of the parties actively and another subset passively. Recall that our comparisons to mixed-security in Sections 4.1.2 and 5.2 only hold in the computational setting. It is unclear if those results hold in the information-theoretic setting. Here, we give some evidence for the contrary.

The main result of this section, is characterizing the types of adversaries, for which we can compute *any multiparty functionality* with statistical/perfect FaF full-security (with and without broadcast). Here we generalize our model to consider adversaries that corrupt parties according to more general adversarial structures, beyond threshold structures as discussed so far. We define our adversarial structures following Fitzi et al. [24] who studied the notion of *mixed adversarial structures*. As defined in Section 5.2, we refer to this type of security as *mixed-security*. They considered a type of adversarial structures called *monotone adversarial structure*. Roughly, this means that turning a malicious party into being semi-honest does not compromise the security of the protocol. As discussed in Section 5.1, this is not generally the case (even for threshold adversaries).

Fitzi et al. [24] characterized the conditions for when general secure computation against mixed adversaries is possible. Somewhat surprisingly, we show that their conditions characterize FaF full-security as well. Furthermore, in the positive direction, we show that same conditions provided by [24] imply *strong* FaF full-security, while in the other direction, we show that if the conditions do not hold, then even *weak* FaF full-security is impossible. An interesting corollary, is that (weak) FaF full-security, strong FaF full-security, and mixed-full-security, are all equivalent as far as general MPC goes. We next provide formal definitions.

A pair $(\mathcal{I},\mathcal{H}) \subseteq \mathcal{P}^2$ is called *valid* if $\mathcal{I}\cap\mathcal{H}=\emptyset$. A *mixed adversarial structure over a set* $\mathcal{P}$ is a set $\mathcal{R} \subseteq \left\{(\mathcal{I},\mathcal{H}) \subseteq \mathcal{P}^2 \colon \mathcal{I}\cap\mathcal{H}=\emptyset\right\}$ of valid pairs. We omit $\mathcal{P}$ when clear from the context. For two valid pairs $(\mathcal{I},\mathcal{H}),(\mathcal{I}',\mathcal{H}')$, we say that $(\mathcal{I}',\mathcal{H}') \leq (\mathcal{I},\mathcal{H})$ if $\mathcal{I}'\cup\mathcal{H}' \subseteq \mathcal{I}\cup\mathcal{H}$ and $\mathcal{I}' \subseteq \mathcal{I}$. A mixed adversarial structure $\mathcal{R}$ is called *monotone*, if for every $(\mathcal{I},\mathcal{H}) \in \mathcal{R}$, it holds that if $(\mathcal{I}',\mathcal{H}') \leq (\mathcal{I},\mathcal{H})$ then $(\mathcal{I}',\mathcal{H}') \in \mathcal{R}$.

We next extend Definition 3.3 for general mixed adversarial structures. We only define security for the statistical and perfect cases.

Definition 7.1 (FaF-security for general mixed adversarial structure)**.** *Let* Π *be a protocol for computing* f *and let* $\mathcal{R}$ *be a mixed adversarial structure. We say that* Π *computes* f *with* statistical $\mathcal{R}$-FaF full-security*, if the following holds for any* $(\mathcal{I},\mathcal{H}) \in \mathcal{R}$*. For every adversary* $\mathcal{A}$*, controlling the set* $\mathcal{I}$ *in the real-world, there exists a adversary* $\mathsf{S}_{\mathcal{A}}$*, controlling* $\mathcal{I}$ *in the ideal model; and for every* semi-honest *adversary* $\mathcal{A}_{\mathcal{H}}$ *controlling the subset* $\mathcal{H}$*, there exists an adversary* $\mathsf{S}_{\mathcal{A},\mathcal{H}}$*, controlling* $\mathcal{H}$ *in the ideal-world, such that*

$$\left\{\text{IDEAL}^{\mathsf{S}_{\mathcal{A}},\mathsf{S}_{\mathcal{H}}}_{1^n,\mathbf{x}}\left(\mathsf{S}_{\mathcal{A}}\right)\right\}_{\mathbf{x}\in\mathcal{X},n\in\mathbb{N}} \overset{S}{\equiv} \left\{\text{REAL}^{\mathcal{A},\mathcal{A}_{\mathcal{H}}}_{1^n,\mathbf{x}}\left(\mathcal{A}\right)\right\}_{\mathbf{x}\in\mathcal{X},n\in\mathbb{N}}, \tag{4}$$

and

$$\left\{\text{IDEAL}^{\mathsf{S}_{\mathcal{A}},\mathsf{S}_{\mathcal{H}}}_{1^n,\mathbf{x}}\left(\mathsf{S}_{\mathcal{H}}\right)\right\}_{\mathbf{x}\in\mathcal{X},n\in\mathbb{N}} \overset{S}{\equiv} \left\{\text{REAL}^{\mathcal{A},\mathcal{A}_{\mathcal{H}}}_{1^n,\mathbf{x}}\left(\mathcal{A}_{\mathcal{H}}\right)\right\}_{\mathbf{x}\in\mathcal{X},n\in\mathbb{N}}. \tag{5}$$

The protocol Π *further computes* f *with* perfect $\mathcal{R}$-FaF full-security *if the distributions are equivalent. Finally, we say that* $\mathcal{R}$ *allows for general* m*-party* statistical *(resp., perfect) FaF fully-secure MPC if for every* m*-party function* f *there exists a protocol* Π *that computes* f *with statistical (resp., perfect)* $\mathcal{R}$*-FaF full-security.*

Definition 7.2 (strong statistical FaF-security)**.** *Let* Π *be a protocol for computing* f *and let* $\mathcal{R}$ *be a mixed adversarial structure. We say that* Π *computes* f *with* strong *statistical* $\mathcal{R}$*-FaF full-security, if the following holds for any* $(\mathcal{I},\mathcal{H}) \in \mathcal{R}$*. For every adversary* $\mathcal{A}$*, controlling a set* $\mathcal{I}$ *in the real-world, there exists a adversary* $\mathsf{S}_{\mathcal{A}}$*, controlling* $\mathcal{I}$ *in the ideal model; and for every* semi-honest *adversary* $\mathcal{A}_{\mathcal{H}}$ *controlling the subset* $\mathcal{H}$*, there exists a adversary* $\mathsf{S}_{\mathcal{A},\mathcal{H}}$*, controlling* $\mathcal{H}$ *in the ideal-world, such that*

$$\left\{\mathrm{IDEAL}^{\mathsf{S}_{\mathcal{A}},\mathsf{S}_{\mathcal{A},\mathcal{H}}}_{1^n,\mathbf{x}}\right\}_{\mathbf{x}\in\mathcal{X},n\in\mathbb{N}} \overset{S}{\equiv} \left\{\mathrm{REAL}^{\mathcal{A},\mathcal{A}_{\mathcal{H}}}_{1^n,\mathbf{x}}\right\}_{\mathbf{x}\in\mathcal{X},n\in\mathbb{N}}. \tag{6}$$

The security is perfect if the distributions above are identical. Finally, we say that $\mathcal{R}$ *allows for general* m*-party* strong statistical *(resp., perfect) FaF fully-secure MPC if for every* m*-party function* f *there exists a protocol* Π *that computes* f *with* strong *statistical (resp., perfect)* $\mathcal{R}$*-FaF full-security.*

7.1 Statistical FaF Full-Security For Monotone Adversarial Structures

In this section, we present the main result for characterizing general FaF fully-secure MPC. We start with defining a family of predicates, called $\mathsf{Q}(a,b)$ for $0 \leq b \leq a$, over the set of monotone adversarial structures. These predicates form a generalization of the condition $at + bh^* < m$ for threshold adversaries.

Definition 7.3. *Let* $\mathcal{R}$ *be a monotone mixed adversarial structure, and let* $0 \leq b \leq a$*. We say that* $\mathcal{R}$ *is* $\mathsf{Q}(a,b)$*, if for all* $(\mathcal{I}_1,\mathcal{H}_1),\ldots,(\mathcal{I}_a,\mathcal{H}_a) \in \mathcal{R}$*, it holds that*

$$\bigcup_{1\leq\ell\leq a}\mathcal{I}_\ell \cup \bigcup_{1\leq\ell\leq b}\mathcal{H}_\ell \neq \mathcal{P}$$

For example, $\mathcal{R}$ is $\mathsf{Q}(3,2)$ if for every $(\mathcal{I}_1,\mathcal{H}_1),(\mathcal{I}_2,\mathcal{H}_2),(\mathcal{I}_3,\mathcal{H}_3) \in \mathcal{R}$ it holds that

$$\mathcal{I}_1 \cup \mathcal{I}_2 \cup \mathcal{I}_3 \cup \mathcal{H}_1 \cup \mathcal{H}_2 \neq \mathcal{P}.$$

Similarly to [24], we are only going to consider $\mathsf{Q}(3,2)$, $\mathsf{Q}(2,2)$, and $\mathsf{Q}(3,0)$.

The following theorem states the main result of this section, providing a characterization of FaF fully-secure MPC secure against monotone mixed adversarial structures.

Theorem 7.4. *Let* $\mathcal{R}$ *be a monotone mixed adversarial structure over a set* $\mathcal{P} = \{\mathrm{P}_1,\ldots,\mathrm{P}_m\}$ *for some* $m \geq 3$*. Then:*

1. $\mathcal{R}$ *allows for general* m*-party strong* statistical *FaF fully-secure MPC, assuming an available broadcast channel, if and only if* $\mathcal{R}$ *is* $\mathsf{Q}(2,2)$*.*
2. $\mathcal{R}$ *allows for general* m*-party strong* statistical *FaF fully-secure MPC (without broadcast) if and only if* $\mathcal{R}$ *is both* $\mathsf{Q}(2,2)$ *and* $\mathsf{Q}(3,0)$*.*
3. $\mathcal{R}$ *allows for general* m*-party strong* perfect *FaF fully-secure MPC if and only if* $\mathcal{R}$ *is* $\mathsf{Q}(3,2)$*. Moreover, the negative direction holds even when assuming the availability of a broadcast channel, while the positive direction holds without a broadcast channel.*

In all cases, both the communication complexity and computation complexity of the resulting protocols is $\mathrm{poly}(|\mathcal{B}|)$*, where* $\mathcal{B}$ *is the set of all maximal elements in* $\mathcal{R}$*. Furthermore, in each of the three cases above, if the corresponding condition is not satisfied, then there exists an access structure for which even* weak *FaF full-security is not achievable.*

Before proving Theorem 7.4, we discuss an interesting corollary. Fitzi et al. [24] showed that the same conditions as those asserted in Theorem 7.4 hold with respect to mixed-security. Somewhat surprisingly, we conclude that the conditions on $\mathcal{R}$ also allow for general (information theoretic) MPC with FaF-security. In particular, no further gains can be made by further restricting the security requirement as in our model as far as general MPC goes (although the simulators are also weaker). This does not rule out specific functionalities f and adversarial structures $\mathcal{R}$ for which mixed-security is possible, while FaF-security is not and vice versa. See Example 5.4 of a protocol that satisfies *computational* mixed-security but is not FaF-secure for the same $\mathcal{R}$. In the other direction, in Section 4.1.2 we gave an example for a 3-party functionality that cannot be computed with $(1,1)$-mixed-security, yet is computable with computational $(1,1)$-FaF full-security.

Corollary 7.5. *Let* $\mathcal{R}$ *denote a monotone mixed adversarial structure over* $\mathcal{P} = \{\mathrm{P}_1, \ldots, \mathrm{P}_m\}$*, where* $m \geq 3$*. Then* $\mathcal{R}$ *allows for general* m*-party statistical (resp., perfect) FaF fully-secure MPC with (resp., without) broadcast if and only if* $\mathcal{R}$ *allows for general* m*-party statistical (resp., perfect) mixed fully-secure MPC with (resp., without) broadcast.*

7.2 Proofs of the Negative Results

In this section, we prove the impossibility results stated in Theorem 7.4. The following lemma formulate the negative direction of Case 1.

Lemma 7.6. *Let* $\mathcal{R}$ *denote a monotone mixed adversarial structure over* $\mathcal{P} = \{\mathrm{P}_1, \ldots, \mathrm{P}_m\}$*, where* $m \geq 3$*. Assume that* $\mathcal{R}$ *is* not $\mathsf{Q}(2,2)$*. Then, there exists an* m*-party functionality* f *which cannot be computed with statistical* $\mathcal{R}$*-FaF full-security. Moreover, this holds even when assuming an available broadcast channel.*

The proof is done using a *player-partitioning argument*, similarly to [24]. For the sake of completeness we provide the proof.

Proof. Since $\mathcal{R}$ is not $\mathsf{Q}(2,2)$, there exist $(\mathcal{I}_1, \mathcal{H}_1), (\mathcal{I}_2, \mathcal{H}_2) \in \mathcal{R}$ such that

$$\mathcal{I}_1 \cup \mathcal{I}_2 \cup \mathcal{H}_1 \cup \mathcal{H}_2 = \mathcal{P}.$$

Then at least one of the sets $\mathcal{I}_1 \cup \mathcal{H}_1$ and $\mathcal{I}_2 \cup \mathcal{H}_2$ is not empty. Furthermore, we can assume without loss of generality that *both* are not empty. Indeed, if one of them is empty, then by monotonicity we could partition the non-empty one into more sets, since $m \geq 3$. Additionally, we may assume that $\mathcal{I}_1 \cup \mathcal{H}_1$ and $\mathcal{I}_2 \cup \mathcal{H}_2$ are disjoint. This follows from the observation that by monotonicity, we could replace $\mathcal{I}_2$ with $\mathcal{I}_2 \setminus (\mathcal{I}_1 \cup \mathcal{H}_1)$ and replace $\mathcal{H}_2$ with $\mathcal{H}_2 \setminus (\mathcal{I}_1 \cup \mathcal{H}_1)$. Furthermore, the monotonicity of $\mathcal{R}$, implies that $(\emptyset, \mathcal{I}_1 \cup \mathcal{H}_1), (\emptyset, \mathcal{I}_2 \cup \mathcal{H}_2) \in \mathcal{R}$.

Now, assume towards contradiction, that for any m-party functionality g there exists a protocol Π_g that computes it with statistical $\mathcal{R}$-FaF full-security (possibly using broadcast). Consider the 2-party symmetric functionality $f(x_1, x_2) = x_1 \vee x_2$, which cannot be computed against a semi-honest adversary corrupting any single party [12]. Fix $i_1 \in \mathcal{I}_1 \cup \mathcal{H}_1$ and $i_2 \in \mathcal{I}_2 \cup \mathcal{H}_2$, and consider the m-party symmetric functionality $g(x_1, \ldots, x_m) = f(x_{i_1}, x_{i_2})$.

Next, consider the following 2-party protocol Π_f for computing f. The parties emulate the execution of Π_g, where for each $i \in \{1,2\}$, party P_i emulates the parties in $\mathcal{I}_i \cup \mathcal{H}_i$. In the emulation, the inputs x_1, x_2 are given to P_{i_1} and P_{i_2} respectively, while other parties start with arbitrary inputs. The parties then output whatever output was computed in Π_g.

The main observation is that any semi-honest adversary $\mathcal{A}_f$ corrupting one of the parties in Π_f, corresponds to the case where only the semi-honest adversary $\mathcal{A}_g$ corrupting some set $\mathcal{H} \subseteq \mathcal{P}$, is present in the emulated protocol Π_g. Indeed, a single semi-honest adversary $\mathcal{A}_f$ corrupting $(\emptyset, \{\mathrm{P}_i\})$, for $i \in \{1,2\}$, corresponds to a semi-honest adversary $\mathcal{A}_g$ corrupting $(\emptyset, \mathcal{I}_i \cup \mathcal{H}_i) \in \mathcal{R}$. Furthermore, the simulators for $\mathcal{A}_g$ directly translate to simulators for $\mathcal{A}$ contradicting the fact that f cannot be securely computed with statistical security against semi-honest adversaries. □

The following two lemmata state the negative direction for Cases 2 and 3 respectively.

Lemma 7.7. *Let $\mathcal{R}$ denote a monotone mixed adversarial structure over $\mathcal{P} = \{\mathrm{P}_1, \ldots, \mathrm{P}_m\}$, where $m \geq 3$. Assume that $\mathcal{R}$ is* not $\mathsf{Q}\,(2,2)$ *or it is* not $\mathsf{Q}\,(3,0)$*. Then, there exists an m-party functionality f which cannot be computed with statistical $\mathcal{R}$-FaF full-security, assuming the parties do not have an available broadcast channel.*

Lemma 7.8. *Let $\mathcal{R}$ denote a monotone mixed adversarial structure over $\mathcal{P} = \{\mathrm{P}_1, \ldots, \mathrm{P}_m\}$, where $m \geq 3$. Assume that $\mathcal{R}$ is* not $\mathsf{Q}\,(3,2)$*. Then, there exists an m-party functionality f which cannot be computed with perfect $\mathcal{R}$-FaF full-security. Moreover, this holds even when assuming an available broadcast channel.*

The proofs of Lemmas 7.7 and 7.8 are done similarly to the proof of Lemma 7.6. For the former we reduce it from the computation of the three-party broadcast functionality, which was shown by [41] to be impossible to compute against a malicious adversary corrupting a single party. The latter is reduced to the computation of the function $f(x_1, x_2, \perp) = (x_1 \wedge x_2, x_1 \wedge x_2, \perp)$, which was shown by [24] to be impossible to compute against the adversarial structure $\{(\emptyset, \{\mathrm{P}_1\}),\ (\emptyset, \{\mathrm{P}_2\}),\ (\{\mathrm{P}_3\}, \emptyset)\}$. We provide the proofs in Appendix A for completeness.

7.3 Proofs of Positive Results

In this section, we prove the positive direction of Theorem 7.4. We prove it by showing that the protocols provided by [24] are *strong* FaF fully-secure. Note that although real-world mixed-adversaries corrupting $(\mathcal{I}, \mathcal{H}) \in \mathcal{R}$ are strictly stronger than their FaF counterpart, it is not straight forward that mixed-security of a protocol Π implies FaF-security. This is due to the fact that an *ideal-world* mixed-adversary is also strictly strogner than its ideal-world FaF counterpart. Indeed, an ideal-world mixed-adversary corrupting $(\mathcal{I}, \mathcal{H})$ might send to the trusted party inputs on behalf of the parties in $\mathcal{I}$ based on the inputs of the parties in $\mathcal{H}$. In fact, Example 5.4 demonstrates that computational mixed-security does not imply computational FaF-security. Finding a counterexample for statistical security remains open. We next state the positive direction of the three cases in Theorem 7.4.

Lemma 7.9. *Let $\mathcal{R}$ be a monotone mixed adversarial structure over a set $\mathcal{P} = \{\mathrm{P}_1, \ldots, \mathrm{P}_m\}$ for some $m \geq 3$. Then:*

1. *If $\mathcal{R}$ is $\mathsf{Q}\,(2,2)$, then $\mathcal{R}$ allows for general m-party strong* statistical *FaF fully-secure MPC, assuming an available broadcast channel.*

2. *If $\mathcal{R}$ is both $\mathsf{Q}(2,2)$ and $\mathsf{Q}(3,0)$, then $\mathcal{R}$ allows for general m-party strong* statistical *FaF fully-secure MPC (without broadcast).*

3. *If $\mathcal{R}$ is $\mathsf{Q}(3,2)$, then $\mathcal{R}$ allows for general m-party strong* perfect *FaF fully-secure MPC (without broadcast).*

In all three cases of Lemma 7.9, the resulting protocol is described in a recursive manner, where the main difference is in the basis of the recursion. We next present an informal description of the idea, followed by a formalized proof. We start with some notations.

Notations. An element $(\mathcal{I},\mathcal{H}) \in \mathcal{R}$ is called maximal if for all $(\mathcal{I}',\mathcal{H}') \in \mathcal{R}$, if $(\mathcal{I},\mathcal{H}) \leq (\mathcal{I}',\mathcal{H}')$ then $(\mathcal{I},\mathcal{H}) = (\mathcal{I}',\mathcal{H}')$. A *basis* of $\mathcal{R}$ is the set of all maximal elements in $\mathcal{R}$. In addition, we let the *monotone closure* of $\mathcal{R}$ be the set

$$\mathsf{mCl}(\mathcal{R}) = \left\{(\mathcal{I},\mathcal{H}) \subseteq \mathcal{P}^2 \colon \mathcal{I} \cap \mathcal{H} = \emptyset,\ \exists(\mathcal{I}',\mathcal{H}') \in \mathcal{R} \text{ such that } (\mathcal{I},\mathcal{H}) \leq (\mathcal{I}',\mathcal{H}')\right\},$$

of valid pairs that are smaller than some element in $\mathcal{R}$. It is not hard to see that $\mathcal{R}$ is monotone if and only if $\mathcal{R} = \mathsf{mCl}(\mathcal{R})$.

7.3.1 The Recursive Protocol

We next describe the recursive protocol and prove its security. We start with giving an informal overview of the construction. Fix a monotone mixed adversarial structure $\mathcal{R}$, and let $\mathcal{B}$ be the basis of $\mathcal{R}$. For the recursive protocol we assume that $|\mathcal{B}| \geq 4$. The parties are going to emulate the 4-party BGW protocol secure against a single malicious party. Let Q_1,Q_2,Q_3, and Q_4 be the virtual parties running the BGW protocol. The virtual parties compute the following functionality. The input of Q_j is defined as follows. Share each x_i in a 2-out-of-4 secret sharing scheme, where x_i is held by P_i. The input of Q_j is its respective shares of each input, i.e., Q_j holds $(x_i[j])_{i=1}^m$. The output is defined as follows. Let $\mathbf{y}$ be the output of $f(\mathbf{x})$. Share each y_i in a 2-out-of-4 secret sharing scheme. The output of Q_j is its respective shares of each y_i, i.e., it receives $(y_i[j])_{i=1}^m$.

The idea in emulating the computation of the 4-party BGW protocol, is to have each value held by a virtual party, shared by the real parties. The secret sharing scheme functionality is denoted by $\mathsf{Sh}_{\mathcal{B}}$. It is a reactive functionality, where the sharing phase shares the secret so that the parties in $\mathcal{I}\cup\mathcal{H}$, where $(\mathcal{I},\mathcal{H}) \in \mathcal{B}$, has no information on the secret. We further denote the reconstruction function of $\mathsf{Sh}_{\mathcal{B}}$ as $\mathsf{Rec}_{\mathcal{B}}$. The functionality is formally described in Figure 1. We first formulate the interaction assuming the parties have access to an idealized functionality that can perform the local computations supposedly done by each virtual party. The functionality takes shares of the $\mathsf{Sh}_{\mathcal{B}}$ scheme, and output shares of the same scheme.

The real parties first share their inputs using two layers of secret sharing schemes. In the first layer, they use a 2-out-of-4 Shamir's secret sharing scheme (and associate the i-th share as the input of the virtual party $\mathrm{Q}_{j'}$). Then, they share each secret in an additional layer of sharing using a $\mathsf{Sh}_{\mathcal{B}}$ secret sharing scheme. Next, the real parties sample together randomness for each of the virtual parties. This will be done using the idealized functionality that sample randomness. Then, during the execution of the protocol, each message that is supposedly sent by one of the four virtual parties, is also going to be computed using the idealized functionality. Finally, to recover the output, the parties will call one last time to the functionality to recover each virtual party's

Sharing phase:

- *Input.* One party, designated as the dealer, provides some input s and no other parties provides any input.
- *Output.* The functionality will share s using [37] secret sharing scheme for general access structures, where the minterms are $\mathcal{I} \cup \mathcal{H} \cup \{\mathrm{P}\}$, where $(\mathcal{I}, \mathcal{H}) \in \mathcal{B}$ and $\mathrm{P} \notin \mathcal{I} \cup \mathcal{H}$. Additionally, the functionality records the value s.

Reconstruction phase:

- *Input.* Every party holds the shares of s received as the output of the previous phase.
- *Output.* If the shares of are consistent with s, then each party receives s from the functionality. Otherwise, each party receives the default value 0.

Figure 1: The $\mathsf{Sh}_{\mathcal{B}}$ secret sharing functionality

output. They can simply apply the reconstruction of the $\mathsf{Sh}_{\mathcal{B}}$ secret sharing to receive a Shamir share of the output, which they can reconstruct as well.

In order to implement the idealized functionalities, the parties are going to use the exact same protocol, i.e., emulating 4-party BGW protocol. This can continue recursively, until the basis protocol, which will differ based on the assumption on the adversarial structure. Each BGW emulation will admit security against "smaller" adversarial structures. Specifically, we partition the basis $\mathcal{B}$ into four subsets $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4$, each of size at least $\lfloor |\mathcal{B}|/4 \rfloor$, and denote $\overline{\mathcal{B}}_j = \mathcal{B} \setminus \mathcal{B}_j$. Then the implementation of each local computation done by virtual party Q_j, will be an emulation of 4-party BGW protocol, that is $\mathsf{mCl}\left(\overline{\mathcal{B}}_j\right)$-FaF fully-secure, that is, it tolerates adversaries corrupting according to the adversarial structure induced by the basis $\overline{\mathcal{B}}_j$. Recall that we assume $|\mathcal{B}| \geq 4$. Therefore, each $\overline{\mathcal{B}}_j$ is a basis for $\mathsf{mCl}\left(\overline{\mathcal{B}}_j\right)$ that is strictly smaller than $\mathcal{B}$. Therefore, it can be inductively replaced with a secure protocol. Moreover, observe that the corrupted set $(\mathcal{I}, \mathcal{H})$ can belong to at most one $\mathcal{B}_j$. In this case, only the messages computed for Q_j are not guaranteed to be correct, or that Q_j could "learn" some information. This can be viewed as if it is the only corrupted party (either malicious or semi-honest) in the 4-party BGW protocol.

We next formalize the above protocol. The idealized functionalities we will use, modeling the inner computation of each virtual party, are denoted $\mathcal{F}^j_{g,\mathcal{B}'}$. It is parametrized by the index of the virtual party Q_j, some function $g : \mathcal{V}_j \mapsto \mathcal{V}_j$ whose domain and range are the set of all possible views of Q_j, and a strict subset $\mathcal{B}' \subset \mathcal{B}$, representing a basis under which the functionality would be implemented using a secure protocol. The functionality $\mathcal{F}^j_{g,\mathcal{B}'}$ is defined in Figure 2.

In the following, for $j \in [4]$ we let $\mathsf{Next}_{j,\ell}$ be the next-message function of round ℓ for Q_j as specified by the BGW protocol. We view $\mathsf{Next}_{j,\ell}$ as a deterministic function, where the randomness of Q_j is given as an input. Additionally, we let $\mathsf{Next}_{j,0}$ be the function which samples randomness for Q_j, and we let $\mathsf{Next}_{j,r+1}$ be the function that computes the output of Q_j, where r is the number of rounds in the BGW protocol. The following lemma, states that it is possible to compute any functionality against any monotone mixed adversarial structure whose basis is of size at least 4, given the appropriate hybrid model.

Input: Suppose that $\mathbf{v}_{\ell,j} = (v_{1,j}, \ldots, v_{\ell,j}) \in \mathcal{V}_j$ is some possible partial view of Q_j. The input for party P_i is $(v_{k,j}[i])_{k=1}^{\ell}$, where each $v_{k,j}[i]$ is the i-th share of $v_{k,j}$ in a $\mathsf{Sh}_{\mathcal{B}}$ secret sharing scheme.
Output: Let $(\mathcal{I}, \mathcal{H}) \in \mathcal{R}$ be the set of corrupted parties.

1. Assume that $(\mathcal{I}, \mathcal{H}) \in \mathsf{mCl}(\mathcal{B}')$ (i.e., FaF-security holds). In this case, the functionality recovers $\mathbf{v}_{\ell,j}$ using $\mathsf{Rec}_{\mathcal{B}}$ and compute $\mathbf{v}_{\ell+1,j} = g(\mathbf{v}_{\ell,j})$. The functionality then share $\mathbf{v}_{\ell+1,j}$ using $\mathsf{Sh}_{\mathcal{B}}$ and send each party its respective share.

2. Assume that $(\mathcal{I}, \mathcal{H}) \notin \mathsf{mCl}(\mathcal{B}')$ and that $(\mathcal{I}, \emptyset) \in \mathsf{mCl}(\mathcal{B}')$ (i.e., standard security against the malicious adversary holds, yet FaF-security is not guaranteed to hold). Similarly to the previous case, the functionality recovers $\mathbf{v}_{\ell,j}$ using $\mathsf{Rec}_{\mathcal{B}}$ and compute $\mathbf{v}_{\ell+1,j} = g(\mathbf{v}_{\ell})$. The functionality then share $\mathbf{v}_{\ell+1,j}$ using $\mathsf{Sh}_{\mathcal{B}}$ and send each party its respective share. In addition, the functionality gives $\mathbf{v}_{\ell,j}$ to the parties in $\mathcal{H}$.

3. Assume that $(\mathcal{I}, \emptyset) \notin \mathsf{mCl}(\mathcal{B}')$ (i.e., standard security is not guaranteed to hold). In this case, send the *inputs of all parties* to the adversary the to receive purported sharing of some value $\mathbf{v}^*_{\ell+1,j}$. Send each party its respective share of $\mathbf{v}^*_{\ell+1,j}$.

Figure 2: Ideal functionality $\mathcal{F}^j_{g,\mathcal{B}'}$

Lemma 7.10. *Let $\mathcal{R}$ denote a monotone mixed adversarial structure over $\mathcal{P} = \{\mathrm{P}_1, \ldots, \mathrm{P}_m\}$, whose basis $\mathcal{B}$ is of size at least 4, and where $m \geq 3$. Partition $\mathcal{B}$ into four sets $\mathcal{B}_1, \ldots, \mathcal{B}_4$, each of size at least $\lfloor |\mathcal{B}|/4 \rfloor$, and let $\overline{\mathcal{B}}_j = \mathcal{B} \setminus \mathcal{B}_j$ for all $j \in [4]$. Then $\mathcal{R}$ allows for general m-party strong perfect FaF fully-secure MPC in the $\left(\mathsf{Sh}_{\mathcal{B}}, \left(\mathcal{F}^j_{\mathsf{Next}_{j,\ell},\overline{\mathcal{B}}_j}\right)_{i,j\in[4],\ell\in\{0,\ldots,r+1\}}\right)$-hybrid model.*

Proof. Fix some m functionality f. We next present the protocol Π^{rec}_f for computing f with strong perfect $\mathcal{R}$-FaF full-security. For brevity, we will write $\mathcal{F}^j_\ell$ for the idealized functionality, instead of $\mathcal{F}^j_{\mathsf{Next}_{j,\ell},\overline{\mathcal{B}}_j}$.

Protocol 7.11 (Π^{rec}_f).
Inputs: The input for party P_i is $x_i \in \mathcal{X}_i$.

Sharing the inputs: *Each party P_i share its input x_i using two layers of secret sharing schemes.*

1. *The outer layer is a 2-out-of-4 Shamir's secret sharing scheme.*
2. *The inner layer is a $\mathsf{Sh}_{\mathcal{B}}$ secret sharing scheme.*

Denote by $x_i[j]$ the j-share of the inner layer scheme. Party P_i then sends each other party its respective share, that is, party P_j receives $x_i[j]$.

Generating randomness: *For each $j \in [4]$, the parties call $\mathcal{F}^j_0$ with no inputs, to receive shares of randomness for the virtual party Q_j.*

Emulating the BGW protocol: *For $\ell = 1$ to r do the following: Suppose that the next message of the BGW protocol is sent from Q_j to $\mathrm{Q}_{j'}$. The parties call $\mathcal{F}^j_\ell$ with their current view as the input. The parties receive an output $\mathbf{v}_{\ell+1,j}$ (in shared form) which they keep as being part of $\mathrm{Q}_{j'}$'s view.*

Recovering the output: *After the interaction of the emulated BGW protocol has ended, the parties call $\mathcal{F}^{j}_{r+1}$ for every $j \in [4]$. Next, the parties recover the output by calling $\mathsf{Rec}_{\mathcal{B}}$ for each of the four outputs of the virtual parties, and then locally apply Shamir's sharing reconstruction (with error-correcting).*

We next show that Π^{rec}_{f} admits strong perfect $\mathcal{R}$-FaF full-security. Fix a malicious adversary $\mathcal{A}$ and a semi-honest adversary $\mathcal{A}_{\mathcal{H}}$, corrupting the subset $\mathcal{I} \subseteq \mathcal{P}$ and $\mathcal{H} \subseteq \mathcal{P}$ respectively, such that $(\mathcal{I}, \mathcal{H}) \in \mathcal{R}$. We first intuitively explain why simulation is possible. Assume that $(\mathcal{I}, \mathcal{H}) \in \mathsf{mCl}\left(\mathcal{B}_j\right)$ for some $j \in [4]$. Observe that since the $\mathcal{B}_j$'s form a partition of $\mathcal{B}$, in all hybrids $\left(\mathcal{F}^{j'}_{\ell}\right)^{r+1}_{\ell=0}$, for every $j' \neq j$, the correct value will be computed, without revealing anything to any of the two adversaries (i.e., Case 1 holds). Therefore, the only set of hybrids for which the (at least) one of the adversaries receive additional information, are $\left(\mathcal{F}^{j}_{\ell}\right)^{r+1}_{\ell=0}$. Intuitively, this can be viewed as if at most one of the virtual parties, namely Q_j, is corrupted (either maliciously or semi-honestly). Moreover, by the monotonicity of $\mathsf{mCl}\left(\mathcal{B}_j\right)$, it follows that $(\mathcal{I}, \emptyset) \in \mathsf{mCl}\left(\mathcal{B}_j\right)$.

We separate into three cases. For the case we will assume that there exists a unique $j \in [4]$ for which $(\mathcal{I}, \emptyset) \in \mathsf{mCl}\left(\mathcal{B}_j\right)$. Note that by monotonicity, this also implies that $(\mathcal{I}, \mathcal{H}) \notin \mathsf{mCl}\left(\mathcal{B}_{j'}\right)$ for any other j'. Intuitively, when the parties call $\mathcal{F}^{j}_{\ell}$ for some ℓ, since $(\mathcal{I}, \emptyset) \notin \mathsf{mCl}\left(\overline{\mathcal{B}}_j\right)$ the hybrid will give the malicious adversary full control over the input and the output. When the parties call $\mathcal{F}^{j'}_{\ell}$ for any other j', it holds that $(\mathcal{I}, \mathcal{H}) \in \mathsf{mCl}\left(\overline{\mathcal{B}}_{j'}\right)$. Therefore, both the malicious adversary and the semi-honest adversary will gain no information. As a result, this case corresponds to the case where the virtual party Q_j is maliciously corrupted in the 4-party BGW protocol, and no additional semi-honest virtual party is present. For the second case, in addition for $(\mathcal{I}, \emptyset)$ being in more than one $\mathsf{mCl}\left(\mathcal{B}_j\right)$, we will assume that $(\mathcal{I}, \mathcal{H})$ does belong to a single $\mathsf{mCl}\left(\mathcal{B}_j\right)$. Analogously to the previous case, on the intuitive level, this corresponds to the case where the virtual party Q_j is semi-honest, and no additional malicious virtual party is present. For the third and final case, we will assume that $(\mathcal{I}, \mathcal{H})$ belongs to a more than one $\mathsf{mCl}\left(\mathcal{B}_j\right)$. In the following we let $\mathsf{S}^{j}_{\mathrm{BGW}}$ be the malicious simulator for the single corrupted party Q_j in the 4-party BGW protocol. The corrupted party Q_j may be malicious or semi-honest. We rely on the fact that $\mathsf{S}^{j}_{\mathrm{BGW}}$ is black-box and straight-line.

First Case. Suppose there exists a unique $j \in [4]$ such that $(\mathcal{I}, \emptyset) \in \mathsf{mCl}\left(\mathcal{B}_j\right)$. Roughly, the malicious simulator will generate the shares for each message based on which virtual party supposedly sent the message, by either using the adversary, the (malicious) BGW simulator, or it will produce shares of some arbitrary message. The semi-honest simulator will complete the shares whenever the parties in $\mathcal{H}$ hold the secret (e.g., the inputs). We next formally describe the malicious simulator $\mathsf{S}_{\mathcal{A}}$ on input $\mathbf{x}_{\mathcal{I}}$ and auxiliary input $z_{\mathcal{A}}$.

Sharing the inputs: The simulator generate shares of 0 for each uncorrupted party, in two layers, and give the adversary $\mathcal{A}$ its respective shares. The adversary then answers with an input $\mathbf{x}^{*}_{\mathcal{I}}$ to be sent to the functionality $\mathsf{Sh}_{\mathcal{B}}$. The simulator then sends it to the trusted party, to receive an output $\mathbf{y}_{\mathcal{I}}$.

Generating randomness: For all $j' \in [4]$, where $j' \neq j$, compute $\mathsf{Next}_{j',0}$, share the output in a $\mathsf{Sh}_{\mathcal{B}}$ secret sharing scheme, and give the adversary $\mathcal{A}$ its respective shares. In addition, query the adversary to receive purported shares of the randomness it chose for Q_j.

Emulating the BGW protocol: For $\ell = 1$ to r do the following.

- If the next message of the BGW protocol is sent from (malicious) Q_j to $\mathrm{Q}_{j'}$, where $j' \neq j$, query the adversary to receive purported shares of the message it chose to send.
- If the next message of the BGW protocol is sent from $\mathrm{Q}_{j'}$ to (malicious) Q_j, where $j' \neq j$, then have the BGW simulator $\mathsf{S}^j_{\mathrm{BGW}}$ compute the ℓ-th message, share it in a $\mathsf{Sh}_{\mathcal{B}}$ secret sharing scheme, and give the adversary $\mathcal{A}$ its respective shares.
- If the next message of the BGW protocol is sent from $\mathrm{Q}_{j'}$ to $\mathrm{Q}_{j''}$, where $j', j'' \neq j$, share some arbitrary message in a $\mathsf{Sh}_{\mathcal{B}}$ secret sharing scheme, and give the adversary $\mathcal{A}$ its respective shares.

If at any point in the simulation, the BGW simulator $\mathsf{S}^j_{\mathrm{BGW}}$ sends its input to the trusted party, give it an output described as follows. Let $\mathbf{y}^* = (y^*_1, \ldots, y^*_m)$ be defined as $y^*_i = y_i$ for every $\mathrm{P}_i \in \mathcal{I}$, and $y^*_i = 0$ otherwise. The output is then a sharing of $\mathbf{y}^*$, where the sharing being used is a 2-out-of-4 Shamir's secret sharing scheme for each entry in $\mathbf{y}^*$. Denote by $\mathbf{y}^*[j']$ the j'-th vector of shares of $\mathbf{y}^*$.

Recovering the output: Query the adversary to receive purported shares of the output it chose for Q_j. For all $j' \in [4]$, where $j' \neq j$, the output of $\mathrm{Q}_{j'}$ will be $\mathbf{y}^*[j']$. Share each $\mathbf{y}^*[j']$ using a $\mathsf{Sh}_{\mathcal{B}}$ secret sharing scheme, and give $\mathcal{A}$ its respective shares. The adversary answers with shares to be sent to $\mathsf{Rec}_{\mathcal{B}}$ for each virtual party $\mathrm{Q}_{j'}$, for $j' \neq j$. Finally, answer $\mathcal{A}$ with $\mathbf{y}^*[j']$, for every $j' \neq j$ and halt.

We now formally describe the semi-honest simulator $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ on input $\mathbf{x}_{\mathcal{H}}$ and auxiliary input $z_{\mathcal{H}}$.

Receive output and ideal-world view of $\mathsf{S}_{\mathcal{A}}$: Receive the output $\mathbf{y}_{\mathcal{H}}$ from the trusted party, and receive r, $\mathbf{x}_{\mathcal{I}}$, $z_{\mathcal{A}}$, $\mathbf{y}_{\mathcal{I}}$ – $\mathsf{S}_{\mathcal{A}}$'s randomness, input, auxiliary input, and output respectively.

Sharing the inputs: For each party $\mathcal{P}_i \in \mathcal{H}$, the simulator completes the shares generated by the adversary to shares of x_i. For each party $\mathcal{P}_i \notin \mathcal{I} \cup \mathcal{H}$, use the shares of 0 generated by $\mathsf{S}_{\mathcal{A}}$.

Generating randomness: For all $j' \in [4]$, where $j' \neq j$, use the shares generated by $\mathsf{S}_{\mathcal{A}}$, and give $\mathcal{H}$ their respective shares. In addition, use the same purported shares of the randomness for Q_j that $\mathcal{A}$ gave to $\mathsf{S}_{\mathcal{A}}$. Send $\mathcal{A}$ the shares it expect to see, to receive the messages it sends to the parties in $\mathcal{H}$.

Emulating the BGW protocol: For $\ell = 1$ to r use the shares generated by $\mathsf{S}_{\mathcal{A}}$ for each of the messages. In addition, query $\mathcal{A}$ to receive the messages it sends to the parties in $\mathcal{H}$.

Recall that $\mathbf{y}^* = (y^*_1, \ldots, y^*_m)$ defined as $y^*_i = y_i$ for every $\mathrm{P}_i \in \mathcal{I}$, and $y^*_i = 0$ otherwise, was shared and given to $\mathsf{S}^j_{\mathrm{BGW}}$ as an output. The sharing used was a 2-out-of-4 Shamir's secret sharing scheme for each entry in $\mathbf{y}^*$. The semi-honest simulator $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ will complete the shares of each y^*_i, where $\mathrm{P}_i \in \mathcal{H}$ is semi-honest, so that the secret is y_i. Let $\mathbf{y}$ be the new shared vector.

Recovering the output: Query the adversary to receive purported shares of the output it chose for Q_j. For all $j' \in [4]$, where $j' \neq j$, the output of $\mathrm{Q}_{j'}$ will be $\mathbf{y}[j']$. Share each $\mathbf{y}[j']$ using a $\mathsf{Sh}_{\mathcal{B}}$ secret sharing scheme, and give $\mathcal{A}$ and $\mathcal{H}$ their respective shares. In addition, give $\mathcal{H}$ the outputs $\mathbf{y}[j']$ for each $j' \neq j$. Finally, send the adversary its respective shares to receive the messages it sends to the parties in $\mathcal{H}$.

We now show that
$$\text{IDEAL}^{\mathsf{S}_{\mathcal{A}},\mathsf{S}_{\mathcal{A},\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}} \equiv \text{REAL}^{\mathcal{A},\mathcal{A}_{\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}}.$$
Observe that in the real world, the view of $\mathcal{A}$ and $\mathcal{H}$ consist of the view of the malicious virtual party Q_j, and additional shares, shared according to $\mathsf{Sh}_{\mathcal{B}}$. Each round where Q_j received messages, is being simulated by $\mathsf{S}^j_{\mathrm{BGW}}$, hence the message that $\mathcal{A}$ receives is distributed the same in both the real and ideal models, conditioned on previous messages. In other rounds, $\mathcal{A}$ and $\mathcal{H}$ get shares that, by the secret sharing properties, are distributed the same in both worlds, conditioned on previous messages. As a result, it follows that the messages that $\mathcal{A}$ chooses that Q_j will send are also distributed the same. Therefore, each round the joint view is distributed exactly the same both the real and the ideal model, conditioned on the previous rounds. Finally, the output of the uncorrupted parties, is determined in the ideal world by the BGW simulator, which is assumed to have the same distribution as the real world output.

Second Case. Suppose that $(\mathcal{I}, \emptyset)$ belongs to at least two of the $\mathsf{mCl}(\mathcal{B}_j)$'s, yet there exists a unique $j \in [4]$ such that $(\mathcal{I}, \mathcal{H}) \in \mathsf{mCl}(\mathcal{B}_j)$. We next formally describe the malicious simulator $\mathsf{S}_{\mathcal{A}}$ on input $\mathbf{x}_{\mathcal{I}}$ and auxiliary input $z_{\mathcal{A}}$. Unlike in the previous case, the malicious adversary has no control over a virtual party, however, the semi-honest parties control a virtual party passively.

Sharing the inputs: The simulator generate shares of 0 for each uncorrupted party, in two layers, and give the adversary $\mathcal{A}$ its respective shares. The adversary then answers with an input $\mathbf{x}^*_{\mathcal{I}}$ to be sent to the functionality $\mathsf{Sh}_{\mathcal{B}}$. The simulator then sends it to the trusted party, to receive an output $\mathbf{y}_{\mathcal{I}}$.

Generating randomness: For all $j' \in [4]$, where $j' \neq j$, compute $\mathsf{Next}_{j',0}$, share the output in a $\mathsf{Sh}_{\mathcal{B}}$ secret sharing scheme, and give the adversary $\mathcal{A}$ its respective shares. In addition, query the adversary to receive purported shares of the randomness it chose for Q_j.

Emulating the BGW protocol: For $\ell = 1$ to r, share some arbitrary message in a $\mathsf{Sh}_{\mathcal{B}}$ secret sharing scheme, and give the adversary $\mathcal{A}$ its respective shares.

Recovering the output: For all $j' \in [4]$, the output of $\mathrm{Q}_{j'}$ will be some arbitrary output, shared using a $\mathsf{Sh}_{\mathcal{B}}$ secret sharing scheme. Give $\mathcal{A}$ its respective shares.

We now formally describe the semi-honest simulator $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ on input $\mathbf{x}_{\mathcal{H}}$ and auxiliary input $z_{\mathcal{H}}$.

Receive output and ideal-world view of $\mathsf{S}_{\mathcal{A}}$: Receive the output $\mathbf{y}_{\mathcal{H}}$ from the trusted party, and receive r, $\mathbf{x}_{\mathcal{I}}$, $z_{\mathcal{A}}$, $\mathbf{y}_{\mathcal{I}}$ – $\mathsf{S}_{\mathcal{A}}$'s randomness, input, auxiliary input, and output respectively.

Sharing the inputs: For each party $\mathcal{P}_i \in \mathcal{H}$, the simulator completes the shares generated by the adversary to shares of x_i. For each party $\mathcal{P}_i \notin \mathcal{I} \cup \mathcal{H}$, use the shares of 0 generated by $\mathsf{S}_{\mathcal{A}}$.

Generating randomness: For all $j' \in [4]$, where $j' \neq j$, use the shares generated by $\mathsf{S}_{\mathcal{A}}$, and give $\mathcal{H}$ their respective shares. In addition, complete the shares of Q_j's randomness, generated by $\mathsf{S}_{\mathcal{A}}$, to shares of a sample from $\mathsf{Next}_{j,0}$, and give $\mathcal{H}$ their respective shares.

Initializing $\mathsf{S}^{j}_{\mathrm{BGW}}$: In order to execute the semi-honest BGW simulator, $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ must compute an output for it. This is done as follows. Let $\mathbf{y}^* = (y^*_1, \ldots, y^*_m)$ be defined as $y^*_i = y_i$ for every $\mathrm{P}_i \in \mathcal{I} \cup \mathcal{H}$, and $y^*_i = 0$ otherwise. Give $\mathsf{S}^{j}_{\mathrm{BGW}}$ shares of each y^*_i, where the sharing being used is a 2-out-of-4 Shamir's secret sharing scheme. Denote by $\mathbf{y}^*[j']$ the j'-th vector of shares of $\mathbf{y}^*$.

Emulating the BGW protocol: For $\ell = 1$ to r do the following.

- If the next message of the BGW protocol is sent from (semi-honest) Q_j to $\mathrm{Q}_{j'}$, where $j' \neq j$, complete the shares generated by $\mathsf{S}_{\mathcal{A}}$ to shares of an output computed by $\mathsf{Next}_{j,\ell}$, and give the output to the parties in $\mathcal{H}$.
- If the next message of the BGW protocol is sent from $\mathrm{Q}_{j'}$ to (semi-honest) Q_j, where $j' \neq j$, complete the shares generated by $\mathsf{S}_{\mathcal{A}}$ to shares of the ℓ-th message computed by the BGW simulator $\mathsf{S}^{j}_{\mathrm{BGW}}$ compute the, and give the parties in $\mathcal{H}$ their respective shares.
- If the next message of the BGW protocol is sent from $\mathrm{Q}_{j'}$ to $\mathrm{Q}_{j''}$, where $j', j'' \neq j$, the use the shares generated by $\mathsf{S}_{\mathcal{A}}$.

Recovering the output: The output of Q_j is $\mathbf{y}^*[j]$, which is given to the parties in $\mathcal{H}$. For all $j' \in [4]$, where $j' \neq j$, the output of $\mathrm{Q}_{j'}$ will be $\mathbf{y}[j']$. Share each $\mathbf{y}[j']$ using a $\mathsf{Sh}_{\mathcal{B}}$ secret sharing scheme, and give the parties in $\mathcal{H}$ their respective shares. Send the adversary its respective shares to receive the messages it sends to the parties in $\mathcal{H}$.

The proof of security for this case is analogue to the first case, and is thus omitted.

Third Case. Suppose that $(\mathcal{I}, \mathcal{H})$ belongs to at least two of the $\mathsf{mCl}(\mathcal{B}_j)$'s. In this case, $\mathsf{S}_{\mathcal{A}}$ is the same simulator from the second case. The semi-honest simulator $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ will simply use the shares generated by $\mathsf{S}_{\mathcal{A}}$. The shares for the inputs and outputs will be completed accordingly. The security of this case easily follows from the properties of the secret sharing scheme. □

7.3.2 Protocols For The Basis Cases

In this section, we provide the basis protocol to be used in order to construct the statistically/perfectly (strong) FaF fully-secure protocols. We start with the basis for the perfect case, then with the basis for the statistical case with broadcast, and finally, we present the case for the statistical security without broadcast.

Recall that for the perfect case, we assume that the adversarial structure $\mathcal{R}$ is $\mathsf{Q}(3,2)$. For the basis case, we assume that its basis $\mathcal{B}$ is of size $|\mathcal{B}| \leq 3$. The protocol is defined as follows.

Protocol 7.12 (Π^{per}_f)**.**
Inputs: The input of party P_i *is* $x_i \in \mathcal{X}_i$*.*

Assume that $|\mathcal{B}| \leq 2$: *In this case, $\mathcal{B} = \{(\mathcal{I}_1, \mathcal{H}_1), (\mathcal{I}_2, \mathcal{H}_2)\}$ (not necessarily distinct). Since $\mathcal{R}$ is $\mathsf{Q}(3,2)$, it holds that $\mathcal{I}_1 \cup \mathcal{H}_1 \cup \mathcal{I}_2 \cup \mathcal{H}_2 \neq \mathcal{P}$. Therefore, there is a party P that does not belong to any $(\mathcal{I}, \mathcal{H}) \in \mathcal{R}$, and hence is not corrupted by any adversary (both malicious and semi-honest). The parties can therefore* send their inputs *to P who can compute the outputs for all other parties (note that we do not need to simulate the view of P).*

Assume that $|\mathcal{B}| = 3$: *In this case, $\mathcal{B} = \{(\mathcal{I}_1, \mathcal{H}_1), (\mathcal{I}_2, \mathcal{H}_2), (\mathcal{I}_3, \mathcal{H}_3)\}$. Since $\mathcal{R}$ is $\mathsf{Q}(3,2)$, it holds that $\mathcal{I}_1 \cup \mathcal{H}_1 \cup \mathcal{I}_2 \cup \mathcal{H}_2 \cup \mathcal{I}_3 \neq \mathcal{P}$. Therefore, there exists a party Q_3 that is either in $\mathcal{H}_3$ or it cannot be corrupted. Similarly, there are parties Q_1 and Q_2 that are either in $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively, or they cannot be corrupted.*

1. *Each party P_i share its input x_i in a 3-out-of-3 secret sharing scheme, and sends $x_i[j]$ to Q_j.*
2. *The three parties Q_1, Q_2, and Q_3 compute the following functionality. Let $\mathbf{y}$ be the output of $f(\mathbf{x})$. Share each y_i in a 3-out-of-3 secret sharing scheme. The output of Q_j is its respective shares of each y_i, i.e., it receives $(y_i[j])_{i=1}^m$. The computation can be done using any 3-party protocol that is secure against a single semi-honest party.*
3. *Each Q_j sends $y_i[j]$ to P_i, who then reconstructs the output.*

We claim that Π_f^{per} admits perfect strong FaF full-security.

Claim 7.13. *Let $\mathcal{R}$ denote a monotone mixed adversarial structure over $\mathcal{P} = \{\mathrm{P}_1, \ldots, \mathrm{P}_m\}$, whose basis $\mathcal{B}$ is of size at most 3, and where $m \geq 3$. If $\mathcal{R}$ is $\mathsf{Q}(3,2)$, then for every m-party functionality f, Π_f^{per} computes it with strong perfect $\mathcal{R}$-FaF full-security.*

Proof Sketch. For the simple case where $|\mathcal{B}| \leq 2$, the security follows from the fact that the party P used in the protocol is never corrupted, and thus it can be viewed as a trusted party. Assume that $\mathcal{B} = \{(\mathcal{I}_1, \mathcal{H}_1), (\mathcal{I}_2, \mathcal{H}_2), (\mathcal{I}_3, \mathcal{H}_3)\}$ is of size 3, and fix the parties Q_1, Q_2, and Q_3 satisfying $\mathrm{Q}_j \in \mathcal{H}_j$ or it cannot be corrupted, for all $j \in \{1,2,3\}$, as done in the protocol. Next, fix a malicious adversary $\mathcal{A}$ and a semi-honest adversary $\mathcal{A}_{\mathcal{H}}$, corrupting the subset $\mathcal{I} \subseteq \mathcal{P}$ and $\mathcal{H} \subseteq \mathcal{P}$ respectively, such that $(\mathcal{I}, \mathcal{H}) \in \mathcal{R}$. The malicious simulator $\mathsf{S}_{\mathcal{A}}$ will simply query $\mathcal{A}$ for the shares it sends to the Q_j's, recover the secret, and send it to the trusted party. The simulator then secret share each output y_i, for $i \in \mathcal{I}$, in a 3-out-of-3 secret sharing scheme, output the shares and halt. We next describe the semi-honest simulator $\mathsf{S}_{\mathcal{A},\mathcal{H}}$. If none of the Q_j's are in $\mathcal{H}$, then $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ operates analogously to $\mathsf{S}_{\mathcal{A}}$. If $\mathrm{Q}_j \in \mathcal{H}$ for some $j \in \{1,2,3\}$, then $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ will generate shares for each uncorrupted party, and use the shares given by $\mathcal{A}$ for each corrupted party. It will then execute the semi-honest simulator guaranteed to exist for Step 2. Finally, it will output the shares it generated, and the messages $\mathcal{A}$ sent to the parties in $\mathcal{H}$.

□

We next state the existence of a protocol in the statistical case with broadcast.

Claim 7.14. *Let $\mathcal{R}$ denote a monotone mixed adversarial structure over $\mathcal{P} = \{\mathrm{P}_1, \ldots, \mathrm{P}_m\}$, whose basis $\mathcal{B}$ is of size at most 3, and where $m \geq 3$. If $\mathcal{R}$ is $\mathsf{Q}(2,2)$, then for every m-party functionality f, there exists a protocol that computes it with strong perfect $\mathcal{R}$-FaF full-security.*

Proof Sketch. The protocol for this case will be similar to Π_f^{per}. If $|\mathcal{B}| \leq 2$ then it is exactly the same protocol, as the $\mathsf{Q}(2,2)$ property guarantees the existence of a party that is never corrupted by any of the adversaries. If $|\mathcal{B}| = 3$, then the only difference is that the $\mathsf{Q}(2,2)$ property guarantees the existence of three parties Q_1, Q_2, and Q_3, so that at most one of them will be corrupted. Unlike in the perfect, here the corrupted party may be malicious. Therefore, under the assumption of an available broadcast channel, the BGW protocol can still be executed among the three designated parties.

□

Finally, we state the claim for the statistical case without broadcast.

Claim 7.15. *Let $\mathcal{R}$ denote a monotone mixed adversarial structure over $\mathcal{P} = \{\mathrm{P}_1, \ldots, \mathrm{P}_m\}$, whose basis $\mathcal{B}$ is of size at most 3, and where $m \geq 3$. If $\mathcal{R}$ is $\mathsf{Q}(2,2)$ and $\mathsf{Q}(3,0)$, then for every m-party functionality f, there exists a protocol that computes it with strong perfect $\mathcal{R}$-FaF full-security.*

Proof Sketch. The protocol for this case will be similar to the statistically secure protocol from the previous case. If $|\mathcal{B}| \leq 2$ then it is exactly the same protocol. If $|\mathcal{B}| = 3$, then the two properties $\mathsf{Q}(2,2)$ and $\mathsf{Q}(3,0)$ guarantee the existence of three parties Q_1, Q_2, and Q_3, so that at most one of them will be semi-honest. Therefore, they can execute the BGW protocol without broadcast, similarly to the perfect case (here, however, only statistical security is achieved since the parties need to compute the broadcast functionality).

□

7.3.3 Protocols For The Secret Sharing Functionality

In this section, we show how to implement the secret sharing functionality $\mathsf{Sh}_{\mathcal{B}}$. Recall that if the adversarial structure $\mathcal{R}$ is $\mathsf{Q}(3,2)$ then the implementation should have perfect security, and if it is $\mathsf{Q}(2,2)$ then the implementation can have statistical security (without assuming the availability of broadcast). We first provide the implementation and proof of security for the perfect setting.

The perfect case. The sharing phase is implemented in a straightforward way, by having the dealer generate shares as done by [37]. For the reconstruction phase, the parties would search for the lexicographically first $(\mathcal{I}, \mathcal{H}) \in \mathcal{R}$ such that the shares of the parties in $\overline{\mathcal{I}}$ are consistent with some value s^*. The parties then recover s^* and output it. To show that the reconstruction works, we will show that for any such $(\mathcal{I}, \mathcal{H})$, the shares will be consistent with the *same* s^*, regardless of what the adversary does. In particular, since this will hold for the parties that are corrupted by the adversary, if the dealer is honest, then the value s^* will be the value s chosen by the dealer.

Claim 7.16. *Assume that $\mathcal{R}$ is $\mathsf{Q}(3,1)$. Then there exists a unique s^* such that for any $(\mathcal{I}, \mathcal{H}) \in \mathcal{R}$, the shares of the parties in $\overline{\mathcal{I}}$ are consistent with s^*.*

Proof. We show that by the $\mathsf{Q}(3,1)$ property, for every $(\mathcal{I}_1, \mathcal{H}_1), (\mathcal{I}_2, \mathcal{H}_2) \in \mathcal{R}$, the set $\mathcal{J} := \overline{\mathcal{I}}_1 \cap \overline{\mathcal{I}}_2$ is qualified with respect to $\mathsf{Sh}_{\mathcal{B}}$. Assume that $\mathcal{J}$ is unqualified. Then $\mathcal{I}' \cup \mathcal{H}' \cup \{\mathrm{P}\} \not\subseteq \mathcal{J}$ for any $(\mathcal{I}', \mathcal{H}') \in \mathcal{B}$ and $\mathrm{P} \notin \mathcal{I}' \cup \mathcal{H}'$. Hence, by monotonicity of $\mathcal{R}$ there exists $(\mathcal{I}', \mathcal{H}') \in \mathcal{R}$ such that $\mathcal{J} = \mathcal{I}' \cup \mathcal{H}'$. By the $\mathsf{Q}(3,1)$ property, it follows that

$$\mathcal{J} \cup \overline{\mathcal{J}} = \mathcal{I}' \cup \mathcal{H}' \cup \overline{\left(\overline{\mathcal{I}}_1 \cap \overline{\mathcal{I}}_2\right)} = \mathcal{I}' \cup \mathcal{H}' \cup \mathcal{I}_1 \cup \mathcal{I}_2 \neq \mathcal{P}$$

yielding a contradiction. Therefore $\mathcal{J}$ is qualified and thus the shares of the parties from this set are consistent with some value s^*. In particular, the shares of the parties from $\overline{\mathcal{I}}_1$, and the shares of the parties from $\overline{\mathcal{I}}_2$ are consistent with s^*. $\square$

The statistical case. Unlike in the perfect case, here we may only assume that $\mathcal{R}$ is $\mathsf{Q}(2,2)$ (we do not assume broadcast in the sharing protocol). The difficulty in this case, lies in the fact that the adversary may change its shares, hence there might exist two $(\mathcal{I},\mathcal{H}) \in \mathcal{R}$ such that the shares of the parties in $\overline{\mathcal{I}}$ are consistent with different secrets. Therefore, we let the dealer compute an authenticated sharing, thereby forcing the adversary to use the real shares it got from the dealer at the reconstruction phase. This can be done using standard techniques. For example, the dealer can sign each share using an information theoretic One-Time MAC, and give the other parties the key for verification. For the sake of clarity of presentation, we will assume that the malicious parties are using the correct shares. We next formally describe the reconstruction phase.

Protocol 7.17.

1. *Each* P_i *checks if its input is of the form* $\left(s[i], \mathrm{tag}_i, \left(\mathrm{key}_j\right)_{j=1}^{m}\right)$, *where* $s[i]$ *is the purported* i*-th share of* s, tag_i *is the tag of* $s[i]$, *and* key_j *is the verification key for the* j*-th tag.*
2. *If so,* P_i *sends* $(s[i], \mathrm{tag}_i)$ *to every other party (a malicious* P_i *may send different values to different parties). Otherwise, it sends an accusation of the dealer.*
3. *If there exists* $(\mathcal{I},\mathcal{H}) \in \mathcal{B}$ *such that the parties in* $\overline{\mathcal{I}}$ *accused the dealer, then disqualify the dealer and output the default value 0.*
4. *Otherwise, for every pair* $(s[j], \mathrm{tag}_j)$ *received, party* P_i *verify that* $\mathrm{Ver}_{\mathrm{key}_j}\left(s[j], \mathrm{tag}_j\right) = 1$. *If it is not 1, then send an accusation of* P_j.
5. *Party* P_j *is then disqualified, if there exists* $(\mathcal{I}_j,\mathcal{H}_j) \in \mathcal{B}$ *such that the parties in* $\overline{\mathcal{I}}_j$ *accused* P_j.
6. *If there exists a set of parties* $\overline{\mathcal{I}}$, *where* $(\mathcal{I},\mathcal{H}) \in \mathcal{B}$, *that were all disqualified, then disqualify the dealer and output the default value 0.*
7. *Otherwise, find the lexicographically first* $(\mathcal{I},\mathcal{H}) \in \mathcal{B}$ *such that the parties in* $\overline{\mathcal{I}}$ *were not disqualified and did not accuse the dealer, and output the secret that is consistent with their shares.*

The following claim asserts that any $\overline{\mathcal{I}}$ is qualified with respect to the sharing scheme.

Claim 7.18. *Assume that* $\mathcal{R}$ *is* $\mathsf{Q}(2,1)$. *Then for every* $(\mathcal{I},\mathcal{H}) \in \mathcal{R}$, *the set* $\mathcal{J} := \overline{\mathcal{I}}$ *is qualified with respect to* $\mathsf{Sh}_{\mathcal{B}}$.

Proof. Assume for the sake of contradiction, that $\mathcal{J}$ is unqualified. Then $\mathcal{I}' \cup \mathcal{H}' \cup \{\mathrm{P}\} \not\subset \mathcal{J}$ for any $(\mathcal{I}',\mathcal{H}') \in \mathcal{B}$ and $\mathrm{P} \notin \mathcal{I}' \cup \mathcal{H}'$. Hence, by monotonicity of $\mathcal{R}$ there exists $(\mathcal{I}',\mathcal{H}') \in \mathcal{R}$ such that $\mathcal{J} = \mathcal{I}' \cup \mathcal{H}'$. By the $\mathsf{Q}(2,1)$ property, it follows that

$$\mathcal{J} \cup \overline{\mathcal{J}} = \mathcal{I}' \cup \mathcal{H}' \cup \mathcal{I} \neq \mathcal{P}$$

yielding a contradiction. Therefore, $\mathcal{J}$ must a qualified set of parties. $\square$

In the following claims, we let $(\mathcal{I}, \mathcal{H}) \in \mathcal{R}$ be the pair of sets of corrupted parties. The next claim states that an uncorrupted dealer will not get disqualified.

Claim 7.19. *Assume that $\mathcal{R}$ is $\mathsf{Q}(2,1)$. If the dealer is uncorrupted, then it will not get disqualified.*

Proof. We separate into two cases. For the first case, let us assume that a set $\overline{\mathcal{I}'}$, for $(\mathcal{I}', \mathcal{H}') \in \mathcal{B}$, accused the dealer. Since the dealer is honest, the only accusations can come from the parties in $\mathcal{I}$, hence $\overline{\mathcal{I}'} \subseteq \mathcal{I}$. However, by Claim 7.18 the set $\overline{\mathcal{I}'}$ is qualified, and therefore $\mathcal{I}$ is qualified as well, yielding a contradiction.

We now the consider the second case in which the dealer is disqualified. Assume that a set of parties $\overline{\mathcal{I}'}$, for $(\mathcal{I}', \mathcal{H}') \in \mathcal{B}$, were disqualified. Since the dealer is honest, the only set of parties that can accuse them is $\mathcal{I}$. Therefore, for every $\mathrm{P}_j \in \overline{\mathcal{I}'}$, there exists $(\mathcal{I}_j, \mathcal{H}_j) \in \mathcal{B}$ such that $\overline{\mathcal{I}}_j \subseteq \mathcal{I}$. Similarly to the previous case, this contradicts Claim 7.18 since $\mathcal{I}$ is unqualified. □

The following claim asserts that the sharing protocol is in fact a VSS protocol.

Claim 7.20. *Assume that $\mathcal{R}$ is $\mathsf{Q}(2,1)$. If the dealer was not disqualified, then all the uncorrupted parties will output some secret s^*. Moreover, if the dealer is honest, then s^* will be the secret s originally shared by the dealer.*

Proof. Since the dealer was not disqualified, for any $(\mathcal{I}', \mathcal{H}') \in \mathcal{B}$, there exists a party in $\overline{\mathcal{I}'}$ that was not disqualified. By Claim 7.18 any $\overline{\mathcal{I}'}$ is qualified, and thus, the parties in $\overline{\mathcal{I}'}$ have shares that consistent with some secret s^*. Since this holds for any such $\mathcal{I}'$, the same holds for the lexicographically first $(\mathcal{I}', \mathcal{H}')$, hence the parties will reconstruct s^*. Moreover, if the dealer is honest, then by Claim 7.19 it will not get disqualified, hence the secret s^* will be the original secret s that the dealer shared. □

7.3.4 Putting it Together

We next combine the above claims to prove Lemma 7.9.

Proof of Lemma 7.9. The proof of all items follow the same inductive argument. Therefore, we only present the proof of Item 1. Fix a mixed adversarial structure $\mathcal{R}$ that is $\mathsf{Q}(2,2)$, and let $\mathcal{B}$ be the basis of $\mathcal{R}$. We next show that $\mathcal{R}$ allows for general MPC assuming an available broadcast channel. We prove this by induction on $|\mathcal{B}|$. The basis case, where $|\mathcal{B}| \leq 3$, follows from Claim 7.14. Assume that the claim holds for all structures with a basis of size strictly less than $|\mathcal{B}| \geq 4$. By Lemma 7.10, for every m-party functionality f there exists a protocol computing f in some hybrid model. Recall that in Lemma 7.10, we let $\mathcal{B}_1, \ldots, \mathcal{B}_4$ be a partition of $\mathcal{B}$, where each $\mathcal{B}_i$ is of size at least $\lfloor |\mathcal{B}|/4 \rfloor$. Since $|\mathcal{B}| \geq 4$, each $\mathcal{B}_i$ is not empty, and is of size that is strictly smaller than $|\mathcal{B}|$. Thus, by the inductive hypothesis, for each $i \in [4]$, every m-party functionality can be computed by some protocol that is $\mathsf{mCl}(\mathcal{B}_i)$-FaF fully-secure. By applying the composition theorem (see Theorem 9.1),[10] we get the existence of a protocol that computes f with $\mathsf{mCl}(\mathcal{B}_i)$-FaF full-security. Since this holds for every i, it also holds for the union of the structures, that is, the protocol is $\mathsf{mCl}(\mathcal{B})$-FaF fully-secure. □

[10]The composition theorem is stated and proved only against threshold adversaries, however, we note that the same proof work for general mixed adversarial structures.

8 Other (Failed) Attempts to Capture FaF-Security

In trying to come up with a definition for a new notion of security, one may expect to come across several attempts that seem to capture the essence of the notion, but in fact fall short in doing so. We think it is instructive to describe some of this failed approaches and exemplify where they come short.

First flawed definition: Recall that in the FaF full-security ideal-model, there are two distinct simulators, one for the malicious adversary and the other for the semi-honest parties. Indeed, the second simulator may use the view (in the ideal-world) of the malicious simulator. Suppose that instead, we require a *single simulator* controlling the parties in $\mathcal{I} \cup \mathcal{H}$. Further suppose that this simulator is restricted so to only be allowed to change inputs for the parties in $\mathcal{I}$ (i.e., the simulator is not allowed to change the inputs for the parties in $\mathcal{H}$).

Security would then be defined to hold if such a simulator is able to simulate the view of any adversary $\mathcal{A}$, and *every* $\mathcal{H} \subset \mathcal{P} \setminus \mathcal{I}$ in the above ideal-world. Since this is required for every $\mathcal{H}$, in particular it should hold for $\mathcal{H} = \emptyset$. A different way to view this approach is that it adopts the ideal-model of the *mixed-adversaries* definition, while considering FaF real-world adversaries, i.e., adversaries controlling only parties in $\mathcal{I}$ (still, being allowed to send *non-prescribed* messages to parties in $\mathcal{H}$).

While, this proposed definition seems to have the nice property of being easy to work with, we argue that it does not fully capture the desired properties that we are aiming for. Intuitively, this definition is too lenient, as it allows the simulator to depend on the inputs of parties in $\mathcal{H}$ (which the real-world adversary should not be able to do). Indeed, we argue that Protocol 5.5 from Example 5.4 constitutes an example for this shortcoming, namely, it is secure according to this definition, yet it should arguably *not* be deemed secure for the desired notion.

Second flawed definition: Suppose that instead of a single ideal-world experiment, where both the malicious simulator and the semi-honest simulator operate together, we have two ideal-world experiments, one for capturing standard security, and the other for trying to capture FaF-security. Specifically, for the first ideal-world experiment there is the standard malicious simulator $\mathsf{S}_{\mathcal{A}}$ that simulates the adversary's view. In the second experiment, in addition to the semi-honest simulator $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ controlling the parties in $\mathcal{H}$, there is another malicious simulator $\mathsf{S}'_{\mathcal{A}}$ who is given the task of sending inputs to the trusted party on behalf of the corrupted parties in $\mathcal{I}$.

Security would then be defined to hold if $\mathsf{S}_{\mathcal{A}}$ is able to simulate the adversary's view, and if $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ is able to simulate the view of the parties in $\mathcal{H}$, while interacting in the same ideal-world as $\mathsf{S}'_{\mathcal{A}}$. Definition 3.3 can be viewed as the same definition, while further requiring that $\mathsf{S}_{\mathcal{A}}$ and $\mathsf{S}'_{\mathcal{A}}$ are the same simulator.

At first glance, there seem to be no reason to require that the two simulators are the same. However, similarly to the first flawed definition, we claim that this notion is to lenient. Intuitively, this is due to the fact that $\mathsf{S}'_{\mathcal{A}}$ could send a different input than $\mathsf{S}_{\mathcal{A}}$ to "help" $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ in its simulation. We next show a concrete example where security according to this definition is possible, yet deeming it secure is undesirable.

Example 8.1. *Consider the following 4-party functionality* $f : \{0,1\} \times \emptyset \times (\{0,1\}^n)^2 \mapsto$

$(\{0,1\}^n \cup \{\perp\})^2 \times \emptyset^2$ *defined as follows*

$$f(b, \perp, x_3, x_4) = \begin{cases} (x_3, \perp, \perp, \perp) & \text{if } b = 0 \\ (\perp, x_4, \perp, \perp) & \text{if } b = 1 \end{cases}$$

We next show a protocol where computing f according to the above definition will be secure against a specific *malicious* P_1 *and* $\mathcal{H} = \{\mathrm{P}_2\}$*, yet simulation is impossible when considering FaF-security.*

Protocol 8.2.

1. P_3 *sends* x_3 *to* P_1

2. P_4 *sends* x_4 *to* P_2

3. P_1 *sends* b *to* P_2

4. If $b = 0$ *then* P_1 *outputs* x_3 *and* P_2 *outputs* $\perp$*. Otherwise* P_1 *outputs* $\perp$ *and* P_2 *outputs* x_4*. In both cases* P_3 *and* P_4 *output* $\perp$*.*

Clearly, the above protocol is not $(1,1)$*-FaF-secure since in the ideal-world either* P_1 *holds* x_3 *or* P_2 *holds* x_4*, but both cannot occur in the same experiment.*

We claim that the protocol is secure according to the definition proposed above, against a malicious P_1 *that always sends* $b = 0$ *to* P_2*. This is highly undesirable since in the real-world* P_1 *learns* x_3 *and* P_2 *learns* x_4*, however, in a single execution of the ideal-world this cannot happen. To simulate* P_1*, the malicious simulator* $\mathsf{S}_{\mathcal{A}}$ *will send* $b = 0$ *to* T *to receive* x_3*, and output it. On the other hand, the other malicious simulator* $\mathsf{S}'_{\mathcal{A}}$ *will send* $b = 1$*, forcing* T *to send* x_4 *to* $\mathsf{S}_{\mathcal{A},\mathrm{P}_2}$*, which it will then output. Since in the real-world* P_2 *always outputs* $\perp$*, it follows that the simulators achieve perfect simulation.*

9 Composition Theorem

In this section, we state and prove a variant of the sequential composition theorem of Canetti [14] for the FaF setting (both weak and strong). We formulate and prove the theorem for the computational setting. Similar composition theorems can be analogously stated and proven statistical and perfect settings, and for the security-with-identifiable-abort model.

Theorem 9.1. *Let* $t, h^*, m \in \mathbb{N}$ *be such that* $t + h^* < m$ *and let* $f_1, \ldots, f_k, g$ *be* m*-party functionalities. Let* $\Pi^{f_1,\ldots,f_k}$ *denote an* m*-party protocol in the* $(f_1, \ldots, f_k)$*-hybrid model, computing* g *with weak (resp., strong) computational* (t, h^*)*-FaF full-security. Let* $\pi_{f_1}, \ldots, \pi_{f_k}$ *be* m*-party protocols computing* $f_1, \ldots, f_k$ *respectively with weak (resp., strong) computational* (t, h^*)*-FaF full-security. Then, the protocol* $\Pi = \Pi_{\pi_{f_1},\ldots,\pi_{f_k}}$*, where each call to each* f_i *is replaced with an execution of* π_{f_i} *computes* g *with weak (resp., strong) computational* (t, h^*)*-FaF full-security.*

Proof Sketch. For simplicity, we assume there is a single idealized functionality f that is being called exactly once at a known round ℓ. The proof for the general case of any number of sub-functionalities follows by a a hybrid argument. We focus on the weak FaF-security setting, as the proof for the strong FaF-security follows similarly (see Remark 9.2 for more detail). Let $\pi = \pi_f$ be a protocol that (t, h^*)-FaF-securely computes f. Let $\mathcal{A}$ be a malicious adversary for protocol Π corrupting a set $\mathcal{I} \subset \mathcal{P}$ of size at most t, and let $\mathcal{A}_{\mathcal{H}}$ be a semi-honest adversary corrupting a set $\mathcal{H} \subset (\mathcal{P} \setminus \mathcal{I})$ of size at most h^*. Similarly to [14] we construct the corresponding adversaries $\mathcal{A}^f$ and $\mathcal{A}^f_{\mathcal{H}}$ that work in the hybrid model (engaging in protocol Π^f) and show how they can simulate $\mathcal{A}$ and $\mathcal{A}_{\mathcal{H}}$'s views in Π^f. The proof would then follow from the weak (resp., strong) FaF-security of Π^f in the hybrid-world.

We next present the construction for $\mathcal{A}^f$ and $\mathcal{A}^f_{\mathcal{H}}$. Towards constructing the hybrid model adversaries, we first construct the stand-alone model adversaries for the protocol π (again, similarly to [14]). Specifically, we construct a malicious adversary $\mathcal{A}_\pi$ controlling $\mathcal{I}$ and a semi-honest adversary $\mathcal{A}_{\pi,\mathcal{H}}$ controlling $\mathcal{H}$, both interacting in π. In the following, when we say global-view, we refer to the joint view of the adversaries $\mathcal{A}$ and $\mathcal{A}_{\mathcal{H}}$, and the remaining parties in $\mathcal{P} \setminus (\mathcal{I} \cup \mathcal{H})$.

Construction of $\mathcal{A}_\pi$: The adversary $\mathcal{A}_\pi$ controls the parties in $\mathcal{I}$. It is given an auxiliary input $z_{\mathcal{A},\pi}$ and does the following. First, it verifies that $z_{\mathcal{A},\pi}$ is a consistent view for the real-world adversary $\mathcal{A}$ until round ℓ (just before the execution of the inner protocol π). That is, this view is consistent with the behavior of $\mathcal{A}$, given the messages it sees from other parties (which are also part of the given view). If this verification fails, then $\mathcal{A}_\pi$ halts and outputs $\perp$.

Otherwise, if the verification goes through, the stand-alone protocol adversary $\mathcal{A}_\pi$ internally runs the real-world adversary $\mathcal{A}$ with auxiliary information $z_{\mathcal{A},\pi}$ and with messages it receives from parties in the stand-alone execution. It then reacts to these messages in the same way the real-world $\mathcal{A}$ reacts to them. Once the execution of the stand-alone terminates, $\mathcal{A}_\pi$ outputs the simulated view of $\mathcal{A}$ so far (i.e., $z_{\mathcal{A},\pi}$ concatenated with the simulated view for the execution in π) and halts.

Construction of $\mathcal{A}_{\pi,\mathcal{H}}$: The adversary $\mathcal{A}_{\pi,\mathcal{H}}$ controls the parties in $\mathcal{H}$, and is given an auxiliary input $z_{\mathcal{H},\pi}$. It is a semi-honest adversary, and thus, it must follow the instructions of the protocol π. Similarly, to $\mathcal{A}_\pi$, once the execution of the stand-alone terminates, $\mathcal{A}_{\pi,\mathcal{H}}$ outputs the simulated view of the real-world semi-honest adversary $\mathcal{A}_{\mathcal{H}}$ so far (including $z_{\mathcal{H},\pi}$) and halts.

By the assumed security of π, there exist two simulators $\mathsf{S}_{\mathcal{A}_\pi}$ and $\mathsf{S}_{\mathcal{A}_\pi,\mathcal{H}}$ that simulate $\mathcal{A}_\pi$ and $\mathcal{A}_{\pi,\mathcal{H}}$'s view respectively. Moreover, $\mathsf{S}_{\mathcal{A}_\pi,\mathcal{H}}$ simulates successfully for every auxiliary information $z_{\mathcal{H},\pi}$ held by the parties in $\mathcal{H}$. In particular, this holds for $z_{\mathcal{H},\pi}$ describing the view of the $\mathcal{A}_{\mathcal{H}}$ that is consistent with $z_{\mathcal{A},\pi}$.

Construction of $\mathcal{A}^f$: The adversary $\mathcal{A}^f$ controls the parties in $\mathcal{I}$. It (internally) invokes the real-world adversary $\mathcal{A}$ and behaves exactly the same until round ℓ (just before the execution of the inner protocol π). At this point $\mathcal{A}$ expects to interact in π. To provide $\mathcal{A}$ with messages from the uncorrupted parties, $\mathcal{A}^f$ will invoke $\mathsf{S}_{\mathcal{A}_\pi}$, with an auxiliary input $z_{\mathcal{A},\pi}$ that equals to the current view of $\mathcal{A}$ (formally, it also sets the inputs of the parties to 0). When $\mathsf{S}_{\mathcal{A}_\pi}$ sends inputs to its trusted party, the adversary $\mathcal{A}^f$ forwards the same inputs to the functionality f, and sends back to $\mathsf{S}_{\mathcal{A}_\pi}$ the output it got from f.

Recall that the output of $\mathsf{S}_{\mathcal{A}_\pi}$ is a view v for the real-world adversary $\mathcal{A}$ that agrees with its view until round ℓ (specifically, the randomness of $\mathcal{A}$ according to v is the same as its randomness used for running $\mathcal{A}$ internally). Thus, $\mathcal{A}^f$ can complete the interaction by (internally) invoking $\mathcal{A}$ on the view v and behaving exactly the same.

Construction of $\mathcal{A}^f_{\mathcal{H}}$: The adversary $\mathcal{A}^f_{\mathcal{H}}$ will be constructed similarly to $\mathcal{A}^f$. The main difference is the inputs it gives to the simulator $\mathsf{S}_{\mathcal{A}_\pi,\mathcal{H}}$ of the stand-alone semi-honest adversary. Since this is a semi-honest adversary, it cannot change its inputs. Note, however, that the inputs are determined by its view, and hence can be computed deterministically.

To conclude the proof of Theorem 9.1, we next show that the hybrid-world adversaries $\mathcal{A}^f$ and $\mathcal{A}^f_{\mathcal{H}}$ simulate the real-world adversaries $\mathcal{A}$ and $\mathcal{A}_{\mathcal{H}}$ respectively, namely, we show that

$$\text{REAL}^{\Pi,\mathcal{A},\mathcal{A}_{\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}}(\mathcal{A}) \overset{c}{\equiv} \text{REAL}^{\Pi^f,\mathcal{A}^f,\mathcal{A}^f_{\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}}\left(\mathcal{A}^f\right), \tag{7}$$

and that

$$\text{REAL}^{\Pi,\mathcal{A},\mathcal{A}_{\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}}(\mathcal{A}_{\mathcal{H}}) \overset{c}{\equiv} \text{REAL}^{\Pi^f,\mathcal{A}^f,\mathcal{A}^f_{\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}}\left(\mathcal{A}^f_{\mathcal{H}}\right). \tag{8}$$

As [14] proved Equation (7) we are left with the task of proving Equation (8). Assume towards contradiction that Equation (8) does not hold, that is, assume that there exists a non-uniform PPTM D distinguishing the left-hand side from the right-hand side with noticeable probability. We show that this imply that π is not (t,h^*)-FaF-secure, specifically, against the adversaries $\mathcal{A}_\pi$ and $\mathcal{A}_{\pi,\mathcal{H}}$ constructed above. We next construct a distinguisher D_π that distinguishes $\text{REAL}^{\pi,\mathcal{A}_\pi,\mathcal{A}_{\pi,\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A},\pi},z_{\mathcal{H},\pi}}(\mathcal{A}_{\pi,\mathcal{H}})$ from $\text{IDEAL}^{f,\mathsf{S}_{\mathcal{A}_\pi},\mathsf{S}_{\mathcal{A}_\pi,\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A},\pi},z_{\mathcal{H},\pi}}(\mathsf{S}_{\mathcal{A}_\pi,\mathcal{H}})$. Here, $z_{\mathcal{A},\pi}$ and $z_{\mathcal{H},\pi}$ are consistent view of $\mathcal{A}$ and $\mathcal{A}_{\mathcal{H}}$, respectively, at the start of round ℓ in the execution of Π. The auxiliary information z of D_π, will be the inputs and random inputs for the parties interacting in Π, that are consistent with $z_{\mathcal{A},\pi}$ and $z_{\mathcal{H},\pi}$. D_π will run Π using z, and will generate the global view v of all parties. It will then hand v over to D and output the same.

Observe that until round ℓ, the malicious adversaries $\mathcal{A}$ and $\mathcal{A}^f$ behave the same, implying that the global view in Π and Π^f are distributed the same. Since z is consistent with $z_{\mathcal{A},\pi}$ and $z_{\mathcal{H},\pi}$, it follows that v is determined solely by the input of D_π. Specifically, if the input of D_π is distributed according to the stand-alone real-world distribution $\text{REAL}^{\pi,\mathcal{A}_\pi,\mathcal{A}_{\pi,\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A},\pi},z_{\mathcal{H},\pi}}(\mathcal{A}_{\pi,\mathcal{H}})$, then v is a global view of the real-world $\text{REAL}^{\Pi,\mathcal{A},\mathcal{A}_{\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}}(\mathcal{A}_{\mathcal{H}})$, and if the input of D_π is distributed according to the stand-alone ideal-world distribution $\text{IDEAL}^{f,\mathsf{S}_{\mathcal{A}_\pi},\mathsf{S}_{\mathcal{A}_\pi,\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A},\pi},z_{\mathcal{H},\pi}}(\mathsf{S}_{\mathcal{A}_\pi,\mathcal{H}})$, then v is a global view of the ideal-world $\text{REAL}^{\Pi^f,\mathcal{A}^f,\mathcal{A}^f_{\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}}\left(\mathcal{A}^f_{\mathcal{H}}\right)$. Thus, D_π and D distinguish with the same probability.

□

Remark 9.2 (Proving the composition theorem for strong FaF-security)**.** *The proof above can be easily adapted to the setting of strong FaF-security, by simply replacing the task of proving Equations* (7) *and* (8) *with the task of proving*

$$\text{REAL}^{\Pi,\mathcal{A},\mathcal{A}_{\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}} \overset{c}{\equiv} \text{REAL}^{\Pi^f,\mathcal{A}^f,\mathcal{A}^f_{\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}}.$$

The reduction to the security of π remains the same and is therefore omitted.

Acknowledgements

We are grateful to Amos Beimel for many helpful discussions.

References

[1] B. Alon, E. Omri, and A. Paskin-Cherniavsky. MPC with friends and foes. In *Advances in Cryptology – CRYPTO 2020*, 2020. To appear.

[2] J. Alwen, A. Shelat, and I. Visconti. Collusion-free protocols in the mediated model. In *Annual International Cryptology Conference*, pages 497–514. Springer, 2008.

[3] J. Alwen, J. Katz, U. Maurer, and V. Zikas. Collusion-preserving computation. In *Advances in Cryptology–CRYPTO 2012*, pages 124–143. Springer, 2012.

[4] G. Asharov, A. Beimel, N. Makriyannis, and E. Omri. Complete characterization of fairness in secure two-party computation of boolean functions. In *TCC 2015*, pages 199–228, 2015.

[5] D. Beaver. Foundations of secure interactive computing. In *Annual International Cryptology Conference*, pages 377–391. Springer, 1991.

[6] D. Beaver. Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority. *Journal of Cryptology*, 4(2):75–122, 1991.

[7] D. Beaver. Minimal-latency secure function evaluation. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 335–350. Springer, 2000.

[8] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 503–513. ACM, 1990.

[9] Z. Beerliová-Trubíniová, M. Fitzi, M. Hirt, U. Maurer, and V. Zikas. Mpc vs. sfe: Perfect security in a unified corruption model. In *Theory of Cryptography Conference*, pages 231–250. Springer, 2008.

[10] A. Beimel, T. Malkin, and S. Micali. The all-or-nothing nature of two-party secure computation. In *Annual International Cryptology Conference*, pages 80–97. Springer, 1999.

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

[12] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1–10, 1988.

[13] F. Benhamouda, H. Lin, A. Polychroniadou, and M. Venkitasubramaniam. Two-round adaptively secure multiparty computation from standard assumptions. In *TCC 2018*, pages 175–205, 2018.

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

[15] R. Canetti and M. Vald. Universally composable security with local adversaries. In *International Conference on Security and Cryptography for Networks*, pages 281–301. Springer, 2012.

[16] R. Canetti, O. Poburinnaya, and M. Venkitasubramaniam. Equivocating yao: constant-round adaptively secure multiparty computation in the plain model. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 497–509. ACM, 2017.

[17] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *26th Annual Symposium on Foundations of Computer Science*, pages 383–395, 1985.

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

[19] R. Cramer, I. Damgård, and Y. Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In *Theory of Cryptography Conference*, pages 342–362. Springer, 2005.

[20] I. Damgård and Y. Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *Annual International Cryptology Conference*, pages 378–394. Springer, 2005.

[21] V. Daza and N. Makriyannis. Designing fully secure protocols for secure two-party computation of constant-domain functions. In *Theory of Cryptography Conference*, pages 581–611. Springer, 2017.

[22] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM review*, 45(4):727–784, 2003.

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

[24] M. Fitzi, M. Hirt, and U. M. Maurer. General adversaries in unconditional multi-party computation. In *ASIACRYPT '99*, pages 232–246, 1999.

[25] S. Garg and A. Sahai. Adaptively secure multi-party computation with dishonest majority. In *Annual Cryptology Conference*, pages 105–123. Springer, 2012.

[26] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. The round complexity of verifiable secret sharing and secure multicast. In *STOC 2001*, pages 580–589, 2001.

[27] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. On 2-round secure multiparty computation. In *Annual International Cryptology Conference*, pages 178–193. Springer, 2002.

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

[29] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *stoc19*, pages 218–229, 1987.

[30] S. D. Gordon and J. Katz. Complete fairness in multi-party computation without an honest majority. In *Theory of Cryptography Conference*, pages 19–35. Springer, 2009.

[31] 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.

[32] S. Halevi, Y. Lindell, and B. Pinkas. Secure computation on the web: Computing without simultaneous interaction. In *Annual Cryptology Conference*, pages 132–150. Springer, 2011.

[33] S. Halevi, Y. Ishai, E. Kushilevitz, and T. Rabin. Best possible information-theoretic mpc. In *TCC 2018*, pages 255–281, 2018.

[34] M. Hirt, U. Maurer, and V. Zikas. Mpc vs. sfe: Unconditional and computational security. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 1–18. Springer, 2008.

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

[36] 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.

[37] M. Ito, A. Saito, and T. Nishizeki. Secret sharing scheme realizing general access structure. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 72(9): 56–64, 1989.

[38] J. Katz and Y. Lindell. Collusion-free multiparty computation in the mediated model. *IACR Cryptology ePrint Archive*, 2008:533, 2008.

[39] C.-Y. Koo. Secure computation with partial message loss. In *Theory of Cryptography Conference*, pages 502–521. Springer, 2006.

[40] E. Kushilevitz, Y. Lindell, and T. Rabin. Information-theoretically secure protocols and security under composition. *SIAM Journal on Computing*, 39(5):2090–2112, 2010.

[41] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[42] N. Makriyannis. On the classification of finite boolean functions up to fairness. In *International Conference on Security and Cryptography for Networks*, pages 135–154. Springer, 2014.

[43] S. Micali and P. Rogaway. Secure computation. In *Annual International Cryptology Conference*, pages 392–404. Springer, 1991.

[44] M. O. Rabin. How to exchange secrets with oblivious transfer, 2005. URL `http://eprint.iacr.org/2005/187`. Harvard University Technical Report 81 talr@watson.ibm.com 12955 received 21 Jun 2005.

[45] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *STOC 1989*, pages 73–85, 1989.

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

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

[48] V. Zikas. *Generalized corruption models in secure multi-party computation*. PhD thesis, ETH Zurich, 2010. URL http://d-nb.info/1005005729.

[49] V. Zikas, S. Hauser, and U. Maurer. Realistic failures in secure multi-party computation. In *Theory of Cryptography Conference*, pages 274–293. Springer, 2009.

A Missing Proofs

Proof of Lemma 4.10. Assume towards contradiction, that for any m-party functionality g there exists a protocol Π_g that computes it with computational (t, h^*)-FaF full-security. Let $f(x_1, x_2, x_3) = (f_i(x_1, x_2, x_3))_{i=1}^3$ be the 3-party functionality guaranteed to exist by Lemma 4.11. Recall that the "moreover" part of Lemma 4.11 states that for any protocol computing f, there is an attacker corrupting one of two parties, while the remaining third party is semi-honest. Partition the set of parties as follows. Let $\mathrm{A} = \{\mathrm{P}_1, \ldots, \mathrm{P}_t\}$, let $\mathrm{B} = \{\mathrm{P}_{t+1}, \ldots, \mathrm{P}_{t+h^*}\}$, and let $\mathrm{C} = \{\mathrm{P}_{t+h^*+1}, \ldots, \mathrm{P}_m\}$. Observe that $|\mathrm{A}| = |\mathrm{C}| = t$ and $|\mathrm{B}| = h^*$. Consider the m-party functionality $g(x_1, \ldots, x_m)$ that outputs $f_1(x_1, x_{t+1}, x_{t+h^*+1})$ to P_1, outputs $f_2(x_1, x_{t+1}, x_{t+h^*+1})$ to P_{t+1}, outputs $f_3(x_1, x_{t+1}, x_{t+h^*+1})$ to P_{t+h^*+1}, and outputs $\perp$ to all other parties.

Next, consider the following 3-party protocol Π_f for computing f. The parties emulate the execution of Π_g, where P_1 emulates the parties in A, P_2 emulates the parties in B, and P_3 emulates the parties in C. The parties then output whatever output was computed in Π_g.

The main observation is that any adversary $\mathcal{A}_f$ corrupting either P_1 or P_3 in Π_f, corresponds to an adversary $\mathcal{A}_g$ corrupting either the set A or C respectively in Π_g. That is, the view and actions of $\mathcal{A}_f$ can be simulated in Π_g. Similarly, any set $\mathcal{H}_f$ of a single semi-honest party in Π_f has the same view as the set $\mathcal{H}_g$ emulated by that party in Π_g. Furthermore, the simulators for $\mathcal{A}_g$ and $\mathcal{H}_g$ directly translate to simulators for $\mathcal{A}_f$ and $\mathcal{H}_f$. This contradicts the fact that f cannot be computed with computational $(1, 1)$-FaF full-security, against malicious adversaries corrupting either P_1 or P_3. $\square$

Proof of Claim 5.6. We first prove the security properties of the protocol. For all of the adversaries, the corresponding simulator we construct will not change the inputs they send to T, unless the adversary changes its input. As a result, the 2-security of the protocol implies that it is also $(1, 1)$-mixed fully secure.

Any adversary corrupting 2 parties from $\{\mathrm{P}_3, \mathrm{P}_4, \mathrm{P}_5\}$, or corrupting P_2 and one party from $\{\mathrm{P}_3, \mathrm{P}_4, \mathrm{P}_5\}$, can be simulated using the composition theorem of Canetti [14] for both Steps 1 and 3. That fact that there is an honest majority implies the existence of the simulator.

Let $\mathcal{A}$ be an adversary corrupting P_1 and P_2. The simulator $\mathsf{S}_{\mathcal{A}}$ works as follows. Compute a pair of keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathrm{Gen}(1^n)$ and send pk to $\mathcal{A}$. Let c_1 and c_2 be the inputs of P_1 and P_2 respectively, that $\mathcal{A}$ chose for the computation of g. If $c_1 = \mathrm{Enc}_{\mathsf{pk}}(x_1, 1)$, $c_2 = \mathrm{Enc}_{\mathsf{pk}}(x_2, 2)$, and $x_1 = x_2$, then send (x_1, x_2) to the trusted party T to receive two shares of x_3, hand them over to $\mathcal{A}$, output the view and halt. Else, if $c_1 = c_2 = \mathrm{Enc}_{\mathsf{pk}}(x_2, 2)$, then send (x_2, x_2) to the trusted party T to receive two shares of x_3. Sample a random string r, hand and the two shares to $\mathcal{A}$, output the view and halt. Otherwise, give to $\mathcal{A}$ two random and independent strings $r_1, r_2 \in \{0,1\}^n$, output the view and halt.

Next, consider an adversary $\mathcal{A}'$ corrupting P_1 and a party from $\{\mathrm{P}_3, \mathrm{P}_4, \mathrm{P}_5\}$. We construct its simulator $\mathsf{S}_{\mathcal{A}'}$ as follows. Compute a pair of keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathrm{Gen}(1^n)$, and use [14] to simulate Step 1. Compute an encryption $c \leftarrow \mathrm{Enc}_{\mathsf{pk}}(0^n, 2)$ and send both pk and c to $\mathcal{A}'$. Let c' be the ciphertext chosen by $\mathcal{A}$ as the input for P_1 in Step 3. Decrypt c' to recover an input x_1', which the simulator then sends to T. Finally, simulate Step 3 using [14] with input c' for P_1. Since the encryption scheme is semantically secure and $\mathcal{A}$ does not hold sk, it follows that the value of c' in the real-world is computationally indistinguishable from its ideal-world value. Moreover, as the encryption scheme is non-malleable, if $x_1 \neq x_2$ then $c' = \mathrm{Enc}_{\mathsf{pk}}(x_2, 1)$ holds with negligible probability. Therefore, in this case the output of P_1 and P_2 will be random strings., as given by

T. In the other case, where $x_1 = x_2$, the output given by T to P_1 and P_2 are sharing of x_3. In addition, the output of g will be a sharing of x_3 as well. Security follows.

We next show that Protocol 5.5 is not $(1,1)$-FaF fully-secure. This follows from the observation that a corrupt P_1 can use c_2 as its input for the computation of g. Thus, party P_2 receives two shares of x_3 in a 2-out-of-2 secret sharing, thereby learning x_3. In the ideal-world, any simulator for P_2 will receive from T a share of x_3 only if $x_1 = x_2$. Since for the case where $x_1 \neq x_2$ this happens only with negligible probability (with the probability being over the choices of the simulator $\mathsf{S}_{\mathcal{A}}$ for the *adversary's input*), we conclude that no such simulator exists. $\square$

Proof of Claim 5.7. We first show that the protocol computes f with weak computational $(1,1)$-FaF full-security. We go over all 6 possible (malicious and semi-honest) corruptions and construct two simulators for each case.

First, suppose $\mathcal{A}$ corrupts P_1. Its simulator $\mathsf{S}_{\mathcal{A}}$ will output a commitment to 0 and halt. By the hiding property of the commitment scheme, the simulator's output will be indistinguishable from the adversary's real-world view. The semi-honest simulator $\mathsf{S}_{\mathcal{A},\mathrm{P}_2}$ will query the adversary on the commitment of its input b to receive the messages it sends to P_2, output them and halt. The other semi-honest simulator $\mathsf{S}_{\mathcal{A},\mathrm{P}_3}$ will receive b from the trusted party T, and do the same as $\mathsf{S}_{\mathcal{A},\mathrm{P}_2}$.

Next, suppose $\mathcal{A}$ corrupts P_2. Its simulator $\mathsf{S}_{\mathcal{A}}$ will ask $\mathcal{A}$ for a commitment and a decommitment, supposedly sent to P_3, and recover the committed value $\hat{b}$ (if the decommitment is invalid or the adversary did not send a message, then choose $\hat{b}$ to be a default value). It then sends $\hat{b}$ to T and halt. By the binding property of the commitment scheme, the adversary cannot decommit to a different value, hence the output distribution of P_3 in both worlds are indistinguishable. The semi-honest simulators $\mathsf{S}_{\mathcal{A},\mathrm{P}_1}$ and $\mathsf{S}_{\mathcal{A},\mathrm{P}_3}$, which hold $\mathsf{S}_{\mathcal{A}}$'s input, randomness, and auxiliary input, query $\mathcal{A}$ to receive the messages it sends to P_1 and P_3, respectively, output them, and halt.

Finally, suppose $\mathcal{A}$ corrupts P_3. Its simulator $\mathsf{S}_{\mathcal{A}}$ will output a commitment and the corresponding decommitment to the output b received from T and halt. The semi-honest simulator $\mathsf{S}_{\mathcal{A},\mathrm{P}_2}$, which holds b as well, will query $\mathcal{A}$ and output the messages received from it. The other simulator $\mathsf{S}_{\mathcal{A},\mathrm{P}_1}$ will do the same with the addition of outputting a commitment to b.

We now show that the protocol is not strong $(1,1)$-FaF fully-secure. Consider an adversary $\mathcal{A}$, which corrupts P_1 and *does nothing*, and let $\mathcal{H} = \{\mathrm{P}_3\}$. Assume towards contradiction that the pair of simulators $\mathsf{S}_{\mathcal{A}}$ and $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ exists so that $\left\{\mathrm{IDEAL}_b^{\mathsf{S}_{\mathcal{A}},\mathsf{S}_{\mathcal{A},\mathcal{H}}}\right\}_{b\in\{0,1\}}$ and $\left\{\mathrm{REAL}_b^{\mathcal{A},\mathcal{A}_{\mathcal{H}}}\right\}_{b\in\{0,1\}}$ are indistinguishable. We show how to construct a sender that breaks the binding property of the commitment scheme. The sender sends to the receiver the commitment c' generated by $\mathsf{S}_{\mathcal{A}}$. The decommitment will be the commitment c and its decommitment d, generated by $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ on input $b = 0$ (and $\mathsf{S}_{\mathcal{A}}$'s randomness). By assumed strong security of the protocol, the view $(c', (c, d))$ of the receiver is indistinguishable from $(c, (c, d))$, and therefore it will output 0. However, the same holds when applying $\mathsf{S}_{\mathcal{A},\mathcal{H}}$ on input $b = 1$ in the decommitment phase. Thus, the sender can force the output of the receiver in the decommitment phase to be any value it chooses with probability $1 - \mu(n)$, for some negligible function $\mu(\cdot)$. $\square$

Proof of Lemma 7.8. Since $\mathcal{R}$ is not $\mathsf{Q}(3,2)$, there exist $(\mathcal{I}_1,\mathcal{H}_1), (\mathcal{I}_2,\mathcal{H}_2), (\mathcal{I}_3,\mathcal{H}_3) \in \mathcal{R}$ such that

$$\mathcal{I}_1 \cup \mathcal{I}_2 \cup \mathcal{I}_3 \cup \mathcal{H}_1 \cup \mathcal{H}_2 = \mathcal{P}.$$

Similarly to the proof of Lemma 7.6, we may assume without loss of generality that all of the sets $\mathcal{I}_1 \cup \mathcal{H}_1$, $\mathcal{I}_2 \cup \mathcal{H}_2$, and $\mathcal{I}_3 \cup \mathcal{H}_3$ are not empty, and furthermore, they are pairwise disjoint.

Now, assume towards contradiction, that for any m-party functionality g there exists a protocol Π_g that computes it with perfect $\mathcal{R}$-FaF full-security (possibly using broadcast). Consider the 3-party functionality $f(x_1, x_2, \bot) = (x_1 \wedge x_2, x_1 \wedge x_2, \bot)$, which was shown by [24] to be impossible to compute against the adversarial structure $\mathcal{R}_3 = \{(\emptyset, \{\mathrm{P}_1\}),\ (\emptyset, \{\mathrm{P}_2\}),\ (\{\mathrm{P}_3\}, \emptyset)\}$. Fix $i_1 \in \mathcal{I}_1 \cup \mathcal{H}_1$ and $i_2 \in \mathcal{I}_2 \cup \mathcal{H}_2$, and consider the m-party symmetric functionality $g(x_1, \ldots, x_m)$ that outputs $x_{i_1} \wedge x_{i_2}$ to P_{i_1} and P_{i_2}, and outputs $\bot$ the all other parties.

Next, consider the following 3-party protocol Π for computing f. The parties emulate the execution of Π_g, where for each $i \in \{1, 2\}$, party P_i emulates the parties in $\mathcal{I}_i \cup \mathcal{H}_i$, and P_3 emulates the parties in $\mathcal{I}_3$. In the emulation, the inputs x_1, x_2 are given to P_{i_1} and P_{i_2} respectively, while other parties start with arbitrary inputs. The parties then output whatever output was computed in Π_g.

The main observation is that any $(\mathcal{I}, \mathcal{H}) \in \mathcal{R}_3$, where $\mathcal{I}$ is corrupted by an adversary $\mathcal{A}$, corresponds to the case where $(\mathcal{I}, \mathcal{H}) \in \mathcal{R}$ and where $\mathcal{I}$ is corrupted by some adversary $\mathcal{A}'$ in the emulated protocol Π_g. Indeed, a single semi-honest adversary $\mathcal{A}$ corrupting $(\emptyset, \{\mathrm{P}_i\})$, for $i \in \{1, 2\}$, corresponds to a semi-honest adversary $\mathcal{A}'$ corrupting $(\emptyset, \mathcal{I}_i \cup \mathcal{H}_i) \in \mathcal{R}$. Additionally, a single malicious adversary $\mathcal{A}$ corrupting $(\{\mathrm{P}_3\}, \emptyset)$, corresponds to a malicious adversary $\mathcal{A}'$ corrupting $(\mathcal{I}_3, \emptyset) \in \mathcal{R}$. Furthermore, the simulators for $\mathcal{A}'$ directly translate to simulators for $\mathcal{A}$ contradicting the fact that f cannot be securely computed with perfect $\mathcal{R}_3$-security. $\square$

Proof of Lemma 7.7. By Lemma 7.6 we may assume that $\mathcal{R}$ is $\mathsf{Q}(2, 2)$. Since $\mathcal{R}$ is not $\mathsf{Q}(3, 0)$, there exist $(\mathcal{I}_1, \mathcal{H}_1), (\mathcal{I}_2, \mathcal{H}_2), (\mathcal{I}_3, \mathcal{H}_3) \in \mathcal{R}$ such that

$$\mathcal{I}_1 \cup \mathcal{I}_2 \cup \mathcal{I}_3 = \mathcal{P}.$$

Similarly to the proof of Lemma 7.6, we may assume without loss of generality that all of the sets $\mathcal{I}_1$, $\mathcal{I}_2$, and $\mathcal{I}_3$ are not empty, and furthermore, they are pairwise disjoint.

Now, assume towards contradiction, that for any m-party functionality g there exists a protocol Π_g that computes it with statistical $\mathcal{R}$-FaF full-security, without using broadcast. Consider the 3-party broadcast functionality $f(x_1, \bot, \bot) = x_1$, which cannot be computed against a single malicious adversary corrupting any single party [41]. Fix $i_1 \in \mathcal{I}_1$ and consider the m-party broadcast functionality $g(x_1, \ldots, x_m) = x_{i_1}$.

Next, consider the following 3-party protocol Π_f for computing f. The parties emulate the execution of Π_g, where for each $i \in \{1, 2, 3\}$, party P_i emulates the parties in $\mathcal{I}_i$. In the emulation, the input x_{i_1} is given to P_{i_1} while the other parties have arbitrary input. The parties then output whatever output was computed in Π_g.

The main observation is that any malicious adversary $\mathcal{A}_f$ corrupting P_i in Π_f, corresponds to a malicious adversary $\mathcal{A}_g$ corrupting $\mathcal{I}_i$ in the emulated protocol Π_g. Furthermore, the simulators for $\mathcal{A}_g$ directly translate to simulators for $\mathcal{A}_f$ contradicting the fact that f cannot be securely computed with statistical 1-security without broadcast. $\square$

B Additional Ideal Model Definitions

B.1 Definition of FaF Ideal Model with Fairness

The FaF ideal model – fairness.

Inputs: Each party P_i holds 1^n and $x_i \in \mathcal{X}_i^n$. The adversaries $\mathcal{A}$ and $\mathcal{A}_{\mathcal{H}}$ are given each an auxiliary input $z_{\mathcal{A}}, z_{\mathcal{H}} \in \{0,1\}^*$ respectively, and x_i for every P_i controlled by them. The trusted party T holds 1^n.

Parties send inputs: Each uncorrupted party $\mathrm{P}_j \in \mathcal{P} \setminus \mathcal{I}$ sends x_j as its input to T. The malicious adversary sends a value $x_i' \in \mathcal{X}_i^n$ as the input for party $\mathrm{P}_i \in \mathcal{I}$. Write $(x_1', \ldots, x_m')$ for the tuple of inputs received by the trusted party.

The trusted party performs computation: The trusted party T selects a random string r and computes $\mathbf{y} = (y_1, \ldots, y_m) = f(x_1' \ldots, x_m'; r)$.

Malicious adversary instructs trusted party to continue or halt: the adversary $\mathcal{A}$ sends either `continue` or $\perp$ to T. If it sent `continue`, then it sends $\mathbf{y}_{\mathcal{I}}$ to $\mathcal{A}$, and for every honest party P_j the trusted party sends it y_j. Otherwise, if $\mathcal{A}$ sent $\perp$, then T sends $\perp$ to all parties and $\mathcal{A}$.

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

Outputs: Each uncorrupted party outputs whatever output it received from T, the parties in $\mathcal{I}$ output nothing. The adversaries output some function of their respective view.

B.2 Definition of (Standard) Adaptive Security

The Adaptive Real Model

The real-world t-adaptive adversary is a non-uniform adversary $\mathcal{A}$ that starts with some randomness. During each round of the protocol, it may choose to corrupt parties, based on its current view, so long as the total number of corrupted parties is at most t. Once a party is corrupted, the party's view (including messages received during the protocol) becomes known to $\mathcal{A}$. Additionally, after the protocol's execution has terminated, the adversary interacts with an *environment* $\mathcal{Z}$. The environment $\mathcal{Z}$ is a non-uniform probabilistic machine, which has the global view of the protocol as an input. $\mathcal{Z}$ and $\mathcal{A}$ interact in rounds, where in each round, $\mathcal{Z}$ request to corrupt some honest party, and $\mathcal{A}$ answers with some arbitrary information. The interaction continues until $\mathcal{Z}$ halts, or t parties have been corrupted. The environment $\mathcal{Z}$ then output some arbitrary value.

We next define the real-world global view for security parameter $n \in \mathbb{N}$, an input sequence $\mathbf{x} = (x_1, \ldots, x_m)$, and auxiliary inputs $z_{\mathcal{A}}, z_{\mathcal{Z}} \in \{0,1\}^*$ with respect to adversary $\mathcal{A}$ and environment $\mathcal{Z}$, respectively. Let $\textsc{out}^{\mathrm{REAL}}_{\mathcal{A},\Pi}(1^n, \mathbf{x})$ denote the outputs of the honest parties in a random execution of Π, while interacting with $\mathcal{A}$. Further let $\textsc{view}^{\mathrm{REAL}}_{\mathcal{A},\Pi}(1^n, \mathbf{x})$ be the adversary's view during an execution of Π, which contains its auxiliary input, its random coins, and the view of the parties it corrupted during the execution of the protocol. In addition, we let $\textsc{view}^{\mathrm{REAL}}_{\mathcal{A},\mathcal{Z},\Pi}(1^n, \mathbf{x})$ be the view of $\mathcal{Z}$ after the interaction with $\mathcal{A}$ ended (this view consists of the view of all the parties it requested to corrupt).

We denote the global view in the adaptive real model by

$$\textsc{real}^{\Pi,\mathcal{A},\mathcal{Z}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{Z}}} = \left(\textsc{view}^{\mathrm{REAL}}_{\mathcal{A},\Pi}(1^n, \mathbf{x}),\ \textsc{view}^{\mathrm{REAL}}_{\mathcal{A},\mathcal{Z},\Pi}(1^n, \mathbf{x}),\ \textsc{out}^{\mathrm{REAL}}_{\mathcal{A},\Pi}(1^n, \mathbf{x})\right).$$

The Adaptive Ideal Model

We next describe the interaction in the *adaptive ideal model*, which specifies the requirements for fully secure computation of the function f with security parameter n, against adaptive adversaries. Let $\mathcal{A}$ be an adversary in the ideal-world, which is given an auxiliary input z and some randomness.

The adaptive ideal model – full-security.

Inputs: Each party P_i holds 1^n and $x_i \in \mathcal{X}_i^n$. The adversary $\mathcal{A}$ is given an auxiliary input $z_{\mathcal{A}} \in \{0,1\}^*$ and some randomness. The trusted party T holds 1^n.

First corruption phase: $\mathcal{A}$ chooses adaptively to corrupt a set $\mathcal{I}_1 \subseteq \mathcal{P}$ of parties. The corruption is done in several iterations. Once a party is corrupted, its input becomes known to $\mathcal{A}$.

Parties send inputs: Each honest party $\mathrm{P}_j \in \mathcal{P} \setminus \mathcal{I}_1$ sends x_j as its input to T. The adversary $\mathcal{A}$ sends a value $x'_i \in \mathcal{X}_i^n$ as the input for party $\mathrm{P}_i \in \mathcal{I}_1$. Write $(x'_1, \ldots, x'_m)$ for the tuple of inputs received by the trusted party.

The trusted party performs computation: The trusted party T selects a random string r and computes $\mathbf{y} = (y_1, \ldots, y_m) = f\left(x'_1 \ldots, x'_m; r\right)$ and sends y_i to each party P_i.

Second corruption phase: $\mathcal{A}$ chooses adaptively to corrupt another set $\mathcal{I}_2 \subseteq \mathcal{P} \setminus \mathcal{I}_1$ of parties. Similarly to the first corruption phase, this is done in several iterations.

Outputs: Each honest party outputs whatever output it received from T, the parties in $\mathcal{I}$ output nothing. $\mathcal{A}$ output some function of their respective view.

Post-protocol corruption: The (non-uniform) environment $\mathcal{Z}$ and the adversary $\mathcal{A}$ interacts in rounds, where in each round, $\mathcal{Z}$ request to corrupt some honest party, and $\mathcal{A}$ answers with some arbitrary response. The interaction continues until $\mathcal{Z}$ halts with an arbitrary output.

We next define the ideal-world global view for security parameter $n \in \mathbb{N}$, an input sequence $\mathbf{x} = (x_1, \ldots, x_m)$, and auxiliary inputs $z_{\mathcal{A}}, z_{\mathcal{Z}} \in \{0,1\}^*$ with respect to adversary $\mathcal{A}$ and environment $\mathcal{Z}$, respectively. Let $\textsc{out}^{\text{IDEAL}}_{\mathcal{A},f}(1^n, \mathbf{x})$ denote the outputs of the honest parties in a random execution of the above ideal-world process, while interacting with $\mathcal{A}$. Further let $\textsc{view}^{\text{IDEAL}}_{\mathcal{A},f}(1^n, \mathbf{x})$ be the (simulated, real-world) view description being the *output* of $\mathcal{A}$ in such a process. In addition, we let $\textsc{view}^{\text{IDEAL}}_{\mathcal{A},\mathcal{Z},f}(1^n, \mathbf{x})$ be the view of $\mathcal{Z}$ after the interaction with $\mathcal{A}$ ended. We denote the global view in the adaptive ideal model by

$$\textsc{ideal}^{f,\mathcal{A},\mathcal{Z}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{Z}}} = \left(\textsc{view}^{\text{IDEAL}}_{\mathcal{A},f}(1^n, \mathbf{x}),\ \textsc{view}^{\text{IDEAL}}_{\mathcal{A},\mathcal{Z},f}(1^n, \mathbf{x}),\ \textsc{out}^{\text{IDEAL}}_{\mathcal{A},f}(1^n, \mathbf{x})\right).$$

We next give the definition for computational adaptive security.

Definition B.1 (adaptive security)**.** *Let* Π *be a protocol for computing* f*. We say that* Π *computes* f *with* computational adaptive t-security*, if the following holds. For every* t*-adaptive non-uniform PPTM adversary* $\mathcal{A}$ *in the real-world and for every non-uniform PPTM environment* $\mathcal{Z}$*, there exists an adaptive non-uniform PPTM adversary* $\mathsf{S}_{\mathcal{A}}$ *in the ideal-world such that*

$$\left\{\textsc{ideal}^{\mathsf{S}_{\mathcal{A}},\mathcal{Z}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{Z}}}\right\}_{\mathbf{x}\in\mathcal{X},z_{\mathcal{A}},z_{\mathcal{Z}}\in\{0,1\}^*,n\in\mathbb{N}} \overset{c}{\equiv} \left\{\textsc{real}^{\mathcal{A},\mathcal{Z}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{Z}}}\right\}_{\mathbf{x}\in\mathcal{X},z_{\mathcal{A}},z_{\mathcal{Z}}\in\{0,1\}^*,n\in\mathbb{N}}. \quad (9)$$

The statistical/perfect variants are obtained naturally from the above definition by replacing computational indistinguishability with statistical distance.