MPC with Friends and Foes[⋆]

Bar Alon, Eran Omri, and Anat Paskin-Cherniavsky

Department of Computer Science, Ariel University
alonbar08@gmail.com, omrier@ariel.ac.il, anatpc@ariel.ac.il

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.

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

[⋆] 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.

in a way that preserves some security properties, even in the face of adversarial behavior by some of the parties. Classical security definitions (cf., [25]) 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. [32], tolerating a single malicious party.

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

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 [11], 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 sharing scheme [39]. 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 [38]. 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 [38] 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 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 multi-party computation. Security is defined via the real vs. ideal paradigm. Here, the

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

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., $\left(\text{VIEW}_{\mathcal{H}}^{\text{REAL}}, \text{Out}^{\text{REAL}}\right)$ is indistinguishable from $\left(\text{VIEW}_{\mathsf{S}_{\mathcal{H}}}^{\text{IDEAL}}, \text{Out}^{\text{IDEAL}}\right)$.

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 [16] 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 [26] 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. Due to space limitations, the proof is given in the full-version [1].

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 (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, 24, 32]). 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. [32], 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.

[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.

Theorem 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 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. [24] 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 [18]. Using share conversion techniques ([17]) and the 2-round verifiable secret sharing (VSS) protocol of [23], they were able to construct a 3-round protocol that tolerates $t < m/5$ corruptions. We follow similar lines as [18]. 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 [18] 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 [23]. 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., [11, 38, 21] . 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. [21], 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 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. [21] 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 [21]. 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 full treatment of information theoretic FaF-security with respect to monotone adversarial structures is deferred to the full-version of the paper [1].

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 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 [21]. 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 2).

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 [37, 5, 6, 13, 25]. The question of achieving (standard) full-security was given quite some attention. See, e.g., [4, 16, 19, 28, 36] for two parties, [4, 11, 27] 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, 14, 34], 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. [21] 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. [31] 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 [35] 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. [41] 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 [40].

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.

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. Other preliminaries are standard and for space considerations are deferred to the full version [1].

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 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, inthe full-versionwe 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 $\mathit{\Pi}$ 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 $\mathit{\Pi}$ 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 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 $\textsc{out}^{\mathrm{REAL}}_{\mathcal{A},\mathit{\Pi}}(1^n, \mathbf{x})$ denote the outputs of the uncorrupted parties (those in $\mathcal{P} \setminus \mathcal{I}$) in a random execution of $\mathit{\Pi}$, with $\mathcal{A}$ corrupting the parties in $\mathcal{I}$. Further let $\textsc{view}^{\mathrm{REAL}}_{\mathcal{A},\mathit{\Pi}}(1^n, \mathbf{x})$ be the adversary's view during an execution of $\mathit{\Pi}$, 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 $\textsc{view}^{\text{REAL}}_{\mathcal{A},\mathcal{A}_{\mathcal{H}},\mathit{\Pi}}(1^n, \mathbf{x})$ be the view of $\mathcal{A}_{\mathcal{H}}$ during an execution of $\mathit{\Pi}$ 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

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

It will be convenient to denote

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

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

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

When $\mathit{\Pi}$ is clear from the context, we will remove it for brevity.

Remark 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 $\mathrm{P}_j \in \mathcal{P} \setminus \mathcal{I}$ 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 $\mathbf{y} = (y_1, \ldots, y_m) = f(x'_1 \ldots, x'_m; r)$ 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 $\mathcal{H} \subset \mathcal{P} \setminus \mathcal{I}$ respectively. Let $\textsc{out}^{\mathrm{IDEAL}}_{\mathcal{A},f}(1^n, \mathbf{x})$ 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 $\textsc{view}^{\mathrm{IDEAL}}_{\mathcal{A},f}(1^n, \mathbf{x})$ be the (simulated, real-world) view description being the *output* of $\mathcal{A}$ in such a process. In addition, we let $\textsc{view}^{\mathrm{IDEAL}}_{\mathcal{A},\mathcal{A}_{\mathcal{H}},f}(1^n, \mathbf{x})$ 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

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

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

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

and

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

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

Definition 1 (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 2 (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 $\mathcal{H} \subset \mathcal{P} \setminus \mathcal{I}$ 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*

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

and

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

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*

$$\begin{aligned} &\left\{ \mathrm{IDEAL}^{\mathsf{S}_{\mathcal{A}},\mathsf{S}_{\mathcal{A},\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}} \right\}_{\mathbf{x}\in\mathcal{X},z_{\mathcal{A}},z_{\mathcal{H}}\in\{0,1\}^*,n\in\mathbb{N}} \\ &\quad \stackrel{c}{\equiv} \left\{ \mathrm{REAL}^{\mathcal{A},\mathcal{A}_{\mathcal{H}}}_{1^n,\mathbf{x},z_{\mathcal{A}},z_{\mathcal{H}}} \right\}_{\mathbf{x}\in\mathcal{X},z_{\mathcal{A}},z_{\mathcal{H}}\in\{0,1\}^*,n\in\mathbb{N}}. \end{aligned} \tag{3}$$

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

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

Remark 2. 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. 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 $\mathbf{y} = (y_1, \ldots, y_m) = f(x'_1 \ldots, x'_m; r)$ 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. *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. 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 [26] 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* [30], 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 5. *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 [26], 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 3 (Residual Function [30, 29]). *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' secret sharing scheme, so that party P_i is required for the reconstruction (this can be done by first sharing in a 2-out-of-2 secret sharing, and then give one of the shares to P_i and share the other among $m' - 1$ parties). 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 1. *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 $\left(f'_{m-t}, \ldots, f'_m\right)$-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 $\left(f'_{m-t}, \ldots, f'_m\right)$-hybrid model.*

Remark 4. 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 1 with Theorem 5 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 1. *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. The proof of Lemma 1 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 2. *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 2, 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 6. *No protocol computes f with $(1,1)$-mixed full-security.*

The proof follows from a simple observation on a result by Ishai et al. [33]. They showed that for any protocol computing the functionality $g(a, \bot, c) = (a \oplus c, \bot, a \oplus c)$, 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 1 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 1

We next provide the proof of Lemma 1. 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' secret sharing scheme, so that party P_i is required for the reconstruction. 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 1). The protocol Π in the real world is described as follows:

Protocol 7
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.*[4]
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

[4] 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.

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.
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. Specifically, we prove the following lemma.

Lemma 2. *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 3. *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 2 is deferred to the full-version. We next give an overview of the proof of Lemma 3. 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 3). 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 $\mathsf{Swap} =$

$\left\{\mathsf{Swap}_n : \{0,1\}^n \times \{0,1\}^{2n} \times \{0,1\}^n \mapsto \{0,1\}^{2n}\right\}_{n\in\mathbb{N}}$ 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

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

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 8 *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 $q = \Pr\left[c_i = (a,c)\right]$.

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 8 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 8.

Proof (of Claim 8). 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 $\frac{1-\mathrm{neg}(n)}{2r+1}$.

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.

Remark 5. 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 the full-version.

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 2. 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 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 $\mathsf{3OR}$ 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 $\mathcal{H} = \{C\}$. In the real-world, $\mathcal{A}$ can reveal b to C. However, in the ideal-world, this cannot be simulated (when $c = 1$).

Remark 6. Example 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 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.

Theorem 9. *Let* $\mathtt{type} \in \{\text{computational, statistical, perfect}\}$ *and let* Π *be an* m*-party protocol computing some* m*-party functionality* f *with* $\mathtt{type}$ *adaptive* $(t + h^*)$*-security. Then* Π *computes* f *with* $\mathtt{type}$ (t, h^*)*-FaF full-security.*

A proof sketch of Theorem 9 is given in the full-version.

By applying recent results on adaptive security, we get that there exist constant-round protocol that are FaF secure-with-abort [15, 12, 22].

5.2 The Relation Between FaF-Security and Mixed-Security

The notion of "mixed adversaries" [21, 40] considers a single entity that corrupts a subset $\mathcal{I}$ maliciously, and another subset $\mathcal{H}$ semi-honestly.[5] A simulator for a

[5] There are various types of mixed adversaries one can consider. For example, [31] also gave the adversary the ability to fail-corrupt parties, based on its adversarial structure. Here, we only consider the notion considered by [21].

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 1 is not $(1, 1)$-mixed secure).

Example 2. 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 [20].

Protocol 10

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 [26].*
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 [26].
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 11 *Protocol 10 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.*

The proof is deferred to the full-version due space considerations.

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

The proof is deferred to the full-versiondue to space considerations. 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.

Acknowledgements

We are grateful to Amos Beimel for many helpful discussions.

Bibliography

[1] B. Alon, E. Omri, and A. Paskin-Cherniavsky. MPC with friends and foes. Cryptology ePrint Archive, Report 2020/701. https://eprint.iacr.org/2020/701.

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

[3] J. Alwen, J. Katz, U. Maurer, and V. Zikas. Collusion-preserving computation. In *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 *Crypto*, 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 *EUROCRYPT 2000*, pages 335–350. Springer, 2000.

[8] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *STOC 1990*, 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] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

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

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

[23] 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.

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

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

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

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

[28] 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.

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

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

[31] M. Hirt, U. Maurer, and V. Zikas. MPC vs. SFE: Unconditional and computational security. In *ASIACRYPT 2008*, pages 1–18. Springer, 2008.

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

[33] 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.

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

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

[36] N. Makriyannis. On the classification of finite boolean functions up to fairness. In *SCN 2014*, pages 135–154. Springer, 2014.

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

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

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

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

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