CS 758 - Project Report
MPC with Friends and Foes

Ankit Kumar Misra

May 13, 2022

1 Motivation

The goal of secure multi-party computation is essentially to enable n mutually distrusting parties to compute some common function of their inputs while preserving certain security properties of these inputs. Classical security definitions are often expressed in terms of the ideal versus real paradigm, wherein a real-world protocol is considered t-secure against a class $\mathcal{C}$ of adversaries if for every real-world adversary $\mathcal{A} \in \mathcal{C}$ corrupting at most t parties, there exists an ideal-world simulator, corrupting the same set of parties, that outputs a view for the real-world adversary that is distributed identically or closely to its view in a random execution of the real-world protocol.

However, with such definitions, there are no restrictions on the views of honest parties participating in the protocol. It is therefore not considered a violation of privacy if honest parties learn about the secret inputs of other honest parties. In fact, there exists a protocol (Ishai et al., CRYPTO 2010) that requires honest parties to reveal their private inputs to other honest parties towards the end of execution, once the corrupted party has been identified. But it is not reasonable to expect all honest parties to agree to share their private inputs, even with honest parties. There is always the practical possibility that an honest party may get corrupted later on, and the information obtained earlier may be used maliciously, thus making that party only semi-honest.

2 Intuition

It thus makes sense to consider a situation where every party is either corrupted by a malicious adversary, or semi-honest by itself. This is where the name of this paper originates; MPC should be secure against friends and foes alike.

There are mainly two ways in which information about secret inputs can reach these uncorrupted but semi-honest parties:

(a) The protocol itself may instruct all honest parties to reveal their private inputs to one another, as in the case mentioned previously.

(b) The malicious adversary may send parts of its view to certain honest parties during the execution of the protocol. This is especially problematic for protocols like BGW that assume an upper bound of t malicious parties and rely on $(t + 1)$-out-of-n secret sharing; if a malicious adversary controlling t parties sends all its shares to any honest party, the latter can compute the secret inputs of all the parties.

One idea to deal with the second issue above is to incorporate erasure of unsolicited messages from the malicious adversary into the protocol itself. That way, a semi-honest party would erase the information leakage during the execution of the protocol itself. However, this is difficult to guarantee due to practical obstacles, like physical limitations on erasures. Sometimes, it may not even be possible to define what exactly is to be erased.

Another possible solution may be to simply raise the threshold on the secret sharing schemes involved. For example, if the protocol needs to tolerate t malicious parties and sets of possibly colluding h^* semi-honest parties simultaneously, then simply using a protocol resistant against $(t+h^*)$ malicious parties might be good enough. But this strategy destroys efficiency, and $(t+h^*)$ malicious security need not imply (t, h^*)-FaF security in general, as proved in this paper.

This paper proposes a new security definition called FaF-security, that extends classical static and malicious MPC security. There are two variants; full-security and security-with-abort. Basically, (t, h^*)-FaF security guarantees that for any malicious adversary $\mathcal{A}$ corrupting t parties, and for any disjoint set of h^* parties, both the view of the adversary and the joint view of the h^* parties can be simulated separately in the ideal model.

3 Defining FaF Security

Consider the following function

$$f = \{f^k : \mathcal{X}_1^k \times \cdots \times \mathcal{X}_n^k \to \mathcal{Y}_1^k \times \cdots \times \mathcal{Y}_n^k\}_{k\in\mathbb{N}}$$

to be computed with secure MPC using a protocol Π. Let $\mathcal{X}^k$ denote $\mathcal{X}_1^k \times \cdots \times \mathcal{X}_n^k$. We now describe the real FaF model, the ideal model for full-security, and the ideal model for security-with-abort.

3.1 The Real FaF Model

Let the protocol Π operate over n parties, belonging to the set $P = \{P_1, \ldots, P_n\}$. Each party P_i holds the security parameter 1^k, along with its private input $x_i \in \mathcal{X}_i^k$. The computation proceeds for some polynomial number of rounds. The number of rounds as a function of the security parameter is denoted by $r(k)$. In each round, the parties either send or receive broadcasts, or communicate with some other party over a secure channel. At the end, the honest parties output values according to the specifications of the protocol. The view of a party is composed of its input, the randomness used, and the messages it receives from other parties during the protocol.

This model has two adversaries:

(a) A malicious adversary $\mathcal{A}$ that controls a subset $\mathcal{I} \subset P$ of the parties, having full access to their view and complete control over their actions. This adversary is non-uniform and is given an input $z_\mathcal{A}$. It can send messages arbitrarily to any other party in any round of the protocol, whenever it desires.

(b) A semi-honest adversary $\mathcal{A}_\mathcal{H}$ that controls a subset $\mathcal{H} \subset P\backslash\mathcal{I}$ of the remaining parties, having full access to their view although not their actions. This adversary is also non-uniform and is given an input $z_\mathcal{H}$.

The real-world global view in this case is

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

where the first term is the view of the malicious adversary $\mathcal{A}$, the second is the view of the semi-honest adversary $\mathcal{A}_\mathcal{H}$, and the third is the set of outputs of all the semi-honest parties, comprising the set $P\backslash\mathcal{I}$.

Note that, although the main idea is to provide security against unsolicited messages sent by the malicious adversary to semi-honest parties, this model cannot guarantee any security against messages sent to semi-honest parties after the completion of the protocol. We only consider messages sent by the malicious adversary to semi-honest parties during the rounds of the protocol itself.

3.2 The Ideal FaF Model for Full-Security

Each party P_i holds 1^k and $x_i \in \mathcal{X}_i^k$. Adversaries $\mathcal{A}$ and $\mathcal{A}_{\mathcal{H}}$ are each given auxiliary inputs along with x_i for each party they control. There is a trusted party T that holds 1^k. Each uncorrupted party $P_j \in P \backslash \mathcal{I}$ sends $x'_j = x_j$ as its input to T, while $\mathcal{A}$ sends some values x'_j for the parties P_j that it controls. Next, T computes the outputs $y_1, \ldots, y_n$ to be sent to the parties, and sends them out. The malicious adversary $\mathcal{A}$ sends to $\mathcal{A}_{\mathcal{H}}$ its randomness, inputs, auxiliary input, and the output it received from T. (This is necessary because the adversary can send parts of its view to semi-honest parties in the real world.) Each uncorrupted party then outputs whatever it received from T, and parties in $\mathcal{I}$ output nothing. Adversaries $\mathcal{A}$ and $\mathcal{A}_{\mathcal{H}}$ output some function of their view.

The global view in the ideal model is given by

$$\mathrm{IDEAL}^{f,\mathcal{A},\mathcal{A}_{\mathcal{H}}}_{1^k,x,z_{\mathcal{A}},z_{\mathcal{H}}} = \Big(\mathrm{VIEW}^{\mathrm{IDEAL}}_{\mathcal{A},f}(1^k,x), \mathrm{VIEW}^{\mathrm{IDEAL}}_{\mathcal{A},\mathcal{A}_{\mathcal{H}},f}(1^k,x), \mathrm{OUT}^{\mathrm{IDEAL}}_{\mathcal{A},f}(1^k,x)\Big)$$

where the first term is the view of the malicious adversary $\mathcal{A}$, the second is the view of the semi-honest adversary $\mathcal{A}_{\mathcal{H}}$, and the third is the set of outputs of all the non actively corrupted parties, comprising the set $P \backslash \mathcal{I}$.

Definition 3.1. We say Π computes f with computational (t, h^*)-FaF full-security if the following holds. For every non-uniform PPT adversary $\mathcal{A}$ controlling a set $\mathcal{I}$ of at most t parties, there exists a non-uniform PPT adversary $S_{\mathcal{A}}$, controlling $\mathcal{I}$ in the ideal model, and for every subset H having size at most h^* of the remaining parties $P \backslash \mathcal{I}$, controlled by a non-uniform semi-honest adversary $\mathcal{A}_{\mathcal{H}}$, there exists a semi-honest adversary $S_{\mathcal{A},\mathcal{H}}$ controlling $\mathcal{H}$ in the ideal world, such that the view of each adversary (taken separately) along with the outputs of all non actively corrupted parties is computationally indistinguishable from the view of its simulator counterpart in the ideal world.

Also note that if the joint view of $\mathcal{A}$ and $\mathcal{A}_{\mathcal{H}}$ in the above definition becomes indistinguishable from the ideal pair of $S_{\mathcal{A}}$ and $S_{\mathcal{A},\mathcal{H}}$, then we say Π computes f with *strong* computational (t, h^*)-FaF full-security.

3.3 The Ideal FaF Model for Security-With-Identifiable-Abort

The model here is almost identical to the one for full-security. However, the malicious adversary here can additionally instruct the trusted party to not send the computed outputs to the honest parties by publishing the identity of any one corrupted party.

Before the trusted party T sends out the outputs to all the parties, the malicious adversary $\mathcal{A}$ can send either `continue` or `abort,`P_j for some corrupted party P_j to T. In the former case, T sends out outputs as usual. In the latter case, T sends `abort,`P_j to all the parties.

4 The Main Theorem

The following is the main theorem proved by this paper.

Theorem 4.1. *Let $t, h^*, n \in \mathbb{N}$. Then, assuming OT and OWP exist, any n-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.*

Here is a sketch of the somewhat lengthy proof described in the paper.

First of all, it is proved that the GMW protocol admits FaF security-with-identifiable-abort, when the secret sharing involved is set to a $(t + h^* + 1)$-out-of-n sharing scheme.

Fix disjoint-party adversaries $\mathcal{A}$ and $\mathcal{H}$ having sizes at most t and h^* respectively. The simulators $S_{\mathcal{A}}$ and $S_{\mathcal{A},\mathcal{H}}$ work very similarly; the only difference is that parties under $S_{\mathcal{A},\mathcal{H}}$ must send the same messages that they are instructed to send by the protocol. Further, in case the adversary did not abort, the message received on every wire held by some party in $\mathcal{I} \cup \mathcal{H}$ is set to the XOR of the output on that wire and the share of that wire held by the corrupted or semi-honest party. The messages sent by $S_{\mathcal{A},\mathcal{H}}$ to $\mathcal{A}$ are consistent with the inputs of the semi-honest parties, which makes the protocol secure.

Next, the paper describes a reduction from residual FaF full-security to FaF security-with-identifiable-abort, in an uncorrupted-majority setting. This residuality condition can be removed if it is guaranteed that $2t + h^* < m$.

Definition 4.1. The residual $f_{S,x}$ of an n-ary function f with respect to a subset S of its inputs x is defined as the function obtained by fixing the values of the inputs at indices in $[n] \backslash S$.

So how is residual FaF full-security defined? Very similarly to FaF full-security; the only difference is that in the ideal world, the two adversaries $\mathcal{A}$ and $\mathcal{A}_{\mathcal{H}}$ receive the residual function $f_{\mathcal{I},x}$ instead of single outputs. The uncorrupted parties in $P \backslash (\mathcal{I} \cup \mathcal{H})$ still receive single outputs from the trusted party T.

Lemma 4.2. *Let $n, t, h^* \in \mathbb{N}$ be such that $t + h^* \leq n(k)$ and that $t < n/2$, and let f be an n-party functionality. Then there exists a protocol Π that computes f with strong perfect (t, h^*)-residual FaF full-security in the $(f'_{n-t}, \ldots, f'_n)$-hybrid model. Moreover, the protocol satisfies the following.*

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

(b) If $2t + h^* < n$ then Π admits (t, h^*)-FaF full-security in the $(f'_{n-t}, \ldots, f'_n)$-hybrid model.

Note that, for an n-party functionality f, and for $n' \in \{n - t, \ldots, n\}$, we define an n'-party functionality $f'_{n'}(x)$ in the security-with-identifiable-abort model. Let $y = (y_1, \ldots, y_n)$ be the output of f. Share each y_i in a $(n - t)$-out-of-n' secret sharing scheme, with respect to party P_i. Then the output of a party P_j is its respective shares of each y_i.

What does it mean to share a secret with respect to a party? This is when any subset of parties having size $(n - t)$ in the above case, which includes the party P_i, can reconstruct the secret. This can be implemented by performing additive secret sharing to generate a mask for the party P_i, followed by simple Shamir secret sharing to generate shares for the rest of the parties.

Note that, in general, increasing the threshold on the secret sharing scheme, for example to $t + h^* + 1$ in this case, is not sufficient to make protocols like GMW fully FaF secure.

Consider the following 3-party functionality:

$$(a, \perp, c) \to a \oplus b \oplus c$$

where $b \leftarrow \{0, 1\}$ and $t = h^* = 1$. Then:

(a) A 2-out-of-3 secret sharing would enable the malicious adversary to send its share to the semi-honest adversary and thus let it learn the secret input of the third honest party.

(b) A 3-out-of-3 secret sharing would enable the malicious adversary to withhold information about the final output from the other 2 parties.

Corollary 4.3. *Let $n, t, h^* \in \mathbb{N}$ be such that $t + h^* \leq n$ and $t < n/2$, and let f be an n-party functionality. Then, assuming OT exists, there exists a protocol Π that computes f with weak computational (t, h^*)-residual FaF full-security.*

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

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

Note that, in the special case that f is a no-input functionality such as coin-tossing, it is easy to see that Π would be able to admit (t, h^*)-FaF full security.

Also interesting is the gap between weak FaF security and security against a mixed adversary that controls one subset of parties maliciously and another subset of parties semi-honestly. The paper uses a result by Ishai et al. to prove that a 3-party functionality

$$f(\perp, \perp, \perp) = (b, \perp, b)$$

where $b \leftarrow \{0, 1\}$ can be computed with computational $(1, 1)$-FaF security, but not with computational $(1, 1)$-mixed security.

With this, the forward direction of Theorem 4.1 is proved. As for the reverse direction:

Lemma 4.4. *Let $n, t, h^* \in \mathbb{N}$ be such that $2t + h^* = n$. Then there exists an n-party functionality such that no protocol computes it with (weak) computational (t, h^*)-FaF full-security. Moreover, the claim holds even assuming the availability of simultaneous broadcast.*

The paper proves this by first proving the result for a 3-party case having $t = h^* = 1$, and then extending it to multiple parties by using a player-partitioning argument.

Lemma 4.5. *Assume that one-way permutation exists. Then there exists a 3-party functionality that no protocol can compute with (weak) computational $(1, 1)$-FaF full-security. Moreover, the following hold:*

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

(b) The claim holds even assuming the availability of simultaneous broadcast.

The paper proves this by defining the following functionality

$$Swap_n(a, (y_A, y_C), c) = \begin{cases} (a, c) & \text{if } f_k(a) = y_A \text{ and } f_k(c) = y_C \\ \perp & \text{otherwise} \end{cases}$$

where f_k is a OWP, and proving by an averaging argument that there must be some round where (A, B) can reconstruct the output with probability non-negligibly higher than (B, C).