Threshold-Optimal MPC With Friends and Foes

Nikolas Melissaris[1], Divya Ravi[1], and Sophia Yakoubov[1*]

Aarhus University, Denmark; {nikolas, divya, sophia.yakoubov}@cs.au.dk

Abstract. Alon *et. al* (Crypto 2020) initiated the study of MPC with *Friends and Foes* (FaF) security, which captures the desirable property that even up to h^* honest parties should learn nothing additional about other honest parties' inputs, even if the t corrupt parties send them extra information. Alon *et. al* describe two flavors of FaF security: *weak* FaF, where the simulated view of up to h^* honest parties should be indistinguishable from their real view, and *strong* FaF, where the simulated view of the honest parties should be indistinguishable from their real view *even in conjunction with the simulated / real view of the corrupt parties*. They give several initial FaF constructions with guaranteed output delivery (GOD); however, they leave some open problems. Their only construction which supports the optimal corruption bounds of $2t+h^* < n$ (where n denotes the number of parties) only offers weak FaF security and takes much more than the optimal three rounds of communication. In this paper, we describe two new constructions with GOD, both of which support $2t + h^* < n$. Our first construction, based on threshold FHE, is the first three-round construction that matches this optimal corruption bound (though it only offers weak FaF security). Our second construction, based on a variant of BGW, is the first such construction that offers strong FaF security (though it requires more than three rounds, as well as correlated randomness). Our final contribution is further exploration of the relationship between FaF security and similar security notions. In particular, we show that FaF security does not imply mixed adversary security (where the adversary can make t active and h^* passive corruptions), and that Best of Both Worlds security (where the adversary can make t active or $t + h^*$ passive corruptions, but not both) is orthogonal to both FaF and mixed adversary security.

Keywords: Secure Computation, Friends and Foes, Guaranteed Output Delivery, Round Complexity

*Funded in part by the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 669255 (MPCPRO) and No 803096 (SPEC).

Table of Contents

1 Introduction

A set of n mutually distrusting parties who have secrets $x_1, \ldots, x_n$ can use *secure multi-party computation* (MPC) [BGW88,Yao86] to compute a joint function $f(x_1, \ldots, x_n)$ of their secrets, without revealing anything more about those secrets to one another. MPC is typically parametrized by a threshold t such that as long as t or fewer participants collude, they cannot subvert the privacy and correctness guarantees of the computation. However, if t parties deviate from the protocol, no guarantees are made about what the remaining $n - t$ parties learn. Many MPC protocols (such as [IKP10,IKKP15,PR18]) make use of this by relying on fall-back protocols where, in the event of cheating, if one or more parties are identified as definitely *not* being one of the t cheaters, they are entrusted with the others' secrets.

Of course, this is not what we would like to use in practice. We would like even our honest peers — who do not collude with some central malicious adversary — not to learn our secrets. Alon *et. al* [AOP20] introduce MPC with Friends and Foes (or MPC with FaF security), which captures exactly this guarantee. Informally, a protocol achieves (t, h^*)-FaF security if, as in the standard definition of MPC, for any adversary $\mathcal{A}$ there exists a simulator $\mathcal{S}_{\mathcal{A}}$ which produces a view indistinguishable from that of the t corrupt parties without seeing the inputs of the honest parties. However, for FaF security, there must additionally exist a simulator $\mathcal{S}_{\mathcal{A}_{\mathcal{H}^*}}$ for every subset of up to h^* of the honest parties which produces a view indistinguishable from that of those honest parties, without seeing the inputs of the *remaining* honest parties. This implies that no matter what messages the corrupt parties send, they can not cause any h^* honest parties to learn more about their peers' inputs than they should.

Alon *et. al* define two degrees of FaF security:

Weak FaF. Here, though the output of $\mathcal{S}_{\mathcal{A}}$ must be indistinguishable from the real view of the t corrupt parties and the output of $\mathcal{S}_{\mathcal{A}_{\mathcal{H}^*}}$ must be indistinguishable from the real view of the h^* honest parties, taken together,

those views may be distinguishable from the set of real views. (That is, the simulated views may not be mutually consistent.)
Strong FaF. Here, the outputs of $\mathcal{S}_{\mathcal{A}}$ and $\mathcal{S}_{\mathcal{A}_{\mathcal{H}^*}}$ must be *jointly* indistinguishable from the real views of the t corrupt parties and h^* honest parties.

One can think of strong FaF as modeling the case where the adversary receives some feedback about what the honest parties learned, and weak FaF as modeling the case where there is no such feedback.

1.1 Prior Work

Alon *et. al* showed some inherent limitations on MPC with FaF security (Section 1.1), and gave some initial constructions (Section 1.1). Their results primarily focus on the notion of guaranteed output delivery (GOD) where corrupt parties cannot prevent the honest parties from obtaining the output.

Limitations Alon *et. al* consider two parameters of MPC protocols: number of rounds, and thresholds. They showed that two-round MPC with weak FaF security (and thus also strong FaF security) is impossible even for the lowest possible thresholds ($t = h^* = 1$); so, three rounds is the best we could hope to achieve. They then showed that even weak FaF security (and thus also strong FaF security) is unachievable for certain thresholds irrespective of the number of rounds. In particular, let n be the number of participants, t the bound on the number of corrupt parties, and h^* the bound on the number of honest parties who should learn nothing about other honest parties' secrets. Alon *et. al* show the following:

- Weak FaF secure MPC with $2t + h^* \geq n$ is impossible.
- Information theoretic (statistical) weak FaF secure MPC is impossible if:
  - $2t + 2h^* \geq n$ (even if broadcast is available).
  - $2t + 2h^* \geq n$ or $3t \geq n$ (when broadcast is *not* available).
- Information theoretic (perfect) weak FaF secure MPC with $3t + 2h^* \geq n$ is impossible (even when broadcast is available).

Constructions Alon *et. al* give several initial constructions of FaF secure MPC with GOD. They describe a round-optimal (three-round) construction that achieves strong FaF security, but only for $5t + 3h^* < n$. They also describe a threshold-optimal ($2t + h^* < n$) construction that only achieves *weak* FaF security. Finally, they show several information theoretic constructions. We summarize all of the constructions in Figure 1.

1.2 Our Contributions

In this paper, we close two of the gaps left open by the constructions of Alon *et. al*. We also extend the study of how FaF security relates to other security notions. The focus of our work is FaF security with GOD.

Construction	FaF Level	Security	Threshold	Rounds	Assumptions	Preprocessing?
From GMW [AOP20]	Weak	Comp	$2t + h^* < n$	Dependent on κ	OT, OWP	no
From DI [AOP20]	Strong	Comp	$5t + 3h^* < n$	3	PRG	no
BGW emulation [AOP20]	Strong	Statistical IT	$2t + 2h^* < n$	Dependent on κ	Broadcast	no
BGW emulation [AOP20]	Strong	Perfect IT	$3t + 2h^* < n$	Dependent on κ	None	no
Ours						
TFHE-FaF	Weak	Comp	$2t + h^* < n$	3	Lattices	no
BGW-BT-Comp	Strong	Comp	$2t + h^* < n$	$O(\kappa)$	Enhanced Trapdoor Permutations	Beaver triples

Fig. 1: Constructions. n denotes the number of participants, t denotes the bound on the number of corruptions, h^* denotes the bound on the number of honest parties against whom we want privacy, and κ denotes the multiplicative depth of the circuit being evaluated.

First, we give a three-round construction that achieves weak FaF security for $2t + h^* < n$ in the CRS (common reference string) model. This is the first construction that is optimal both in terms of the number of rounds and in terms of the threshold (even though it does not achieve the stronger notion of FaF). Second, we give a construction that achieves strong FaF security for $2t+h^* < n$. This is the first strong FaF construction to achieve the optimal threshold (though the number of rounds depends on the multiplicative depth of the function being computed). A caveat of our second construction is that it relies on correlated randomness.

Finally, we further the study of the relationship between FaF security and other related notions of security. Recalling the standard notions, while actively corrupt parties are completely controlled by the adversary and may deviate arbitrarily from the protocol; passively corrupt parties follow the protocol steps but leak their internal states to the adversary. Alon *et. al* showed that *Mixed Adversary* security (where the adversary can make t active corruptions and h^* passive ones) does not imply FaF security in the computational setting. We show the other direction; that FaF security does not imply mixed adversary security. We additionally consider *Best of Both Worlds* (BoBW) security [IKLP06,Kat07] (where the adversary can either make t active corruptions or $t+h^*$ passive ones, but not both). We show that FaF security does not imply BoBW security, and vice versa. These results are summarized in Figure 2.

Technical Overview

Three-Round Weak FaF Construction Our three-round construction is based on decentralized threshold FHE (described in Section 4), and follows the blueprint of Gordon *et. al* [GLS15]. In the first round, the participants exchange public keys. They then encrypt their inputs *to the set of all participants' public keys*, and broadcast the resulting ciphertexts in the second round. Once they receive one another's ciphetexts, they perform the homomorphic computation of the function locally, and broadcast their individual partial decryptions in the third

round. Everyone is then able to locally combine these partial decryptions and obtain the output. Gordon *et. al* show that this construction achieves guaranteed output delivery in the presence of a dishonest minority. We show that it has weak FaF security as long as $2t + h^* < n$.

Strong FaF Construction Our strong FaF construction is based on BGW [BGW88]. We proceed in three steps; first, we show that BGW with Beaver triple preprocessing [Bea92] achieves guaranteed output delivery in the presence of an adaptive mixed adversary making t fail-stop corruptions (where fail-stop corruptions are similar to passive corruptions, except that the parties may additionally choose to abort at any step) and h^* passive corruptions, as long as $2t+h^* < n$. We then apply the compiler of Canetti *et. al* [CLOS02], which relies on adaptively secure commitments and zero knowledge proofs, to instead allow our mixed adversary to make t *active* corruptions and h^* passive corruptions. Finally, we rely on the observation of Alon *et. al* that adaptive security implies strong FaF security to obtain our result.

Relation of FaF to Other Notions We consider FaF security, BoBW security and mixed adversary security. We describe several protocols that achieve some of these notions but not others, which, taken together with the results of Alon *et. al*, shows that all three notions are incomparable. In Figure 2 we summarize what we know about the relationship of FaF, BoBW and mixed adversaries.

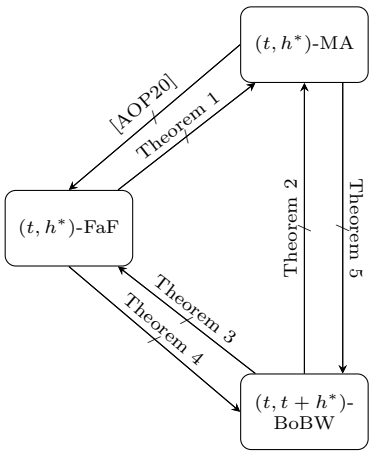

Fig. 2: Relationships of FaF to other notions. MA denotes security against mixed adversaries; BoBW denotes (active / passive) best of both worlds security.

1.3 Organization

In Section 2, we recall the definitions of FaF security. In Section 3, we describe our results about the relationship of FaF, BoBW and mixed adversary security. In Section 4, we describe decentralized threshold fully homomorphic encryption (dTFHE), which we use in one of our constructions. In Section 5 and Section 6, we describe our round optimal weak FaF and strong FaF constructions, respectively.

1.4 Notation

We use λ to denote the security parameter. By $\mathsf{poly}(\lambda)$ we denote a polynomial function in λ. By $\mathsf{negl}(\lambda)$ we denote a negligible function; that is, a function f such that $f(\lambda) < \frac{1}{p(\lambda)}$ holds for any polynomial $p(\cdot)$ and sufficiently large λ. We denote the set $\{1, \ldots, k\}$ by $[k]$ (or, equivalently, by $[1, \ldots, k]$).

2 Definitions

In this section, we recall the definitions given by Alon *et. al* [AOP20] of Friends and Foes security. They consider a classical adversary $\mathcal{A}$ corrupting t of parties, who would like to leak unauthorized information to some honest parties. So, we really have *two separate adversaries* in this setting:

1. The adversary $\mathcal{A}$ who *actively* corrupts a subset $\mathcal{I} \subseteq [n]$ of the parties, meaning that she can instruct the parties in $\mathcal{I}$ to arbitrarily deviate from the protocol. ($\mathcal{A}$ is given auxiliary input $y_{\mathcal{A}}$.)
2. An adversary $\mathcal{A}_{\mathcal{H}^*}$ who *passively* corrupts a subset $\mathcal{H}^* \subseteq [n] \setminus \mathcal{I}$ of the honest parties. ($\mathcal{A}_{\mathcal{H}^*}$ is given auxiliary input $y_{\mathcal{H}^*}$.)

For security parameter λ, inputs $\boldsymbol{x} = (x_1, \ldots x_n)$ and auxiliary inputs $y_{\mathcal{A}}, y_{\mathcal{H}^*}$, we define the following random variables for a real-world execution of protocol Π that computes a function f:

$\mathsf{OUT}^{\mathsf{REAL}}_{\mathcal{A},\Pi}(1^\lambda, \boldsymbol{x})$ is the output of the non-active parties (where, non-active refers to the honest and passively corrupt parties) $\mathcal{H} = [n] \setminus \mathcal{I}$.
$\mathsf{VIEW}^{\mathsf{REAL}}_{\mathcal{A},\Pi}(1^\lambda, \boldsymbol{x})$ is $\mathcal{A}$'s view during an execution of the protocol.
$\mathsf{VIEW}^{\mathsf{REAL}}_{\mathcal{A},\mathcal{A}_{\mathcal{H}^*},\Pi}(1^\lambda, \boldsymbol{x})$ is $\mathcal{A}_{\mathcal{H}^*}$'s view during an execution of the protocol. Since $\mathcal{A}$'s best strategy in order to leak information is to send her entire view, we assume that $\mathcal{A}_{\mathcal{H}^*}$'s view includes $\mathcal{A}$'s view.

We can now formalize the global view of the real world execution of Π:

$$\mathsf{REAL}^{\Pi,\mathcal{A},\mathcal{A}_{\mathcal{H}^*}}_{1^\lambda,\boldsymbol{x},y_{\mathcal{A}},y_{\mathcal{H}^*}} = \left(\mathsf{VIEW}^{\mathsf{REAL}}_{\mathcal{A},\Pi}(1^\lambda, \boldsymbol{x}), \mathsf{VIEW}^{\mathsf{REAL}}_{\mathcal{A},\mathcal{A}_{\mathcal{H}^*},\Pi}(1^\lambda, \boldsymbol{x}), \mathsf{OUT}^{\mathsf{REAL}}_{\mathcal{A},\Pi}(1^\lambda, \boldsymbol{x})\right)$$

It is useful to define the following projection of the global view to the view of each of the adversaries and the non-active parties' output:

$$\mathsf{REAL}^{\Pi,\mathcal{A},\mathcal{A}_{\mathcal{H}^*}}_{1^\lambda,\boldsymbol{x},y_{\mathcal{A}},y_{\mathcal{H}^*}}(\mathcal{A}) = \left(\mathsf{VIEW}^{\mathsf{REAL}}_{\mathcal{A},\Pi}(1^\lambda, \boldsymbol{x}), \mathsf{OUT}^{\mathsf{REAL}}_{\mathcal{A},\Pi}(1^\lambda, \boldsymbol{x})\right)$$

and

$$\mathsf{REAL}^{\Pi,\mathcal{A},\mathcal{A}_{\mathcal{H}^*}}_{1^\lambda,\boldsymbol{x},y_{\mathcal{A}},y_{\mathcal{H}^*}}(\mathcal{A}_{\mathcal{H}^*}) = \left(\mathsf{VIEW}^{\mathsf{REAL}}_{\mathcal{A},\mathcal{A}_{\mathcal{H}^*},\Pi}(1^\lambda,\boldsymbol{x}), \mathsf{OUT}^{\mathsf{REAL}}_{\mathcal{A},\Pi}(1^\lambda,\boldsymbol{x})\right).$$

Similarly we can define the following random variables for an ideal-world execution:

$\mathsf{OUT}^{\mathsf{IDEAL}}_{\mathcal{A},f}(1^\lambda,\boldsymbol{x})$ is the output of the non-active parties in $\mathcal{H}$.
$\mathsf{VIEW}^{\mathsf{IDEAL}}_{\mathcal{A},f}(1^\lambda,\boldsymbol{x})$ is $\mathcal{A}$'s simulated view.
$\mathsf{VIEW}^{\mathsf{IDEAL}}_{\mathcal{A},\mathcal{A}_{\mathcal{H}^*},f}(1^\lambda,\boldsymbol{x})$ is $\mathcal{A}_{\mathcal{H}^*}$'s simulated view.

As before we can formalize the global view of the ideal world execution:

$$\mathsf{IDEAL}^{f,\mathcal{A},\mathcal{A}_{\mathcal{H}^*}}_{1^\lambda,\boldsymbol{x},y_{\mathcal{A}},y_{\mathcal{H}^*}} = \left(\mathsf{VIEW}^{\mathsf{IDEAL}}_{\mathcal{A},f}(1^\lambda,\boldsymbol{x}), \mathsf{VIEW}^{\mathsf{IDEAL}}_{\mathcal{A},\mathcal{A}_{\mathcal{H}^*},f}(1^\lambda,\boldsymbol{x}), \mathsf{OUT}^{\mathsf{IDEAL}}_{\mathcal{A},f}(1^\lambda,\boldsymbol{x})\right)$$

We define the following projections:

$$\mathsf{IDEAL}^{f,\mathcal{A},\mathcal{A}_{\mathcal{H}^*}}_{1^\lambda,\boldsymbol{x},y_{\mathcal{A}},y_{\mathcal{H}^*}}(\mathcal{A}) = \left(\mathsf{VIEW}^{\mathsf{IDEAL}}_{\mathcal{A},f}(1^\lambda,\boldsymbol{x}), \mathsf{OUT}^{\mathsf{IDEAL}}_{\mathcal{A},f}(1^\lambda,\boldsymbol{x})\right)$$

and

$$\mathsf{IDEAL}^{f,\mathcal{A},\mathcal{A}_{\mathcal{H}^*}}_{1^\lambda,\boldsymbol{x},y_{\mathcal{A}},y_{\mathcal{H}^*}}(\mathcal{A}_{\mathcal{H}^*}) = \left(\mathsf{VIEW}^{\mathsf{IDEAL}}_{\mathcal{A},\mathcal{A}_{\mathcal{H}^*},f}(1^\lambda,\boldsymbol{x}), \mathsf{OUT}^{\mathsf{IDEAL}}_{\mathcal{A},f}(1^\lambda,\boldsymbol{x})\right).$$

2.1 FaF Security.

We say that a protocol Π computes a functionality f with (t,h^*)-FaF security if the two following simulators exist for any adversary $\mathcal{A}$ that statically corrupts at most t parties:

- A simulator $\mathcal{S}_{\mathcal{A}}$ which simulates $\mathcal{A}$'s view in the real world, and
- A simulator $\mathcal{S}_{\mathcal{A}_{\mathcal{H}^*}}$ which simulates the view of any subset $\mathcal{H}^*$ of size at most h^* of the honest parties, such that when given $\mathcal{S}_{\mathcal{A}}$'s entire state, $\mathcal{S}_{\mathcal{A}_{\mathcal{H}^*}}$ can generate a view that is indistinguishable from the real world view of $\mathcal{H}^*$.

We say that $\mathcal{S}_{\mathcal{A}_{\mathcal{H}^*}}$ is given the entire state of $\mathcal{S}_{\mathcal{A}}$ because in the real world, nothing stops an adversary from sending her entire view to one (or more) honest parties.

FaF Functionality with GOD In an ideal evaluation of the function f, the parties interact with the functionality as follows:

Inputs. Each party P_i is given input x_i. Adversary $\mathcal{A}$ is given auxiliary input $y_{\mathcal{A}} \in \{0,1\}^*$ and x_i for all $i \in \mathcal{I}$. Adversary $\mathcal{A}_{\mathcal{H}^*}$ is given auxiliary input $y_{\mathcal{H}^*} \in \{0,1\}^*$ and x_i for all $i \in \mathcal{H}^*$.

Parties Send Input. All non-active parties (i.e. the honest and passive parties) $i \in [n] \setminus \mathcal{I}$ send their inputs x_i to the functionality. $\mathcal{A}$ chooses inputs x'_i for $i \in \mathcal{I}$ as the input of each corrupt party and sends it to the functionality. For non-active parties i, we define $x'_i := x_i$.

Computation. The functionality computes $z = (z_1, \ldots, z_n) = f(x'_1, \ldots, x'_n)$ and sends z_i to each party i.

$\mathcal{A}_{\mathcal{H}^*}$ receives $\mathcal{A}$'s state. $\mathcal{A}_{\mathcal{H}^*}$ receives $\mathcal{A}$'s randomness, inputs, auxiliary input, and z_i for $i \in \mathcal{I}$.

Output. Each non-active party i outputs z_i, while the corrupted parties output nothing. $\mathcal{A}_{\mathcal{H}^*}$ and $\mathcal{A}$ output some function of their view.

Weak and Strong FaF Definitions In the following, we use $\equiv$ to denote computational indistinguishability.

Definition 1 (Weak FaF). *Let Π be a protocol for computing f. We say that Π computes f with computational* weak *(t, h^*)-FaF security (with GOD), if the following holds. For every non-uniform PPT adversary $\mathcal{A}$ controlling a set $\mathcal{I} \subset [n]$ of size at most t in the real world, there exists a non-uniform PPT simulator $\mathcal{S}_{\mathcal{A}}$ controlling $\mathcal{I}$ in the ideal world; and for every subset of the remaining parties $\mathcal{H}^* \subset [n] \setminus \mathcal{I}$ of size at most h^* controlled by a non-uniform passive PPT adversary $\mathcal{A}_{\mathcal{H}^*}$ there exists a non uniform PPT simulator $\mathcal{S}_{\mathcal{A}_{\mathcal{H}^*}}$, controlling $\mathcal{H}^*$ in the ideal world such that*

$$\mathsf{IDEAL}^{\mathcal{S}_{\mathcal{A}}, \mathcal{S}_{\mathcal{A}_{\mathcal{H}^*}}}_{1^\lambda, \boldsymbol{x}, y_{\mathcal{A}}, y_{\mathcal{H}}}(\mathcal{S}_{\mathcal{A}}) \equiv \mathsf{REAL}^{\mathcal{A}, \mathcal{A}_{\mathcal{H}}}_{1^\lambda, \boldsymbol{x}, y_{\mathcal{A}}, y_{\mathcal{H}}}(\mathcal{A})$$

and

$$\mathsf{IDEAL}^{\mathcal{S}_{\mathcal{A}}, \mathcal{S}_{\mathcal{A}_{\mathcal{H}^*}}}_{1^\lambda, \boldsymbol{x}, y_{\mathcal{A}}, y_{\mathcal{H}}}(\mathcal{S}_{\mathcal{A}_{\mathcal{H}^*}}) \equiv \mathsf{REAL}^{\mathcal{A}, \mathcal{A}_{\mathcal{H}}}_{1^\lambda, \boldsymbol{x}, y_{\mathcal{A}}, y_{\mathcal{H}}}(\mathcal{A}_{\mathcal{H}^*})$$

for any set of inputs $\boldsymbol{x} \in (\{0,1\}^*)^n$, any auxiliary inputs $(y_{\mathcal{A}}, y_{\mathcal{H}^*}) \in (\{0,1\}^*)^2$, and any large enough security parameter $\lambda \in \mathbb{N}$.

Definition 2 (Strong FaF). *For $\mathcal{A}, \mathcal{S}_{\mathcal{A}}, \mathcal{S}_{\mathcal{A}_{\mathcal{H}^*}}$ defined as in Definition 1, we say that Π computes f with computational* strong *(t, h^*)-FaF security (with GOD), if*

$$\mathsf{IDEAL}^{\mathcal{S}_{\mathcal{A}}, \mathcal{S}_{\mathcal{A}_{\mathcal{H}^*}}}_{1^\lambda, \boldsymbol{x}, y_{\mathcal{A}}, y_{\mathcal{H}}} \equiv \mathsf{REAL}^{\mathcal{A}, \mathcal{A}_{\mathcal{H}}}_{1^\lambda, \boldsymbol{x}, y_{\mathcal{A}}, y_{\mathcal{H}}}$$

for any set of inputs $\boldsymbol{x} \in (\{0,1\}^*)^n$, any auxiliary inputs $(y_{\mathcal{A}}, y_{\mathcal{H}^*}) \in (\{0,1\}^*)^2$, and any large enough security parameter $\lambda \in \mathbb{N}$.

The main difference is that in the strong notion of FaF security we want the simulated views of $\mathcal{A}$ and $\mathcal{A}_{\mathcal{H}^*}$ to be indistinguishable from the real views *even when taken together.*

3 Relation of FaF to Other Notions

Somewhat surprisingly, Alon *et. al* show that standard security against a static adversary making $t + h^*$ active corruptions does not imply (t, h^*) FaF security. Informally, this is because the simulator $\mathcal{S}_{\mathcal{A}_{\mathcal{H}^*}}$ for the honest parties is not allowed to choose which input to send to the ideal functionality, so it does not have as much power as the standard security simulator $\mathcal{S}$. Security against a (t, h^*) mixed adversary making t active corruptions and h^* passive corruptions also does not imply (t, h^*) FaF security. This is because the mixed adversary simulator can decide the active parties' inputs based on the passive parties' inputs; however, the FaF simulator $\mathcal{S}_{\mathcal{A}}$ does not know any of the honest parties' inputs when simulating.

On the other hand, Alon *et. al* show that security against an *adaptive* adversary making $t + h^*$ active corruptions *does* imply (t, h^*) FaF security. This is because a simulator for an adaptive adversary needs to be able to handle corruptions which occur after the end of the protocol execution, at which point it cannot choose the input even for actively corrupt parties. We observe that the proof given by Alon *et. al* also shows that security against an adaptive *mixed* (t, h^*) adversary making t active corruptions and h^* passive corruptions implies (t, h^*) FaF security, and we use this in our strong FaF construction.

Here, we further explore the relationship between FaF security and other security notions. First, we show the other direction: that (t, h^*) FaF security does not imply security against a (t, h^*) mixed adversary making t active corruptions and h^* passive corruptions, making the FaF and mixed adversary models incomparable. We do so by giving an example (Example 1) of a protocol that achieves (t, h^*) FaF security but not (t, h^*) mixed adversary security. We also consider $(t, t + h^*)$ Best of Both Worlds (BoBW) security, where the same protocol must tolerate either t active corruptions *or* $t + h^*$ passive corruptions (but not both). We show that BoBW is incomparable to both FaF and mixed adversaries.

Example 1 ($\Pi_{\neg\mathsf{MA}}$). Consider a function $f(x_1, \ldots, x_n)$, and a protocol Π_{FaF} that computes any function with (t, h^*) FaF security. Now, consider a function $g((x_1, \rho_1), \ldots, (x_n, \rho_n))$, where each party P_i has an additional input ρ_i. g returns $(x_1, \ldots, x_n)$ to everyone if at least $t + 1$ of the ρ_i's are equal, and returns $f(x_1, \ldots, x_n)$ to everyone otherwise. The following protocol $\Pi_{\neg\mathsf{MA}}$ computes $f(x_1, \ldots, x_n)$ with (t, h^*) FaF security but not with security against a (t, h^*) mixed adversary.

1. Each party P_i chooses ρ_i uniformly at random from a large space.
2. The parties use Π_{FaF} to compute $g((x_1, \rho_1), \ldots, (x_n, \rho_n))$.
3. Each party outputs the value returned by Π_{FaF}.

Theorem 1. *Protocol $\Pi_{\neg\mathsf{MA}}$ (Example 1) computes $f(x_1, \ldots, x_n)$ with (t, h^*) FaF security but not with security against a (t, h^*) mixed adversary making t active corruptions and h^* passive corruptions.*

Proof (Sketch). $\Pi_{\neg\mathsf{MA}}$ achieves FaF security, because the adversary cannot possibly guess the honest parties' randomly chosen values ρ_i, and so cannot exploit the additional leakage given by the output of g.

However, the mixed adversary knows h^* passive party inputs and randomly chosen ρ_i's, and can choose one of those to set corrupt parties' ρ_i's to. This allows the mixed adversary to learn all parties' inputs, and is clearly insecure.

Remark 1. We observe that since the above reduction (Example 1) is information-theoretic, plugging in the statistically-secure FaF protocol of [AOP20] to instantiate Π_{FaF} would yield a statistically-secure protocol $\Pi_{\neg\mathsf{MA}}$ that satisfies (t, h^*) FaF security but not (t, h^*) mixed security. This shows a separation between statistical FaF and mixed security at the protocol level, which was left as an open question in [AOP20] (the only known separation at the protocol level was for computational security).

Theorem 2. *If Π_{FaF} from Example 1 is replaced with a protocol Π_{BoBW} which has $(t, t+h^*)$ BoBW security, then protocol $\Pi_{\neg\mathsf{MA}}$ (Example 1) computes $f(x_1, \ldots, x_n)$ with $(t, t + h^*)$ BoBW security, but not with security against a (t, h^*) mixed adversary making t active corruptions and h^* passive corruptions.*

Proof (Sketch). $\Pi_{\neg\mathsf{MA}}$ achieves BoBW security, since an adversary making just t corruptions cannot guess honest parties' ρ_i values and thus cannot exploit the additional leakage, and an adversary making $t + h^*$ corruptions cannot dishonestly choose passive parties' values ρ_i to be equal.

However, as in the proof of Theorem 1, the mixed adversary knows h^* passive party inputs and randomly chosen ρ_i's, and can choose one of those to set corrupt parties' ρ_i's to.

We next show that BoBW security does not imply FaF security, by giving an example (Example 2) of a protocol that achieves $(t, t + h^*)$ BoBW security but not (t, h^*) FaF security.

Example 2 ($\Pi_{\neg\mathsf{FaF}}$). Consider a function $f(x_1, \ldots, x_n)$, and a protocol Π_{BoBW} that computes any function with $(t, t+h^*)$ BoBW security. The following protocol $\Pi_{\neg\mathsf{FaF}}$ computes $f(x_1, \ldots, x_n)$ with $(t, t + h^*)$ BoBW security, but not with (t, h^*) FaF security.

1. The parties use Π_{BoBW} to compute $f(x_1, \ldots, x_n)$.
2. If a party P_i receives a special "attack" message from t parties, it sends it's input x_i to the *other* $n - t$ parties. (Note that *sending* an "attack" message is not part of the instructions.)
3. Each party outputs the value returned by Π_{BoBW}.

Theorem 3. *Protocol $\Pi_{\neg\mathsf{FaF}}$ (Example 2) computes $f(x_1, \ldots, x_n)$ with $(t, t+h^*)$ BoBW security, but not with (t, h^*) FaF security.*

Proof (Sketch). $\Pi_{\neg\mathsf{FaF}}$ achieves $(t, t+h^*)$ BoBW security: if there are only passive corruptions, no party will send an "attack" message, and thus the second step of $\Pi_{\neg\mathsf{FaF}}$ will never come into play. If there are only t active corruptions, they are able to trigger the attack, but will not learn the honest parties' inputs because those inputs are only sent to parties who didn't send attack messages. $\Pi_{\neg\mathsf{FaF}}$ does not achieve (t, h^*) FaF security: the t corrupt parties can easily trigger an attack, causing all the honest parties to learn one another's inputs.

It remains to show that neither FaF or mixed adversary security imply BoBW security. This follows from the fact that in both FaF and mixed adversary security, the simulator can choose the inputs of the actively corrupt parties. However, in BoBW security, in the case where the adversary only makes passive corruptions, the simulator is unable to choose the inputs of any parties.

Example 3 ($\Pi_{\neg\mathsf{BoBW}}$). Consider a function $f(x_1, \ldots, x_n)$, and a protocol Π_{FaF} that computes f with (t, h^*) FaF security. Now, consider a function $g((x_1, y_1), \ldots, (x_n, y_n))$, where each party P_i has an additional input y_i. g returns $y_i \wedge f(x_1, \ldots, x_n)$ to each party P_i. It returns $\perp$ to everyone else. The following protocol $\Pi_{\neg\mathsf{BoBW}}$ computes $g((x_1, y_1), \ldots, (x_n, y_n))$ with (t, h^*) FaF security but not with $(t, t + h^*)$ BoBW security.

1. The parties use Π_{FaF} to compute $z = f(x_1, \ldots, x_n)$.
2. Each party P_i outputs $y_i \wedge z$.

Theorem 4. *Protocol $\Pi_{\neg\mathsf{BoBW}}$ (Example 3) computes $g((x_1, y_1), \ldots, (x_n, y_n))$ with (t, h^*) FaF security but not with $(t, t + h^*)$ BoBW security.*

Proof (Sketch). $\Pi_{\neg\mathsf{BoBW}}$ achieves (t, h^*) FaF security: a simulator can always set one of the actively corrupt parties' auxiliary inputs y_i to be 1 to learn the output of f.

However, it does *not* achieve BoBW security: in the case where the adversary can only make passive corruptions, if all parties' auxiliary inputs y_i are 0, the simulator does not learn the output of f, which it needs in order to simulate successfully.

Theorem 5. *If Π_{FaF} from Example 3 is replaced with a protocol Π_{MA} which has (t, h^*) mixed adversary security, then protocol $\Pi_{\neg\mathsf{BoBW}}$ (Example 3) computes $g((x_1, y_1), \ldots, (x_n, y_n))$ with (t, h^*) mixed adversary security but not with $(t, t + h^*)$ BoBW security.*

The proof is the same as the proof of Theorem 4.

Remark 2. We design the function g in such a way that *any* party can learn the output of f by setting their auxiliary input y_i to 1. This gives us FaF and mixed adversary security; no matter whom the adversary actively corrupts, the simulator can use that party to learn the output of f, and simulate for the rest.

4 Building Block: Decentralized Threshold FHE

We recap the definitions of d-out-of-n decentralized threshold fully homomorphic encryption (dTFHE) as presented by Boneh et al. [BGG+18].

Syntax A dTFHE scheme is a tuple of PPT algorithms ($\mathsf{DistGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PDec}, \mathsf{Combine}, \mathsf{SimPDec}$) with the following syntax:

$\mathsf{DistGen}(1^\lambda, 1^\kappa, i; \rho_i) \to (\mathtt{pk}_i, \mathtt{sk}_i)$**:** On input the security parameter λ, a depth bound κ, party index i and randomness ρ_i, the distributed setup outputs a public-secret key pair $(\mathtt{pk}_i, \mathtt{sk}_i)$ for party i. The public key of the scheme is denoted by $\mathtt{pk} = (\mathtt{pk}_1 \parallel \mathtt{pk}_2 \parallel \ldots \parallel \mathtt{pk}_n)$.

$\mathsf{Enc}(\mathtt{pk}, m; \rho) \to \mathtt{c}$**:** On input a public key $\mathtt{pk}$ and a plaintext m in the message space $\mathcal{M}$, the randomized algorithm outputs a ciphertext $\mathtt{c}$.

$\mathsf{Eval}(\mathtt{pk}, C, \mathtt{c}_1, \ldots, \mathtt{c}_k) \to \mathtt{c}$**:** On input a public key $\mathtt{pk}$, a circuit $C : \mathcal{M}^k \to \mathcal{M}$ of depth at most κ, and a set of k ciphertexts $\mathtt{c}_1, \ldots, \mathtt{c}_k$ (where $k = \mathsf{poly}(\lambda)$), the evaluation algorithm outputs an encrypted evaluation $\mathtt{c}$.

$\mathsf{PDec}(\mathtt{pk}, \mathtt{sk}_i, \mathtt{c}) \to \mathtt{d}_i$**:** On input the public key $\mathtt{pk}$, a ciphertext $\mathtt{c}$ and a secret key $\mathtt{sk}_i$ the algorithm outputs a partial decryption $\mathtt{d}_i$.

$\mathsf{Combine}(\mathtt{pk}, \{\mathtt{d}_i\}_{i \in S}) \to m \backslash \perp$**:** On input a public key $\mathtt{pk}$ and a set partial decryptions $\{\mathtt{d}_i\}_{i \in S}$ where $S \subseteq [n]$, the combination algorithm outputs a plaintext m or the symbol $\perp$.

$\mathsf{SimPDec}(\mathtt{c}, \mathtt{pk}, \{\mathtt{sk}_i\}_{i \in \mathcal{I}}, z) \to \{\mathtt{d}_i\}_{i \in [n] \setminus \mathcal{I}}$**:** On input a ciphertext $\mathtt{c}$, the public key $\mathtt{pk}$, the secret keys of at most d parties, and the target plaintext z, the simulated decryption algorithm outputs partial decryptions on behalf of the rest of the parties which are consistent with $\mathtt{c}$ decrypting to z.

Properties As in a standard homomorphic encryption scheme, we require that a dTFHE scheme satisfies correctness and security, which we describe informally below. We give the formal definitions in Appendix A.

Correctness. Informally, a dTFHE scheme is said to be *correct* if combining at least $d+1$ partial decryptions of any honestly generated ciphertext output by the evaluation algorithm returns the correct evaluation of the corresponding circuit on the underlying plaintexts.

Semantic Security. Informally, a dTFHE scheme satisfies *semantic security* if no PPT adversary can distinguish between encryptions of a pair of (adversarially chosen) plaintext messages m_0 and m_1 of the same length, even given the secret keys corresponding to a subset $\mathcal{I}$ of the parties for any set $\mathcal{I}$ of size at most d. Since we use the dTFHE scheme as a tool in our

semi-malicious MPC construction[1], we define the notion with respect to a semi-malicious adversary $\mathcal{A}$.

Simulation Security. Informally, a dTFHE scheme satisfies *simulation security* if there exists an efficient algorithm $\mathsf{SimPDec}$ that takes as input a ciphertext $\mathtt{c}$, the public key $\mathtt{pk}$, the secret keys of at most d parties and the target plaintext z, and outputs a set of partial decryptions on behalf of the rest of the parties such that its output is computationally indistinguishable from the output of the real algorithm PDec that outputs partial decryptions of the ciphertext $\mathtt{c}$ using the corresponding secret keys for the same subset of parties. Similar to semantic security, we define this notion with respect to a semi-malicious adversary $\mathcal{A}$.

5 Three-Round MPC with Weak FaF and Guaranteed Output Delivery

In this section, we present a three-round MPC construction in the CRS model (where it is assumed that parties have access to a common reference string at the beginning of the protocol execution) that achieves weak FaF security and GOD when $n > 2t + h^*$. This is round-optimal, following the impossibility of two-round MPC with weak FaF security and GOD shown in [AOP20] (which holds even in the CRS model). Specifically, their result shows that there are functionalities that cannot be computed with $(1, 1)$ weak FaF security and GOD in less than three rounds, for any $n \geq 3$.

Our construction is based on the three-round construction of Gordon *et. al* [GLS15] that achieves standard security with GOD against $t < n/2$ active corruptions. At a high-level, this construction uses the tool of distributed threshold fully homomorphic encryption scheme (dTFHE) with threshold t and proceeds as follows. First, the distributed setup allows the parties to obtain their individual public / secret key pairs. Each of them broadcasts their public key. Next, the parties broadcast encryptions of their input, which can be homomorphically evaluated to compute an encryption of the output. In the last round, the parties compute and broadcast partial decryptions of the output ciphertext, which can be combined to obtain the output.

We observe that the above construction admits (t, h^*) weak FaF security and GOD when $n > 2t + h^*$, if a dTFHE scheme with threshold $(t + h^*)$ is used instead. Intuitively, security of such a dTFHE scheme ensures that the joint view of the active and passively corrupt parties comprising of $(t+h^*)$ secret keys does not reveal any information about the inputs of the honest parties (beyond the output of computation). Correctness of such a scheme ensures that even if up to t parties abort, guaranteed output delivery is achieved because the partial decryptions sent by the remaining $n - t > t + h^*$ parties suffice to compute the output.

[1] where semi-malicious security [AJL+12] refers to security against an adversary who needs to follow the protocol specification, but has the liberty to decide the input and random coins in each round.

Similar to the work of Gordon *et. al* [GLS15], we present a three-round construction $\Pi^{\mathsf{sm}}_{\mathsf{wFaF}}$ that is secure against semi-malicious adversaries. Semi-malicious security was introduced by Aashrov *et. al* [AJL+12] and subsequently used in many works as a stepping-stone on the way to achieving active security. Recall that a semi-malicious adversary needs to follow the protocol specification, but has the liberty to decide the input and random coins in each round. Additionally, the parties controlled by the semi-malicious adversary may choose to abort at any time. To upgrade a semi-malicious construction to achieve active security, the general round-preserving compiler of Asharov *et. al* uses UC NIZKs (non-interactive zero-knowledge proofs) in the CRS model.

We give a formal description of the protocol $\Pi^{\mathsf{sm}}_{\mathsf{wFaF}}$ below.

Inputs: Each party P_i has an input $x_i \in \{0,1\}^\lambda$.
Output: $f(x_1, \ldots, x_n)$, where the function f is represented by a circuit C.
Tools: A $(t+h^*)$-out-of-n dTFHE scheme ($\mathsf{DistGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PDec}, \mathsf{Combine}, \mathsf{SimPDec}$). Such a scheme can be built based on LWE [BGG+18,GLS15].

Round 1: Each party P_i does the following:
- Computes $(\mathtt{pk}_i, \mathtt{sk}_i) \leftarrow \mathsf{DistGen}(1^\lambda, 1^\kappa, i; \rho_i)$ using randomness ρ_i.
- Broadcasts $\mathtt{pk}_i$.

Round 2: All parties set $\mathtt{pk} := (\mathtt{pk}_1 || \ldots || \mathtt{pk}_n)$ (where a default public key is used corresponding to parties who have aborted in the first round).
Each party P_i does the following:
- Computes the encryption of its input as $\mathtt{c}_i \leftarrow \mathsf{Enc}(\mathtt{pk}, x_i)$.
- Broadcasts $\mathtt{c}_i$.

Round 3: Each party P_i does the following:
- Computes the homomorphic evaluation of the circuit C on the ciphertexts as $\mathtt{c} \leftarrow \mathsf{Eval}(\mathtt{pk}, C, \mathtt{c}_1, \ldots, \mathtt{c}_n)$, where $\mathtt{c}_j$ is computed using default input and randomness if P_j aborted during the previous rounds.
- Computes her own partial decryption as $\mathtt{d}_i \leftarrow \mathsf{PDec}(\mathtt{pk}, \mathtt{sk}_i, \mathtt{c})$.
- Broadcasts the partial decryption $\mathtt{d}_i$.

Output Computation: Let $S \subseteq [n]$ denote the set of parties who have not yet aborted. Each party combines the partial decryptions broadcast by parties in S to obtain the output as $z \leftarrow \mathsf{Combine}(\mathtt{pk}, \{\mathtt{d}_j\}_{j \in S})$.

Protocol $\Pi^{\mathsf{mal}}_{\mathsf{wFaF}}$. Let $\Pi^{\mathsf{mal}}_{\mathsf{wFaF}}$ denote the three-round construction obtained by applying the compiler of Asharov *et. al* [AJL+12] to the three-round protocol $\Pi^{\mathsf{sm}}_{\mathsf{wFaF}}$ in the semi-malicious setting. In particular, in every round of $\Pi^{\mathsf{mal}}_{\mathsf{wFaF}}$, each party executes the actions of the corresponding round of $\Pi^{\mathsf{sm}}_{\mathsf{wFaF}}$ along with a non-interactive zero-knowledge proving that she is following the protocol consistently with respect to certain random coins. This compiler is round-preserving and preserves security of the underlying construction (i.e. if the underlying protocol achieves GOD, so does the compiled protocol).

We state the formal theorem below.

Theorem 6. *Let f be an efficiently computable n-party function and let $n > 2t + h^*$. Assuming a setup with CRS and the existence of a $(t + h^*)$-out-of-n decentralized threshold fully homomorphic encryption scheme, the three-round protocol $\Pi^{\mathsf{mal}}_{\mathsf{wFaF}}$ achieves (t, h^*) weak FaF security with guaranteed output delivery.*

Proof. To prove the theorem, we construct the simulators $\mathcal{S}_{\mathcal{A}}$ and $\mathcal{S}_{\mathcal{A}_{\mathcal{H}^*}}$ for $\Pi^{\mathsf{sm}}_{\mathsf{wFaF}}$ in the semi-malicious setting. This suffices to complete the proof, since active security would directly follow from the result of [AJL+12].

We begin with the description of $\mathcal{S}_{\mathcal{A}}$. Let $\mathcal{I}$ and $\mathcal{H}^*$ denote the set of indices of the t semi-malicious corrupt parties and the remaining parties respectively.

First Round: $\mathcal{S}_{\mathcal{A}}$ simulates public keys of honest parties by honestly generating key pairs.
Second Round: $\mathcal{S}_{\mathcal{A}}$ simulates input ciphertexts by broadcasting $\mathsf{c}_j \leftarrow \mathsf{Enc}(\mathtt{pk}, 0)$ on behalf of P_j, where $j \in [n] \setminus \mathcal{I}$.
Third Round: $\mathcal{S}_{\mathcal{A}}$ does the following:

- Computes $\mathsf{c} \leftarrow \mathsf{Eval}(\mathtt{pk}, C, \mathsf{c}_1, \ldots, \mathsf{c}_n)$ honestly, where c_i is computed using default input and randomness if P_i ($i \in \mathcal{I}$) aborted during the previous rounds.
- Reads the witness tape of the semi-malicious adversaries to learn the inputs x_i and secret keys $\mathtt{sk}_i$ for each $i \in \mathcal{I}$. If P_i has aborted, x_i is set as the default input.
- Invokes the ideal functionality $\mathcal{F}$ on inputs x_i for $i \in \mathcal{I}$ and receives the output z.
- Runs the simulated decryption algorithm to obtain partial decryptions as $\{\mathsf{d}_j\}_{j\in[n]\setminus\mathcal{I}} \leftarrow \mathsf{SimPDec}(\mathsf{c}, \mathtt{pk}, \{\mathtt{sk}_i\}_{i\in\mathcal{I}}, z)$.
- Broadcasts the partial decryption d_j on behalf of party P_j ($j \in [n] \setminus \mathcal{I}$).

Consider the following sequence of hybrids:

Hyb_0**:** Same as the real world execution of $\Pi^{\mathsf{sm}}_{\mathsf{wFaF}}$.

Hyb_1**:** Same as Hyb_0, except that the partial decryptions for honest parties are computed as $\mathsf{SimPDec}(\mathsf{c}, \mathtt{pk}, \{\mathtt{sk}_i\}_{i\in\mathcal{I}}, z)$ instead of $\mathsf{PDec}(\mathtt{pk}, \mathtt{sk}_i, \mathsf{c})$. It follows from simulation security of the dTFHE scheme that this hybrid is indistinguishable from the previous hybrid.

Hyb_2**:** Same as Hyb_1, except that the input ciphertexts of honest parties are computed as $\mathsf{c}_j \leftarrow \mathsf{Enc}(\mathtt{pk}, 0)$ using a dummy input 0 for $j \in [n]\setminus\mathcal{I}$, instead of using the actual input x_j. It follows from the semantic security of the dTFHE scheme that this hybrid is indistinguishable from the previous hybrid.

Since Hyb_2 corresponds to the ideal execution and every pair of consecutive hybrids are indistinguishable, this completes the proof that the view of $\mathcal{A}$ in the real world is indistinguishable from her view in the ideal world execution.

Next, suppose we fix the adversary $\mathcal{A}$ corrupting the parties in $\mathcal{I}$ where $\mathcal{I}$ is of size at most t, and let $\mathcal{H}^* \subseteq [n] \setminus \mathcal{I}$ of size at most h^*. The passive simulator $\mathcal{S}_{\mathcal{A}_{\mathcal{H}^*}}$ works very similarly to $\mathcal{S}_{\mathcal{A}}$. The only difference is that the messages the

simulator sends to the adversary on behalf of the parties in $\mathcal{H}^*$ are the actual messages computed as per protocol specifications. For instance, the input ciphertexts will be computed as encryptions of the real inputs of parties in $\mathcal{H}^*$ (unlike in $\mathcal{S}_\mathcal{A}$ where it was computed as encryptions of dummy input 0). Similarly, the partial decryptions of the parties in $\mathcal{H}^*$ would be computed honestly using their secret keys (not as output of $\mathsf{SimPDec}$). Lastly, the sequence of hybrids and indistinguishability can be argued as above; it follows from the semantic and simulation security of the dTFHE scheme having threshold $(t + h^*)$.

Remark 3. Note that the protocol described in this section is not strongly FaF secure; this is because the simulator $\mathcal{S}_\mathcal{A}$ does not know the honest parties' inputs, and instead encrypts a default value on their behalf. However, this simulation cannot then be consistent with an honest party's simulated view, since their view must include their real input and randomness that maps that input to the ciphertext they broadcast.

6 Optimal-Threshold MPC with Strong FaF and Guaranteed Output Delivery

Alon *et. al* [AOP20] showed that even weak FaF with GOD is impossible if $2t+h^* \geq n$. Their proof holds even if arbitrary correlated randomness is available to the parties. So, the best that we could possible hope for is strong FaF with GOD and $2t + h^* + 1 = n$.

In this section we prove that BGW with Beaver triple preprocessing and augmented with adaptive zero knowledge proofs achieves exactly this. We do this in three steps, as described in Figure 3. First, in Section 6.1, we prove that BGW with Beaver triple preprocessing achieves security with GOD against adaptive mixed (fail-stop / passive) adversaries. Second, in Section 6.2, we use the compiler of Canetti *et. al* [CLOS02] to show that our protocol against adaptive mixed (fail-stop / passive) adversaries can be augmented with adaptive commitments and zero knowledge proofs of correct behavior in order to achieve security against adaptive mixed (*active* / passive) adversaries. Third, we invoke a theorem from Alon *et. al* [AOP20] to argue that any such protocol achieves GOD with strong FaF security.

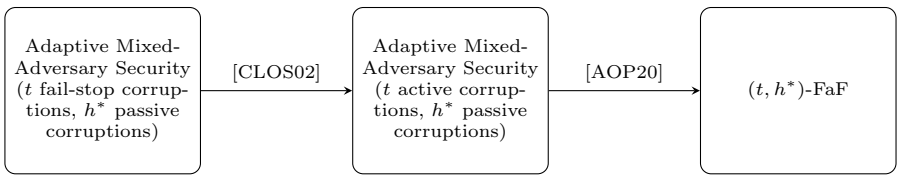

Fig. 3: Getting (t, h^*)-FaF Security from BGW

We restate the theorem of Alon *et. al* [AOP20] which connects adaptive security to strong FaF below. Alon *et. al* prove that $(t + h^*)$ adaptive security implies (t, h^*) FaF; however, their proof can be strengthened (with no modifications necessary) to show that *adaptive mixed security with a corruption budget of t active corruptions and h^* passive corruptions* implies (t, h^*) FaF.

Theorem 7 ([AOP20], Theorem 5.3). *Let* $\mathsf{type} \in \{\text{computational}, \text{statistical}, \text{perfect}\}$*, and let* Π *be an n-party protocol computing some n-party functionality f with* type *adaptive mixed security with a corruption budget of t active corruptions and h^* passive corruptions. Then* Π *computes f with* type *strong (t, h^*)-FaF-security.*

6.1 Adaptive BGW Against Mixed (Fail-Stop / Passive) Adversaries

We first recall BGW (Section 6.1), and how Beaver triples can be used to improve the corruption threshold (Section 6.1).

Brief Overview of BGW Without Preprocessing Let $\mathbb{F}$ be a finite field. A secret value s is secret shared using a polynomial $f_s(x) \in \mathbb{F}[x]$ of degree d such that $f_s(0) = s$ and each party P_i holds $f_s(i)$. The evaluation of the circuit then proceeds gate by gate. In order to add two secret shared values x and y, each party P_i can locally add the shares $f_x(i)$ and $f_y(i)$ that they are holding to get $f_{x+y}(i) = f_x(i) + f_y(i)$. This is a valid sharing of $x + y$, because f_{x+y} is of the same degree as f_x and f_y and $f_{x+y}(0) = f_x(0) + f_y(0) = x + y$. To reconstruct an output z, all parties broadcast their share $f_z(i)$, and everyone interpolates the polynomial.

Multiplication gates pose more of a challenge. If a party P_i computes $f'_{xy}(i) = f_x(i)f_y(i)$, she gets a point on a polynomial f'_{xy} such that $f'_{xy}(0) = xy$, which is what we wanted. The caveat is that f'_{xy} is of degree $2d$; if the degree keeps growing in this way, it will be too high to admit interpolation given only $n - t$ points (which is necessary if we would like to withstand fail-stop corruptions). Additional work needs to be done to reduce the degree of this polynomial: $2d+1$ parties P_i need to reshare their points $f'_{xy}(i)$ using a new d-degree polynomial (that is, party P_i will pick a random polynomial r_i of degree t such that $r_i(0) = f'_{xy}(i)$ and send $r_i(j)$ to all other parties P_j). Each party P_j can locally compute $f_{xy}(j) = \sum_{i \in \mathcal{R}} \lambda_i r_i(j)$ where the λ_i's are the appropriate Lagrange coefficients.

Threshold Requirements Now, consider the (t, h^*)-FaF setting. In order to withstand fail-stop corruptions, even during degree reduction, we need $2d < n - t$. In order to have FaF security, we need $t + h^* \leq d$. We thus need

$$t + h^* < \frac{n - t}{2}$$

$$\Rightarrow 3t + 2h^* < n,$$

which is not optimal.

Brief Overview of BGW With Preprocessing In order to approach the optimal threshold, we change the way we do multiplication to rely on *Beaver triple pre-processing* [Bea92]. We give each party shares $f_a(i), f_b(i), f_c(i)$ of a randomly chosen a and b, and of their product $c = ab$. We use these shares in order to multiply x and y as follows. Each party P_i computes $f_\delta(i) = f_x(i) - f_a(i)$ and $f_\epsilon(i) = f_y(i) - f_b(i)$ and broadcasts these values. All parties reconstruct $\delta = x - a$ and $\epsilon = y - b$, and compute $f_{xy}(i) = f_c(i) + \epsilon f_x(i) + \delta f_y(i) - \delta\epsilon$. We can see that f_{xy} has the same degree d, and that $f_{xy}(0) = c + \epsilon x + \delta y - \delta\epsilon = c + (y-b)x + (x-a)y - (x-a)(y-b) = c + xy - xb + yx - ya - xy + xb + ya - ab = xy$, because $c = ab$.

Threshold Requirements As in Section 6.1, in order to have FaF security, we need $t + h^* \leq d$. However, in order to withstand fail-stop corruptions, now we only need $d < n - t$. Putting these together, we need $t + h^* < n - t \Rightarrow 2t + h^* < n$, which is optimal.

Adaptive Security of BGW With Preprocessing We now prove that BGW with Preprocessing with $2t + h^* < n$ and with $d = t + h^*$ is adaptively secure.

Theorem 8. *The construction summarized in Section 6.1 with $2t + h^* < n$ and with $d = t + h^*$ achieves security with guaranteed output delivery against an adaptive adversary with a budget of t fail-stop corruptions and h^* passive corruptions.*

We follow the blueprint of Damgård and Nielsen [DN14] for proving the adaptive security of BGW. We start by describing a simulator $\mathcal{S}_{static}$ for a static adversary, who is given the inputs of the passive and the fail-stop corrupted parties (which are similar to passive corruptions except that the adversary can choose to abort them at any step in the real world and substitute their input with a default input in the ideal world). $\mathcal{S}_{static}$ interacts with the adversary on behalf of the set $\mathcal{H}$ of the $n - t - h^*$ honest parties. She shares the input 0 on behalf of honest parties $P_i \in \mathcal{H}$. If a fail-stop party P_i fails to share an input, that party implicitly gives the degree 0 sharing of 0, where every share is 0 (thereby, its default input can be considered as 0). $\mathcal{S}_{static}$ forwards these inputs to the ideal functionality to obtain the output. $\mathcal{S}_{static}$ follows the protocol on behalf of the honest parties up until it's time to reconstruct the output. When it's time to reconstruct the output, $\mathcal{S}_{static}$ computes the difference $\delta = z - z'$, where z' is the computed output (shared on the polynomial $f_{z'}$), and z is the output dictated by the ideal functionality. $\mathcal{S}_{static}$ chooses a random polynomial Δ of degree d such that $\Delta(0) = \delta$ and $\Delta(i) = 0$ for all $i \in \mathcal{I}$. Notice that $f_{z'} + \Delta$ is a sharing f_z of the desired output $z = z' + \delta$. $\mathcal{S}_{static}$ uses shares $f_z(i)$ on behalf of honest parties $P_i \in \mathcal{H}$. (We make use of an assumption that the output z is produced by a multiplication gate. This forces the polynomial f_z used for the output to be a random degree d polynomial with the only constraint being that $f_z(0) = z$.)

We now describe the simulator $\mathcal{S}$ for an adaptive adversary. $\mathcal{S}$ starts out much like $\mathcal{S}_{static}$, by interacting with the adversary on behalf of the initial set $\mathcal{H}$ of honest parties, and maintaining a record of their views (including their shares of all intermediate values). However, unlike $\mathcal{S}_{static}$, at any point $\mathcal{S}$ may be asked to explain the view of one of these honest parties P_i, in the event that P_i becomes corrupt. It is insufficient for $\mathcal{S}$ to hand over the current simulated view of P_i, since $\mathcal{S}$ used the input 0 on behalf of P_i; this makes the simulated view clearly distinguishable from the real view, where the real input would have been used. So, $\mathcal{S}$ must adjusts the shares she has stored to account for the use of the real input. Upon the corruption of party P_i, the simulator makes the following adjustments:

Shares of Input x_i: When party P_i is corrupted, $\mathcal{S}$ learns the real input x_i. Let $\mathcal{H}$ be the set of parties who were honest before the corruption of party P_i, and $\mathcal{I}$ be the set of parties who were corrupt. Note that $|\mathcal{I}| < t + h^* = d$; so, the views of the parties in $\mathcal{I}$ contain no information about $\{f_{x_i}(j)\}_{j\in\mathcal{H}}$, even if she knew x_i. $\mathcal{S}$ cannot change the shares of parties in $\mathcal{I}$, so she picks a random difference polynomial Δ s.t. $\Delta(0) = x_i$ and $\Delta(i) = 0$ for $i \in \mathcal{I}$. $\mathcal{S}$ updates the sharing polynomial as $f_{x_i} := f_{x_i} + \Delta$.

Addition or Multiplication by a Constant: In this case we only have local computation, so $\mathcal{S}$ simply recomputes the shares of the honest parties that were affected by the change to f_{x_i}.

Multiplication: Recall that for the multiplication of x by y each party j has published her share of δ, which is $f_\delta(j) = f_x(j) - f_a(j)$, and her share ϵ, which is $f_\epsilon(j) = f_y(j) - f_b(j)$ (where a, b and c is the Beaver triple s.t. $c = ab$). We will consider only x, a and δ; the case for y, b and ϵ is analogous. Since the adversary already saw $\delta = x - a$, and x might have changed, $\mathcal{S}$ must adjust a accordingly. Let x_{old} be the old value of x (shared on $f_{x_{old}}$), and a_{old} be the old value of a (shared on $f_{a_{old}}$). $\mathcal{S}$ defines $a := a_{old} + (x - x_{old})$, and lets $f_a := f_{a_{old}} + f_x - f_{x_{old}}$. Observe that $\delta = x_{old} - a_{old} = x_{old} - (a - (x - x_{old})) = x_{old} - a + x - x_{old} = x - a$, as desired. The shares on f_a are now consistent with the shares of δ previously published.

Note that we have changed the values of a and b, but not the value of c; so, it may no longer be the case that $c = ab$. However, this is not a problem, since the view of the adversary has no information about c (as she has insufficient shares).

Output Reconstruction: For an output z, the adversary has already seen all of the points on f_z. So, now we need to fix P_i's view to be consistent with $f_z(i)$. Recall that we assume that the output is produced by a multiplication gate, which uses a Beaver triple a, b, c. Let c_{old} be the old value of c (shared on $f_{c_{old}}$). Let a (shared on f_a) and b (shared on f_b) be the rest of that Beaver triple, and let x (shared on f_x) and y (shared on f_y) be the two inputs to the multiplication gate. Recall that $\delta = x - a$ and $\epsilon = y - b$ are fixed. $\mathcal{S}$ defines $c := z - \Big(\epsilon x + \delta y - \delta\epsilon\Big)$, and $f_c := f_z - (\epsilon f_x + \delta f_y - \delta\epsilon)$. This is consistent with what the adversary has seen.

As before, it may no longer be the case that $c = ab$; however, this is not a problem, since the adversary will have seen too few shares to be able to tell.

6.2 Adaptive BGW Against Mixed (Active / Passive) Adversaries

We now discuss how the security of the construction in Section 6.1 can be boosted to achieve guaranteed output delivery against a mixed adaptive adversary controlling t parties *actively* and h^* parties passively. This can be done by using the generic compiler of Canetti *et. al* [CLOS02] that transforms a protocol secure against adaptive fail-stop corruptions to a protocol secure against adaptive active corruptions. At a high-level, this compiler follows the GMW compiler paradigm [GMW87] where the parties (a) run an augmented coin-tossing protocol to obtain their respective uniformly distributed random tapes and commitments to other parties' random tapes, (b) commit to their inputs, (c) run the underlying fail-stop adaptively secure MPC protocol, while proving in each round using zero-knowledge that the computations have been done correctly. We can use the same compiler to upgrade security of the construction in Section 6.1 (achieving adaptive security against t fail-stop corruptions and h^* passive corruptions) to adaptive security against t active corruptions and h^* passive corruptions with a minor simplification. Since our underlying protocol satisfies perfect correctness (i.e. the protocol results in the correct output when everyone executes the protocol steps honestly, irrespective of the choice of random tapes of the parties), the augmented coin-tossing protocol used to determine the random tapes of the parties in the compiler of Canetti *et. al* can be avoided. The rest of the compiler remains the same; it relies on adaptively secure commitment and zero-knowledge tools (which can be based on enhanced trapdoor permutations).

We argue that this simplified compiler preserves guaranteed output delivery. This is because, whenever an actively corrupt party misbehaves in the compiled protocol (for instance, a party aborts or the zero-knowledge proof showing the correctness of her actions in round r of the underlying protocol fails), such a scenario can be translated to an analogous scenario in the underlying protocol where the same party is fail-stop corrupt and stops communicating in round r. It is now easy to see that since the underlying protocol achieved GOD against t fail-stop and h^* passive corruptions, the same guarantees must hold against t active and h^* passive corruptions in the compiled protocol as well.

We state the formal theorem below.

Theorem 9. *The construction summarized in Section 6.2 with $2t + h^* < n$ and with $d = t + h^*$ achieves security with guaranteed output delivery against an adaptive adversary with a budget of t active corruptions and h^* passive corruptions.*

References

AJL+12. Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David

Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, Heidelberg, April 2012.

AOP20. Bar Alon, Eran Omri, and Anat Paskin-Cherniavsky. MPC with friends and foes. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 677–706. Springer, Heidelberg, August 2020.

Bea92. Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 420–432. Springer, Heidelberg, August 1992.

BGG+18. Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 565–596. Springer, Heidelberg, August 2018.

BGW88. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988.

CLOS02. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002.

DN14. Ivan Damgård and Jesper Buus Nielsen. Adaptive versus static security in the UC model. In Sherman S. M. Chow, Joseph K. Liu, Lucas C. K. Hui, and Siu-Ming Yiu, editors, *ProvSec 2014*, volume 8782 of *LNCS*, pages 10–28. Springer, Heidelberg, October 2014.

GLS15. S. Dov Gordon, Feng-Hao Liu, and Elaine Shi. Constant-round MPC with fairness and guarantee of output delivery. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 63–82. Springer, Heidelberg, August 2015.

GMW87. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

IKKP15. Yuval Ishai, Ranjit Kumaresan, Eyal Kushilevitz, and Anat Paskin-Cherniavsky. Secure computation with minimal interaction, revisited. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 359–378. Springer, Heidelberg, August 2015.

IKLP06. Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. On combining privacy with guaranteed output delivery in secure multiparty computation. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 483–500. Springer, Heidelberg, August 2006.

IKP10. Yuval Ishai, Eyal Kushilevitz, and Anat Paskin. Secure multiparty computation with minimal interaction. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 577–594. Springer, Heidelberg, August 2010.

Kat07. Jonathan Katz. On achieving the "best of both worlds" in secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 11–20. ACM Press, June 2007.

PR18. Arpita Patra and Divya Ravi. On the exact round complexity of secure three-party computation. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 425–458. Springer, Heidelberg, August 2018.

Yao86. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

A Decentralized Threshold FHE: Formal Definitions

In this appendix, we present the formal definitions of decentralized threshold FHE.

Correctness. A dTFHE scheme is *correct* if for all sufficiently large λ, all $k = \mathsf{poly}(\lambda)$, circuits $C : \mathcal{M}^k \to \mathcal{M}$ of depth at most κ and $m_i \in \mathcal{M}$ for $i \in [k]$, the following condition holds:
Let $(\mathtt{pk}_j, \mathtt{sk}_j) \leftarrow \mathsf{DistGen}(1^\lambda, 1^\kappa, j)$ for all $j \in [n]$, $\mathtt{pk} = (\mathtt{pk}_1 || \ldots || \mathtt{pk}_n)$; let $\mathsf{c}_i \leftarrow \mathsf{Enc}(\mathtt{pk}, m_i)$ for all $i \in [k]$; compute $\mathsf{c} \leftarrow \mathsf{Eval}(\mathtt{pk}, C, \mathsf{c}_1, \ldots, \mathsf{c}_k)$. For any $S \subseteq [n]$, $|S| > d$,

$$\mathsf{Pr}[\mathsf{Combine}(\mathtt{pk}, \{\mathsf{PDec}(\mathtt{pk}, \mathtt{sk}_j, \mathsf{c})\}_{j \in S}) = C(m_1, \ldots, m_k)] \geq 1 - \mathsf{negl}(\lambda).$$

Semantic Security. A dTFHE scheme is *semantically secure* if for all sufficiently large security parameters λ, all depth bound κ and any PPT semi-malicious adversary $\mathcal{A}$, there exists a negligible function negl such that the probability that $\mathcal{A}$ wins the game below is less than $\frac{1}{2} + \mathsf{negl}(\lambda)$.

Adversary $\mathcal{A}$		Challenger $\mathcal{C}$
$\mathcal{I} \subset [n], \lvert\mathcal{I}\rvert \leq d$		
	$\xrightarrow{\mathcal{I}, \{\rho_i\}_{i \in \mathcal{I}}}$	
	$\xrightarrow{(\mathrm{m}_0, \mathrm{m}_1) \in \mathcal{M}^2}$	
		for $i \in \mathcal{I}$:
		$\quad(\mathtt{pk}_i, \mathtt{sk}_i) \leftarrow \mathsf{DistGen}(1^\lambda, 1^\kappa, i; \rho_i)$
		for $i \in [n] \setminus \mathcal{I}$:
		$\quad$sample ρ_i
		$\quad(\mathtt{pk}_i, \mathtt{sk}_i) \leftarrow \mathsf{DistGen}(1^\lambda, 1^\kappa, i; \rho_i)$
		Set $\mathtt{pk} = (\mathtt{pk}_1 \parallel \mathtt{pk}_2 \parallel \ldots \parallel \mathtt{pk}_n)$
		Sample a bit $b \leftarrow \{0,1\}$
		$\mathsf{c} \leftarrow \mathsf{Enc}(\mathtt{pk}, m_b)$
	$\xleftarrow{\mathtt{pk}, \mathsf{c}, \{\mathtt{sk}_i\}_{i \in \mathcal{I}}}$	
	$\xrightarrow{b\prime}$	
$\mathcal{A}$ wins if $b = b\prime$		

Simulation Security. A dTFHE scheme satisfies *simulation security* if there exists a simulator $\mathsf{SimPDec}$ such that for all sufficiently large security parameters λ, all depth bound κ, and any PPT semi-malicious adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins the game below is less than $\frac{1}{2} + \mathsf{negl}(\lambda)$.

Adversary $\mathcal{A}$		Challenger $\mathcal{C}$
$\mathcal{I} \subset [n], \lvert\mathcal{I}\rvert \leq d$		
	$\mathcal{I}, \{\rho_i\}_{i\in\mathcal{I}}$	
	$(\mathrm{m}_1, \ldots, \mathrm{m}_\mathrm{k}) \in \mathcal{M}^\mathrm{k}$ $\longrightarrow$	
	$(\rho'_1, \ldots, \rho'_k)$	
		for $i \in \mathcal{I}$:
		$(\mathtt{pk}_i, \mathtt{sk}_i) \leftarrow \mathsf{DistGen}(1^\lambda, 1^\kappa, i; \rho_i)$
		for $i \in [n] \setminus \mathcal{I}$:
		sample ρ_i
		$(\mathtt{pk}_i, \mathtt{sk}_i) \leftarrow \mathsf{DistGen}(1^\lambda, 1^\kappa, i; \rho_i)$
		Set $\mathtt{pk} = (\mathtt{pk}_1 \parallel \mathtt{pk}_2 \parallel \ldots \parallel \mathtt{pk}_n)$
		$\mathsf{c}_i \leftarrow \mathsf{Enc}\,(\mathtt{pk}, m_i; \rho'_i)\ \forall i \in [k]$
	$\mathtt{pk}, \{\mathsf{c_i}\}_{\mathrm{i}\in[\mathrm{k}]}$ $\longleftarrow$	
	$\mathrm{C} : \mathcal{M}^\mathrm{k} \rightarrow \mathcal{M}$ $\longrightarrow$	
		$\mathsf{c} \leftarrow \mathsf{Eval}\,(\mathtt{pk}, C, \mathsf{c}_1, \ldots, \mathsf{c}_k)$
		Sample a bit $b \leftarrow \{0,1\}$
		if $b = 0$:
		for $i \in [n] \setminus \mathcal{I}$:
		$\mathtt{d}_i \leftarrow \mathsf{PDec}\,(\mathtt{pk}, \mathtt{sk}_i, \mathsf{c})$
		if $b = 1$:
		$z \leftarrow C(m_1, \ldots, m_k)$
		$\{\mathtt{d}_j\}_{j\in[n]\setminus\mathcal{I}} \leftarrow \mathsf{SimPDec}\,(\mathsf{c}, \mathtt{pk}, \{\mathtt{sk}_i\}_{i\in\mathcal{I}}, z)$
	$\{\mathtt{d_j}\}_{\mathrm{j}\in[\mathrm{n}]\setminus\mathcal{I}}$ $\longleftarrow$	
	$\mathrm{b}\prime$ $\longrightarrow$	
	$\mathcal{A}$ wins if $b = b\prime$	