

# Threshold Multi-Key FHE and Applications to MPC

Saikrishna Badrinarayanan\*    Aayush Jain\*    Nathan Manohar\*    Amit Sahai\*

February 27, 2019

## Abstract

In a multi-key FHE scheme (MFHE), first introduced by Lopez-Alt et. al. (STOC '12), and constructed by Clear-McGoldrick (CRYPTO '15) and Mukherjee-Wichs (EUROCRYPT '16), any message encrypted using a public key  $pk_i$  can be “expanded” so that the resulting ciphertext is encrypted with respect to a set of public keys  $(pk_1, \dots, pk_n)$ . Such expanded ciphertexts can be homomorphically evaluated with respect to any circuit to generate a ciphertext  $ct$ . Then, this ciphertext  $ct$  can be partially decrypted using a secret key  $sk_i$  (corresponding to the public key  $pk_i$ ) to produce a partial decryption  $p_i$ . Finally, these partial decryptions  $\{p_i\}_{i \in [n]}$  can be combined to recover the output.

However, this definition of MFHE works only for  $n$ -out-of- $n$  access structures and thus each node in the system is a point of failure. In some cases, it may be useful to be able to decrypt even when one only possesses a subset of partial decryptions (say  $t$  out of  $n$ ). In order to solve this problem, we introduce a new notion of multi-key FHE designed to handle arbitrary access patterns that can reconstruct the output. We call it a threshold multi-key FHE scheme (TMFHE). We give a formal definition and present a construction for any access structure given by a monotone boolean formula, assuming LWE. Using TMFHE, we present two new results for MPC:

- We revisit the mixed adversary model proposed by Fitzi et. al. (CRYPTO '98, ASIACRYPT '99) and present a three-round MPC protocol in this model.
- We also achieve independently interesting consequences in the traditional MPC model. We construct the *first round-optimal* (three rounds) MPC protocol in the *plain model* (without a CRS) that achieves *guaranteed output delivery* (GOD) and is malicious-secure in the presence of an honest majority of parties. Previously, Gordon et al. [CRYPTO' 15] constructed a three-round protocol that achieves GOD in the honest majority setting, but required a CRS. They also showed that it is impossible to construct two-round protocols for the same even in the presence of a CRS. Thus, our result is the first three-round GOD protocol that does not require any trusted setup, and it is optimal in terms of round complexity.

Furthermore, all of our above protocols have communication complexity proportional only to the depth of the circuit being evaluated and are reusable.

---

\*UCLA and Center for Encrypted Functionalities. {saikrishna, aayushjain, nmanohar, sahai}@cs.ucla.edu. Research supported in part from a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1619348, 1228984, 1136174, and 1065276, BSF grant 2012378, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the ARL under Contract W911NF-15-C-0205. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

# 1 Introduction

Traditionally, encryption has been seen as a way to ensure confidentiality of a communication channel between a unique sender and a unique receiver. However, with the emergence of cloud computing, the notion of homomorphic encryption has seen tremendous growth. Starting with the breakthrough work of Gentry [Gen09], Fully Homomorphic Encryption (FHE) has been extensively studied over a long sequence of works [Gen09, BV11b, BV11a, BGV12, GSW13]. In an FHE scheme, given a public key  $pk$  and a ciphertext of a message  $m$  encrypted using this public key, a user can homomorphically evaluate this ciphertext with respect to any circuit  $C$  to generate a new ciphertext  $ct$  that is an encryption of  $C(m)$  without learning anything about the message. Then, the decryptor, using the secret key  $sk$  can decrypt this message to recover the output  $C(m)$ . FHE schemes and the techniques used to build them have given rise to several surprising applications in cryptography such as in delegation [KRR14], multilinear maps [GGH13a], cryptographic obfuscation [GGH<sup>+</sup>13b] and function secret sharing [BGG<sup>+</sup>17]. However, traditionally, FHE schemes are single-key in nature: that is, they can be used to perform arbitrary computation on data encrypted using the same public key.

**Multi-Key FHE.** Lopez-Alt et al. [LTV12] introduced the notion of Multi-Key Fully Homomorphic Encryption. Informally, in a multi-key FHE scheme, any message encrypted using a public key  $pk_i$  can be “expanded” so that the resulting ciphertext is encrypted with respect to a set of public keys  $(pk_1, \dots, pk_n)$ . Such expanded ciphertexts can be homomorphically evaluated with respect to any circuit to generate a ciphertext  $ct$ . Then, this ciphertext  $ct$  can be partially decrypted using a secret key  $sk_i$  (corresponding to the public key  $pk_i$ ) to produce a partial decryption  $p_i$ . Finally, these partial decryptions  $\{p_i\}_{i \in [n]}$  can be combined to recover the output. In addition to semantic security of encryption, a multi-key FHE scheme also requires that given any expanded (and possibly evaluated) ciphertext  $ct$  encrypting a message  $m$ , any set of  $(n - 1)$  secret keys  $\{sk_i\}_{i \neq i^*}$  for any  $i^*$ , and the message  $m$ , it is possible to statistically simulate the partial decryption  $p_{i^*}$ . Multi-key FHE has been extensively studied [CM15, MW16, PS16, BHP17] and has proven particularly useful in the context of building round efficient secure multiparty computation protocols.

**Threshold Multi-Key FHE.** However, none of the existing multi-key FHE schemes enable the output to be reconstructed unless all the  $n$  partial decryptions are given out and hence they only “work” for  $n$ -out-of- $n$  access structures. However, in some cases, it may be useful to be able to decrypt even when one only possesses a subset of partial decryptions (say  $t$  out of  $n$ ). In order to solve this problem, we introduce a new notion of multi-key FHE designed to handle arbitrary access patterns that can reconstruct the output. We call this new notion as Threshold Multi-Key FHE.<sup>1</sup>

At first glance, it is not even clear how to define such a notion. The most direct approach leads to a definition that is impossible to achieve. Consider for example the  $n/2$ -out-of- $n$  access structure. In this case, any evaluator can expand a ciphertext encrypting a message  $m$  with respect to public key  $pk_n$  to a ciphertext  $ct$  with respect to the set of public keys  $(pk_1, \dots, pk_n)$ . Then, the evaluator can use secret keys  $sk_1, \dots, sk_{n/2}$  to learn the value of  $m$ , as the set  $\{1 \dots, n/2\}$  satisfies the access structure. However, in doing so, an adversary can learn  $m$  without knowing  $sk_n$ , breaking

---

<sup>1</sup>We remark that in fact, some existing standard multi-key FHE schemes [MW16] also interchangeably used the term threshold multi-key FHE for their primitive. We overload this term here to denote our stronger notion.

the semantic security of the encryption scheme with respect to  $(pk_n, sk_n)$  and leading to a notion that provides no security.

Although we seem to have arrived at a notion that is not meaningful at all, we note that the issue with the above approach is that a ciphertext encrypted with respect to a public key  $pk$  can be expanded to one encrypted with respect to many public keys. However, if we prevent ciphertexts from being expanded, there is hope of achieving a meaningful notion. Expanding on this idea, we arrive at the following (informal) definition. Any party can generate its own key pair  $(pk, sk)$ . Any encryptor can compute  $ct \leftarrow \text{Encrypt}(pk_1, \dots, pk_n, \mathbb{A}, m)$ . Given two (or more) ciphertexts encrypted with respect to the same set of public keys and the same access structure  $\mathbb{A}$ , it is possible to homomorphically evaluate a circuit on these ciphertexts and partially decrypt the resulting ciphertext using any secret key  $sk_i$  to recover a partial decryption  $p_i$ . Given  $\{p_i\}_{i \in B}$  for some  $B$  satisfying  $\mathbb{A}$ , one can reconstruct the output. Roughly, we require two security guarantees from the scheme.

1. Given  $\{sk_i\}_{i \in S}$  for some  $S \notin \mathbb{A}$ ,

$$\text{Encrypt}(pk_1, \dots, pk_n, \mathbb{A}, m_0) \approx_c \text{Encrypt}(pk_1, \dots, pk_n, \mathbb{A}, m_1)$$

for any two equal length messages  $m_0, m_1$ .

2. Given a ciphertext  $ct$  for an underlying message  $m$  and  $\{sk_i\}_{i \in S}$  for any maximally unqualified set<sup>2</sup>  $S \notin \mathbb{A}$  (for example  $(n/2 - 1)$  of the parties for the example above), it is possible to statistically simulate a partial decryption  $p_i$  for any  $i \in [n]$ .

For technical reasons, we require a more nuanced security definition, and we refer the reader to [Section 4](#) for the details. In this work, we first formally define this new notion of Threshold Multi-Key FHE and then show to construct the primitive from the Learning with Errors (LWE) assumption. Formally, we show the following theorem:

**Theorem 1** (Informal). *Assuming LWE, there exists a secure threshold multi-key FHE scheme for the class of access structures  $\mathbb{A}$  induced by a monotone boolean formula.*

In [Section 2](#), we describe the techniques used in our construction. Our next contribution is two applications of threshold multi-key FHE in the context of MPC.

## 1.1 Applications to round efficient MPC

Secure multi-party computation (MPC) [[Yao82](#), [Yao86](#), [GMW87](#)] has been a problem of fundamental interest in cryptography. In an MPC protocol, a set of mutually distrusting parties can evaluate a function on their joint inputs while maintaining privacy of their respective inputs. Over the last few decades, much of the work related to MPC has been devoted to achieving stronger security guarantees and improving efficiency with respect to various parameters such as round complexity and communication complexity. In this work, we further advance our understanding of this landscape with threshold multi-key FHE being the main technical tool.

---

<sup>2</sup>By maximally unqualified set  $S$ , we mean that for any  $i \in [n] \setminus S$ ,  $(S \cup \{i\}) \in \mathbb{A}$ . Similarly, a set  $S$  is minimally qualified if for any  $i \in [n]$ ,  $(S \setminus \{i\}) \notin \mathbb{A}$ .

**MPC with General Mixed Adversaries.** In traditional MPC, every party is required to remain online and participate completely in the protocol execution. This applies not only to “classical” MPC protocols where every party has to participate and send a message in every round of the protocol, but also to other interesting variants such as protocols in the client-server setting where all the servers are required to remain active until the end of the protocol execution. We refer the reader to [Section 1.2](#) for a more detailed comparison with related works. In other words, traditional MPC protocols decide to treat a “lazy” party that just aborts midway into the protocol execution as a corrupt party that is colluding with the other corrupt parties, and this is addressed in different ways. In some cases, all parties abort the protocol execution while in other cases, the “lazy” party is just discarded and all the other parties compute the function on their joint inputs alone. We believe that such an outlook is undesirable as there are several reasons why even an honest party might have to abort and turn “lazy” during the execution of a protocol without having to be deemed as colluding with the corrupt parties. A few potential reasons include:

- Connectivity - A party might lose connectivity and hence be unable to continue the protocol.
- Computational resources - A computationally weak party might be unable to perform intensive computation and hence be forced to exit the protocol.
- Interest - At some point, a party might just lose interest in that protocol execution due to other higher priority tasks that come up.

Motivated by the above realistic scenario, we study MPC in the presence of general mixed adversaries introduced by Fitzi et al. [[FHM98](#), [FHM99](#)]. In this setting, a general mixed adversary  $\mathcal{A}$  is allowed to corrupt three sets of parties ( $\mathcal{A}_{\text{Mal}}$ ,  $\mathcal{A}_{\text{Sh}}$ ,  $\mathcal{A}_{\text{Fc}}$ ) such that the following holds: (i) The set of parties in  $\mathcal{A}_{\text{Mal}}$  are maliciously corrupted meaning that the adversary can choose to behave using any arbitrary polynomial time algorithm on behalf of each of them. (ii) The set of parties in  $\mathcal{A}_{\text{Sh}}$  are corrupted in a semi-honest manner and so the adversary is required to follow the protocol execution honestly on behalf of each of them. (iii) The set of parties in  $\mathcal{A}_{\text{Fc}}$  are corrupted in a fail-corrupt manner meaning that for each party in this set, the adversary can specify when that party is required to abort the protocol execution. Till then, these parties follow the protocol execution honestly. Note that the adversary never gets to see the inputs or internal state of any of the fail-corrupt parties and hence these parties capture our motivation of “honest but lazy” parties - where their laziness is enforced by the adversary in the security game. We then construct an MPC protocol that is secure against such a general mixed adversary. As a stepping stone to that, we first consider the weaker setting of a semi-malicious mixed adversary that corrupts the sets ( $\mathcal{A}_{\text{Sm}}$ ,  $\mathcal{A}_{\text{Sh}}$ ,  $\mathcal{A}_{\text{Fc}}$ ) of parties such that the first set of parties  $\mathcal{A}_{\text{Sm}}$  is only corrupted in a semi-malicious manner - that is, on behalf of each party in this set, the adversary can pick any arbitrary randomness of its choice but using this randomness, the party is required to execute the protocol honestly. We define this formally in the technical sections.

In this work, our goal is to build round efficient MPC protocols in this model. Prior to our work, much of the focus in this model was to obtain feasibility results, understand under what corruption patterns is secure computation even possible and to improve the communication complexity. To the best of our understanding, the exact round complexity of MPC protocols in this model has not been studied so far. We refer to [Section 1.2](#) for a more detailed discussion on the prior work in this model. In this work, we initiate the study of round-efficient MPC in the mixed adversary model by constructing a three round MPC protocol in this setting. Formally, we show the following results:

**Theorem 2** (Informal). *Assuming learning with errors (LWE), for any function  $f$ , for any access structure  $\mathbb{A}$  induced by a monotone boolean formula, there exists a three-round MPC protocol in the plain model that is secure against a semi-malicious mixed adversary  $\mathcal{A} = (\mathcal{A}_{Sm}, \mathcal{A}_{Sh}, \mathcal{A}_{Fc})$  such that  $(\mathcal{A}_{Sm} \cup \mathcal{A}_{Sh}) \notin \mathbb{A}$ .*

We bootstrap the above protocol to be secure against general mixed adversaries by further assuming UC-secure non-interactive zero knowledge arguments (NIZKs) in the CRS model. We know that UC-secure NIZKs can be constructed based on the existence of Zaps [GOS12]. Formally, we get the following theorem:

**Theorem 3** (Informal). *Assuming LWE and Zaps, for any function  $f$ , for any access structure  $\mathbb{A}$  induced by a monotone boolean formula, there exists a three-round MPC protocol in the CRS model that is secure against a general mixed adversary  $\mathcal{A} = (\mathcal{A}_{Mal}, \mathcal{A}_{Sh}, \mathcal{A}_{Fc})$  such that  $(\mathcal{A}_{Mal} \cup \mathcal{A}_{Sh}) \notin \mathbb{A}$ .*

Hirt et al. [HMZ08] showed that in the setting of a general mixed adversary, MPC is possible if and only if  $(2 \times |\mathcal{A}_{Mal}| + |\mathcal{A}_{Sh}| + |\mathcal{A}_{Fc}|) < N$ . Thus, by appropriately setting our access structure  $\mathbb{A}$  to satisfy this constraint, our resulting protocol is both *optimal* in terms of the best possible corruption we can tolerate and also round efficient.

Also, note that if we set the access structure  $\mathbb{A}$  to be the  $N$  out of  $N$  access structure, this will give us a protocol in the standard MPC model against a malicious adversary that can corrupt up to all but one of the parties.

Further, both the above protocols satisfy the following additional salient features:

- **Depth-Proportional Communication Complexity:** For any function  $f$ , the communication complexity of the protocol is  $\text{poly}(\lambda, d, N, \ell_{\text{inp}})$  where  $N$  is the number of parties,  $\lambda$  is the security parameter,  $\ell_{\text{inp}}$  is the input length for each party,  $d$  is the depth of the circuit computing  $f$ .
- **Reusability:** The computation phase of both protocols consists of only a single round and can be reused across an unbounded polynomial number of executions to compute different functions on the same fixed joint inputs of all the parties.
- **Robustness:** Additionally, our protocol enjoys the property that the output of the computation is a function of the joint inputs of all parties, including those that aborted after a “certain point” - our protocol is divided into two phases - an input commitment phase and a computation phase and we refer to the end of the input commitment phase as this point. That is, in the scenario where the adversary corrupts a set of parties in a fail-corrupt manner, for every fail-corrupt party  $P_i$  that aborts after the input commitment phase, its input  $y_i$  that is used to compute the final output  $C(y_1, \dots, y_n)$  is set to be its actual input  $x_i$  used in the protocol so far and not a default input  $\perp$ . Recall that this is in line with our original motivation where we wish to not discard lazy but honest parties and deem them to be corrupt.

**Traditional MPC: Guaranteed Output Delivery.** Guaranteed output delivery is a fundamental property in the study of secure multiparty computation. Informally, a protocol is said to possess this property if, at the end of the protocol execution, no matter the behavior of the adversary, the honest parties can still compute the output of the function on all their joint inputs

(with either a default or the actual input for each of the malicious parties). In our next application, we focus on the problem of constructing malicious-secure MPC protocols in the standard model that achieve guaranteed output delivery (and hence fairness). Several works over the last few decades [BOGW88, Cle86, GIKR02, DI05, AJLA<sup>+</sup>12, CL14, GLS15] have studied this problem and we refer to the related work section for a more detailed list. Cleve [Cle86] showed that we cannot construct fair MPC protocols unless there are an honest majority of parties. Therefore, we restrict attention to this scenario. Gordon et al. [GLS15] constructed a three-round MPC protocol in the CRS model that achieves guaranteed output delivery in the presence of an honest majority. Further, they showed that even in the CRS model, a two-round protocol for the same is impossible. In this paper, we answer the following open problem:

*“Can we construct a three-round MPC protocol in the plain model that achieves guaranteed output delivery in the presence of an honest majority?”*

Formally, we show the following theorem:

**Theorem 4** (Informal). *Assuming  $LWE$  and  $Zaps$ , for any function  $f$ , there exists a three-round malicious-secure MPC protocol in the plain model that achieves guaranteed output delivery in the presence of an honest majority.*

Thus, our result is optimal in terms of the number of rounds and the absence of a trusted setup.

Additionally, as in the previous section, our protocol also allows for reusability of the third round to compute several different functions and the communication complexity of the protocol is only proportional to the depth of the function being computed. We remark that the protocol of Gordon et al. [GLS15] also satisfies depth-proportional communication complexity but is not reusable.

## 1.2 Related Work

**Client-Server MPC.** Secure computation in the client-server setting has been a widely studied problem [FKN94, IK97, NPS99, DI05, BCD<sup>+</sup>09, IKP10, KMR11, CKKC13]. The key differences from our model are the following: (i) in a client server setting, the identity of the server/servers and clients are decided a priori. As a result, the parties who perform the computation (the servers) are decided in advance while in our setting, any set of “non-lazy” parties can run the computation phase. (ii) In the client server model, all the clients can essentially turn “lazy” after submitting their messages to the server but we typically crucially require all the servers to take part in the computation to receive meaningful output. Once again, this is different from our setting.

**Dishonest Majority MPC in the Plain Model.** A long sequence of works constructed constant-round MPC protocols against dishonest majority based on a variety of assumptions and techniques (see, e.g., [KO04, Pas04, PW10, Wee10, Goy11, GMPP16, ACJ17, BHP17, COSV17a, COSV17b] [BGI<sup>+</sup>, JKKR17, BGJ<sup>+</sup>17, GKP17, BGJ<sup>+</sup>, HHPV17, BL18]). We stress that while the exact round complexity of MPC in the dishonest majority setting has been extensively studied, it is not clear or analyzed whether any of these protocols are also secure in the more general framework of a general mixed adversary.

**MPC with General Mixed Adversaries.** Fitzi et al. [FHM98] introduced the notion of MPC in the presence of a general mixed adversary. Starting with their work, a series of papers [FHM98, FHM99, Hir01, IKLP06, BFH<sup>+</sup>08, HMZ08, ZHM09, SPCR09, Zik10] studied and established lower bounds for corruption patterns under which MPC is feasible. Another line of work [CPA<sup>+</sup>08, HT13, LO14] was focused on improving the communication complexity of MPC protocols for various functionalities in the general mixed adversary setting with various corruption patterns.

**MPC with Guaranteed Output Delivery.** There have been a variety of prior works regarding MPC with guaranteed output delivery and/or fairness in the broadcast model. [BOGW88] constructed MPC protocols with fairness, and [CL14] studied the relationship between fairness and guaranteed output delivery in MPC protocols. There have also been a variety of works constructing MPC protocols with guaranteed output delivery. [DI05] constructed a three-round MPC protocol with guaranteed output delivery that is secure against an adversary that can corrupt less than one fifth of the parties. [AJLA<sup>+</sup>12] constructed five-round MPC protocols with guaranteed output delivery secure against an adversary that corrupts a minority of parties from LWE and NIZKs. Subsequently, Gordon et. al [GLS15] constructed a three-round MPC protocol with guaranteed output delivery in the CRS model from LWE and NIZKs. Furthermore, [GLS15] showed that achieving guaranteed output delivery in two rounds, even in the CRS model, is impossible. This built upon a previous result [GIKR02] that had ruled out such protocols in the plain model when the adversary can corrupt more than a single party.

**Concurrent Work.** Recently, in an independent and concurrent work, Ananth et. al [ACGJ18] also constructed a three-round MPC protocol with guaranteed output delivery in the plain model. The techniques differ from ours, and their construction is from PKE and Zaps, while ours is from LWE and Zaps. However, our construction also has depth-proportional communication and is reusable, properties not present in their scheme.

## 2 Technical Overview

We first describe the challenges involved in defining and constructing our new primitive of threshold multi-key FHE in the next subsection. This is followed by the techniques involved in the applications to round-efficient MPC protocols in the subsequent subsections.

### 2.1 Threshold Multi-Key FHE (TMFHE)

In order to construct TMFHE, one could try many approaches to build on top of existing multi-key FHE schemes. For example, one could try the following. Given any set of public keys  $(pk_1, \dots, pk_n)$ , generate ciphertexts  $ct_S \leftarrow \text{Encrypt}(\{pk_i\}_{i \in S}, m)$  for all minimally valid sets  $S \in \mathbb{A}$ . However, such an approach is not feasible for access structures such as  $n/2$ -out-of- $n$  as then the encryptor has to compute encryptions for roughly  $\binom{n}{n/2}$  subsets, which is super-polynomial.

To overcome this limitation, we use the tool of threshold FHE introduced in the work of Boneh et al. [BGG<sup>+</sup>17]. In a threshold FHE scheme, the setup algorithm samples a single public key  $\text{fpk}$  and  $n$  secret key shares  $(\text{fsk}_1, \dots, \text{fsk}_n)$  for a secret key  $\text{fsk}$  that are shared according to the access structure  $\mathbb{A}$ . Using the public key  $\text{fpk}$ , an encryptor can encrypt a message  $m$  to receive a ciphertext  $ct$  (which may be evaluated). This ciphertext can then be partially decrypted independently using



key shares  $sk_i$  to compute a partial decryption  $p_i$ . Then using these  $\{p_i\}_{i \in S}$  for any set  $S \in \mathbb{A}$ , one can recover  $m$ . Security properties are two fold:

- Given  $\{sk_i\}_{i \in S}$  for some  $S \notin \mathbb{A}$ ,  $\text{Encrypt}(pk, \mathbb{A}, m_0) \approx_c \text{Encrypt}(pk, \mathbb{A}, m_1)$  for any two equal length messages  $m_0, m_1$ .
- Second, given a ciphertext  $ct$  with underlying message  $m$  and  $\{sk_i\}_{i \in S}$  for any maximally unqualified  $S \notin \mathbb{A}$ , it is possible to statistically simulate partial decryptions  $p_i$  for any  $i \in [n]$ .

We make the following useful observations about threshold FHE which will aid us in our construction.

1. The setup algorithm of the scheme of [BGG<sup>+</sup>17] first samples  $(pk, sk) \leftarrow \text{FHE.Setup}(1^\lambda)$  and then secret shares  $sk$  according to the access structure using a “special purpose” secret sharing scheme to compute shares  $(sk_1, \dots, sk_n)$  so that the reconstruction involves just addition of some subset of shares. Looking ahead to the security proof, this feature allows us to easily simulate partial decryptions.
2. The encryption procedure just involves encrypting the message  $m$  using an underlying FHE scheme.
3. The underlying FHE scheme can be instantiated using most of the known homomorphic encryption schemes satisfying a few general properties.

Thus, we observe that, in particular, the multi-key FHE schemes of both [MW16, BHP17], can be used to instantiate the underlying FHE scheme in threshold FHE. This can then be used to evaluate on multiple ciphertexts encrypted with respect to different public keys - since, using multi-key FHE, one can expand on various ciphertexts and evaluate jointly on them. However, at this point, it is still not clear how to compute (or simulate) partial decryptions, especially since the threshold FHE construction of [BGG<sup>+</sup>17] only handled underlying FHE schemes where the ciphertext was encrypted with respect to a single public key. However, we observe the following property of the multi-key FHE schemes of both [MW16, BHP17]. Suppose we have two ciphertexts,  $ct_1$  and  $ct_2$  that are encrypted under public keys  $\text{fpk}_1$  and  $\text{fpk}_2$ , respectively. In the multi-key FHE scheme, we can expand these ciphertexts to  $\hat{ct}_1$  and  $\hat{ct}_2$ , each encrypted under the set of public keys  $\{\text{fpk}_1, \text{fpk}_2\}$ . If the secret keys corresponding to  $\text{fpk}_1$  and  $\text{fpk}_2$  are  $\text{fsk}_1$  and  $\text{fsk}_2$ , respectively, then the secret key for decryption of  $\hat{ct}_1$  and  $\hat{ct}_2$  (and any ciphertext computed by evaluating on these ciphertexts) is  $[\text{fsk}_1, \text{fsk}_2]$ . In a standard threshold FHE scheme, the secret key would be secret shared across  $n$  parties. For simplicity, assume that we secret share according to the  $n$  out of  $n$  access structure. Let party  $i$ 's shares of  $\text{fsk}_1$  and  $\text{fsk}_2$  be denoted by  $\text{fsk}_{1,i}$  and  $\text{fsk}_{2,i}$ , respectively. Since the decryption procedure of the multi-key FHE scheme is linear and the secret sharing of  $\text{fsk}_1$  and  $\text{fsk}_2$  is also linear and, crucially, with respect to the *same* access structure, one could have party  $i$  partially decrypt by running the decryption procedure of the multi-key FHE scheme using the secret key  $[\text{fsk}_{1,i}, \text{fsk}_{2,i}]$ . Given these partial decryptions, one could combine them to recover the message by adding them as specified by the reconstruction procedure of the secret sharing scheme.

The above gives intuition as to how one might construct threshold multi-key FHE, but several points are still unclear. In particular, we noted that in order to achieve a meaningful notion, we want an encryptor to encrypt with respect to a public key set and an access structure. The idea is that the public key set that an encryptor encrypts with respect to is *not* a public key set of



the underlying MFHE scheme, but rather simply a set of public keys for a public-key encryption scheme. These public keys serve as a means to send the corresponding multi-key FHE secret key shares to the other parties. At a high level, encryption works by generating a multi-key FHE public key  $\text{fpk}$  and secret key shares  $\text{fsk}_1, \dots, \text{fsk}_n$  corresponding to the access structure  $\mathbb{A}$ . The encryptor then encrypts  $\text{fsk}_i$  under  $pk_i$  and includes this in the ciphertext. This allows a set of parties satisfying the access structure to use their secret keys  $sk_i$  of the public-key scheme to recover the necessary  $\text{fsk}_i$ 's to decrypt the ciphertext. Furthermore, as we noted above, standard multi-key FHE expansion and evaluation will result in a ciphertext that can be decrypted by concatenating the secret key shares for each of the ciphertexts.

The above discussion is highly simplified and is meant to provide the reader with some intuition behind our construction. We ignored various subtle points and refer the reader to the main technical sections for the details. As a consequence of our techniques, we are able to directly simulate partial decryptions against an adversary that corrupts *any* set  $S \notin \mathbb{A}$ , not only a maximally unqualified one. The constructions of [MW16, BHP17] could only simulate against a maximally unqualified set ( $N - 1$  out of the  $N$  parties in their case) and relied on a transformation to achieve simulation security against any unqualified corrupted set.

## 2.2 MPC against General Mixed Adversaries

Recall that a general mixed adversary is one which corrupts three sets of parties ( $\mathcal{A}_{\text{Mal}}, \mathcal{A}_{\text{Sh}}, \mathcal{A}_{\text{Fc}}$ ) that behave as follows: the set of parties in  $\mathcal{A}_{\text{Mal}}$  are completely malicious and can behave arbitrarily as per the adversary's choice, the set of parties in  $\mathcal{A}_{\text{Sh}}$  are corrupted in a semi-honest manner meaning that they are required to follow the protocol behavior correctly and the set of parties in  $\mathcal{A}_{\text{Fc}}$  are corrupted in a fail-corrupt manner meaning that for each party in this set, the adversary can choose to abort the protocol execution at any point. Crucially, the adversary does not get to see the internal state of any fail-corrupt party. Intuitively, we can imagine these fail-corrupt parties as honest "lazy" parties whose aborting/laziness is controlled by the adversary. In this work, we focus on the setting of static corruptions where the adversary is required to specify all three sets a priori. Of course, note that for each fail-corrupt party, the adversary still has the luxury to determine adaptively when each party is expected to abort.

Our three-round MPC protocol secure against a general mixed adversary follows the same recipe as in the works of Mukherjee and Wichs [MW16] and Brakerski et al. [BHP17] who construct MPC protocols from multi-key FHE. We adapt it to instead use the underlying system as a threshold multi-key FHE scheme. Further, we will parameterize our protocol using an access structure  $\mathbb{A}$  which will be used to run the setup of the threshold multi-key FHE scheme. In our protocol,  $\mathbb{A}$  is used to identify/constrain the corruption pattern as follows. We will show that for any access structure  $\mathbb{A}$ , our protocol is secure against a general mixed adversary  $\mathcal{A} = (\mathcal{A}_{\text{Mal}}, \mathcal{A}_{\text{Sh}}, \mathcal{A}_{\text{Fc}})$  such that  $(\mathcal{A}_{\text{Mal}} \cup \mathcal{A}_{\text{Sh}}) \notin \mathbb{A}$ .

Let's first consider the simpler setting where the first set of corrupted parties  $\mathcal{A}_{\text{Mal}}$  can only be semi-malicious. That is, on behalf of each of them, the adversary can pick randomness of its choice but the parties are required to follow the protocol behavior honestly using this randomness. A more formal definition is given in [Appendix B](#). The overall structure of our MPC protocol with respect to any access structure is the following:

- In round 1, each party generates its parameters and public key for the threshold multi-key FHE scheme.

- In round 2, each party individually encrypts its input with respect to the combined set of public keys and access structure and broadcasts the ciphertext.
- All parties can now homomorphically compute a threshold multi-key FHE encryption of the output, with respect to the functionality under consideration. Then, each party broadcasts a partial decryption of the output using its secret key. The partial decryptions can be combined to recover the output in plaintext.

It can be readily observed from the definition of threshold multi-key FHE that this protocol satisfies correctness and security even in the presence of a general mixed adversary (with semi-malicious corruptions), where some lazy honest parties could drop off from the protocol execution at any point as determined by the fail-corrupt corruption.

One key difference from the previous works [MW16, BHP17] is the following: in the standard model MPC protocols of [MW16, BHP17], due to the design of the multi-key FHE primitive, the protocol is secure only against a semi-malicious adversary that corrupts all but one party. They then need to transform it to a protocol that is secure against an adversary that can corrupt any arbitrary number of parties up to all but one of them. In our MPC protocol, the security guarantee given by the threshold multi-key FHE scheme allows us to prove a more general statement that our protocol is in fact secure with respect to corruption of an arbitrary subset not satisfying the access pattern (given by any monotone boolean access structure) in addition to any arbitrary collection of fail-corrupt corruptions. In particular, if we consider an  $N$  out of  $N$  threshold access structure, our protocol (without an additional transformation) is secure against an adversary that can semi-maliciously corrupt any arbitrary subset of parties up to  $N - 1$  of them.

Finally, using the standard transformation of adding UC-secure non-interactive zero knowledge arguments (NIZKs), we achieve a MPC protocol secure even against general mixed adversaries with malicious corruptions in the common reference string (CRS) model.

### 2.3 Standard MPC: Guaranteed Output Delivery

First, note that our three-round MPC protocol secure against general mixed adversaries in the presence of a CRS is also secure in the standard MPC setting in the presence of an honest majority by considering an adversary that doesn't corrupt any party in a fail-corrupt manner and setting the access structure of the corruption pattern (and hence, of the underlying threshold multi-key FHE scheme) to be an honest majority threshold access structure. Further, observe that this protocol, in the CRS model, in the honest majority setting, already achieves guaranteed output delivery! The reason being that since the access structure is an  $\lfloor N/2 + 1 \rfloor$  out of  $N$  threshold access structure and there are no fail-corrupt parties, we have that all the honest parties together are in the access structure of the underlying threshold multi-key FHE scheme by themselves and hence can compute the output even if the adversary aborts midway. However, the big issue is that our protocol requires a CRS to compute the NIZK arguments and we need to get rid of that to achieve a plain model construction.

**Round One: Malicious.** To do so, the first crucial observation we make is that the underlying semi-malicious protocol (without a NIZK) in the plain model, is already in fact secure against an adversary that can behave maliciously only in the first round. The reason is that the first round message, which consists of the adversary's parameters for the threshold multi-key FHE scheme,

is simply a random matrix and a public key. To argue semi-malicious security, we only need the following two properties:

- The honest parties’ matrices are generated uniformly at random.<sup>3</sup>
- The simulator, before the beginning of round three of the protocol, only needs to know the randomness used by the adversary in the second round to generate its ciphertext. In particular, the simulator does not need to know a corresponding secret key for the public key sent by the adversary in round 1.

As a result, we did not require the input or randomness used by the adversary to generate its round one messages, and hence our protocol is secure against an adversary that can behave maliciously in round one.

**Multi-String NIZK.** Armed with the above property, we note that our protocol no longer needs to prove correctness of round one messages using a NIZK. Therefore, we will use the first round messages of all parties to try to collectively generate a valid CRS that can then be used to generate the NIZKs and achieve a construction in the plain model. The notion of multi-string NIZKs, introduced in the work of Groth and Ostrovsky [GO07] exactly fits this requirement. That is, in a multi-string NIZK argument system, a set of parties can each generate one CRS that can then be combined to compute one unified CRS which is used to compute NIZKs. The guarantee is that as long as an honest majority of the individual CRS strings are honestly generated, the argument system is correct and secure<sup>4</sup>

In our protocol, we can use this primitive as follows: in round 1, each party generates an individual CRS for the multi-string NIZK system. At the end of round 1, all parties can combine the above set of CRS strings to compute one unified CRS that can then be used to compute NIZKs. Security of the multi-string NIZK system holds since we work in the honest majority setting and thus we achieve a plain model construction.

### 3 Preliminaries

We denote the security parameter by  $\lambda$ . For an integer  $n \in \mathbb{N}$ , we use  $[n]$  to denote the set  $\{1, 2, \dots, n\}$ . We use  $\mathcal{D}_0 \cong_c \mathcal{D}_1$  to denote that two distributions  $\mathcal{D}_0, \mathcal{D}_1$  are computationally indistinguishable. We use  $\text{negl}(\lambda)$  to denote a function that is negligible in  $\lambda$ . We use  $x \leftarrow \mathcal{A}$  to denote that  $x$  is the output of a randomized algorithm  $\mathcal{A}$ , where the randomness of  $\mathcal{A}$  is sampled from the uniform distribution. We use PPT as an abbreviation for probabilistic polynomial-time. Whenever we write  $\{x_j\}_{j \in S}$  for a set of parties  $S$ , we assume that the party  $j$  that  $x_j$  corresponds to is included in  $x_j$ . When we say an error distribution is  $E$ -bounded, we mean that the errors are in  $[-E, E]$ .

#### 3.1 Multi-Key Fully Homomorphic Encryption

Multi-key FHE was first introduced in [LTV12] and then constructed by [CM15, MW16, BHP17]. We recall the definition of a multi-key FHE in Appendix A.1. Our definition is inspired from [BHP17].

---

<sup>3</sup>This was a wonderful observation made in the work of Brakerski et al. [BHP17].

<sup>4</sup>As is the case with compiling semi-malicious protocols into malicious secure ones, we need the NIZK to be zero-knowledge and simulation-extractable.

### 3.2 Multi-String NIZKs

In order to achieve malicious security in the plain model, we will make use of simulation-extractable multi-string NIZKs [GO07]. We recall the definition of this primitive in [Appendix A.2](#).

### 3.3 MPC with General Mixed Adversary

We formally define the notion of secure multiparty computation against a general mixed adversary as define in the works of [FHM99, FHM98] in [Appendix B](#).

### 3.4 Guaranteed Output Delivery

Consider  $N$  parties  $P_1, \dots, P_N$  each with inputs  $x_1, \dots, x_N$  respectively who wish to run an MPC protocol  $\pi$  to securely evaluate a function  $f$  on their joint inputs. The protocol  $\pi$  is said to possess the guaranteed output delivery property in the presence of a class of adversaries  $\mathbf{Adv}$ , if for all possible sets of inputs  $\{x_1, \dots, x_N\}$ , for any function  $f$ , the following holds: Let  $S$  denote the set of honest parties. At the end of the execution of  $\pi$ , no matter the behavior of the adversary, each honest party in  $S$  computes the same output  $f(y_1, \dots, y_n)$  where  $y_i = x_i$  for every honest party  $P_i$  and  $y_i = x_i/\perp$  for every corrupt party  $P_i$ .

### 3.5 Additional Preliminaries

In this work, we will also use results regarding statistical distance and secret sharing. Preliminaries on these topics can be found in [Appendix A](#).

## 4 Threshold Multi-Key FHE: Definition

In this section, we present the definition of threshold multi-key fully homomorphic encryption (TMFHE) in the plain model with distributed setup<sup>5</sup>. TMFHE will be the main building block in our MPC protocols.

**Definition 1** (TMFHE). *Let  $P = \{P_1, \dots, P_N\}$  be a set of parties and let  $\mathcal{S}$  be a class of efficient access structures on  $P$ . A threshold multi-key fully homomorphic encryption scheme supporting up to  $N$  parties is a tuple of PPT algorithms*

$$\text{TMFHE} = (\text{DistSetup}, \text{KeyGen}, \text{Enc}, \text{Eval}, \text{PartDec}, \text{FinDec})$$

*satisfying the following specifications:*

$\text{params}_i \leftarrow \text{DistSetup}(1^\lambda, 1^d, 1^N, i)$ : *It takes as input a security parameter  $\lambda$ , a circuit depth  $d$ , the maximal number of parties  $N$ , and a party index  $i$ . It outputs the public parameters  $\text{params}_i$  associated with the  $i$ th party. We define  $\text{params} = \text{params}_1 || \dots || \text{params}_N$ .*

---

<sup>5</sup>Note that we can instead define TMFHE with a single trusted setup, which will allow us to construct MPC protocols in the CRS model as in [MW16]. However, our main focus is on the plain model, and therefore, we use decentralized setup as in [BHP17].

$(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$ : It takes as input the security parameter  $\lambda$  and outputs a key pair  $(pk, sk)$ .

$ct \leftarrow \text{Encrypt}(\text{params}, pk_1, \dots, pk_N, \mathbb{A}, m)$ : It takes as input the public parameters  $\text{params}$ , public keys  $pk_1, \dots, pk_N$ , an access structure  $\mathbb{A}$  over  $P$  and a plaintext  $m \in \{0, 1\}^\lambda$  and outputs a ciphertext  $ct$ . Throughout, we will assume that all ciphertexts include the public parameters and the public keys and access structure that they are encrypted under.

$\hat{ct} \leftarrow \text{Eval}(C, ct_1, \dots, ct_\ell)$ : It takes as input a boolean circuit  $C: (\{0, 1\}^\lambda)^\ell \rightarrow \{0, 1\} \in \mathcal{C}_\lambda$  of depth  $\leq d$  and ciphertexts  $ct_1, \dots, ct_\ell$  for  $\ell \leq N$ . It outputs an evaluated ciphertext  $\hat{ct}$ . Note that  $N$  is the maximal number of supported parties.

$p_i \leftarrow \text{PartDec}(i, sk, \hat{ct})$ : It takes as input an index  $i$ , a secret key  $sk$  and an evaluated ciphertext  $\hat{ct}$  and outputs a partial decryption  $p_i$ .

$\hat{\mu} \leftarrow \text{FinDec}(B)$ : It takes as input a set  $B = \{p_i\}_{i \in S}$  for some  $S \subseteq \{P_1, \dots, P_N\}$  where we recall that we identify a party  $P_i$  with its index  $i$ . It deterministically outputs a plaintext  $\hat{\mu} \in \{0, 1, \perp\}$ .

We require that for any parameters  $\{\text{params}_i \leftarrow \text{DistSetup}(1^\lambda, 1^d, 1^N, i)\}_{i \in [N]}$ , any key pairs  $\{(pk_i, sk_i) \leftarrow \text{KeyGen}(1^\lambda)\}_{i \in [N]}$ , any supported access structure  $\mathbb{A}$  over  $P$ , any plaintexts  $m_1, \dots, m_\ell \in \{0, 1\}^\lambda$  for  $\ell \leq N$ , and any boolean circuit  $C: (\{0, 1\}^\lambda)^\ell \rightarrow \{0, 1\} \in \mathcal{C}_\lambda$  of depth  $\leq d$ , the following is satisfied:

**Correctness.** Let  $ct_i = \text{Encrypt}(\text{params}, pk_1, \dots, pk_N, \mathbb{A}, m_i)$  for  $1 \leq i \leq \ell$ ,  $\hat{ct} = \text{Eval}(C, ct_1, \dots, ct_\ell)$ , and  $B = \{\text{PartDec}(i, sk_i, \hat{ct})\}_{i \in S}$ . With all but negligible probability in  $\lambda$  over the coins of  $\text{DistSetup}$ ,  $\text{KeyGen}$ ,  $\text{Encrypt}$ , and  $\text{PartDec}$ ,

$$\text{FinDec}(B) = \begin{cases} C(m_1, \dots, m_\ell), & S \in \mathbb{A} \\ \perp & S \notin \mathbb{A}. \end{cases}$$

**Compactness of Ciphertexts.** There exists a polynomial,  $\text{poly}$ , such that  $|ct| \leq \text{poly}(\lambda, d, N)$  for any ciphertext  $ct$  generated from the algorithms of TMFHE.

**Simulation Security.** There exists a stateful PPT algorithms  $\text{Sim}_1, \text{Sim}_2$  such that for any PPT adversary  $\mathcal{A}$ , we have that the experiments  $\text{Expt}_{\mathcal{A}, \text{Real}}(1^\lambda, 1^d, 1^n)$  and  $\text{Expt}_{\mathcal{A}, \text{Sim}}(1^\lambda, 1^d, 1^n)$  as defined below are statistically close as a function of  $\lambda$  over the coins of all the algorithms. The experiments are defined as follows:

$\text{Expt}_{\mathcal{A}, \text{Real}}(1^\lambda, 1^d, 1^n)$ :

1. On input the security parameter  $1^\lambda$ , a circuit depth  $1^d$ , and the maximal number of parties  $1^n$ , the adversary  $\mathcal{A}$  outputs a number of parties  $N \leq n$ , a set  $S \subseteq [N]$  and an access structure  $\mathbb{A} \in \mathbb{S}$  over  $N$  parties such that  $S \notin \mathbb{A}$ .
2. For  $i \notin S$ , run  $\text{DistSetup}(1^\lambda, 1^d, 1^N, i) \rightarrow \text{params}_i$ . The adversary is given  $\{\text{params}_i\}_{i \notin S}$ . Sample key pairs  $\text{KeyGen}(1^\lambda) \rightarrow (pk_i, sk_i)$  for  $i \notin S$ . The adversary is given  $\{pk_i\}_{i \notin S}$ .
3. For each  $i \in S$ , the adversary either outputs  $\text{params}_i$  and randomness  $r_i^{\text{KeyGen}}$  used to generate  $(pk_i, sk_i)$  or  $\perp$ .

4. Let  $S_{\text{params}} \subseteq [N]$  be the set of parties  $P_i$  for which  $\text{params}_i$  is defined and let  $S_1 = S \cap S_{\text{params}}$ . The adversary then outputs messages  $m_1, \dots, m_\ell \in \{0, 1\}^\lambda$  and a set  $L \subseteq S_{\text{params}} \setminus S_1$  of indices with  $|L| = \ell$  for some  $\ell \leq |S_{\text{params}} \setminus S_1|$ .
5.  $\text{params}$  is set to the concatenation of the  $\text{params}_i$ 's for  $i \in S_{\text{params}}$ . For  $i \in S_1$ , run  $\text{KeyGen}(1^\lambda; r_i^{\text{KeyGen}})$  to obtain  $(pk_i, sk_i)_{i \in S_1}$ . Let  $\mathcal{PK} = \{pk_i\}_{i \in S_{\text{params}}}$ . Let  $\mathbb{A}'$  be the restriction of  $\mathbb{A}$  to the parties in  $S_{\text{params}}$ . The adversary is given  $ct_i \leftarrow \text{Enc}(\text{params}, \mathcal{PK}, \mathbb{A}', m_i)$  for  $i \in L$ .
6. For all  $i \in S_1$ , the adversary either outputs a pair  $(m_i, r_i^{\text{Encrypt}})$  for a message  $m_i$  and randomness used for encryption  $r_i^{\text{Encrypt}}$  or  $\perp$ . For the  $i \in S_1$  for which  $(m_i, r_i^{\text{Encrypt}})$  is defined, set  $ct_i = \text{Enc}(\text{params}, \mathcal{PK}, \mathbb{A}', m_i; r_i^{\text{Encrypt}})$ . Let  $S_{ct} \subseteq S_{\text{params}}$  be the set of indices for which  $ct_i$  is defined.
7. The adversary issues polynomially many queries of the form  $(S'_k, S_{ct,k}, C_k : (\{0, 1\}^\lambda)^{s_k} \rightarrow \{0, 1\})$ , where  $S'_k \subseteq S_{\text{params}} \setminus S_1$ ,  $S_{ct,k} \subseteq S_{ct}$ ,  $C_k \in \mathcal{C}$ , and  $s_k = |S_{ct,k}| \leq |S_{ct}|$ . Let  $\mathcal{CT}_k = \{ct_i\}_{i \in S_{ct,k}}$  and let the evaluated ciphertext be  $\hat{c}_k \leftarrow \text{Eval}(C_k, \mathcal{CT}_k)$ . After each query, the adversary receives  $p_{i,k} \leftarrow \text{PartDec}(i, sk_i, \hat{c}_k)$  for all  $i \in S'_k$ .
8.  $\mathcal{A}$  outputs out. The output of the experiment is out.

$\text{Expt}_{\mathcal{A}, \text{Sim}}(1^\lambda, 1^d, 1^n)$ :

1. On input the security parameter  $1^\lambda$ , a circuit depth  $1^d$ , and the maximal number of parties  $1^n$ , the adversary  $\mathcal{A}$  outputs a number of parties  $N \leq n$ , a set  $S \subseteq [N]$  and an access structure  $\mathbb{A} \in \mathbb{S}$  over  $N$  parties such that  $S \notin \mathbb{A}$ .
2. For  $i \notin S$ , run  $\text{DistSetup}(1^\lambda, 1^d, 1^N, i) \rightarrow \text{params}_i$ . The adversary is given  $\{\text{params}_i\}_{i \notin S}$ . Sample key pairs  $\text{KeyGen}(1^\lambda) \rightarrow (pk_i, sk_i)$  for  $i \notin S$ . The adversary is given  $\{pk_i\}_{i \notin S}$ .
3. For each  $i \in S$ , the adversary either outputs  $\text{params}_i$  and randomness  $r_i^{\text{KeyGen}}$  used to generate  $(pk_i, sk_i)$  or  $\perp$ .
4. Let  $S_{\text{params}} \subseteq [N]$  be the set of parties  $P_i$  for which  $\text{params}_i$  is defined and let  $S_1 = S \cap S_{\text{params}}$ . The adversary then outputs messages  $m_1, \dots, m_\ell \in \{0, 1\}^\lambda$  and a set  $L \subseteq S_{\text{params}} \setminus S_1$  of indices with  $|L| = \ell$  for some  $\ell \leq |S_{\text{params}} \setminus S_1|$ .
5.  $\text{params}$  is set to the concatenation of the  $\text{params}_i$ 's for  $i \in S_{\text{params}}$ . For  $i \in S_1$ , run  $\text{KeyGen}(1^\lambda; r_i^{\text{KeyGen}})$  to obtain  $(pk_i, sk_i)_{i \in S_1}$ . Let  $\mathcal{PK} = \{pk_i\}_{i \in S_{\text{params}}}$ . Let  $\mathbb{A}'$  be the restriction of  $\mathbb{A}$  to the parties in  $S_{\text{params}}$ . Run  $(\{ct_i\}_{i \in L}, \text{state}) \leftarrow \text{Sim}_1(\text{params}, \mathcal{PK}, \mathbb{A}', S_1, L)$  and give  $\{ct_i\}_{i \in L}$  to the adversary.
6. For all  $i \in S_1$ , the adversary either outputs a pair  $(m_i, r_i^{\text{Encrypt}})$  for a message  $m_i$  and randomness used for encryption  $r_i^{\text{Encrypt}}$  or  $\perp$ . For the  $i \in S_1$  for which  $(m_i, r_i^{\text{Encrypt}})$  is defined, set  $ct_i = \text{Enc}(\text{params}, \mathcal{PK}, \mathbb{A}', m_i; r_i^{\text{Encrypt}})$ . Let  $S_{ct} \subseteq S_{\text{params}}$  be the set of indices for which  $ct_i$  is defined.
7. The adversary issues polynomially many queries of the form  $(S'_k, S_{ct,k}, C_k : (\{0, 1\}^\lambda)^{s_k} \rightarrow \{0, 1\})$ , where  $S'_k \subseteq S_{\text{params}} \setminus S_1$ ,  $S_{ct,k} \subseteq S_{ct}$ ,  $C_k \in \mathcal{C}$ , and  $s_k = |S_{ct,k}| \leq |S_{ct}|$ . Let  $\mathcal{CT}_k = \{ct_i\}_{i \in S_{ct,k}}$  and let the evaluated ciphertext be  $\hat{c}_k \leftarrow \text{Eval}(C_k, \mathcal{CT}_k)$ . After each query, the adversary receives  $\{p_{i,k}\}_{i \in S'_k} \leftarrow \text{Sim}_2(\text{state}, \mu_k, \hat{c}_k, S_1, S'_k, \{sk_i\}_{i \in S_1})$ , where  $\mu_k = C_k(\{m_i\}_{i \in S_{ct,k}})$  if  $S_1 \cup S'_k \in \mathbb{A}'$  and  $\mu_k = \perp$  otherwise.

8. *A outputs out. The output of the experiment is out.*

The above security notion is inspired by the simulation-security definitions of multi-key FHE [MW16, BHP17]. However, looking ahead to our MPC protocol, we will need some stronger guarantees from the TMFHE scheme. In our MPC protocol, the adversary is allowed to choose which honest parties abort in each round and is rushing, so he is allowed to control the randomness of corrupted parties as a function of the honest parties. We capture this by allowing the simulator of the TMFHE scheme to be stateful. Additionally, since the adversary in MPC is rushing, it is allowed to see the honest parameters/ciphertexts before it picks its parameters/ciphertexts.

## 5 Threshold Multi-Key FHE: Construction

In this section, we construct threshold multi-key FHE as defined in Section 4. Formally, we show the following.

**Theorem 5** (TMFHE). *Assuming LWE, there exists a secure threshold multi-key FHE scheme for the class of access structures  $\{0, 1\}$ -LSSSD. In particular, there exists a secure TMFHE scheme for any access structure induced by a monotone boolean formula and any  $t$  out of  $N$  access structure.*

We will construct threshold multi-key FHE using several ingredients. First, we will initialize a multi-key FHE scheme using the construction in [BHP17]. Then, we will utilize the techniques in the construction of threshold FHE in [JRS17], which shows how to transform a generic FHE scheme satisfying several properties into a threshold FHE scheme. We observe that the multi-key FHE construction of [BHP17] is “compatible” with the thresholdizing transformation described in [JRS17]. Finally, we will need a public key encryption scheme to tie everything together.

Examining the construction of [JRS17], we note that it is compatible with a generic FHE scheme where

1. The secret key  $sk$  is a vector in  $\mathbb{Z}_q^m$  for some prime  $q$ .
2. The decryption function  $\text{Dec}$  can be broken into two algorithms  $\text{Dec}_0, \text{Dec}_1$  where  $\text{Dec}_0(sk, ct)$  computes a linear function in  $sk$  and  $ct$  to output  $\mu \lceil q/2 \rceil + e$  for some bounded error  $e \in [-E, E]$  with  $E \ll q$ , where  $ct$  is an encryption of  $\mu$ .  $\text{Dec}_1$  then takes this resulting value and rounds to recover  $\mu$ .

We note that the construction of multi-key FHE in [BHP17] satisfies these required properties. Furthermore, it satisfies the following additional properties that will be useful to note in the construction.

1. An evaluated ciphertext  $\hat{ct}$  that encrypts a bit  $\mu$  with respect to public keys  $pk_1, \dots, pk_\ell$  is a matrix that satisfies

$$\vec{s} \cdot \hat{ct} \approx \mu \vec{s} \cdot G$$

for a gadget matrix  $G$  and  $\vec{s} = (sk_1 || \dots || sk_\ell)$ , where  $sk_i$  is the secret key corresponding to public key  $pk_i$ . Each  $sk_i$  is of the form  $(s_i || 1)$ .

2. There exists a low-norm vector  $\vec{v}$  such that  $G\vec{v} = (0, 0, \dots, \lceil q/2 \rceil)^T$ . Decryption proceeds by evaluating  $\vec{s} \cdot \hat{ct} \cdot \vec{v}$  and then outputs 1 if the resulting value is closer to  $\lceil q/2 \rceil$  than 0 and 0 otherwise.



Furthermore, [JRS17] shows the following result.

**Theorem 6** ([JRS17]). *For any access structure  $\mathbb{A}$  on  $N$  parties induced by a monotone boolean formula, there exists a  $\{0, 1\}$ -LSSSD scheme of a vector  $s \in \mathbb{Z}_q^m$  where each party  $P$  receives at most  $w$  shares of the form  $s_i \in \mathbb{Z}_q^m$  for  $w = \text{poly}(N)$ .*

## 5.1 Construction

Let MFHE = (DistSetup, KeyGen, Enc, Eval, PartDec, FinDec) be a multi-key FHE scheme instantiated with the construction in [BHP17]. Let PKE = (Setup, Enc, Dec) be a public-key encryption scheme. Let  $\chi^{sm}$  denote the uniform distribution on the interval  $[-E_{sm}, E_{sm}]$  for a value  $E_{sm}$  to be determined.

Our threshold multi-key FHE construction TMFHE is given as follows:

DistSetup( $1^\lambda, 1^d, 1^N, i$ ): Run MFHE.DistSetup( $1^\lambda, 1^d, 1^N, i$ )  $\rightarrow$   $\text{params}_i$  and output  $\text{params}_i$ .

KeyGen( $1^\lambda$ ): Run PKE.Setup( $1^\lambda$ )  $\rightarrow$   $(pk, sk)$  and output  $(pk, sk)$ .

Encrypt( $\text{params}, pk_1, \dots, pk_N, \mathbb{A}, m$ ): Run MFHE.KeyGen( $\text{params}$ )  $\rightarrow$   $(\text{fpk}, \text{fsk})$ . Apply the  $\{0, 1\}$ -LSSSD scheme associated with  $\mathbb{A}$  to  $\text{fsk}$  to arrive at  $\{\text{fsk}_{i,j}\}_{i \in [N], j \in [w]}$  for some  $w = \text{poly}(N)$ . Set  $ct' \leftarrow \text{MFHE.Enc}(\text{fpk}, m)$  and for  $i \in [N]$ , set  $ct_i = \text{PKE.Enc}(pk_i, \{\text{fsk}_{i,j}\}_{j \in [w]})$ . Output

$$ct = (ct', ct_1, \dots, ct_N).$$

Eval( $C, ct_1, \dots, ct_\ell$ ): Parse  $ct_i$  as  $(ct'_i, ct_{i,1}, \dots, ct_{i,N})$ . Let  $\text{fpk}_i$  be the MFHE public key associated with  $ct'_i$ . Run MFHE.Eval( $C, ct'_1, \dots, ct'_\ell$ )  $\rightarrow \hat{ct}'$ . Output

$$\hat{ct} = (\hat{ct}', \{ct_{i,j}\}_{(i,j) \in [\ell] \times [N]}).$$

PartDec( $i, sk, \hat{ct}$ ): Parse  $\hat{ct}$  as  $(\hat{ct}', \{ct_{k,j}\}_{(k,j) \in [\ell] \times [N]})$ . For every  $k \in [\ell]$ , run PKE.Dec( $sk, ct_{k,i}$ )  $\rightarrow \{\text{fsk}_{k,i,j}\}_{j \in [w]}$ . For  $t \in [w]$ , compute

$$(\text{fsk}_{1,i,t} || \text{fsk}_{2,i,t} || \dots || \text{fsk}_{\ell,i,t}) \cdot \hat{ct}' \cdot \vec{v} + e_t^{sm} \rightarrow p'_t,$$

where  $e_t^{sm} \leftarrow \chi^{sm}$  and  $\vec{v}$  is the low-norm vector used for decryption in [BHP17] described above. Output  $p_i = (i, \{p'_t\}_{t \in [w]})$ .

FinDec( $B$ ): Parse  $B$  as  $\{(i, \{p'_t\}_{t \in [w]})\}_{i \in S}$  for some set  $S$  of indices. If  $S \notin \mathbb{A}$ , output  $\perp$ . If  $S \in \mathbb{A}$ , apply the  $\{0, 1\}$ -LSSSD reconstruction to get  $\approx \hat{\mu} \lceil q/2 \rceil$ . Then, round to recover  $\hat{\mu}$ .

## 5.2 Correctness and Compactness

Correctness follows from the correctness of the underlying MFHE scheme and the  $\{0, 1\}$ -LSSSD scheme. Let  $\hat{ct}$  be a correctly generated evaluated ciphertext with MFHE ciphertext component  $\hat{ct}'$  and let  $B = \{p_i\}_{i \in S} = \{\text{PartDec}(i, sk_i, \hat{ct})\}_{i \in S}$  for some set of parties  $S$  as specified in the definition of correctness. If  $S \notin \mathbb{A}$ , then FinDec( $B$ ) =  $\perp$  as desired. If  $S \in \mathbb{A}$ , then by the correctness of the

$\{0, 1\}$ -LSSSD reconstruction procedure, there exists some subset of shares that sum to the secret. In other words, given  $\{p_i\}_{i \in S} = \{(i, \{p'_{i,t}\}_{t \in [w]})\}_{i \in S}$ , there exist sets  $W_i \subseteq [w]$  such that

$$\sum_{i \in [N]} \sum_{t \in W_i} p'_{i,t} = (\text{fsk}_1 \| \text{fsk}_2 \| \dots \| \text{fsk}_N) \cdot \hat{c}t' \cdot \vec{v} + \sum_{i \in [N]} \sum_{t \in W_i} e_{i,t}^{sm}.$$

Note that this reconstruction procedure works even with the concatenation of secrets and multiplying by  $\hat{c}t'$  because each of the  $\text{fsk}_i$ 's is shared with respect to the same secret sharing scheme and the reconstruction procedure is linear. This gives

$$\mu \lceil q/2 \rceil + e + \sum_{i \in [N]} \sum_{t \in W_i} e_{i,t}^{sm},$$

where  $e$  is the error incurred by the underlying MFHE scheme. If

$$\left| e + \sum_{i \in [N]} \sum_{t \in W_i} e_{i,t}^{sm} \right| < q/4,$$

then rounding will correctly recover  $\mu$ . Since  $e$  is sampled from an  $E$ -bounded distribution and each  $e_{i,t}^{sm}$  from an  $E_{sm}$ -bounded one, if  $E + NwE_{sm} < q/4$ , then correctness will be satisfied.

Compactness follows immediately from the construction and the compactness of the underlying schemes.

### 5.3 Security

For notational simplicity, we will prove security in the game where the adversary only submits a single circuit  $C$ . The proof naturally extends to the full definition where the adversary is allowed to submit polynomially many circuits, due to the adaptive nature of the result in Proposition 1. We make a note in the proof showing this extension. We will prove security via a series of hybrid games. We use red text to denote the difference between the current hybrid and the previous one. One thing to note is that in the security game, each party generates their parameters  $\text{params}_i$  with respect to the number of parties  $N$ . However, some parties may abort and not output any parameters, which leads to encryption being done with respect to a set of parties of size  $N' \leq N$ . Therefore, it is necessary for parameters generated with respect to  $N$  parties to be able to be used for encryptions with respect to  $N'$  parties. This is not an issue in our construction because we observe that the  $\text{params}_i$  of each party in the MFHE construction in [BHP17] and, therefore, also in our TMFHE construction, are simply random matrices  $A_i$  of a size dependent on  $N$ . Therefore, truncating the matrix to the appropriate size for a scheme with  $N'$  parties is equivalent to having run the distributed setup algorithm for  $N'$  parties.

Hyb<sub>0</sub> : This is the same as the “real” experiment. Namely,

$$\text{Hyb}_0(1^\lambda, 1^d, 1^n) = \text{Expt}_{\mathcal{A}, \text{Real}}(1^\lambda, 1^d, 1^n):$$

1. On input the security parameter  $1^\lambda$ , a circuit depth  $1^d$ , and the maximal number of parties  $1^n$ , the adversary  $\mathcal{A}$  outputs a number of parties  $N \leq n$ , a set  $S \subseteq [N]$  and an access structure  $\mathbb{A} \in \mathbb{S}$  over  $N$  parties such that  $S \notin \mathbb{A}$ .

2. For  $i \notin S$ , run  $\text{DistSetup}(1^\lambda, 1^d, 1^N, i) \rightarrow \text{params}_i$ . The adversary is given  $\{\text{params}_i\}_{i \notin S}$ . Sample key pairs  $\text{KeyGen}(1^\lambda) \rightarrow (pk_i, sk_i)$  for  $i \notin S$ . The adversary is given  $\{pk_i\}_{i \notin S}$ .
3. For each  $i \in S$ , the adversary either outputs  $\text{params}_i$  and randomness  $r_i^{\text{KeyGen}}$  used to generate  $(pk_i, sk_i)$  or  $\perp$ .
4. Let  $S_{\text{params}} \subseteq [N]$  be the set of parties  $P_i$  for which  $\text{params}_i$  is defined and let  $S_1 = S \cap S_{\text{params}}$ . The adversary then outputs messages  $m_1, \dots, m_\ell \in \{0, 1\}^\lambda$  and a set  $L \subseteq S_{\text{params}} \setminus S_1$  of indices with  $|L| = \ell$  for some  $\ell \leq |S_{\text{params}} \setminus S_1|$ .
5.  $\text{params}$  is set to the concatenation of the  $\text{params}_i$ 's for  $i \in S_{\text{params}}$ . For  $i \in S_1$ , run  $\text{KeyGen}(1^\lambda; r_i^{\text{KeyGen}})$  to obtain  $(pk_i, sk_i)_{i \in S_1}$ . Let  $\mathcal{PK} = \{pk_i\}_{i \in S_{\text{params}}}$ . Let  $\mathbb{A}'$  be the restriction of  $\mathbb{A}$  to the parties in  $S_{\text{params}}$ . The adversary is given  $ct_i \leftarrow \text{Enc}(\text{params}, \mathcal{PK}, \mathbb{A}', m_i)$  for  $i \in L$ .
6. For all  $i \in S_1$ , the adversary either outputs a pair  $(m_i, r_i^{\text{Encrypt}})$  for a message  $m_i$  and randomness used for encryption  $r_i^{\text{Encrypt}}$  or  $\perp$ . For the  $i \in S_1$  for which  $(m_i, r_i^{\text{Encrypt}})$  is defined, set  $ct_i = \text{Enc}(\text{params}, \mathcal{PK}, \mathbb{A}', m_i; r_i^{\text{Encrypt}})$ . Let  $S_{ct} \subseteq S_{\text{params}}$  be the set of indices for which  $ct_i$  is defined.
7. The adversary outputs a circuit  $C : (\{0, 1\}^\lambda)^s \rightarrow \{0, 1\}$  along with a subset  $S'_{ct} \subseteq S_{ct}$  with  $C \in \mathcal{C}$  and  $s = |S'_{ct}| \leq |S_{ct}|$ . Let  $\mathcal{CT} = \{ct_i\}_{i \in S'_{ct}}$  and let the evaluated ciphertext be  $\hat{ct} \leftarrow \text{Eval}(C, \mathcal{CT})$ .
8. The adversary outputs a set  $S' \subseteq S_{\text{params}} \setminus S_1$ . For all  $i \in S'$ , the adversary is given  $p_i \leftarrow \text{PartDec}(i, sk_i, \hat{ct})$ .
9.  $\mathcal{A}$  outputs **out**. The output of the experiment is **out**.

Hyb<sub>1</sub> : This is the same as Hyb<sub>0</sub> except we expand the TMFHE encryption and partial decryption procedures according to our construction.

Hyb<sub>1</sub>( $1^\lambda, 1^d, 1^n$ ):

1. On input the security parameter  $1^\lambda$ , a circuit depth  $1^d$ , and the maximal number of parties  $1^n$ , the adversary  $\mathcal{A}$  outputs a number of parties  $N \leq n$ , a set  $S \subseteq [N]$  and an access structure  $\mathbb{A} \in \mathbb{S}$  over  $N$  parties such that  $S \notin \mathbb{A}$ .
2. For  $i \notin S$ , run  $\text{DistSetup}(1^\lambda, 1^d, 1^N, i) \rightarrow \text{params}_i$ . The adversary is given  $\{\text{params}_i\}_{i \notin S}$ . Sample key pairs  $\text{KeyGen}(1^\lambda) \rightarrow (pk_i, sk_i)$  for  $i \notin S$ . The adversary is given  $\{pk_i\}_{i \notin S}$ .
3. For each  $i \in S$ , the adversary either outputs  $\text{params}_i$  and randomness  $r_i^{\text{KeyGen}}$  used to generate  $(pk_i, sk_i)$  or  $\perp$ .
4. Let  $S_{\text{params}} \subseteq [N]$  be the set of parties  $P_i$  for which  $\text{params}_i$  is defined and let  $S_1 = S \cap S_{\text{params}}$ . The adversary then outputs messages  $m_1, \dots, m_\ell \in \{0, 1\}^\lambda$  and a set  $L \subseteq S_{\text{params}} \setminus S_1$  of indices with  $|L| = \ell$  for some  $\ell \leq |S_{\text{params}} \setminus S_1|$ .
5.  $\text{params}$  is set to the concatenation of the  $\text{params}_i$ 's for  $i \in S_{\text{params}}$ . For  $i \in S_1$ , run  $\text{KeyGen}(1^\lambda; r_i^{\text{KeyGen}})$  to obtain  $(pk_i, sk_i)_{i \in S_1}$ . Let  $\mathcal{PK} = \{pk_i\}_{i \in S_{\text{params}}}$ . Let  $\mathbb{A}'$  be the restriction of  $\mathbb{A}$  to the parties in  $S_{\text{params}}$ .  
For  $i \in L$ , run  $\text{MFHE.KeyGen}(\text{params}) \rightarrow (\text{fpk}_i, \text{fsk}_i)$ . Apply the secret sharing scheme associated with  $\mathbb{A}'$  to  $\text{fsk}_i$  to arrive at  $\{\text{fsk}_{i,j,k}\}_{j \in S_{\text{params}}, k \in [w]}$  for some  $w = \text{poly}(n)$ . Set

$ct'_i \leftarrow \text{MFHE.Enc}(\text{fpk}_i, m_i)$  and for  $j \in S_{\text{params}}$ , set  $ct_{i,j} = \text{PKE.Enc}(pk_j, \{\text{fsk}_{i,j,k}\}_{k \in [w]})$ . The adversary is given

$$ct_i = (ct'_i, \{ct_{i,j}\}_{j \in S_{\text{params}}}).$$

6. For all  $i \in S_1$ , the adversary either outputs a pair  $(m_i, r_i^{\text{Encrypt}})$  for a message  $m_i$  and randomness used for encryption  $r_i^{\text{Encrypt}}$  or  $\perp$ . For the  $i \in S_1$  for which  $(m_i, r_i^{\text{Encrypt}})$  is defined, set  $ct_i = \text{Enc}(\text{params}, \mathcal{PK}, \mathbb{A}', m_i; r_i^{\text{Encrypt}})$ . Let  $S_{ct} \subseteq S_{\text{params}}$  be the set of indices for which  $ct_i$  is defined.
7. The adversary outputs a circuit  $C : (\{0, 1\}^\lambda)^s \rightarrow \{0, 1\}$  along with a subset  $S'_{ct} \subseteq S_{ct}$  with  $C \in \mathcal{C}$  and  $s = |S'_{ct}| \leq |S_{ct}|$ . Let  $\mathcal{CT} = \{ct_i\}_{i \in S'_{ct}}$  and let the evaluated ciphertext be  $\hat{ct} \leftarrow \text{Eval}(C, \mathcal{CT})$ .
8. The adversary outputs a set  $S' \subseteq S_{\text{params}} \setminus S_1$ .

Parse  $\hat{ct}$  as  $(\hat{ct}', \{ct_{i,j}\}_{(i,j) \in S'_{ct} \times S_{\text{params}}})$ . Define  $S_{\text{shares}}$  as the set of all the indices of the secret shares corresponding to the parties in  $S_1$  under the secret sharing scheme associated with  $\mathbb{A}'$ . Notationally, these are  $\{(j, k)\}_{j \in S_1, k \in [w]}$ . Define  $S'_{\text{shares}}$  in an analogous manner for the set  $S'$ . For  $(i, j) \in S'_{ct} \setminus L \times S_1$ , decrypt  $ct_{i,j}$  using  $sk_j$  to recover  $\{\text{fsk}_{i,j,k}\}_{i \in S'_{ct} \setminus L, j \in S_1, k \in [w]}$ . For  $(j, k) \in S'_{\text{shares}}$ , compute

$$\tilde{p}_{j,k} = (\text{fsk}_{I_1,j,k} || \text{fsk}_{I_2,j,k} || \dots || \text{fsk}_{I_s,j,k}) \cdot \hat{ct}' \cdot \vec{v},$$

where  $\vec{v}$  is the low-norm vector used for decryption in [BHP17] and the  $I_i$ 's are the ordered sequence of indices in  $S'_{ct}$ . Then, for  $(j, k) \in S'_{\text{shares}}$ , set

$$p'_{j,k} = \tilde{p}_{j,k} + e_{j,k}^{sm},$$

where  $e_{j,k}^{sm} \leftarrow \chi^{sm}$ . For all  $j \in S'$ , give the adversary

$$p_j = (j, \{p'_{j,k}\}_{k \in [w]}).$$

9.  $\mathcal{A}$  outputs out. The output of the experiment is out.

Hyb<sub>2</sub> : This is the same as Hyb<sub>1</sub> except that for all  $i \in L, j \notin S_1$ , we set the encrypted  $\text{fsk}_{i,j,k}$ 's to 0. Note that these are the secret shares that the adversary is not able to recover.

Hyb<sub>2</sub>( $1^\lambda, 1^d, 1^n$ ):

1. On input the security parameter  $1^\lambda$ , a circuit depth  $1^d$ , and the maximal number of parties  $1^n$ , the adversary  $\mathcal{A}$  outputs a number of parties  $N \leq n$ , a set  $S \subseteq [N]$  and an access structure  $\mathbb{A} \in \mathbb{S}$  over  $N$  parties such that  $S \notin \mathbb{A}$ .
2. For  $i \notin S$ , run  $\text{DistSetup}(1^\lambda, 1^d, 1^N, i) \rightarrow \text{params}_i$ . The adversary is given  $\{\text{params}_i\}_{i \notin S}$ . Sample key pairs  $\text{KeyGen}(1^\lambda) \rightarrow (pk_i, sk_i)$  for  $i \notin S$ . The adversary is given  $\{pk_i\}_{i \notin S}$ .
3. For each  $i \in S$ , the adversary either outputs  $\text{params}_i$  and randomness  $r_i^{\text{KeyGen}}$  used to generate  $(pk_i, sk_i)$  or  $\perp$ .

4. Let  $S_{\text{params}} \subseteq [N]$  be the set of parties  $P_i$  for which  $\text{params}_i$  is defined and let  $S_1 = S \cap S_{\text{params}}$ . The adversary then outputs messages  $m_1, \dots, m_\ell \in \{0, 1\}^\lambda$  and a set  $L \subseteq S_{\text{params}} \setminus S_1$  of indices with  $|L| = \ell$  for some  $\ell \leq |S_{\text{params}} \setminus S_1|$ .
5.  $\text{params}$  is set to the concatenation of the  $\text{params}_i$ 's for  $i \in S_{\text{params}}$ . For  $i \in S_1$ , run  $\text{KeyGen}(1^\lambda; r_i^{\text{KeyGen}})$  to obtain  $(pk_i, sk_i)_{i \in S_1}$ . Let  $\mathcal{PK} = \{pk_i\}_{i \in S_{\text{params}}}$ . Let  $\mathbb{A}'$  be the restriction of  $\mathbb{A}$  to the parties in  $S_{\text{params}}$ .

For  $i \in L$ , run  $\text{MFHE.KeyGen}(\text{params}) \rightarrow (\text{fpk}_i, \text{fsk}_i)$ . Apply the secret sharing scheme associated with  $\mathbb{A}'$  to  $\text{fsk}_i$  to arrive at  $\{\text{fsk}_{i,j,k}\}_{j \in S_{\text{params}}, k \in [w]}$  for some  $w = \text{poly}(n)$ . Set  $ct'_i \leftarrow \text{MFHE.Enc}(\text{fpk}_i, m_i)$  and for  $j \in S_{\text{params}}$ , set  $ct_{i,j} = \text{PKE.Enc}(pk_j, \{\text{fsk}_{i,j,k}\}_{k \in [w]})$  if  $j \in S_1$  and  $ct_{i,j} = \text{PKE.Enc}(pk_j, \vec{0})$  if  $j \notin S_1$ , where  $\vec{0}$  is an all 0 encryption of the same length as  $w$  secret key shares. The adversary is given

$$ct_i = (ct'_i, \{ct_{i,j}\}_{j \in S_{\text{params}}}).$$

6. For all  $i \in S_1$ , the adversary either outputs a pair  $(m_i, r_i^{\text{Encrypt}})$  for a message  $m_i$  and randomness used for encryption  $r_i^{\text{Encrypt}}$  or  $\perp$ . For the  $i \in S_1$  for which  $(m_i, r_i^{\text{Encrypt}})$  is defined, set  $ct_i = \text{Enc}(\text{params}, \mathcal{PK}, \mathbb{A}', m_i; r_i^{\text{Encrypt}})$ . Let  $S_{ct} \subseteq S_{\text{params}}$  be the set of indices for which  $ct_i$  is defined.
7. The adversary outputs a circuit  $C : (\{0, 1\}^\lambda)^s \rightarrow \{0, 1\}$  along with a subset  $S'_{ct} \subseteq S_{ct}$  with  $C \in \mathcal{C}$  and  $s = |S'_{ct}| \leq |S_{ct}|$ . Let  $\mathcal{CT} = \{ct_i\}_{i \in S'_{ct}}$  and let the evaluated ciphertext be  $\hat{ct} \leftarrow \text{Eval}(C, \mathcal{CT})$ .
8. The adversary outputs a set  $S' \subseteq S_{\text{params}} \setminus S_1$ .

Parse  $\hat{ct}$  as  $(\hat{ct}', \{ct_{i,j}\}_{(i,j) \in S'_{ct} \times S_{\text{params}}})$ . Define  $S_{\text{shares}}$  as the set of all the indices of the secret shares corresponding to the parties in  $S_1$  under the secret sharing scheme associated with  $\mathbb{A}'$ . Notationally, these are  $\{(j, k)\}_{j \in S_1, k \in [w]}$ . Define  $S'_{\text{shares}}$  in an analogous manner for the set  $S'$ . For  $(i, j) \in S'_{ct} \setminus L \times S_1$ , decrypt  $ct_{i,j}$  using  $sk_j$  to recover  $\{\text{fsk}_{i,j,k}\}_{i \in S'_{ct} \setminus L, j \in S_1, k \in [w]}$ . For  $(j, k) \in S'_{\text{shares}}$ , compute

$$\tilde{p}_{j,k} = (\text{fsk}_{I_1,j,k} || \text{fsk}_{I_2,j,k} || \dots || \text{fsk}_{I_s,j,k}) \cdot \hat{ct}' \cdot \vec{v},$$

where  $\vec{v}$  is the low-norm vector used for decryption in [BHP17] and the  $I_i$ 's are the ordered sequence of indices in  $S'_{ct}$ . Then, for  $(j, k) \in S'_{\text{shares}}$ , set

$$p'_{j,k} = \tilde{p}_{j,k} + e_{j,k}^{sm},$$

where  $e_{j,k}^{sm} \leftarrow \chi^{sm}$ . For all  $j \in S'$ , give the adversary

$$p_j = (j, \{p'_{j,k}\}_{k \in [w]}).$$

9.  $\mathcal{A}$  outputs out. The output of the experiment is out.

**Hyb<sub>3</sub>** : This is the same as **Hyb<sub>2</sub>** except that for all  $j \in S'$ , the partial decryptions given to the adversary are simulated.

**Hyb<sub>3</sub>**( $1^\lambda, 1^d, 1^n$ ):

1. On input the security parameter  $1^\lambda$ , a circuit depth  $1^d$ , and the maximal number of parties  $1^n$ , the adversary  $\mathcal{A}$  outputs a number of parties  $N \leq n$ , a set  $S \subseteq [N]$  and an access structure  $\mathbb{A} \in \mathbb{S}$  over  $N$  parties such that  $S \notin \mathbb{A}$ .
2. For  $i \notin S$ , run  $\text{DistSetup}(1^\lambda, 1^d, 1^N, i) \rightarrow \text{params}_i$ . The adversary is given  $\{\text{params}_i\}_{i \notin S}$ . Sample key pairs  $\text{KeyGen}(1^\lambda) \rightarrow (pk_i, sk_i)$  for  $i \notin S$ . The adversary is given  $\{pk_i\}_{i \notin S}$ .
3. For each  $i \in S$ , the adversary either outputs  $\text{params}_i$  and randomness  $r_i^{\text{KeyGen}}$  used to generate  $(pk_i, sk_i)$  or  $\perp$ .
4. Let  $S_{\text{params}} \subseteq [N]$  be the set of parties  $P_i$  for which  $\text{params}_i$  is defined and let  $S_1 = S \cap S_{\text{params}}$ . The adversary then outputs messages  $m_1, \dots, m_\ell \in \{0, 1\}^\lambda$  and a set  $L \subseteq S_{\text{params}} \setminus S_1$  of indices with  $|L| = \ell$  for some  $\ell \leq |S_{\text{params}} \setminus S_1|$ .
5.  $\text{params}$  is set to the concatenation of the  $\text{params}_i$ 's for  $i \in S_{\text{params}}$ . For  $i \in S_1$ , run  $\text{KeyGen}(1^\lambda; r_i^{\text{KeyGen}})$  to obtain  $(pk_i, sk_i)_{i \in S_1}$ . Let  $\mathcal{PK} = \{pk_i\}_{i \in S_{\text{params}}}$ . Let  $\mathbb{A}'$  be the restriction of  $\mathbb{A}$  to the parties in  $S_{\text{params}}$ .  
For  $i \in L$ , run  $\text{MFHE.KeyGen}(\text{params}) \rightarrow (\text{fpk}_i, \text{fsk}_i)$ . Apply the secret sharing scheme associated with  $\mathbb{A}'$  to  $\text{fsk}_i$  to arrive at  $\{\text{fsk}_{i,j,k}\}_{j \in S_{\text{params}}, k \in [w]}$  for some  $w = \text{poly}(n)$ . Set  $ct'_i \leftarrow \text{MFHE.Enc}(\text{fpk}_i, m_i)$  and for  $j \in S_{\text{params}}$ , set  $ct_{i,j} = \text{PKE.Enc}(pk_j, \{\text{fsk}_{i,j,k}\}_{k \in [w]})$  if  $j \in S_1$  and  $ct_{i,j} = \text{PKE.Enc}(pk_j, \vec{0})$  if  $j \notin S_1$ , where  $\vec{0}$  is an all 0 encryption of the same length as  $w$  secret key shares. The adversary is given

$$ct_i = (ct'_i, \{ct_{i,j}\}_{j \in S_{\text{params}}}).$$

6. For all  $i \in S_1$ , the adversary either outputs a pair  $(m_i, r_i^{\text{Encrypt}})$  for a message  $m_i$  and randomness used for encryption  $r_i^{\text{Encrypt}}$  or  $\perp$ . For the  $i \in S_1$  for which  $(m_i, r_i^{\text{Encrypt}})$  is defined, set  $ct_i = \text{Enc}(\text{params}, \mathcal{PK}, \mathbb{A}', m_i; r_i^{\text{Encrypt}})$ . Let  $S_{ct} \subseteq S_{\text{params}}$  be the set of indices for which  $ct_i$  is defined.
7. The adversary outputs a circuit  $C : (\{0, 1\}^\lambda)^s \rightarrow \{0, 1\}$  along with a subset  $S'_{ct} \subseteq S_{ct}$  with  $C \in \mathcal{C}$  and  $s = |S'_{ct}| \leq |S_{ct}|$ . Let  $\mathcal{CT} = \{ct_i\}_{i \in S'_{ct}}$  and let the evaluated ciphertext be  $\hat{ct} \leftarrow \text{Eval}(C, \mathcal{CT})$ .
8. The adversary outputs a set  $S' \subseteq S_{\text{params}} \setminus S_1$ .

Parse  $\hat{ct}$  as  $(\hat{ct}', \{ct_{i,j}\}_{(i,j) \in S'_{ct} \times S_{\text{params}}})$ . Define  $S_{\text{shares}}$  as the set of all the indices of the secret shares corresponding to the parties in  $S_1$  under the secret sharing scheme associated with  $\mathbb{A}'$ . Notationally, these are  $\{(j, k)\}_{j \in S_1, k \in [w]}$ . Define  $S'_{\text{shares}}$  in an analogous manner for the set  $S'$ . If  $S_1 \cup S' \notin \mathbb{A}'$ , set  $S_{\text{max}} = S_{\text{shares}} \cup S'_{\text{shares}}$ . Else, set  $S_{\text{max}}$  to be a maximally unqualified set of shares with  $S_{\text{shares}} \subseteq S_{\text{max}} \subseteq S_{\text{shares}} \cup S'_{\text{shares}}$ . If  $S_1 \cup S' \in \mathbb{A}'$ , set  $\mu = C(\{m_i\}_{i \in S'_{ct}})$ . Else, set  $\mu = \perp$ .

For  $(i, j) \in S'_{ct} \setminus L \times S_1$ , decrypt  $ct_{i,j}$  using  $sk_j$  to recover  $\{\text{fsk}_{i,j,k}\}_{i \in S'_{ct} \setminus L, j \in S_1, k \in [w]}$ .

For  $(j, k) \in S_{\text{max}}$ , compute

$$\tilde{p}_{j,k} = (\text{fsk}_{I_1,j,k} \parallel \text{fsk}_{I_2,j,k} \parallel \dots \parallel \text{fsk}_{I_s,j,k}) \cdot \hat{ct}' \cdot \vec{v},$$

where  $\vec{v}$  is the low-norm vector used for decryption in [BHP17] and the  $I_i$ 's are the ordered sequence of the indices in  $S'_{ct}$ . Then, for every  $(j, k) \in S'_{\text{shares}} \setminus S_{\text{max}}$ , let  $T_{j,k} \subseteq S_{\text{max}} \cup \{(j, k)\}$  be a minimal valid share set containing  $(j, k)$ . Then, set

$$\tilde{p}_{j,k} = \mu \lceil q/2 \rceil - \sum_{(\alpha, \beta) \neq (j, k) \in T_{j,k}} \tilde{p}_{\alpha, \beta}.$$

Then, for  $(j, k) \in S'_{\text{shares}}$ , set

$$p'_{j,k} = \tilde{p}_{j,k} + e_{j,k}^{sm},$$

where  $e_{j,k}^{sm} \leftarrow \chi^{sm}$ . For all  $j \in S'$ , give the adversary

$$p_j = (j, \{p'_{j,k}\}_{k \in [w]}).$$

9.  $\mathcal{A}$  outputs out. The output of the experiment is out.

**Hyb<sub>4</sub>** : This is the same as **Hyb<sub>3</sub>** except that for all  $i \in L$ , the secret key shares are generated with respect to 0 rather than  $\text{fsk}_i$ .

**Hyb<sub>4</sub>**( $1^\lambda, 1^d, 1^n$ ):

1. On input the security parameter  $1^\lambda$ , a circuit depth  $1^d$ , and the maximal number of parties  $1^n$ , the adversary  $\mathcal{A}$  outputs a number of parties  $N \leq n$ , a set  $S \subseteq [N]$  and an access structure  $\mathbb{A} \in \mathbb{S}$  over  $N$  parties such that  $S \notin \mathbb{A}$ .
2. For  $i \notin S$ , run  $\text{DistSetup}(1^\lambda, 1^d, 1^N, i) \rightarrow \text{params}_i$ . The adversary is given  $\{\text{params}_i\}_{i \notin S}$ . Sample key pairs  $\text{KeyGen}(1^\lambda) \rightarrow (pk_i, sk_i)$  for  $i \notin S$ . The adversary is given  $\{pk_i\}_{i \notin S}$ .
3. For each  $i \in S$ , the adversary either outputs  $\text{params}_i$  and randomness  $r_i^{\text{KeyGen}}$  used to generate  $(pk_i, sk_i)$  or  $\perp$ .
4. Let  $S_{\text{params}} \subseteq [N]$  be the set of parties  $P_i$  for which  $\text{params}_i$  is defined and let  $S_1 = S \cap S_{\text{params}}$ . The adversary then outputs messages  $m_1, \dots, m_\ell \in \{0, 1\}^\lambda$  and a set  $L \subseteq S_{\text{params}} \setminus S_1$  of indices with  $|L| = \ell$  for some  $\ell \leq |S_{\text{params}} \setminus S_1|$ .
5.  $\text{params}$  is set to the concatenation of the  $\text{params}_i$ 's for  $i \in S_{\text{params}}$ . For  $i \in S_1$ , run  $\text{KeyGen}(1^\lambda; r_i^{\text{KeyGen}})$  to obtain  $(pk_i, sk_i)_{i \in S_1}$ . Let  $\mathcal{PK} = \{pk_i\}_{i \in S_{\text{params}}}$ . Let  $\mathbb{A}'$  be the restriction of  $\mathbb{A}$  to the parties in  $S_{\text{params}}$ .  
For  $i \in L$ , run  $\text{MFHE.KeyGen}(\text{params}) \rightarrow (\text{fpk}_i, \text{fsk}_i)$ . **Apply the secret sharing scheme associated with  $\mathbb{A}'$  to 0 to arrive at  $\{\text{fsk}_{i,j,k}\}_{j \in S_{\text{params}}, k \in [w]}$  for some  $w = \text{poly}(n)$ .** Set  $ct'_i \leftarrow \text{MFHE.Enc}(\text{fpk}_i, m_i)$  and for  $j \in S_{\text{params}}$ , set  $ct_{i,j} = \text{PKE.Enc}(pk_j, \{\text{fsk}_{i,j,k}\}_{k \in [w]})$  if  $j \in S_1$  and  $ct_{i,j} = \text{PKE.Enc}(pk_j, \vec{0})$  if  $j \notin S_1$ , where  $\vec{0}$  is an all 0 encryption of the same length as  $w$  secret key shares. The adversary is given

$$ct_i = (ct'_i, \{ct_{i,j}\}_{j \in S_{\text{params}}}).$$

6. For all  $i \in S_1$ , the adversary either outputs a pair  $(m_i, r_i^{\text{Encrypt}})$  for a message  $m_i$  and randomness used for encryption  $r_i^{\text{Encrypt}}$  or  $\perp$ . For the  $i \in S_1$  for which  $(m_i, r_i^{\text{Encrypt}})$  is defined, set  $ct_i = \text{Enc}(\text{params}, \mathcal{PK}, \mathbb{A}', m_i; r_i^{\text{Encrypt}})$ . Let  $S_{ct} \subseteq S_{\text{params}}$  be the set of indices for which  $ct_i$  is defined.
7. The adversary outputs a circuit  $C : (\{0, 1\}^\lambda)^s \rightarrow \{0, 1\}$  along with a subset  $S'_{ct} \subseteq S_{ct}$  with  $C \in \mathcal{C}$  and  $s = |S'_{ct}| \leq |S_{ct}|$ . Let  $\mathcal{CT} = \{ct_i\}_{i \in S'_{ct}}$  and let the evaluated ciphertext be  $\hat{ct} \leftarrow \text{Eval}(C, \mathcal{CT})$ .



8. The adversary outputs a set  $S' \subseteq S_{\text{params}} \setminus S_1$ .

Parse  $\hat{ct}$  as  $(\hat{ct}', \{ct_{i,j}\}_{(i,j) \in S'_{ct} \times S_{\text{params}}})$ . Define  $S_{\text{shares}}$  as the set of all the indices of the secret shares corresponding to the parties in  $S_1$  under the secret sharing scheme associated with  $\mathbb{A}'$ . Notationally, these are  $\{(j,k)\}_{j \in S_1, k \in [w]}$ . Define  $S'_{\text{shares}}$  in an analogous manner for the set  $S'$ . If  $S_1 \cup S' \notin \mathbb{A}'$ , set  $S_{\text{max}} = S_{\text{shares}} \cup S'_{\text{shares}}$ . Else, set  $S_{\text{max}}$  to be a maximally unqualified set of shares with  $S_{\text{shares}} \subseteq S_{\text{max}} \subseteq S_{\text{shares}} \cup S'_{\text{shares}}$ . If  $S_1 \cup S' \in \mathbb{A}'$ , set  $\mu = C(\{m_i\}_{i \in S'_{ct}})$ . Else, set  $\mu = \perp$ .

For  $(i,j) \in S'_{ct} \setminus L \times S_1$ , decrypt  $ct_{i,j}$  using  $sk_j$  to recover  $\{\text{fsk}_{i,j,k}\}_{i \in S'_{ct} \setminus L, j \in S_1, k \in [w]}$ .

For  $(j,k) \in S_{\text{max}}$ , compute

$$\tilde{p}_{j,k} = (\text{fsk}_{I_1,j,k} \| \text{fsk}_{I_2,j,k} \| \dots \| \text{fsk}_{I_s,j,k}) \cdot \hat{ct}' \cdot \vec{v},$$

where  $\vec{v}$  is the low-norm vector used for decryption in [BHP17] and the  $I_i$ 's are the ordered sequence of the indices in  $S'_{ct}$ . Then, for every  $(j,k) \in S'_{\text{shares}} \setminus S_{\text{max}}$ , let  $T_{j,k} \subseteq S_{\text{max}} \cup \{(j,k)\}$  be a minimal valid share set containing  $(j,k)$ . Then, set

$$\tilde{p}_{j,k} = \mu \lceil q/2 \rceil - \sum_{(\alpha,\beta) \neq (j,k) \in T_{j,k}} \tilde{p}_{\alpha,\beta}.$$

Then, for  $(j,k) \in S'_{\text{shares}}$ , set

$$p'_{j,k} = \tilde{p}_{j,k} + e_{j,k}^{sm},$$

where  $e_{j,k}^{sm} \leftarrow \chi^{sm}$ . For all  $j \in S'$ , give the adversary

$$p_j = (j, \{p'_{j,k}\}_{k \in [w]}).$$

9.  $\mathcal{A}$  outputs out. The output of the experiment is out.

Hyb<sub>5</sub> : This is the same as Hyb<sub>4</sub> except that for all  $i \in L$ , the ciphertexts given to the adversary contain MFHE encryptions of 0 rather than  $m_i$ .

Hyb<sub>5</sub>( $1^\lambda, 1^d, 1^n$ ):

1. On input the security parameter  $1^\lambda$ , a circuit depth  $1^d$ , and the maximal number of parties  $1^n$ , the adversary  $\mathcal{A}$  outputs a number of parties  $N \leq n$ , a set  $S \subseteq [N]$  and an access structure  $\mathbb{A} \in \mathbb{S}$  over  $N$  parties such that  $S \notin \mathbb{A}$ .
2. For  $i \notin S$ , run  $\text{DistSetup}(1^\lambda, 1^d, 1^N, i) \rightarrow \text{params}_i$ . The adversary is given  $\{\text{params}_i\}_{i \notin S}$ . Sample key pairs  $\text{KeyGen}(1^\lambda) \rightarrow (pk_i, sk_i)$  for  $i \notin S$ . The adversary is given  $\{pk_i\}_{i \notin S}$ .
3. For each  $i \in S$ , the adversary either outputs  $\text{params}_i$  and randomness  $r_i^{\text{KeyGen}}$  used to generate  $(pk_i, sk_i)$  or  $\perp$ .
4. Let  $S_{\text{params}} \subseteq [N]$  be the set of parties  $P_i$  for which  $\text{params}_i$  is defined and let  $S_1 = S \cap S_{\text{params}}$ . The adversary then outputs messages  $m_1, \dots, m_\ell \in \{0, 1\}^\lambda$  and a set  $L \subseteq S_{\text{params}} \setminus S_1$  of indices with  $|L| = \ell$  for some  $\ell \leq |S_{\text{params}} \setminus S_1|$ .

5.  $\text{params}$  is set to the concatenation of the  $\text{params}_i$ 's for  $i \in S_{\text{params}}$ . For  $i \in S_1$ , run  $\text{KeyGen}(1^\lambda; r_i^{\text{KeyGen}})$  to obtain  $(pk_i, sk_i)_{i \in S_1}$ . Let  $\mathcal{PK} = \{pk_i\}_{i \in S_{\text{params}}}$ . Let  $\mathbb{A}'$  be the restriction of  $\mathbb{A}$  to the parties in  $S_{\text{params}}$ .

For  $i \in L$ , run  $\text{MFHE.KeyGen}(\text{params}) \rightarrow (\text{fpk}_i, \text{fsk}_i)$ . Apply the secret sharing scheme associated with  $\mathbb{A}'$  to 0 to arrive at  $\{\text{fsk}_{i,j,k}\}_{j \in S_{\text{params}}, k \in [w]}$  for some  $w = \text{poly}(n)$ . Set  $ct'_i \leftarrow \text{MFHE.Enc}(\text{fpk}_i, 0^\lambda)$  and for  $j \in S_{\text{params}}$ , set  $ct_{i,j} = \text{PKE.Enc}(pk_j, \{\text{fsk}_{i,j,k}\}_{k \in [w]})$  if  $j \in S_1$  and  $ct_{i,j} = \text{PKE.Enc}(pk_j, \vec{0})$  if  $j \notin S_1$ , where  $\vec{0}$  is an all 0 encryption of the same length as  $w$  secret key shares. The adversary is given

$$ct_i = (ct'_i, \{ct_{i,j}\}_{j \in S_{\text{params}}}).$$

6. For all  $i \in S_1$ , the adversary either outputs a pair  $(m_i, r_i^{\text{Encrypt}})$  for a message  $m_i$  and randomness used for encryption  $r_i^{\text{Encrypt}}$  or  $\perp$ . For the  $i \in S_1$  for which  $(m_i, r_i^{\text{Encrypt}})$  is defined, set  $ct_i = \text{Enc}(\text{params}, \mathcal{PK}, \mathbb{A}', m_i; r_i^{\text{Encrypt}})$ . Let  $S_{ct} \subseteq S_{\text{params}}$  be the set of indices for which  $ct_i$  is defined.
7. The adversary outputs a circuit  $C : (\{0, 1\}^\lambda)^s \rightarrow \{0, 1\}$  along with a subset  $S'_{ct} \subseteq S_{ct}$  with  $C \in \mathcal{C}$  and  $s = |S'_{ct}| \leq |S_{ct}|$ . Let  $\mathcal{CT} = \{ct_i\}_{i \in S'_{ct}}$  and let the evaluated ciphertext be  $\hat{ct} \leftarrow \text{Eval}(C, \mathcal{CT})$ .
8. The adversary outputs a set  $S' \subseteq S_{\text{params}} \setminus S_1$ .

Parse  $\hat{ct}$  as  $(\hat{ct}', \{ct_{i,j}\}_{(i,j) \in S'_{ct} \times S_{\text{params}}})$ . Define  $S_{\text{shares}}$  as the set of all the indices of the secret shares corresponding to the parties in  $S_1$  under the secret sharing scheme associated with  $\mathbb{A}'$ . Notationally, these are  $\{(j, k)\}_{j \in S_1, k \in [w]}$ . Define  $S'_{\text{shares}}$  in an analogous manner for the set  $S'$ . If  $S_1 \cup S' \notin \mathbb{A}'$ , set  $S_{\text{max}} = S_{\text{shares}} \cup S'_{\text{shares}}$ . Else, set  $S_{\text{max}}$  to be a maximally unqualified set of shares with  $S_{\text{shares}} \subseteq S_{\text{max}} \subseteq S_{\text{shares}} \cup S'_{\text{shares}}$ . If  $S_1 \cup S' \in \mathbb{A}'$ , set  $\mu = C(\{m_i\}_{i \in S'_{ct}})$ . Else, set  $\mu = \perp$ .

For  $(i, j) \in S'_{ct} \setminus L \times S_1$ , decrypt  $ct_{i,j}$  using  $sk_j$  to recover  $\{\text{fsk}_{i,j,k}\}_{i \in S'_{ct} \setminus L, j \in S_1, k \in [w]}$ .

For  $(j, k) \in S_{\text{max}}$ , compute

$$\tilde{p}_{j,k} = (\text{fsk}_{I_1,j,k} \| \text{fsk}_{I_2,j,k} \| \dots \| \text{fsk}_{I_s,j,k}) \cdot \hat{ct}' \cdot \vec{v},$$

where  $\vec{v}$  is the low-norm vector used for decryption in [BHP17] and the  $I_i$ 's are the ordered sequence of the indices in  $S'_{ct}$ . Then, for every  $(j, k) \in S'_{\text{shares}} \setminus S_{\text{max}}$ , let  $T_{j,k} \subseteq S_{\text{max}} \cup \{(j, k)\}$  be a minimal valid share set containing  $(j, k)$ . Then, set

$$\tilde{p}_{j,k} = \mu \lceil q/2 \rceil - \sum_{(\alpha, \beta) \neq (j,k) \in T_{j,k}} \tilde{p}_{\alpha, \beta}.$$

Then, for  $(j, k) \in S'_{\text{shares}}$ , set

$$p'_{j,k} = \tilde{p}_{j,k} + e_{j,k}^{sm},$$

where  $e_{j,k}^{sm} \leftarrow \chi^{sm}$ . For all  $j \in S'$ , give the adversary

$$p_j = (j, \{p'_{j,k}\}_{k \in [w]}).$$

9.  $\mathcal{A}$  outputs out. The output of the experiment is out.

**Simulator:** Note that the simulator is implicit in  $\text{Hyb}_5$ . Namely,  $\text{Sim}_1$  is the algorithm in Step 5 to generate the ciphertexts and  $\text{Sim}_2$  is the algorithm in Step 8 used to generate the partial decryptions. The state passed from  $\text{Sim}_1$  to  $\text{Sim}_2$  is

$$\text{state} = \{\text{fsk}_{i,j,k}\}_{i \in L, j \in S_{\text{params}}, k \in [w]},$$

the shares generated by  $\text{Sim}_1$  for these indices when secret sharing 0.

**Remark 1.** Note that although  $\text{Sim}_2$  is given  $\{sk_i\}_{i \in S_1}$ , it only uses these secret keys to recover  $\{\text{fsk}_{i,j,k}\}_{i \in S'_{\text{ct}} \setminus L, j \in S_1, k \in [w]}$ . If  $\text{Sim}_2$  was instead given  $\{(m_i, r_i^{\text{Encrypt}})\}_{i \in S'_{\text{ct}} \setminus L}$ , it could simulate in the same manner by using  $(m_i, r_i^{\text{Encrypt}})$ 's to run the adversary's encryption computation and recover the secret key shares  $\{\text{fsk}_{i,j,k}\}_{i \in S'_{\text{ct}} \setminus L, j \in S_1, k \in [w]}$ . This observation will be useful later when analyzing the security of our maliciously-secure MPC protocol in the plain model.

**Lemma 1.**  $\text{Hyb}_0$  and  $\text{Hyb}_1$  are computationally indistinguishable.

*Proof.* These two hybrids are identical; we merely expanded the TMFHE encryption and partial decryption procedures for an easier comparison with future hybrids.  $\square$

**Lemma 2.**  $\text{Hyb}_1$  and  $\text{Hyb}_2$  are computationally indistinguishable.

*Proof.* This follows from the semantic security of the underlying public-key encryption scheme. Suppose there was an adversary  $\mathcal{A}$  that can distinguish between these two hybrids. Then, if we make a sequence of intermediate hybrids, where we switch a single  $\text{fsk}_{i,j,k}$  encryption to 0 in successive hybrids,  $\mathcal{A}$  can distinguish between two neighboring intermediate hybrids in this sequence.  $\mathcal{A}'$  can break the semantic security of PKE by interacting with  $\mathcal{A}$  according to these intermediate hybrids. When it needs to either give an encryption of  $\text{fsk}_{i,j,k}$  or 0,  $\mathcal{A}'$  submits these two messages to its challenger and receives an encryption of one of them, which it feeds to  $\mathcal{A}$ . If  $\mathcal{A}$  can distinguish between the intermediate hybrids, then  $\mathcal{A}'$  also can distinguish between an encryption of  $\text{fsk}_{i,j,k}$  and an encryption of 0, contradicting the security of PKE.  $\square$

**Lemma 3.** Assuming  $E/E_{sm} < \text{negl}(\lambda)$ , then  $\text{Hyb}_2$  and  $\text{Hyb}_3$  are statistically indistinguishable.

*Proof.* The only difference in the adversary's view between  $\text{Hyb}_2$  and  $\text{Hyb}_3$  is that in  $\text{Hyb}_2$ , all the partial decryptions for  $(j, k) \in S'_{\text{shares}}$  are generated using the real secret key shares, whereas in  $\text{Hyb}_3$ , the partial decryptions for  $(j, k) \in S_{\text{max}} \cap S'_{\text{shares}}$  are generated using the real secret key shares, but the partial decryptions for  $(j, k) \in S'_{\text{shares}} \setminus (S_{\text{max}} \cap S'_{\text{shares}})$  are simulated using  $\mu$ . Therefore, the distributions of  $\tilde{p}_{j,k}$  and  $p'_{j,k}$  for  $(j, k) \in S_{\text{max}} \cap S'_{\text{shares}}$  in  $\text{Hyb}_2$  and  $\text{Hyb}_3$  are identical. For the remaining  $(j, k) \in S'_{\text{shares}}$ , note that by the properties of a  $\{0, 1\}$ -LSSSD scheme and the linearity of computing the  $\tilde{p}_{j,k}$ 's, there exists a minimal valid share set  $T_{j,k} \subseteq S_{\text{max}} \cup \{(j, k)\}$  such that

$$\sum_{(\alpha, \beta) \in T_{j,k}} \tilde{p}_{\alpha, \beta} = \mu \lceil q/2 \rceil + e$$

for some  $E$ -bounded noise  $e$ . Therefore, it follows that

$$\tilde{p}_{j,k} = \mu \lceil q/2 \rceil + e - \sum_{(\alpha, \beta) \in T_{j,k} \setminus \{(j,k)\}} \tilde{p}_{\alpha, \beta}.$$

This is the value of the  $\tilde{p}_{j,k}$  computed in  $\text{Hyb}_2$ , whereas in  $\text{Hyb}_3$ , the value is

$$\tilde{p}_{j,k} = \mu \lceil q/2 \rceil - \sum_{(\alpha,\beta) \in T_{j,k} \setminus \{(j,k)\}} \tilde{p}_{\alpha,\beta}.$$

Setting  $\tilde{p}_{j,k}$  to be the value computed in  $\text{Hyb}_3$ , it follows that in  $\text{Hyb}_3$ , the adversary receives the value

$$\tilde{p}_{j,k} + e_{j,k}^{sm}$$

and in  $\text{Hyb}_2$ , the adversary receives the value

$$\tilde{p}_{j,k} + e + e_{j,k}^{sm}$$

for  $e_{j,k}^{sm} \leftarrow \chi^{sm}$  uniformly at random for each  $(j,k) \in S'_{\text{shares}}$ . Since

$$(\tilde{p}_{j,k} + e) - \tilde{p}_{j,k} = e \in [-E, E],$$

it follows from Proposition 1 and Lemma 7 that the statistical distance between  $\text{Hyb}_2$  and  $\text{Hyb}_3$  is  $\leq nwE/E_{sm} \leq \text{poly}(n)E/E_{sm} = \text{negl}(\lambda)$ . Note that the adaptive nature of the adversary in Proposition 1 allows indistinguishability to extend to the case of multiple circuits, where the adversary may choose the circuit queries adaptively.  $\square$

**Lemma 4.**  *$\text{Hyb}_3$  and  $\text{Hyb}_4$  are computationally indistinguishable.*

*Proof.* This follows from the fact that the secret sharing scheme associated with  $\mathbb{A}$  is information-theoretically secure. In both  $\text{Hyb}_3$  and  $\text{Hyb}_4$ , only shares associated with unqualified sets are used. Since unqualified sets reveal no information about the secret, these two games must be indistinguishable.  $\square$

**Lemma 5.**  *$\text{Hyb}_4$  and  $\text{Hyb}_5$  are computationally indistinguishable.*

*Proof.* This follows from the semantic security of the underlying MFHE scheme. Suppose there is an adversary  $\mathcal{A}$  that can distinguish between these two hybrids. Then, consider a sequence of  $\ell$  intermediate hybrids where in neighboring hybrids, we switch one of the encryptions of  $m_i$  to an encryption of  $0^\lambda$ . There must exist two neighboring intermediate hybrids that  $\mathcal{A}$  can distinguish between.  $\mathcal{A}'$  can break the semantic security of the MFHE scheme by interacting with  $\mathcal{A}$  according to these hybrids. When  $\mathcal{A}'$  would need to generate an encryption of either  $m_i$  or 0 depending on which intermediate hybrid it is running,  $\mathcal{A}'$  submits  $m_i$  and 0 as two messages to its challenger and receives an encryption of one of them, which it uses to continue interacting with  $\mathcal{A}$ . If  $\mathcal{A}$  can distinguish between these two hybrid, then  $\mathcal{A}'$  will be able to distinguish between MFHE encryptions of  $m_i$  and 0, contradicting the semantic security of MFHE.  $\square$

## 5.4 Instantiation

In order for correctness to hold, we required that  $E + NwE_{sm} < q/4$ . For security, we required that  $NwE/E_{sm} = \text{negl}(\lambda)$ . Recall that  $w = \text{poly}(N)$ . Let  $W = \text{poly}(N)$  be an upper bound for the set of access structures supported by the scheme. Then, setting  $E/E_{sm} < \lambda^{-\log_2 \lambda}$  and  $E_{sm} < q/8NW$  gives us an instantiation that satisfies both correctness and security. The MFHE scheme of [BHP17] can be instantiated with such properties assuming a variant of the learning with errors assumption, which is as hard as approximating the shortest vector problem to within a subexponential factor.

## 6 Three-Round MPC with General Mixed Adversaries

In this section, we initiate the study of the exact round complexity of MPC in the presence of a general mixed adversary by constructing a three round MPC protocol secure in this model, which was defined in [Appendix B](#). Our protocol supports all functionalities computable by polynomial-sized circuits and is parameterized by an access structure  $\mathbb{A}$  in the class  $\{0, 1\}$ -LSSSD. The protocol is secure against all general mixed adversaries  $\mathcal{A} = (\mathcal{A}_{\text{Mal}}, \mathcal{A}_{\text{Sh}}, \mathcal{A}_{\text{Fc}})$  such that  $(\mathcal{A}_{\text{Mal}} \cup \mathcal{A}_{\text{Sh}}) \notin \mathbb{A}$  and there is no restriction on the set  $\mathcal{A}_{\text{Fc}}$ . As a stepping stone, we first construct a protocol secure against a semi-malicious mixed adversary (defined in [Appendix B](#)) with a similar access structure requirement on the adversarial corruption pattern. Formally, we show the following results.

**Theorem 7.** *Assuming  $\text{LWE}$ , for any access structure  $\mathbb{A} \in \{0, 1\}$ -LSSSD, there exists a three-round MPC protocol in the plain model for any functionality  $f$  computable by a polynomial-sized circuit that is secure against a semi-malicious mixed adversary  $\mathcal{A} = (\mathcal{A}_{\text{Sm}}, \mathcal{A}_{\text{Sh}}, \mathcal{A}_{\text{Fc}})$  such that  $(\mathcal{A}_{\text{Sm}} \cup \mathcal{A}_{\text{Sh}}) \notin \mathbb{A}$ . Furthermore, the protocol is reusable and has communication complexity  $\text{poly}(\lambda, d, N)$ , where  $d$  is the depth of the circuit computing  $f$ .*

From [Theorem 7](#), we can apply UC-secure NIZKs [[DSDCO<sup>+</sup>01](#), [GOS12](#)] to obtain the following theorem.

**Theorem 8.** *Assuming  $\text{LWE}$  and  $\text{DLIN}$ , for any access structure  $\mathbb{A} \in \{0, 1\}$ -LSSSD, there exists a three-round MPC protocol in the CRS model for any functionality  $f$  computable by a polynomial-sized circuit that is secure against a general mixed adversary  $\mathcal{A} = (\mathcal{A}_{\text{Mal}}, \mathcal{A}_{\text{Sh}}, \mathcal{A}_{\text{Fc}})$  such that  $(\mathcal{A}_{\text{Mal}} \cup \mathcal{A}_{\text{Sh}}) \notin \mathbb{A}$ . Furthermore, the protocol is reusable and has communication complexity  $\text{poly}(\lambda, d, N)$ , where  $d$  is the depth of the circuit computing  $f$ .*

Hirt et al. [[HMZ08](#)] showed that in the setting of a general mixed adversary, MPC is possible if and only if  $(2 \times |\mathcal{A}_{\text{Mal}}| + |\mathcal{A}_{\text{Sh}}| + |\mathcal{A}_{\text{Fc}}|) < N$ . Thus, by appropriately setting our access structure  $\mathbb{A}$  to satisfy this constraint, our resulting protocol is both *optimal* in terms of the best possible corruption we can tolerate and also round efficient.

We note that our results in the mixed adversary setting are in fact broader and more general than the setting of standard MPC. For example, instantiating [Theorem 8](#) with the  $N$  out of  $N$  access structure, we arrive at the following corollary below. Another instantiation would be to the setting of honest majority when we consider the  $\frac{N}{2}$  out of  $N$  access structure and we defer that to the next section where it is explored in more detail.

**Corollary 1.** *Assuming  $\text{LWE}$  and  $\text{DLIN}$ , there exists a three-round MPC protocol in the CRS model for any functionality  $f$  computable by a polynomial-sized circuit that is secure against malicious adversaries that corrupt up to  $N - 1$  parties. Furthermore, the protocol is reusable and has communication complexity  $\text{poly}(\lambda, d, N)$ , where  $d$  is the depth of the circuit computing  $f$ .*

### 6.1 Plain Model: Semi-Malicious Mixed Adversary

We first describe a three-round MPC protocol for any access structure  $\mathbb{A} \in \{0, 1\}$ -LSSSD that is secure against a semi-malicious mixed adversary  $\mathcal{A} = (\mathcal{A}_{\text{Sm}}, \mathcal{A}_{\text{Sh}}, \mathcal{A}_{\text{Fc}})$  such that  $(\mathcal{A}_{\text{Sm}} \cup \mathcal{A}_{\text{Sh}}) \notin \mathbb{A}$ .

### 6.1.1 Construction

**Notation:**

- Consider  $N$  parties  $P_1, \dots, P_N$  with inputs  $x_1, \dots, x_N$ , respectively, who wish to evaluate a boolean circuit  $C$  with depth  $\leq d$  on their joint inputs. Let  $\lambda$  denote the security parameter and without loss of generality, assume  $|x_i| = \lambda$  for all  $i \in [N]$ .
- Let  $\text{TMFHE} = (\text{DistSetup}, \text{KeyGen}, \text{Enc}, \text{Eval}, \text{PartDec}, \text{FinDec})$  be the previously constructed threshold multi-key FHE scheme. Let  $\mathbb{A}$  be the agreed-upon access structure.

**Protocol:** We now describe the construction of our three-round MPC protocol  $\Pi$  that is secure against a semi-malicious mixed adversary.

- **Input Commitment Phase:**

- **Round 1:** Each party  $P_i$  does the following:

1. Run  $\text{TMFHE.DistSetup}(1^\lambda, 1^d, 1^N, i)$  to obtain  $\text{params}_i$ .
2. Run  $\text{TMFHE.KeyGen}(1^\lambda)$  to compute  $(pk_i, sk_i)$ .
3. Output  $(\text{params}_i, pk_i)$ .

- **Round 2:** Each party  $P_i$  does the following:

1. Parse the message (if one was sent) from  $P_j$  as  $(\text{params}_j, pk_j)$ . Let  $S_1 \subseteq [N]$  be the set of parties that sent a message in round 1.
2. Truncate each  $\text{params}_j$  for  $j \in S_1$  to the appropriate size given  $|S_1|^6$ . Set  $\text{params}$  as the concatenation of the truncated  $\text{params}_j$ 's for  $j \in S_1$ . Set  $\mathcal{PK} = \{pk_j\}_{j \in S_1}$ . Let  $\mathbb{A}'$  be the access structure induced by restricting  $\mathbb{A}$  to the parties in  $S_1$ .
3. Run  $\text{TMFHE.Encrypt}(\text{params}, \mathcal{PK}, \mathbb{A}', x_i)$  to compute  $ct_i$ .
4. Output  $ct_i$ .

- **Computation Phase:**

- **Round 3:** Each party  $P_i$  does the following:

1. Parse the previous message (if one was sent) from  $P_j$  as  $ct_j$ . Let  $S_2 \subseteq [N]$  be the set of parties that sent a message in round 2. Let  $\mathcal{CT} = \{ct_j\}_{j \in S_2}$ . Let  $C'$  be the circuit induced by hardcoding the inputs to  $C$  corresponding to parties not in  $S_2$  to be  $0^\lambda$ .
2. Run  $\text{TMFHE.Eval}(C', \mathcal{CT})$  to obtain  $\hat{ct}$ .
3. Run  $\text{TMFHE.PartDec}(i, sk_i, \hat{ct})$  to obtain  $p_i$ .
4. Output  $p_i$ .

- **Output Computation:** Each party  $P_i$  does the following:

---

<sup>6</sup>Note that the  $\text{params}_i$  of each party in the MFHE construction in [BHP17] and, therefore, also in our TMFHE construction, are simply random matrices  $A_i$  of a size dependent on  $N$ . Therefore, truncating the matrix to the appropriate size for a scheme with  $|S_1|$  parties is equivalent to having run the distributed setup algorithm for  $|S_1|$  parties.

1. Parse the previous message (if one was sent) from  $P_j$  as  $p_j$ . Let  $S_3 \subseteq [N]$  be the set of parties that sent a message in round 3.
2. Take any set  $S \subseteq S_3$  with  $S \in \mathbb{A}$  and run  $\text{TMFHE.FinDec}(B)$  where  $B = \{p_j\}_{j \in S}$  to recover  $\hat{\mu}$ . If no such set exists, output  $\perp$ .

**Correctness.** Correctness follows immediately from the correctness of the underlying TMFHE scheme. In particular, let  $S \subseteq [N]$  be the set of parties that finished the input commitment phase and let  $S' \subseteq S$  be the set of parties that finished the computation phase. Note that  $C'(\{x_i\}_{i \in S}) = f(y_1, \dots, y_N)$  where  $y_i = x_i$  if  $i \in S$  and  $0^\lambda$  otherwise. Furthermore, if  $S' \in \mathbb{A}$ , then  $S' \in \mathbb{A}'$  and therefore running  $\text{TMFHE.FinDec}$  will correctly recover  $f(y_1, \dots, y_N)$  as desired.

## 6.2 Security

We will first give a description of the simulator and then argue indistinguishability between the real and ideal worlds.

**Simulator:** The simulator  $\text{Sim}$  is given the security parameter  $\lambda$  and an auxiliary input  $z$ . Let  $f$  be representable by a circuit  $C$  of depth  $\leq d$ .  $\text{Sim}$  proceeds as follows:

- **Before Protocol Execution:** From the semi-malicious mixed adversary  $\text{Adv}$ ,  $\text{Sim}$  receives a tuple of sets  $(\mathcal{A}_{\text{Sm}}, \mathcal{A}_{\text{Sh}}, \mathcal{A}_{\text{Fc}})$  of corrupted parties.
- **Input Commitment Phase (Round 1):** For every fail-corrupt party that  $\text{Adv}$  wishes to abort in this round,  $\text{Sim}$  instructs the corresponding party. For each honest and each fail-corrupt party not yet instructed to abort,  $P_i$ ,  $\text{Sim}$  does the following:
  1. Run  $\text{TMFHE.DistSetup}(1^\lambda, 1^d, 1^N, i)$  to compute  $\text{params}_i$ .
  2. Run  $\text{TMFHE.KeyGen}(1^\lambda)$  to compute  $(pk_i, sk_i)$ .
  3. Give  $(\text{params}_i, pk_i)$  as  $P_i$ 's round 1 message to  $\text{Adv}$ .

$\text{Sim}$  then receives round 1 messages from  $\text{Adv}$  on behalf of every party in the sets  $\mathcal{A}_{\text{Sm}}$  and  $\mathcal{A}_{\text{Sh}}$ .

- **Input Commitment Phase (Round 2):** For every fail-corrupt party that  $\text{Adv}$  wishes to abort in this round,  $\text{Sim}$  instructs the corresponding party. Then,  $\text{Sim}$  parses the message (if one was sent) from party  $P_j$  as  $(\text{params}_j, pk_j)$ . Let  $S_1 \subseteq [N]$  be the set of parties that sent a message in round 1. It truncates each  $\text{params}_j$  to the appropriate size for  $|S_1|$  parties and sets  $\text{params}$  as the concatenation of the truncated  $\text{params}_j$ 's for all  $j \in S_1$ . Let  $\mathcal{PK}$  denote  $\{pk_j\}_{j \in S_1}$ . Let  $\mathbb{A}'$  be the access structure induced by restricting  $\mathbb{A}$  to the parties in  $S_1$ . Let  $S_{\text{hon}}^2$  be the set of honest and fail-corrupt parties that send a message in round 2. Let  $S_{\text{corr}}^1$  be the set of corrupted (semi-malicious and semi-honest) parties that sent a message in round 1.  $\text{Sim}$  does the following:
  1. Run  $\text{Sim}_1(\text{params}, \mathcal{PK}, \mathbb{A}', S_{\text{corr}}^1, S_{\text{hon}}^2)$  to compute  $(\{ct_i\}_{i \in S_{\text{hon}}^2}, \text{state})$ , where  $\text{Sim}_1$  is the first algorithm of the TMFHE simulator.
  2. Give  $ct_i$  as  $P_i$ 's round 2 message to  $\text{Adv}$  for  $i \in S_{\text{hon}}^2$ .



Let  $S_2 \subseteq [N]$  be the set of parties that sent a round 2 message. For semi-maliciously and semi-honestly corrupted parties  $P_i$  in  $S_2$ , Sim receives the input  $x_i$  used by Adv and sends it to the trusted party. For the fail-corrupt parties that already aborted, Sim sends  $0^\lambda$  to the trusted party.

- **Query to Ideal Functionality:** Sim receives the output  $b$  from the trusted party.
- **Computation Phase (Round 3):** For every fail-corrupt party that Adv wishes to abort in this round, Sim instructs the corresponding party. Let  $\mathcal{CT} = \{ct_j\}_{j \in S_2}$ . Let  $C'$  be the circuit induced by hardcoding the inputs to  $C$  corresponding to aborted fail-corrupt parties as  $0^\lambda$ . Let  $S_{\text{hon}}^3$  be the set of honest and fail-corrupt parties that have not yet been told to abort in round 3 by Adv. For corrupted (semi-honest and semi-malicious) parties  $P_i$  in  $S_{\text{corr}}^1$ , Sim extracts the secret keys  $sk_i$  that they generated. Sim does the following
  1. Run  $\text{Sim}_2(\text{state}, b, \hat{ct}, S_{\text{corr}}^1, S_{\text{hon}}^2, \{sk_i\}_{i \in S_{\text{corr}}^1})$  to compute  $\{p_j\}_{j \in S_{\text{hon}}^2}$ , where  $\text{Sim}_2$  is the second algorithm of the TMFHE simulator and  $\hat{ct}$  is the ciphertext obtain by evaluating  $C'$  on the ciphertexts in  $\mathcal{CT}$ .
  2. For  $j \in S_{\text{hon}}^3$ , give  $p_j$  as  $P_j$ 's round 3 message to Adv.
- **Output to Honest Parties:** Let  $S_3 \subseteq [N]$  be the set of parties that sent a round 3 message. If  $S_3 \in \mathbb{A}'$ , then Sim tells the trusted party to send  $b$  to all honest parties. If  $S_3 \notin \mathbb{A}'$ , then Sim does not give  $b$  to any honest party.

**Lemma 6.** For any semi-malicious mixed adversary  $\text{Adv} = (\mathcal{A}_{\text{Sm}}, \mathcal{A}_{\text{Sh}}, \mathcal{A}_{\text{Fc}})$ , for the above simulator Sim, for any  $(\mathcal{A}_{\text{Sm}} \cup \mathcal{A}_{\text{Sh}}) \notin \mathbb{A}$ ,

$$|Pr[\mathcal{D}(\text{REAL}_{\Pi, \text{Adv}(z)}(\lambda, \vec{x})) = 1] - Pr[\mathcal{D}(\text{IDEAL}_{f, \text{Sim}(z)}(\lambda, \vec{x})) = 1]| \leq \text{negl}(\lambda)$$

for any PPT distinguisher  $\mathcal{D}$ .

*Proof.* Suppose there was some semi-malicious mixed adversary  $\text{Adv} = (\mathcal{A}_{\text{Sm}}, \mathcal{A}_{\text{Sh}}, \mathcal{A}_{\text{Fc}})$  with  $(\mathcal{A}_{\text{Sm}} \cup \mathcal{A}_{\text{Sh}}) \notin \mathbb{A}$  for which there existed a distinguisher  $\mathcal{D}$  that could distinguish between the real and ideal world experiments. Then, there exists an adversary  $\text{Adv}'$  that could break the security of the underlying TMFHE scheme.  $\text{Adv}'$  proceeds as follows.

1.  $\text{Adv}'$  runs Adv, which outputs a tuple of sets  $(\mathcal{A}_{\text{Sm}}, \mathcal{A}_{\text{Sh}}, \mathcal{A}_{\text{Fc}})$  of corrupted parties.
2. Adv outputs a set of fail-corrupt parties  $S_{\text{inp}}^1 \subseteq \mathcal{A}_{\text{Fc}}$  that will abort in round 1 (they will never send a message). Let  $S_{\text{parties}} = [N] \setminus S_{\text{inp}}^1$  and let  $N' = |S_{\text{parties}}|$ .  $\text{Adv}'$  outputs  $N' \leq N$  as its number of parties, the corrupted set  $S = (\mathcal{A}_{\text{Sm}} \cup \mathcal{A}_{\text{Sh}}) \subseteq S_{\text{parties}}$ , and the access structure  $\mathbb{A}'$  induced by restricting  $\mathbb{A}$  to the parties in  $S_{\text{parties}}$ .
3. For  $i \in S_{\text{parties}} \setminus S$ ,  $\text{Adv}'$  receives  $(\text{params}_i, pk_i)$  and gives this to Adv as  $P_i$ 's round 1 message.
4. For each  $j \in S$ , Adv will output  $(\text{params}_j, pk_j)$ . By running Adv,  $\text{Adv}'$  is able to determine the randomness  $r_j^{\text{KeyGen}}$  used by Adv to generate  $pk_j$  and outputs  $(\text{params}_j, r_j^{\text{KeyGen}})$ .
5. Let  $S_{\text{hon}}^2$  be the set of honest and fail-corrupt parties that will send a round 2 message.  $\text{Adv}'$  outputs this set along with the inputs  $x_i \in \{0, 1\}^\lambda$  for  $i \in S_{\text{hon}}^2$ . Adv' is given  $ct_i$  for  $i \in S_{\text{hon}}^2$  and gives this to Adv as  $P_i$ 's round 2 message.

6. By running  $\text{Adv}$ ,  $\text{Adv}'$  is able to extract the input  $x_i$  and randomness  $r_i^{\text{Encrypt}}$  used by  $\text{Adv}$  for each  $i \in S$ .  $\text{Adv}'$  outputs  $(x_i, r_i^{\text{Encrypt}})$  for all  $i \in S$ .
7. Let  $S_2 = (S_{\text{hon}}^2 \cup \mathcal{A}_{\text{Sm}} \cup \mathcal{A}_{\text{Sh}})$  be the set of parties that sent a round 2 message. Let  $C'$  be the circuit induced by  $C$  by setting the input of all parties that did not send a round 2 message to  $0^\lambda$ .  $\text{Adv}'$  outputs  $C'$  along with  $S_2$ .
8. Let  $S_{\text{hon}}^3$  be the set of honest and fail-corrupt parties that send a round 3 message.  $\text{Adv}'$  outputs  $S_{\text{hon}}^3$  and receives partial decryptions  $p_i$  for  $i \in S_{\text{hon}}^3$ .  $\text{Adv}'$  gives these to  $\text{Adv}$  as  $P_i$ 's round 3 message.  $\text{Adv}$  outputs some function of its view and  $\text{Adv}'$  outputs the same value along with  $\{x_i\}_{i \notin S}$ .

If  $\text{Adv}'$  is interacting with the real TMFHE security game, it simulates the real world experiment for  $\Pi$  exactly for some fixed inputs. Similarly, if  $\text{Adv}'$  is interacting with the simulated TMFHE security game, it simulates the ideal world experiment for  $\Pi$  exactly. Therefore, the existence of  $\text{Adv}$  would result in an adversary that could break the security of the TMFHE scheme, a contradiction.  $\square$

### 6.3 Properties

**Communication Complexity.** To see that the protocol has communication complexity  $\text{poly}(\lambda, d, N)$ , note that the round 1 message is clearly of size  $\text{poly}(\lambda, d, N)$ . So is the round 2 message due to the compactness of the TMFHE scheme. Similarly, the size of  $\hat{c}t$  is  $\text{poly}(\lambda, d, N)$  and, therefore, so too is the partial decryption.

**Reusability.** Reusability means that given the transcript of the input commitment phase, the computation phase can be run any polynomial number of times on different functions using the same transcript for the input commitment phase to compute the different functionalities. Reusability follows from the following:

1. The input commitment phase of  $\Pi$  is function-independent.
2. Our TMFHE simulator can simulate partial decryptions for a polynomial number of adaptively chosen circuit queries.

**Robustness.** Additionally, our protocol enjoys the property that the output of the computation is a function of the joint inputs of all parties, including those that aborted after the input commitment phase was completed. That is, in the scenario where the adversary corrupts a set of parties in a fail-corrupt manner, for every fail-corrupt party  $P_i$  that aborts after the input commitment phase, its input  $y_i$  that is used to compute the final output  $C(y_1, \dots, y_n)$  is set to be its actual input  $x_i$  used in the protocol so far and not a default input  $\perp$ . Recall that this is in line with our motivation for studying this setting where an honest but lazy party is not entirely discarded and its input is still considered in the computation if it aborted after the input commitment phase.

## 6.4 CRS Model: General Mixed Adversary

The MPC protocol described above in the plain model is only secure against semi-malicious mixed adversaries. That is, the adversary can only corrupt some subset  $\mathcal{A}_{\text{Sm}}$  of the parties semi-maliciously, some subset  $\mathcal{A}_{\text{Sh}}$  in a semi-honest manner and another subset  $\mathcal{A}_{\text{Fc}}$  in a fail-corrupt manner. However, our goal now is to allow the adversary to corrupt the first subset  $\mathcal{A}_{\text{Sm}}$  maliciously. That is, for each party in the set  $\mathcal{A}_{\text{Sm}}$ , the adversary can behave arbitrarily while the behaviour of the semi-honest and fail-corrupt corruptions remain unaffected.

Now, if we work in the common reference string (CRS) model, we can use UC-secure non-interactive zero knowledge (NIZK) arguments [DSDCO<sup>+</sup>01, GOS12] to convert the protocol to one that is secure against general mixed adversaries. That is, the NIZK would help to transform and upgrade security against the semi-malicious set of parties  $\mathcal{A}_{\text{Sm}}$  in the earlier protocol into one that is secure even against an adversary that behaves maliciously on behalf of those parties. A generic transformation was shown in [AJLA<sup>+</sup>12] to convert from a protocol where the adversary only corrupts in a semi-malicious manner to one in which the adversary can corrupt maliciously, and we refer the reader to their paper for full details. We note that their transformation trivially extends to our setting as well - where the adversary can additionally also corrupt some set of parties in a semi-honest manner and some in a fail-corrupt manner. Furthermore, the transformation does not affect the communication complexity of the protocol since the statement being proved (and witness length) is proportional only to the depth of the circuit being computed. Also, reusability is also maintained since the CRS allows the NIZK to be reused. Similarly, the robustness property also trivially extends.

## 7 MPC with Guaranteed Output Delivery

In this section, we construct a three-round honest-majority MPC protocol with fairness and guaranteed output delivery in the plain model that is secure against malicious adversaries. Informally, by fairness, we mean that the adversary is unable to learn any output without the honest parties also learning the output. By guaranteed output delivery, we mean that the honest parties will learn some meaningful output with respect to their inputs regardless of the adversary's behavior. Note that guaranteed output delivery implies fairness. We refer to [Gol04] for the formal definitions. This is the first construction of such a protocol in the plain model as previous constructions [GLS15] operated in the common reference string (CRS) model. Furthermore, [Cle86] showed that achieving fairness and guaranteed output delivery is impossible without an honest majority and [GLS15] showed that 2-round MPC protocols with fairness and guaranteed output delivery are impossible, even in the CRS model. Therefore, our protocol is round-optimal and supports the maximal number of corruptions possible to achieve guaranteed output delivery. Furthermore, our protocol inherits the properties of the MPC protocol of Section 6 in that it has depth-proportional communication as well as reusability of the first two rounds. Formally, we show the following.

**Theorem 9.** *Assuming LWE and Zaps, there exists a three-round honest-majority MPC protocol with guaranteed output delivery in the plain model for any functionality  $f$  computable by a polynomial-sized circuit and is secure against malicious adversaries. Furthermore, the protocol is reusable and has communication complexity  $\text{poly}(\lambda, d, N)$ , where  $d$  is the depth of the circuit computing  $f$ ,  $N$  is the number of parties and  $\lambda$  denotes the security parameter and input length.*

In order to prove [Theorem 9](#), we take the MPC protocol of [Section 6](#) that is secure against a semi-malicious mixed adversary and view it in the standard model with honest majority. That is, we fix  $\mathbb{A}$  to be the  $\lfloor N/2 + 1 \rfloor$  out of  $N$  threshold access structure and restrict the adversary to corrupt at most  $(\lfloor N/2 + 1 \rfloor - 1)$  parties semi-maliciously and no extra party is corrupted either in a semi-honest or fail-corrupt manner. From [Theorem 7](#), it follows immediately that such a protocol is semi-malicious secure in the presence of an honest majority. Moreover, this protocol will have guaranteed output delivery (and therefore also fairness) since the set of honest parties always constitutes a majority and will satisfy  $\mathbb{A}$ . However, we observe that if the underlying public-key encryption scheme is such that any string is a valid public key (such schemes are called dense cryptosystems [[DSP92](#)]), then the protocol is secure even against adversaries that are allowed to behave *maliciously* in round 1, but only semi-maliciously in rounds 2 and 3. After noting this, we are able to upgrade to security against malicious adversaries by utilizing multi-string NIZKs [[GO07](#)]. We simply have each party send a reference string  $\text{crs}$  in round 1 and then require each party to also provide a NIZK argument in rounds 2 and 3 using these  $\text{crs}$ 's to ensure that they submitted a valid message in that round.

## 7.1 Semi-Maliciously Secure MPC with Honest Majority

We first describe the three-round honest majority MPC protocol with fairness and guaranteed output delivery that is secure against an adversary that behaves maliciously in round 1 and semi-maliciously in rounds 2 and 3.

### 7.1.1 Construction

The construction is the same as the MPC protocol described in [Section 6](#), except we now view it in the standard model where there are no fail-corrupt or semi-honest corruptions. Furthermore, we make the following changes.

1. In the instantiation of the TMFHE scheme used in the protocol, we use the previously constructed threshold multi-key FHE scheme from [Section 5](#) with the underlying PKE scheme instantiated with one where any string is a valid public key (a dense cryptosystem). Such a PKE scheme can also be built from LWE.
2. We fix  $\mathbb{A}$  as the  $\lfloor N/2 + 1 \rfloor$  out of  $N$  threshold access structure.

**Correctness, Communication Complexity, and Reusability.** Correctness, depth-proportional communication complexity, and reusability follow immediately from the properties of the protocol in [Section 6](#).

### 7.1.2 Security

Semi-malicious security is immediate as the protocol of [Section 6](#) was secure against a stronger semi-malicious general adversary that was also allowed to corrupt additional parties in a fail-corrupt manner beyond the  $(\lfloor N/2 + 1 \rfloor - 1)$  semi-malicious corruptions. To see that the above protocol is also secure against an adversary that can behave maliciously in round 1, we observe the following. As shown previously, the protocol is able to handle an adversary that aborts some of its parties in a fail-corrupt manner. Furthermore, since  $\text{params}_i$  in the MFHE construction in [[BHP17](#)]

is simply a matrix  $A_i$  of random entries and every string  $pk_i$  is a valid public key, it follows that every output of a malicious adversary could also have been output by a semi-malicious adversary that chose the appropriate randomness (we can simply truncate the message or pad it with 0's if the malicious adversary sends a message of inappropriate length). The difference in the proof is that the simulator does not receive the randomness  $r_i^{\text{KeyGen}}$  used by the adversary to compute the round 1 message for a corrupted party and therefore does not receive  $sk_i$  for corrupted parties. However, as we saw in [Section 5.3](#), the simulator does not need to know  $sk_i$ . Rather, it suffices to know  $(x_i, r_i^{\text{Encrypt}})$ , the input and randomness used to compute a corrupted party's round 2 message in order to simulate. Therefore, an analogous simulator and proof can be used to show security against this adversary.

## 7.2 Malicious Security

In this section, we show [Theorem 9](#). Using a simulation-extractable multi-string NIZK in the plain model, we are able to upgrade the previous MPC protocol to one that is secure against malicious adversaries. Since the previous protocol was secure against a malicious adversary in round 1, we can have each party send a reference string in the first round and then send a multi-string NIZK argument in rounds 2 and 3 using the reference strings sent in round 1 to prove that the messages sent were computed correctly. Recall from [Section 3](#) that the security properties of a multi-string NIZK argument hold as long as a majority of the individual reference strings were generated honestly. Formally, the construction is as follows.

### 7.2.1 Construction

#### Notation:

- Consider  $N$  parties  $P_1, \dots, P_N$  with inputs  $x_1, \dots, x_N$ , respectively, who wish to evaluate a boolean circuit  $C$  with depth  $\leq d$  on their joint inputs. Let  $\lambda$  denote the security parameter and without loss of generality, assume  $|x_i| = \lambda$  for all  $i \in [N]$ .
- Let  $\text{TMFHE} = (\text{DistSetup}, \text{KeyGen}, \text{Enc}, \text{Eval}, \text{PartDec}, \text{FinDec})$  be the previously constructed threshold multi-key FHE scheme from [Section 5](#) with the underlying PKE scheme instantiated with one where any string is a valid public key (a dense cryptosystem). Let  $\mathbb{A}$  be the  $\lfloor N/2 + 1 \rfloor$  out of  $N$  threshold access structure. Let  $\text{NIZK} = (\text{Gen}, \text{Prove}, \text{Verify})$  be a simulation-extractable multi-string NIZK.

**Protocol:** We now describe the construction of our three-round MPC protocol  $\Pi$  with guaranteed output delivery that is secure against malicious adversaries. To compare against our previous protocol in [Section 6](#), we highlight the changes in red.

- **Round 1:** Each party  $P_i$  does the following:
  1. Run  $\text{TMFHE.DistSetup}(1^\lambda, 1^d, 1^N, i)$  to obtain  $\text{params}_i$ .
  2. Run  $\text{TMFHE.KeyGen}(1^\lambda)$  to compute  $(pk_i, sk_i)$ .
  3. Run  $\text{NIZK.Gen}(1^\lambda)$  to compute  $\text{crs}_i$ .
  4. Output  $(\text{params}_i, pk_i, \text{crs}_i)$ .

- **Round 2:** Each party  $P_i$  does the following:
  1. Parse the message (if one was sent) from  $P_j$  as  $(\text{params}_j, pk_j, \text{crs}_j)$  by appropriately truncating or padding with 0's if it was of incorrect length. Let  $S_1 \subseteq [N]$  be the set of parties that sent a message in round 1.
  2. Truncate each  $\text{params}_j$  for  $j \in S_1$  to the appropriate size given  $|S_1|$ . Set  $\text{params}$  as the concatenation of the truncated  $\text{params}_j$ 's for  $j \in S_1$ . Set  $\mathcal{PK} = \{pk_j\}_{j \in S_1}$ . Let  $\mathcal{CRS} = \{\text{crs}_j\}_{j \in S_1}$ . Let  $\mathbb{A}'$  be the access structure induced by restricting  $\mathbb{A}$  to the parties in  $S_1$ .
  3. Sample randomness  $r_i$  and run  $\text{TMFHE.Encrypt}(\text{params}, \mathcal{PK}, \mathbb{A}', x_i; r_i)$  to compute  $ct_i$ .
  4. Run  $\text{NIZK.Prove}(\mathcal{CRS}, y_i, (x_i, r_i))$  to compute  $\pi_i$ , where  $y_i$  is the statement that there exists some input  $x$  and randomness  $r$  such that  $\text{TMFHE.Encrypt}(\text{params}, \mathcal{PK}, \mathbb{A}', x; r) = ct_i$ .
  5. Output  $(ct_i, \pi_i)$ .
  
- **Round 3:** Each party  $P_i$  does the following:
  1. Parse the previous message (if one was sent) from  $P_j$  as  $(ct_j, \pi_j)$  and check that  $\text{NIZK.Verify}(\mathcal{CRS}, y_j, \pi_j) = 1$ . Let  $S_2 \subseteq S_1$  be the set of parties that sent a message in round 2 that passed the verification. Let  $\mathcal{CT} = \{ct_j\}_{j \in S_2}$ . Let  $C'$  be the circuit induced by hardcoding the inputs to  $C$  corresponding to parties not in  $S_2$  to be  $0^\lambda$ .
  2. Run  $\text{TMFHE.Eval}(C', \mathcal{CT})$  to compute  $\hat{ct}$ .
  3. Sample randomness  $r'_i$  and run  $\text{TMFHE.PartDec}(i, sk_i, \hat{ct}; r'_i)$  to compute  $p_i$ .
  4. Run  $\text{NIZK.Prove}(\mathcal{CRS}, z_i, (sk_i, r'_i))$  to compute  $\pi'_i$ , where  $z_i$  is the statement that there exists some randomness  $r, r'$  such that  $\text{TMFHE.KeyGen}(1^\lambda; r) = (pk_i, sk)$  and  $\text{TMFHE.PartDec}(i, sk, \hat{ct}; r') = p_i$ .
  5. Output  $(p_i, \pi'_i)$ .
  
- **Output Computation:** Each party  $P_i$  does the following:
  1. Parse the previous message (if one was sent) from  $P_j$  as  $(p_j, \pi'_j)$  and check that  $\text{NIZK.Verify}(\mathcal{CRS}, z_j, \pi'_j) = 1$ . Let  $S_3 \subseteq S_2$  be the set of parties that sent a message in round 3 that passed verification.
  2. Take any set  $S \subseteq S_3$  with  $S \in \mathbb{A}'$  and run  $\text{TMFHE.FinDec}(B)$  where  $B = \{p_j\}_{j \in S}$  to recover  $\hat{\mu}$ . If no such set exists, output  $\perp$ .

**Correctness and Communication Complexity.** Correctness follows from the correctness of the protocol in Section 6 and perfect completeness of the multi-string NIZK. Depth-proportional communication complexity follows from the fact that the communication complexity of the protocol in Section 6 was  $\text{poly}(\lambda, d, N)$  and the size of the NIZK reference strings and proofs are  $\text{poly}(\lambda, d, N)$  because the evaluated ciphertext can be computed publicly and the NIZK is only used to prove correctness of encryption and partial decryption, which only depends on the depth of the function.

**Fairness and Guaranteed Output Delivery.** Guaranteed output delivery follows from the fact that the underlying access structure is the  $\lfloor N/2 + 1 \rfloor$  out of  $N$  majority threshold access structure and the honest parties are always a majority. By soundness of the multi-string NIZK, an adversary cannot cheat and submit an invalid ciphertext as its round 2 message since this message will be discarded with overwhelming probability. The output recovered is the same as that in the protocol of Section 6. Namely, they compute  $C(y_1, \dots, y_N)$  where  $y_i = x_i$  if  $P_i$  sent valid messages in rounds 1 and 2 and  $y_i = 0^\lambda$  otherwise.

## 7.2.2 Security

We provide a description of the simulator.

**Simulator:** The simulator  $\text{Sim}$  is given the security parameter  $\lambda$  and an auxiliary input  $z$ . Let  $f$  be representable by a circuit  $C$  of depth  $\leq d$ . Let  $\text{ExtGen}, \text{Ext}, \text{SimProve}$  be the extraction and simulation algorithms associated with the simulation-extractable multi-string NIZK.  $\text{Sim}$  proceeds as follows:

- **Before Protocol Execution:**  $\text{Sim}$  receives a set  $S \subseteq [N]$  of corrupted parties.
- **Round 1:** For each honest party  $P_i$ ,  $\text{Sim}$  does the following:
  1. Run  $\text{TMFHE.DistSetup}(1^\lambda, 1^d, 1^N, i)$  to compute  $\text{params}_i$ .
  2. Run  $\text{TMFHE.KeyGen}(1^\lambda)$  to compute  $(pk_i, sk_i)$ .
  3. Run  $\text{ExtGen}(1^\lambda)$  to compute  $(\text{crs}_i, \tau_i, \xi_i)$ .
  4. Give  $(\text{params}_i, pk_i, \text{crs}_i)$  as  $P_i$ 's round 1 message to  $\text{Adv}$ .

$\text{Sim}$  then receives round 1 messages from  $\text{Adv}$ .

- **Round 2:**  $\text{Sim}$  parses the message (if one was sent) from party  $P_j$  as  $(\text{params}_j, pk_j, \text{crs}_j)$ . Let  $S_1 \subseteq [N]$  be the set of parties that sent a message in round 1. It truncates each  $\text{params}_j$  to the appropriate size for  $|S_1|$  parties and sets  $\text{params}$  as the concatenation of the truncated  $\text{params}_j$ 's for all  $j \in S_1$ . Let  $\mathcal{PK}$  denote  $\{pk_j\}_{j \in S_1}$ . Let  $\mathcal{CRS}$  denote  $\{\text{crs}_j\}_{j \in S_1}$ . Let  $\mathbb{A}'$  be the access structure induced by restricting  $\mathbb{A}$  to the parties in  $S_1$ . Let  $S_{\text{hon}}$  be the set of honest parties. Let  $T = \{\tau_j\}_{j \in S_{\text{hon}}}$ . Let  $E = \{\xi_j\}_{j \in S_{\text{hon}}}$ . Let  $S_{\text{corr}}^1$  be the set of corrupted parties that sent a message in round 1.  $\text{Sim}$  does the following:
  1. Run  $\text{Sim}_1(\text{params}, \mathcal{PK}, \mathbb{A}', S_{\text{corr}}^1, S_{\text{hon}})$  to obtain  $(\{ct_i\}_{i \in S_{\text{hon}}}, \text{state})$ , where  $\text{Sim}_1$  is the first algorithm of the TMFHE simulator.
  2. For each honest party  $P_i$ , run  $\text{SimProve}(\mathcal{CRS}, T, y_i)$  to compute  $\pi_i$  where  $y_i$  is the statement that there exists some input  $x$  and randomness  $r$  such that  $\text{TMFHE.Encrypt}(\text{params}, \mathcal{PK}, \mathbb{A}', x; r) = ct_i$ .
  3. Give  $(ct_i, \pi_i)$  as  $P_i$ 's round 2 message to  $\text{Adv}$  for  $i \in S_{\text{hon}}$ .
- **Query to Ideal Functionality:**

1. Parse the round 2 message (if one was sent) from  $P_j$  as  $(ct_j, \pi_j)$  and check that  $\text{NIZK.Verify}(\mathcal{CRS}, y_j, \pi_j) = 1$ . Let  $S_2 \subseteq S_1$  be the set of parties that sent a round 2 message that passed verification. For corrupted parties  $P_j$  in  $S_2$ , **Sim** runs  $\text{Ext}(\mathcal{CRS}, E, y_j, \pi_j)$  to extract a witness  $(x_j, r_j)$  used by **Adv** and sends  $x_j$  to the trusted party as  $P_j$ 's input. For corrupted parties  $P_j$  that are not in  $S_2$ , **Sim** sends  $0^\lambda$  to the trusted party.
  2. **Sim** receives the output  $b$  from the trusted party.
- **Round 3:** Let  $\mathcal{CT} = \{ct_j\}_{j \in S_2}$ . Let  $C'$  be the circuit induced by hardcoding the inputs to  $C$  corresponding to parties not in  $S_2$  as  $0^\lambda$ . Let  $S_{\text{corr}}^2$  be the set of corrupted parties that sent a round 2 message that passed verification. **Sim** does the following
    1. Run  $\text{Sim}_2(\text{state}, b, \hat{ct}, S_{\text{corr}}^1, S_{\text{hon}}, \{(x_i, r_i)\}_{i \in S_{\text{corr}}^2})$  to obtain  $\{p_j\}_{j \in S_{\text{hon}}}$ , where  $\text{Sim}_2$  is the second algorithm of the modified TMFHE simulator that uses the  $(x_i, r_i)$ 's of the corrupted parties round 2 messages to simulate and  $\hat{ct}$  is the evaluated ciphertext obtained by evaluating  $C'$  on the ciphertexts in  $\mathcal{CT}$ .
    2. For  $j \in S_{\text{hon}}$ , run  $\text{SimProve}(\mathcal{CRS}, T, z_j)$  to compute  $\pi'_j$  where  $z_j$  is the statement that there exists some randomness  $r, r'$  such that  $\text{TMFHE.KeyGen}(1^\lambda; r) = (pk_j, sk)$  and  $\text{TMFHE.PartDec}(j, sk, \hat{ct}; r') = p_j$ .
    3. For  $j \in S_{\text{hon}}$ , give  $(p_j, \pi'_j)$  as  $P_j$ 's round 3 message to **Adv**.
  - **Output to Honest Parties:** **Sim** tells the trusted party to send  $b$  to all honest parties.

Security with respect to this simulator follows from the properties of the simulation-extractable multi-string NIZK and the security of the underlying TMFHE scheme with respect to  $\text{Sim}_1, \text{Sim}_2$ .

**Reusability.** Reusability follows from the following:

1. The reusability of the protocol in [Section 6](#).
2. The NIZK in round 3 can be generated afresh for different invocations of the protocol while preserving security.

## References

- [ACGJ18] Prabhanjan Ananth, Arka Rai Choudhuri, Aarushi Goel, and Abhishek Jain. Round-optimal secure multiparty computation with honest majority. in *CRYPTO 2018*. Cryptology ePrint Archive, Report 2018/572, 2018. <https://eprint.iacr.org/2018/572>.
- [ACJ17] Prabhanjan Ananth, Arka Rai Choudhuri, and Abhishek Jain. A new approach to round-optimal secure multiparty computation. In *CRYPTO*, pages 468–499, 2017.
- [AJLA<sup>+</sup>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 *EUROCRYPT*, 2012.



- [BCD<sup>+</sup>09] Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In *Financial Cryptography and Data Security*, Berlin, Heidelberg, 2009.
- [Bei96] Amos Beimel. Phd thesis. *Israel Institute of Technology, Technion, Haifa, Israel*, 1996.
- [BFH<sup>+</sup>08] Zuzana Beerliová-Trubíniová, Matthias Fitzi, Martin Hirt, Ueli M. Maurer, and Vassilis Zikas. MPC vs. SFE: perfect security in a unified corruption model. In *TCC*, pages 231–250, 2008.
- [BGG<sup>+</sup>17] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M.R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2017, 2017.
- [BGI<sup>+</sup>] Saikrishna Badrinarayanan, Sanjam Garg, Yuval Ishai, Amit Sahai, and Akshay Wadia. Two-message witness indistinguishability and secure computation in the plain model from new assumptions. In *ASIACRYPT*.
- [BGJ<sup>+</sup>] Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Amit Sahai. Promise zero knowledge and its applications to round optimal MPC. *IACR Cryptology ePrint Archive*, 2017:1088. In *CRYPTO 2018*.
- [BGJ<sup>+</sup>17] Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Dakshita Khurana, and Amit Sahai. Round optimal concurrent MPC via strong simulation. In *TCC*, pages 743–775, 2017.
- [BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325, 2012.
- [BHP17] Zvika Brakerski, Shai Halevi, and Antigoni Polychroniadou. Four round secure computation without setup. In *Theory of Cryptography*, 2017.
- [BL18] Fabrice Benhamouda and Huijia Lin. k-round mpc from k-round ot via garbled interactive circuits. *EUROCRYPT*, 2018.
- [BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC*, 1988.
- [BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106, 2011.
- [BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *CRYPTO*, pages 505–524, 2011.
- [CKKC13] Seung Geol Choi, Jonathan Katz, Ranjit Kumaresan, and Carlos Cid. Multi-client non-interactive verifiable computation. In *Theory of Cryptography*, 2013.

- [CL14] Ran Cohen and Yehuda Lindell. Fairness versus guaranteed output delivery in secure multiparty computation. In *ASIACRYPT*, 2014.
- [Cle86] R. Cleve. Limits on the security of coin flips when half the processors are faulty. In *STOC*, New York, NY, USA, 1986.
- [CM15] Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled fhe from learning with errors. In *CRYPTO*, 2015.
- [COSV17a] Michele Ciampi, Rafail Ostrovsky, Siniscalchi, and Ivan Visconti. Delayed-input non-malleable zero knowledge and multi-party coin tossing in four rounds. In *TCC*, 2017.
- [COSV17b] Michele Ciampi, Rafail Ostrovsky, Siniscalchi, and Ivan Visconti. Round-optimal secure two-party computation from trapdoor permutations. In *TCC*, 2017.
- [CPA<sup>+</sup>08] Ashish Choudhary, Arpita Patra, B. V. Ashwinkumar, K. Srinathan, and C. Pandu Rangan. Perfectly reliable and secure communication tolerating static and mobile mixed adversary. In *ICITS*, pages 137–155, 2008.
- [DI05] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *CRYPTO*, Berlin, Heidelberg, 2005.
- [DSDCO<sup>+</sup>01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, pages 566–598, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.
- [DSP92] A. De Santis and G. Persiano. Zero-knowledge proofs of knowledge without interaction. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, SFCS '92, pages 427–436, Washington, DC, USA, 1992. IEEE Computer Society.
- [FHM98] Matthias Fitzi, Martin Hirt, and Ueli M. Maurer. Trading correctness for privacy in unconditional multi-party computation (extended abstract). In *CRYPTO*, pages 121–136, 1998.
- [FHM99] Matthias Fitzi, Martin Hirt, and Ueli M. Maurer. General adversaries in unconditional multi-party computation. In *ASIACRYPT*, pages 232–246, 1999.
- [FKN94] Uri Feige, Joe Killian, and Moni Naor. A minimal model for secure computation (extended abstract). In *STOC*, New York, NY, USA, 1994.
- [Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.
- [GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, pages 1–17, 2013.

- [GGH<sup>+</sup>13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, pages 40–49. IEEE Computer Society, 2013.
- [GIKR02] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. On 2-round secure multiparty computation. In *CRYPTO*, 2002.
- [GKP17] Sanjam Garg, Susumu Kiyoshima, and Omkant Pandey. On the exact round complexity of self-composable two-party computation. In *EUROCRYPT*, pages 194–224, 2017.
- [GLS15] S. Dov Gordon, Feng-Hao Liu, and Elaine Shi. Constant-round MPC with fairness and guarantee of output delivery. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 63–82, 2015.
- [GMPP16] Sanjam Garg, Pratyay Mukherjee, Omkant Pandey, and Antigoni Polychroniadou. The exact round complexity of secure computation. In *EUROCRYPT*, pages 448–476, 2016.
- [GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *STOC*, pages 218–229. ACM, 1987.
- [GO07] Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007*, pages 323–341, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
- [Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.
- [GOS12] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11:1–11:35, June 2012.
- [Goy11] Vipul Goyal. Constant round non-malleable protocols using one way functions. In *STOC*, 2011.
- [GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, pages 75–92, 2013.
- [HHPV17] Shai Halevi, Carmit Hazay, Antigoni Polychroniadou, and Muthuramakrishnan Venkatasubramanian. Round-optimal secure multi-party computation. 2017. In *CRYPTO 2018*.
- [Hir01] Martin Hirt. *Multi party computation: efficient protocols, general adversaries, and voting*. PhD thesis, ETH Zurich, Zürich, Switzerland, 2001.
- [HMZ08] Martin Hirt, Ueli M. Maurer, and Vassilis Zikas. MPC vs. SFE : Unconditional and computational security. In *ASIACRYPT*, pages 1–18, 2008.

- [HT13] Martin Hirt and Daniel Tschudi. Efficient general-adversary multi-party computation. In *ASIACRYPT*, pages 181–200, 2013.
- [IK97] Yuval Ishai and Eyal Kushilevitz. Private simultaneous messages protocols with applications. In *ISTCS*, Washington, DC, USA, 1997.
- [IKLP06] Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. On combining privacy with guaranteed output delivery in secure multiparty computation. In *CRYPTO*, pages 483–500, 2006.
- [IKP10] Yuval Ishai, Eyal Kushilevitz, and Anat Paskin. Secure multiparty computation with minimal interaction. In *CRYPTO*, Berlin, Heidelberg, 2010.
- [JKKR17] Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Ron Rothblum. Distinguisher-dependent simulation in two rounds and its applications. In *CRYPTO*, pages 158–189, 2017.
- [JRS17] Aayush Jain, Peter M. R. Rasmussen, and Amit Sahai. Threshold fully homomorphic encryption. Cryptology ePrint Archive, Report 2017/257, 2017. <https://eprint.iacr.org/2017/257>.
- [KMR11] Seny Kamara, Payman Mohassel, and Mariana Raykova. Outsourcing multiparty computation. Cryptology ePrint Archive, Report 2011/272, 2011. <https://eprint.iacr.org/2011/272>.
- [KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *CRYPTO*, pages 335–354, 2004.
- [KRR14] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to delegate computations: the power of no-signaling proofs. In *STOC*, pages 485–494, 2014.
- [LO14] Joshua Lampkins and Rafail Ostrovsky. Communication-efficient MPC for general adversary structures. In *SCN*, pages 155–174, 2014.
- [LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *STOC*, pages 1219–1234, 2012.
- [LW11] Allison B. Lewko and Brent Waters. Decentralizing attribute-based encryption. In *EUROCRYPT*, 2011.
- [MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multikey fhe. In *EUROCRYPT*, 2016.
- [NPS99] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *EC*, 1999.
- [Pas04] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 232–241, 2004.

- [PS16] Chris Peikert and Sina Shiehian. Multi-key FHE from lwe, revisited. In *TCC Part II*, 2016.
- [PW10] Rafael Pass and Hoeteck Wee. Constant-round non-malleable commitments from sub-exponential one-way functions. In *EUROCRYPT*, pages 638–655, 2010.
- [SPCR09] Kannan Srinathan, Arpita Patra, Ashish Choudhary, and C. Pandu Rangan. Unconditionally secure message transmission in arbitrary directed synchronous networks tolerating generalized mixed adversary. In *ASIACCS*, pages 171–182, 2009.
- [Wee10] Hoeteck Wee. Black-box, round-efficient secure computation via non-malleability amplification. In *FOCS*, pages 531–540, 2010.
- [Yao82] Andrew C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pages 160–164, Washington, DC, USA, 1982. IEEE Computer Society.
- [Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.
- [ZHM09] Vassilis Zikas, Sarah Hauser, and Ueli M. Maurer. Realistic failures in secure multi-party computation. In *TCC*, pages 274–293, 2009.
- [Zik10] Vassilis Zikas. *Generalized corruption models in secure multi-party computation*. PhD thesis, ETH Zurich, 2010.

## A Deferred Preliminaries

### A.1 Multi-Key FHE

In this section, we present the definition of multi-key fully homomorphic encryption in the plain model with distributed setup as found in [BHP17].

**Definition 2** (MFHE). *A multi-key fully homomorphic encryption scheme is a tuple of PPT algorithms*

$$\text{MFHE} = (\text{DistSetup}, \text{KeyGen}, \text{Enc}, \text{Eval}, \text{PartDec}, \text{FinDec})$$

*satisfying the following specifications:*

$\text{params}_i \leftarrow \text{DistSetup}(1^\lambda, 1^d, 1^N, i)$ : *It takes as input a security parameter  $\lambda$ , a circuit depth  $d$ , the maximal number of parties  $N$ , and a party index  $i$ . It outputs the public parameters  $\text{params}_i$  associated with the  $i$ th party, where  $\text{params}_i \in \{0, 1\}^{\text{poly}(\lambda, d, N)}$  for some polynomial  $\text{poly}$ . We assume implicitly that all the following algorithms take the public parameters of all parties as input, where we define  $\text{params} = \text{params}_1 || \dots || \text{params}_N$ .*

$(pk, sk) \leftarrow \text{KeyGen}(\text{params})$ : *It takes as input the public parameters  $\text{params}$  and outputs a key pair  $(pk, sk)$ .*

$ct \leftarrow \text{Encrypt}(pk, m)$ : *It takes as input a public key  $pk$  and a plaintext  $m \in \{0, 1\}^\lambda$  and outputs a ciphertext  $ct$ . Throughout, we will assume that all ciphertexts include the public key(s) that they are encrypted under.*

$\hat{ct} \leftarrow \text{Eval}(C, ct_1, \dots, ct_\ell)$ : It takes as input a boolean circuit  $C: (\{0, 1\}^\lambda)^\ell \rightarrow \{0, 1\} \in \mathcal{C}_\lambda$  of depth  $\leq d$  and ciphertexts  $ct_1, \dots, ct_\ell$  for  $\ell \leq N$ . It outputs an evaluated ciphertext  $\hat{ct}$ .

$p_i \leftarrow \text{PartDec}(i, sk, \hat{ct})$ : It takes as input an index  $i$ , a secret key  $sk$  and an evaluated ciphertext  $\hat{ct}$  and outputs a partial decryption  $p_i$ .

$\hat{\mu} \leftarrow \text{FinDec}(p_1, \dots, p_\ell)$ : It takes as input partial decryptions  $p_1, \dots, p_\ell$  and deterministically outputs a plaintext  $\hat{\mu} \in \{0, 1, \perp\}$ .

We require that for any parameters  $\{\text{params}_i \leftarrow \text{Setup}(1^\lambda, 1^d, 1^N, i)\}_{i \in [N]}$ , any key pairs  $\{(pk_i, sk_i) \leftarrow \text{KeyGen}(\text{params})\}_{i \in [N]}$ , any plaintexts  $m_1, \dots, m_\ell \in \{0, 1\}^\lambda$  for  $\ell \leq N$ , any sequence  $I_1, \dots, I_\ell \in [N]$  of indices, and any boolean circuit  $C: \{0, 1\}^\ell \rightarrow \{0, 1\} \in \mathcal{C}_\lambda$  of depth  $\leq d$ , the following is satisfied:

**Correctness.** Let  $ct_i = \text{Encrypt}(pk_{I_i}, m_i)$  for  $1 \leq i \leq \ell$ ,  $\hat{ct} = \text{Eval}(C, ct_1, \dots, ct_\ell)$ , and  $p_i = \text{PartDec}(i, sk_{I_i}, \hat{ct})$  for all  $i \in [\ell]$ . With all but negligible probability in  $\lambda$  over the coins of  $\text{Setup}$ ,  $\text{KeyGen}$ ,  $\text{Encrypt}$ , and  $\text{PartDec}$ ,

$$\text{FinDec}(p_1, \dots, p_\ell) = C(m_1, \dots, m_\ell).$$

**Compactness of Ciphertexts.** There exists a polynomial,  $\text{poly}$ , such that  $|ct| \leq \text{poly}(\lambda, d, N)$  for any ciphertext  $ct$  generated from the algorithms of MFHE.

**Semantic Security of Encryption.** Any PPT adversary  $\mathcal{A}$  has only negligible advantage as a function of  $\lambda$  over the coins of all the algorithms in the following game:

1. On input the security parameter  $1^\lambda$ , a circuit depth  $1^d$ , and the number of parties  $1^N$ , the adversary  $\mathcal{A}$  outputs a non-corrupted party  $i$ .
2. Run  $\text{DistSetup}(1^\lambda, 1^d, 1^N, i) \rightarrow \text{params}_i$ . The adversary is given  $\text{params}_i$ .
3. The adversary outputs  $\{\text{params}_j\}_{j \in [N] \setminus \{i\}}$ .
4.  $\text{params}$  is set to  $\text{params}_1 || \dots || \text{params}_N$ . Run  $\text{KeyGen}(\text{params}) \rightarrow (pk_i, sk_i)$ . The adversary is given  $pk_i$ .
5. The adversary outputs two messages  $m_0, m_1 \in \{0, 1\}^\lambda$ .
6. The adversary is given  $ct \leftarrow \text{Encrypt}(pk_i, m_b)$  for a random  $b \in \{0, 1\}$ .
7. The adversary outputs  $b'$  and wins if  $b = b'$ .

**Simulation Security.** There exists a stateful PPT algorithm  $\text{Sim}$  such that for any PPT adversary  $\mathcal{A}$ , we have that the experiments  $\text{Expt}_{\mathcal{A}, \text{Real}}(1^\lambda, 1^d, 1^N)$  and  $\text{Expt}_{\mathcal{A}, \text{Sim}}(1^\lambda, 1^d, 1^N)$  as defined below are statistically close as a function of  $\lambda$  over the coins of all the algorithms. The experiments are defined as follows:

$\text{Expt}_{\mathcal{A}, \text{Real}}(1^\lambda, 1^d, 1^N)$ :

1. On input the security parameter  $1^\lambda$ , a circuit depth  $1^d$ , and the number of parties  $1^N$ , the adversary  $\mathcal{A}$  a non-corrupted party  $i$ .
2. Run  $\text{DistSetup}(1^\lambda, 1^d, 1^N, i) \rightarrow \text{params}_i$ . The adversary is given  $\text{params}_i$ .
3. The adversary outputs  $\{\text{params}_j\}_{j \in [N] \setminus \{i\}}$ .

4.  $\text{params}$  is set to  $\text{params}_1 || \dots || \text{params}_N$ . Sample  $N - 1$  key pairs  $\text{KeyGen}(\text{params}) \rightarrow (pk_j, sk_j)$  for  $j \in [N] \setminus \{i\}$ . The adversary is given  $\{(pk_j, sk_j)\}_{j \in [N] \setminus \{i\}}$ .
5. The adversary outputs randomness  $r_i^{\text{KeyGen}}$  used to generate  $(pk_i, sk_i)$ ,  $m_1, \dots, m_\ell \in \{0, 1\}^\lambda$ ,  $I_1, \dots, I_\ell \in [N]$ , and a set of circuits  $\{C_k : (\{0, 1\}^\lambda)^\ell \rightarrow \{0, 1\}\}_{k \in [t]}$  with each  $C_k \in \mathcal{C}$  for some  $\ell \leq N$  and some  $t = \text{poly}(\lambda, d, N)$ .
6. Set  $(pk_i, sk_i) \leftarrow \text{KeyGen}(\text{params}; r_i^{\text{KeyGen}})$ . The adversary is given  $ct_j \leftarrow \text{Enc}(pk_{I_j}, m_j)$  for  $1 \leq j \leq \ell$  and the evaluated ciphertexts  $\hat{ct}_k \leftarrow \text{Eval}(C_k, ct_1, \dots, ct_\ell)$  for all  $k \in [t]$ .
7. The adversary is given  $p_{i,k} \leftarrow \text{PartDec}(i, sk_i, \hat{ct}_k)$  for all  $k \in [t]$ .
8.  $\mathcal{A}$  outputs  $\text{out}$ . The output of the experiment is  $\text{out}$ .

$\text{Expt}_{\mathcal{A}, \text{Sim}}(1^\lambda, 1^d, 1^N)$ :

1. On input the security parameter  $1^\lambda$ , a circuit depth  $1^d$ , and the number of parties  $1^N$ , the adversary  $\mathcal{A}$  a non-corrupted party  $i$ .
2. Run  $\text{DistSetup}(1^\lambda, 1^d, 1^N, i) \rightarrow \text{params}_i$ . The adversary is given  $\text{params}_i$ .
3. The adversary outputs  $\{\text{params}_j\}_{j \in [N] \setminus \{i\}}$ .
4.  $\text{params}$  is set to  $\text{params}_1 || \dots || \text{params}_N$ . Sample  $N - 1$  key pairs  $\text{KeyGen}(\text{params}) \rightarrow (pk_j, sk_j)$  for  $j \in [N] \setminus \{i\}$ . The adversary is given  $\{(pk_j, sk_j)\}_{j \in [N] \setminus \{i\}}$ .
5. The adversary outputs randomness  $r_i^{\text{KeyGen}}$  used to generate  $(pk_i, sk_i)$ ,  $m_1, \dots, m_\ell \in \{0, 1\}^\lambda$ ,  $I_1, \dots, I_\ell \in [N]$ , and a set of circuits  $\{C_k : (\{0, 1\}^\lambda)^\ell \rightarrow \{0, 1\}\}_{k \in [t]}$  with each  $C_k \in \mathcal{C}$  for some  $\ell \leq N$  and some  $t = \text{poly}(\lambda, d, N)$ .
6. Set  $(pk_i, sk_i) \leftarrow \text{KeyGen}(\text{params}; r_i^{\text{KeyGen}})$ . The adversary is given  $ct_j \leftarrow \text{Enc}(pk_{I_j}, m_j)$  for  $1 \leq j \leq \ell$  and the evaluated ciphertexts  $\hat{ct}_k \leftarrow \text{Eval}(C_k, ct_1, \dots, ct_\ell)$  for all  $k \in [t]$ .
7. Define  $\mu_k = C_k(m_1, \dots, m_\ell)$ . For all  $k \in [t]$ , the adversary is given  $p_{i,k} \leftarrow \text{Sim}(\mu_k, \hat{ct}_k, i, \{sk_j\}_{j \in [N] \setminus \{i\}})$ .
8.  $\mathcal{A}$  outputs  $\text{out}$ . The output of the experiment is  $\text{out}$ .

## A.2 Multi-String NIZK

We adapt the definition from [GO07]. Let  $\mathcal{R}$  be an efficiently computable binary relation and  $\mathcal{L}$  an NP-language of statements  $x$  such that  $(x, w) \in \mathcal{R}$  for some witness  $w$ .

**Definition 3** (Multi-String NIZK). *A multi-string NIZK using  $N$  strings for a language  $\mathcal{L}$  is a tuple of PPT algorithms*

$$\text{NIZK} = (\text{Gen}, \text{Prove}, \text{Verify})$$

*satisfying the following specifications:*

$\text{crs} \leftarrow \text{Gen}(1^\lambda)$ : *It takes as input the security parameter  $\lambda$  and outputs a uniformly random string  $\text{crs}$ .*

$\pi \leftarrow \text{Prove}(\text{CRS}, x, w)$ : *It takes as input a set of  $N$  random strings  $\text{CRS}$ , a statement  $x$ , and a witness  $w$ . It outputs a proof  $\pi$ .*

$\{0, 1\} \leftarrow \text{Verify}(\text{CRS}, x, \pi)$ : *It takes as input a set of  $N$  random strings  $\text{CRS}$ , a statement  $x$ , and a proof  $\pi$ . It outputs 1 if it accepts  $\pi$  and 0 if it rejects it.*

We require that the algorithms satisfy the following properties for all non-uniform PPT adversaries  $\mathcal{A}$ :

**Perfect Completeness.**

$$\Pr \left[ \begin{array}{l} S := \emptyset; (\mathcal{CRS}, x, w) \leftarrow \mathcal{A}^{\text{Gen}}; \pi \leftarrow \text{Prove}(\mathcal{CRS}, x, w) : \\ \text{Verify}(\mathcal{CRS}, x, \pi) = 0 \text{ and } (x, w) \in \mathcal{R} \end{array} \right] = 0,$$

where  $\text{Gen}$  is an oracle that on a query  $q$  outputs  $\text{crs}_q \leftarrow \text{Gen}(1^\lambda)$  and sets  $S := S \cup \{\text{crs}_q\}$ . Note that this says that even if the adversary arbitrarily picks all the random strings, perfect completeness still holds.

**Soundness.**

$$\Pr \left[ \begin{array}{l} S := \emptyset; (\mathcal{CRS}, x, \pi) \leftarrow \mathcal{A}^{\text{Gen}} : \\ \text{Verify}(\mathcal{CRS}, x, \pi) = 1 \text{ and } x \notin \mathcal{L} \text{ and } |\mathcal{CRS} \cap S| > N/2 \end{array} \right] \leq \text{negl}(\lambda),$$

where  $\text{Gen}$  is an oracle that on a query  $q$  outputs  $\text{crs}_q \leftarrow \text{Gen}(1^\lambda)$  and sets  $S := S \cup \{\text{crs}_q\}$ . Note that this says that as long as at least half of the random strings are honestly generated, the adversary cannot forge a proof except with negligible probability.

**Composable Zero-Knowledge.** There exist PPT algorithms  $\text{SimGen}, \text{SimProve}$  such that

$$\Pr[\text{crs} \leftarrow \text{Gen}(1^\lambda) : \mathcal{A}(\text{crs}) = 1] \cong_c \Pr[(\text{crs}, \tau) \leftarrow \text{SimGen}(1^\lambda) : \mathcal{A}(\text{crs}) = 1]$$

and

$$\Pr \left[ \begin{array}{l} S := \emptyset; (\mathcal{CRS}, x, w) \leftarrow \mathcal{A}^{\text{SimGen}}(1^\lambda); \pi \leftarrow \text{Prove}(\mathcal{CRS}, x, w) : \\ \mathcal{A}(\pi) = 1 \text{ and } (x, w) \in \mathcal{R} \text{ and } |\mathcal{CRS} \cap S| > N/2 \end{array} \right]$$

$$\cong_c$$

$$\Pr \left[ \begin{array}{l} S := \emptyset; (\mathcal{CRS}, x, w) \leftarrow \mathcal{A}^{\text{SimGen}}(1^\lambda); \pi \leftarrow \text{SimProve}(\mathcal{CRS}, T, x) : \\ \mathcal{A}(\pi) = 1 \text{ and } (x, w) \in \mathcal{R} \text{ and } |\mathcal{CRS} \cap S| > N/2 \end{array} \right],$$

where  $T$  is the set containing all simulation trapdoors  $\tau$  generated by  $\text{SimGen}$ . Note that this is saying that random strings with simulation trapdoors can be generated that are indistinguishable from honestly generated random strings and that using these trapdoors, it is possible to simulate a proof that is indistinguishable from a real proof even to an adversary that possesses all the simulation trapdoors.

In this work, we will deal with multi-string NIZKs that are simulation-extractable. Informally, this means that it is possible to extract a witness from an adversary's proof even if the adversary is allowed to see many simulated proofs. Formally, we have the following definition from [GO07].

**Definition 4** (Simulation-Extractable Multi-String NIZK). A simulation-extractable multi-string NIZK is a multi-string NIZK with the following additional property.



**Simulation-Extractability.** *There exist PPT algorithms  $\text{ExtGen}, \text{Ext}$  such that  $\text{ExtGen}(1^\lambda)$  outputs  $(\text{crs}, \tau, \xi)$ , a random string, a simulation trapdoor, and an extraction key, such that the output distribution  $(\text{crs}, \tau)$  is identical to that of  $\text{SimGen}$  and*

$$\Pr \left[ \begin{array}{l} S := \emptyset; Q := \emptyset; (\mathcal{CRS}, x, \pi) \leftarrow \mathcal{A}^{\text{ExtGen}', \text{SimProve}}(1^\lambda); \\ w \leftarrow \text{Ext}(\mathcal{CRS}, E, x, \pi) : \\ (x, \pi) \notin Q \text{ and } (x, w) \notin \mathcal{R} \text{ and } \text{Verify}(\mathcal{CRS}, x, \pi) = 1 \\ \text{and } |\mathcal{CRS} \cap S| > N/2 \end{array} \right] \leq \text{negl}(\lambda),$$

where  $\text{ExtGen}'$  is an oracle that runs  $\text{ExtGen}$  to generate  $(\text{crs}, \tau, \xi)$ , outputs  $(\text{crs}, \tau)$  and sets  $S := S \cup \{\text{crs}\}$ ,  $\text{SimProve}$  outputs a proof  $\pi$  for a statement  $x$  given the set of simulation trapdoors and sets  $Q := Q \cup \{(x, \pi)\}$ , and  $E$  is the set of the  $\xi$ 's generated by  $\text{ExtGen}$ .

[GO07] show the following.

**Theorem 10** ([GO07]). *Assuming Zaps, there exist simulation-extractable multi-string NIZKs.*

### A.3 Statistical Distance

In this section, we state results related to the statistical closeness of distributions that will be used in the security proof of our TMFHE construction. This section was adapted from one in [JRS17], and we defer the reader to their paper for the proofs of these results.

**Definition 5** (Statistical Distance). *Let  $E$  be a finite set,  $\Omega$  a probability space, and  $X, Y: \Omega \rightarrow E$  random variables. We define the statistical distance between  $X$  and  $Y$  to be the function  $\Delta$  defined by*

$$\Delta(X, Y) = \frac{1}{2} \sum_{e \in E} \left| \Pr_X(X = e) - \Pr_Y(Y = e) \right|.$$

**Proposition 1** ([JRS17]). *Let  $E$  be a finite set,  $\Omega$  a probability space, and let  $\{X_s^b\}_{s \in S, b \in \{0,1\}}$  be a family of random variables  $X_s^b: \Omega \rightarrow E$  indexed by an element  $s \in S$  and a state  $b \in \{0,1\}$ . Further, assume that for every  $s \in S$  we have  $\Delta(X_s^0, X_s^1) \leq \delta$ . Now, for a stateful PPT algorithm  $\mathcal{A}$ , define the following experiment:*

$\text{Expt}_{\mathcal{A}, b, m}$ :

- The algorithm  $\mathcal{A}$  issues  $m$  queries. Each query is an element  $s_i \in S$  and after each query,  $\mathcal{A}$  receives in return  $x_i \leftarrow X_{s_i}^b$  sampled independently of the other samples.
- The output of the experiment is  $(s_1, x_1), \dots, (s_m, x_m)$ .

Then  $\Delta(\text{Expt}_{\mathcal{A}, 0, m}, \text{Expt}_{\mathcal{A}, 1, m}) \leq m\delta$ .

Another useful lemma is the following, which demonstrates a technique to “smudge” or hide the presence of error ( $e_1$  in the lemma) by adding a much larger error. While no values are specifically given in the statement of the lemma,  $B_1$  is meant to be negligible compared to  $B_2$  such that the statistical distance between the two distributions is negligible.

**Lemma 7** (Smudging Lemma [MW16]). *Let  $B_1, B_2 \in \mathbb{N}$ . For any  $e_1 \in [-B_1, B_1]$  let  $E_1$  and  $E_2$  be independent random variables uniformly distributed on  $[-B_2, B_2]$  and define the two stochastic variables  $X_1 = E_1 + e_1$  and  $X_2 = E_2$ . Then  $\Delta(E_1, E_2) < B_1/B_2$ .*

## A.4 Secret Sharing

Throughout this paper we will use secret sharing terminology and techniques. This section provides an introduction to the basic terms, notation, and concepts that will be needed later. Large portions of this section were taken verbatim from [JRS17].

### A.4.1 Secret Sharing Basics.

We assume that the reader is familiar with the notion of an information theoretic secret sharing scheme and, in particular, the Shamir secret sharing scheme. We now describe concepts about access structures and specific secret sharing schemes that we consider in this paper. We adapt some definitions from [LW11].

**Definition 6** (Monotone Access Structure). *Let  $P = \{P_1, \dots, P_N\}$  be a set of parties. A collection  $\mathbb{A} \subseteq \mathcal{P}(P)$  is monotone if whenever we have sets  $B, C$  satisfying  $B \in \mathbb{A}$  and  $B \subseteq C \subseteq P$  then  $C \in \mathbb{A}$ . A monotone access structure on  $P$  is a monotone collection  $\mathbb{A} \subseteq \mathcal{P}(P) \setminus \emptyset$ . The sets in  $\mathbb{A}$  are called the valid sets and the sets in  $\mathcal{P}(P) \setminus \mathbb{A}$  are called the invalid sets.*

**Definition 7** (Restriction of Access Structure). *Let  $P = \{P_1, \dots, P_N\}$  be a set of parties and  $\mathbb{A}$  be an access structure over these parties. Let  $P_S \subseteq P$  be a subset of these parties. We say that  $\mathbb{A}'$  is the access structure induced by restricting  $\mathbb{A}$  to the parties in  $P_S$  if  $\mathbb{A}'$  is an access structure on  $P_S$  such that a set  $A \in \mathbb{A}'$  for some  $A \subseteq P_S$  if and only if  $A \in \mathbb{A}$ .*

For ease of notation, we will generally identify a party with its index. Further, since this presentation will only consider monotone access structures, the terms monotone access structure and simply access structure will be used interchangeably throughout the text.

**Notation 1.** *Let  $P = \{P_1, \dots, P_N\}$  be a set of parties and let  $S$  be a subset of  $P$ . We denote by  $\mathbf{X}_S$  the vector  $\mathbf{X}_S = (x_1, \dots, x_N)$  where  $x_i = 1$  if  $P_i \in S$  and  $x_i = 0$  otherwise.*

**Definition 8** (Efficient Access Structure). *Let  $P = \{P_1, \dots, P_N\}$  be a set of parties and  $\mathbb{A} \subseteq \mathcal{P}(P)$  a monotone access structure on  $P$ . We say that  $\mathbb{A}$  is efficient if there exists a polynomial size circuit  $f_{\mathbb{A}}: \{0, 1\}^N \rightarrow \{0, 1\}$  such that for all  $S \subseteq P$ ,  $f_{\mathbb{A}}(\mathbf{X}_S) = 1$  if and only if  $S \in \mathbb{A}$ .*

**Definition 9** (Class of Monotone Access Structures). *Let  $P = \{P_1, \dots, P_N\}$  be a set of parties. A class of monotone access structures is a collection  $\mathbb{S} = \{\mathbb{A}_1, \dots, \mathbb{A}_t\} \subseteq \mathcal{P}(\mathcal{P}(P))$  of monotone access structures on  $P$ .*

Being interested in secret sharing, we will only consider efficient access structures in this work. One of the most canonical classes of access structures is the  $t$ -out-of- $n$  class.

**Definition 10** ( $t$ -out-of- $n$  secret sharing). *Let  $P = \{P_1, \dots, P_N\}$  be a set of parties. An access structure  $\mathbb{A}$  is a  $t$ -out-of- $n$  access structure if for every  $S \subseteq P$ ,  $S \in \mathbb{A}$  if and only if  $|S| \geq t$ .*

A more general form of secret sharing is linear secret sharing.

**Definition 11** (Linear Secret Sharing Scheme (LSSS)). *Let  $P = \{P_1, \dots, P_N\}$  be a set of parties. The class of access structures LSSS (or LSSS $_N$  to emphasize the number of parties) consists of all access structures  $\mathbb{A}$  such that there exists a secret sharing scheme  $\Pi$  satisfying:*

1. For a prime  $p$ , the share of each party  $P_i$  is a vector  $\vec{w}_i \in \mathbb{Z}_p^{n_i}$  for some  $n_i \in \mathbb{N}$ .

2. There exists a matrix  $M \in \mathbb{Z}_p^{\ell \times n}$ ,  $\ell = \sum_{i=1}^N n_i$  called the share matrix for  $\Pi$  with size polynomial in the number of parties and such that for a secret  $s$ , the shares are generated as follows. Values  $r_2, \dots, r_n \in \mathbb{Z}_p$  are chosen at random and the vector  $\vec{v} = M \cdot (s, r_2, \dots, r_n)^T$  is generated. Now, denote by  $T_i \subseteq [\ell]$ ,  $1 \leq i \leq N$  a partition of  $[\ell]$  such that  $T_i$  has size  $|T_i| = n_i$  and is associated with party  $P_i$ . The share of  $P_i$  is the vector  $\vec{w}_i = (v_i)_{i \in T_i}$ .

3. For any set  $S \subseteq P$ ,  $S \in \mathbb{A}$  if and only if

$$(1, 0, \dots, 0) \in \text{span}(\{M[i]\}_{i \in \bigcup_{j \in S} T_j})$$

over  $\mathbb{Z}_p$  where  $M[i]$  denotes the  $i$ th row of  $M$ .

We denote by  $\text{LSSS}_N$  the class of linear secret sharing schemes on  $N$  parties.

Note that keeping with the notation of the LSSS definition above, any set of parties  $S \subseteq P$  such that  $S \in \mathbb{A}$  can recover the secret by finding coefficients  $\{c_i\}_{i \in \bigcup_{j \in S} T_j}$  satisfying

$$\sum_{i \in \bigcup_{j \in S} T_j} c_i M[i] = (1, 0, \dots, 0).$$

Given such coefficients, the secret can be recovered simply by computing

$$s = \sum_{i \in \bigcup_{j \in S} T_j} c_i v_i.$$

Since such coefficients can be found in time polynomial in the size of  $M$  using linear algebra, LSSS is a class of efficient access structures [Bei96]. Further, LSSS has the property that it information theoretically hides the value  $s$ , i.e. for any secrets  $s_0$  and  $s_1$ , it holds that the distributions of shares  $\{\vec{w}_i\}_{i \in S}$  for a set  $S \notin \mathbb{A}$ , are identical.

In our application of linear secret sharing, we will always be sharing a vector over  $\mathbb{Z}_p$ ,  $\vec{s} \in \mathbb{Z}_p^n$  instead of just a single element of  $\mathbb{Z}_p$ . Simply linearly secret sharing each entry of the vector  $\vec{s}$  using fresh randomness for each entry yields shares  $\vec{s}_1, \dots, \vec{s}_\ell \in \mathbb{Z}_p^n$ . It is easy to see that the secret  $\vec{s} \in \mathbb{Z}_p^n$  can now be reconstructed as a linear combination of the shares  $\vec{s}_i$  using the same coefficients as for a single field element. Further, information theoretical hiding is maintained.

#### A.4.2 $\{0, 1\}$ -LSSS and $\{0, 1\}$ -LSSSD.

For the purposes of this paper, we will need a more restricted class of access structures. The access structures of the class  $\{0, 1\}$ -LSSS are those that can be realized as LSSS schemes such that each party only has one share and such that it always is possible to only use recovery coefficients  $c_i \in \{0, 1\}$ .

**Definition 12** ( $\{0, 1\}$ -Linear Single Share Scheme ( $\{0, 1\}$ -LSSS)). *Let  $P = \{P_1, \dots, P_N\}$  be a set of parties. The set  $\{0, 1\}$ -LSSS $_N \subseteq \text{LSSS}_N$  is the collection of access structures  $\mathbb{A} \in \text{LSSS}_N$  such that there exists an efficient linear secret sharing scheme  $\Pi$  for  $\mathbb{A}$  satisfying the following:*

1. For a prime  $p$ , the share of each party  $P_i$  consists of a single element  $w_i \in \mathbb{Z}_p$ .
2. Let  $s$  be a secret and let  $w_i \in \mathbb{Z}_p$  be the share of party  $P_i$  for each  $i$ . For every valid set  $S \in \mathbb{A}$ , there exist a subset  $S' \subseteq S$  such that  $s = \sum_{i \in S'} w_i$ .

In our application, we will need  $\{0, 1\}$ -LSSS schemes that work over a certain prime  $q$  corresponding to the modulus of an FHE scheme. The constructions of later sections will be designed in a way that allows for the access structure to work over any modulus, but for now we will denote by  $\{0, 1\}$ -LSSS $^q$  the set of access structures contained in  $\{0, 1\}$ -LSSS that can be realized as secret sharing schemes by a share matrix over  $\mathbb{Z}_q$ .

That every access structure  $\mathbb{A} \in \{0, 1\}$ -LSSS is efficient follows directly from the efficiency of the LSSS class. However, it is not obvious that the set  $S'$  of the above definition can be found efficiently given any  $S \subseteq P$ . To see that this is indeed the case, we first establish a lemma.

**Definition 13** (Maximal Invalid Share Set). *Let  $P = \{P_1, \dots, P_N\}$  be a set of parties and  $\mathbb{A}$  be a monotone access structure on  $P$ . A set  $S \subseteq P$  is a maximal invalid share set if  $S \notin \mathbb{A}$  but for every  $p \in P \setminus S$ ,  $S \cup \{p\} \in \mathbb{A}$ .*

**Definition 14** (Minimal Valid Share Set). *Let  $P = \{P_1, \dots, P_N\}$  be a set of parties and  $\mathbb{A}$  be a monotone access structure on  $P$ . A set  $S \subseteq P$  is a minimal valid share set if  $S \in \mathbb{A}$  and for every  $S' \subsetneq S$ ,  $S' \notin \mathbb{A}$ .*

Although the following lemma is trivial to show it will turn out to be a useful observation both for the efficiency of reconstruction of  $\{0, 1\}$ -LSSS and for our construction.

**Lemma 8** ([JRS17]). *Let  $P = \{P_1, \dots, P_N\}$  be a set of parties,  $\mathbb{A} \in \{0, 1\}$ -LSSS, and  $\Pi$  be a linear secret sharing scheme as specified in the definition of  $\{0, 1\}$ -LSSS. Let  $s$  be a secret, let  $w_i \in \mathbb{Z}_p$  be the share of party  $P_i$  for each  $i$ , and let  $S \subseteq P$  be a minimal valid share set of  $\mathbb{A}$ . Then  $s = \sum_{i \in S} w_i$ .*

Finally, the following lemma shows that given a linear secret sharing scheme  $P_i$  for  $\mathbb{A} \in \{0, 1\}$ -LSSS, we can find recovery coefficients efficiently. However, it is worth noting that this does not mean that deciding whether an access structure belongs to  $\{0, 1\}$ -LSSS is feasible. In our applications we will instead specifically construct secret sharing schemes that belong to  $\{0, 1\}$ -LSSS.

**Lemma 9** ([JRS17]). *Finding recovery coefficients  $c_i \in \{0, 1\}$  in a linear secret sharing scheme  $\Pi$  for an access structure  $\mathbb{A} \in \{0, 1\}$ -LSSS can be done efficiently.*

In applications, we will need the following access structure, which removes the constraint on the number of shares per party, but retains the overall property of the shares.

**Definition 15** (Derived  $\{0, 1\}$ -LSSS ( $\{0, 1\}$ -LSSSD)). *Let  $P = \{P_1, \dots, P_N\}$  be a set of parties. We denote by  $\{0, 1\}$ -LSSSD $_N$  the class of access structures  $\mathbb{A} \in \text{LSSS}_N$  such that there exists an  $\ell \in \mathbb{N}$  polynomial in  $N$  and an access structure  $\mathbb{B} \in \{0, 1\}$ -LSSS $_{\ell n}$  for parties  $P' = \{P'_1, \dots, P'_{N\ell}\}$  such that we can associate the party  $P_i \in P$  with the parties  $P'_{\ell(i-1)}, P'_{\ell(i-1)+1}, \dots, P'_{\ell i} \in P'$  as follows. For every  $S \subseteq [N]$ ,  $S \in \mathbb{A}$  if and only if the set  $S'$  of parties of  $P'$  associated with a party in  $S$ ,  $S' \in \mathbb{B}$ . More precisely, for every  $S \subseteq [N]$ ,*

$$\bigcup_{i \in S} \{P_i\} \in \mathbb{A} \text{ if and only if } \bigcup_{i \in S} \{P'_{\ell(i-1)}, P'_{\ell(i-1)+1}, \dots, P'_{\ell i}\} \in \mathbb{B}.$$

In other words, a  $\{0, 1\}$ -LSSSD scheme is a secret sharing scheme where the shares satisfy a  $\{0, 1\}$ -LSSS scheme, but each party receives multiple shares.

**Theorem 11** ([JRS17]). *The class of access structures  $\{0, 1\}$ -LSSSD $_N$  contains all those induced by monotone boolean formulas, which, in turn contains all  $t$  out of  $N$  threshold access structures.*

In this work, all access structures will be those in the class  $\{0, 1\}$ -LSSSD.

## B MPC with General Mixed Adversaries: Definition

In this section, we formally define the notion of secure multiparty computation against a general mixed adversary as defined in the works of Fitzi et al. [FHM98, FHM99]. Recall that a general mixed adversary  $\mathcal{A} = (\mathcal{A}_{\text{Mal}}, \mathcal{A}_{\text{Sh}}, \mathcal{A}_{\text{Fc}})$  is one that corrupts a set of parties  $\mathcal{A}_{\text{Mal}}$  maliciously, a set of parties  $\mathcal{A}_{\text{Sh}}$  in a semi-honest manner and a set of parties  $\mathcal{A}_{\text{Fc}}$  in a fail-corrupt manner. While the former two notions are quite standard, we recall that in a fail-corrupt corruption, the adversary can instruct the corrupted party to stop its protocol execution at any point. Note that in the case of fail-corrupt corruption, the adversary does not get to learn the internal state of the corrupted parties at any point.

We now present the formal definition of an MPC protocol secure against a general mixed adversary  $\mathcal{A} = (\mathcal{A}_{\text{Mal}}, \mathcal{A}_{\text{Sh}}, \mathcal{A}_{\text{Fc}})$  with static corruption.

**Syntax.** A multi-party protocol is cast by specifying a random process that maps pairs of inputs to pairs of outputs (one for each party). We refer to such a process as a functionality. The security of a protocol is defined with respect to a functionality  $f$ . In particular, let  $N$  denote the number of parties. A non-reactive  $N$ -party functionality  $f$  is a (possibly randomized) mapping of  $N$  inputs to  $N$  outputs. A multiparty protocol with security parameter  $\lambda$  for computing a non-reactive functionality  $f$  is a protocol running in time  $\text{poly}(\lambda)$  and satisfying the following correctness requirement: if parties  $P_1, \dots, P_N$  with inputs  $(x_1, \dots, x_N)$  respectively, all run an honest execution of the protocol, then the joint distribution of the outputs  $y_1, \dots, y_N$  of the parties is statistically close to  $f(x_1, \dots, x_N)$ . The above can also be extended to the setting of reactive functionalities.

### B.1 Defining Security.

Informally, the security requirement is similar to that in standard multi-party computation where we consider only a single adversary type - either malicious or semi-honest. The difference here is that the adversary is additionally allowed to specify different sets  $(\mathcal{A}_{\text{Mal}}, \mathcal{A}_{\text{Sh}}, \mathcal{A}_{\text{Fc}})$  of parties apriori that will respectively correspond to malicious/semi-honest/ fail-corrupt corruptions. Furthermore, for each party in the fail-corrupt set, the adversary can adaptively decide when that party would abort the computation. We say that our protocol is secure if For simplicity, we will consider only static corruptions which is the focus of this work.

Formally, the security of a multi-party computation protocol against a general mixed adversary with respect to a functionality  $f$  is defined by comparing the real-world execution of the protocol with an ideal-world evaluation of  $f$  by a trusted party. More concretely, it is required that for every adversary  $\text{Adv} = (\mathcal{A}_{\text{Mal}}, \mathcal{A}_{\text{Sh}}, \mathcal{A}_{\text{Fc}})$ , which attacks the real execution of the protocol, there exists an ideal world adversary  $\text{Sim}$ , which can *achieve the same effect* in the ideal-world. Let's denote  $\vec{x} = (x_1, \dots, x_n)$ .

**The real execution.** In the real world execution of the n-party protocol  $\Pi$  for computing  $f$ ,  $\Pi$  is executed in the presence of an adversary  $\text{Adv}$ . The honest parties follow the instructions of  $\Pi$ . Initially, the  $\text{Adv}$  is given as input the security parameter  $\lambda$  and some auxiliary information  $z$ . Then,  $\text{Adv}$  outputs a tuple of sets  $\mathcal{A}_{\text{Mal}}, \mathcal{A}_{\text{Sh}}, \mathcal{A}_{\text{Fc}} \subseteq [N]$  of parties to corrupt and gets as input the inputs of all the parties in the sets  $\mathcal{A}_{\text{Mal}}$  and  $\mathcal{A}_{\text{Sh}}$ .  $\text{Adv}$  sends all messages in place of corrupted parties in the sets  $\mathcal{A}_{\text{Mal}}$  and  $\mathcal{A}_{\text{Sh}}$ . For each party in the set  $\mathcal{A}_{\text{Mal}}$ , it may follow an arbitrary

polynomial-time strategy. For each party in the set  $\mathcal{A}_{\text{Sh}}$ , the adversary is required to execute the protocol honestly. For each party in the set  $\mathcal{A}_{\text{Fc}}$ , the adversary can choose to instruct that party to abort the execution at any point in the protocol. Once again, note that the adversary does not learn the internal state of any fail-corrupt party.

The interaction of  $\text{Adv}$  in the protocol  $\Pi$  defines a random variable  $\text{REAL}_{\Pi, \text{Adv}(z)}(\lambda, \vec{x})$ , where  $\vec{x} = (x_1, \dots, x_N)$ , whose value is determined by the coin tosses of the adversary and the honest parties. This random variable contains the output of the adversary (which may be an arbitrary function of its view subject to the restriction on the semi-honest parties' behaviour) as well as the outputs of the honest parties. We let  $\text{REAL}_{\Pi, \text{Adv}(z)}$  denote the distribution ensemble  $\{\text{REAL}_{\Pi, \text{Adv}(z)}(\lambda, \vec{x})\}_{\lambda \in \mathbb{N}, \vec{x} \in (\{0,1\}^\lambda)^N, z \in \{0,1\}^*}$ .

**The ideal execution.** In the ideal execution of the  $n$ -party protocol  $\Pi$  for computing function  $f$ , an ideal world adversary  $\text{Sim}$  interacts with a trusted party. The ideal execution proceeds as follows.

- **Adversary picks corrupted sets:**  $\text{Sim}$  is given the security parameter  $\lambda$  and an auxiliary input  $z$  and outputs a tuple of sets  $\mathcal{A}_{\text{Mal}}, \mathcal{A}_{\text{Sh}}, \mathcal{A}_{\text{Fc}} \subseteq [N]$  of parties to corrupt.
- **Parties send inputs to the trusted party:** The parties send their inputs to the trusted party, and we let  $x'_i$  denote the value sent by  $P_i$ . Note that for each party  $P_i$  in  $\mathcal{A}_{\text{Sh}}$ , the adversary is required to send its actual input  $x_i$ . For each party  $P_k$  in  $\mathcal{A}_{\text{Fc}}$ , the adversary can decide whether  $P_k$  should send its input or not but the adversary can't change the input. For each party in  $\mathcal{A}_{\text{Mal}}$ , the adversary is free to interact as it wishes.
- **Trusted party sends output to the adversary:** For every party  $P_i$  whose input it did not receive, the trusted party sets  $y_i$  to  $0^\lambda$ . For other parties that did send their inputs, the trusted party sets  $y_i = x'_i$ . The trusted party outputs  $f(y_1, \dots, y_N)$  to  $\text{Sim}$ .
- **Adversary chooses to deliver output to other parties:** For each honest party  $P_i$ ,  $\text{Sim}$  instructs the trusted party whether or not to deliver output to  $P_i$ .
- **Outputs:**  $\text{Sim}$  outputs an arbitrary function of its view, and the honest parties output the value obtained from the trusted party (or  $\perp$  if no value is given).

The interaction of  $\text{Sim}$  with the trusted party defines a random variable  $\text{IDEAL}_{f, \text{Sim}(z)}(\lambda, \vec{x})$ , which we use to denote the distribution ensemble

$$\{\text{IDEAL}_{f, \text{Sim}(z)}(\lambda, \vec{x})\}_{\lambda \in \mathbb{N}, \vec{x} \in (\{0,1\}^\lambda)^N, z \in \{0,1\}^*}.$$

Having defined the real and the ideal worlds, we now proceed to define our notion of security.

**Definition 16.** *Let  $\lambda$  be the security parameter. Let  $f$  be an  $N$ -party functionality and  $\Pi$  be an  $N$ -party protocol for  $N \in \mathbb{N}$  for computing  $f$ .*

- *We say that  $\Pi$  securely computes  $f$  in the presence of general mixed adversaries if for every PPT general mixed adversary  $\text{Adv}$ , there exists a PPT simulator  $\text{Sim}$  such that for every PPT distinguisher  $\mathcal{D}$ , the following quantity is negligible in  $\lambda$  if  $S \notin \mathbb{A}$ :*

$$|Pr[\mathcal{D}(\text{REAL}_{\Pi, \text{Adv}(z)}(\lambda, \vec{x})) = 1] - Pr[\mathcal{D}(\text{IDEAL}_{f, \text{Sim}(z)}(\lambda, \vec{x})) = 1]|$$

where  $\vec{x} = \{x_i\}_{i \in [N]} \in (\{0,1\}^\lambda)^N$  and  $z \in \{0,1\}^*$ .

## B.2 Security against Semi-Malicious Mixed Adversaries

**Semi-Malicious Adversary.** We take the definition of a semi-malicious adversary almost verbatim from [AJLA<sup>+</sup>12]. A semi-malicious adversary is modeled as an interactive Turing machine (ITM) which, in addition to the standard tapes, has a special witness tape. In each round of the protocol, whenever the adversary produces a new protocol message  $m$  on behalf of some party  $P_i$ , it must also write to its special witness tape some pair  $(x, r)$  of input  $x$  and randomness  $r$  that explains its behavior. More specifically, all of the protocol messages sent by the adversary on behalf of  $P_i$  up to that point, including the new message  $m$ , must exactly match the honest protocol specification for  $P_i$  when executed with input  $x$  and randomness  $r$ . Note that the witnesses given in different rounds need not be consistent. Also, we assume that the attacker is rushing and hence may choose the message  $m$  and the witness  $(x, r)$  in each round adaptively, after seeing the protocol messages of the honest parties in that round (and all prior rounds). Lastly, the adversary may also choose to abort the execution on behalf of  $P_i$  in any step of the interaction.

**Semi-Malicious Mixed Adversaries.** We now consider a weaker adversarial setting when compared to the general mixed adversary called a semi-malicious mixed adversary. Here, the adversarial structure is similar to a general mixed adversary except that it can not pick a set of parties to be malicious but instead, those parties can only be semi-malicious. That is, for any semi-malicious mixed adversary  $\mathcal{A} = (\mathcal{A}_{\text{Sm}}, \mathcal{A}_{\text{Sh}}, \mathcal{A}_{\text{Fc}})$ ,  $\mathcal{A}_{\text{Sm}}$  denotes the set of parties that are semi-maliciously corrupted,  $\mathcal{A}_{\text{Sh}}$  denotes the set of parties that are corrupted in a semi-honest manner and  $\mathcal{A}_{\text{Fc}}$  denotes the set of fail-corrupt corruptions.

**Definition 17.** Let  $\lambda$  be the security parameter. Let  $f$  be an  $N$ -party functionality and  $\Pi$  be an  $N$ -party protocol for  $N \in \mathbb{N}$  for computing  $f$ .

- We say that  $\Pi$  securely computes  $f$  in the presence of semi-malicious mixed adversaries if for every PPT semi-malicious mixed adversary  $\text{Adv}$ , there exists a PPT simulator  $\text{Sim}$  such that for every PPT distinguisher  $\mathcal{D}$ , the following quantity is negligible in  $\lambda$  if  $S \notin \mathbb{A}$ :

$$|Pr[\mathcal{D}(\text{REAL}_{\Pi, \text{Adv}(z)}(\lambda, \vec{x})) = 1] - Pr[\mathcal{D}(\text{IDEAL}_{f, \text{Sim}(z)}(\lambda, \vec{x})) = 1]|$$

where  $\vec{x} = \{x_i\}_{i \in [N]} \in (\{0, 1\}^\lambda)^N$  and  $z \in \{0, 1\}^*$ .

## C Multi-Key FHE Construction in [BHP17]

Since we frequently refer to the multi-key FHE construction in [BHP17], we give the construction here. This section is taken verbatim from [BHP17].

**A “Dual” LWE-Based Multi-Key FHE with Distributed Setup.** For our protocol, we use an adaption of the “dual” of the multi-key FHE scheme from [CM15, MW16]. Just like the “primal” version, our scheme uses the GSW FHE scheme [GSW13], and its security is based on the hardness of LWE.

Recall that the LWE problem is parametrized by integers  $n, m, q$  (with  $m > n \log q$ ) and a distribution  $\chi$  over  $\mathbb{Z}$  that produces whp integers much smaller than  $q$ . The LWE assumption says that given a random matrix  $A \in \mathbb{Z}_q^{n \times m}$ , the distribution  $sA + e$  with random  $s \in \mathbb{Z}_q^n$  and  $e \leftarrow \chi^m$  is indistinguishable from uniform in  $\mathbb{Z}_q^m$ .



For the “dual” GSW scheme below, we use parameters  $n < m < w < q$  with  $m > n \log q$  and  $w > m \log q$ , and two error distributions  $\chi, \chi'$  with  $\chi'$  producing much larger errors than  $\chi$  (but still much smaller than  $q$ ). Specifically, consider the distribution

$$\chi'' = \{a \leftarrow \{0, 1\}^m, b \leftarrow \chi^m, c \leftarrow \chi', \text{output } c - \langle a, b \rangle\}.$$

We need the condition that the statistical distance between  $\chi'$  and  $\chi''$  is negligible (in the security parameter  $n$ ). This condition holds, for example, if  $\chi, \chi'$  are discrete Gaussian distributions around zero with parameters  $p, p'$ , respectively, such that  $p'/p$  is super-polynomial (in  $n$ ).

**Distributed Setup**  $\text{params}_i \leftarrow \text{MFHE.DistSetup}(1^\kappa, 1^N, i)$ : Set the parameters  $q = \text{poly}(N)n^{\omega(1)}$  (as needed for FHE correctness),  $m > (Nn + 1) \log q + 2\kappa$ , and  $w = m \log q$ .<sup>7</sup> Sample and output a random matrix  $A_i \in \mathbb{Z}_q^{(m-1) \times n}$ .

**Key Generation**  $(pk_i, sk_i) \leftarrow \text{MFHE.KeyGen}(\text{params}, i)$ : Recall that  $\text{params} = \{\text{params}_i\}_{i \in [N]} = \{A_i\}_{i \in [N]}$ . The public key of party  $i$  is a sequence of vectors  $pk_i = \{b_{i,j}\}_{j \in [N]}$  to be formally defined below. The corresponding secret key is a *low-norm vector*  $t_i \in \mathbb{Z}_q^m$ .

We will define  $b_{i,j}, t_i$  such that for  $B_{i,j} = \begin{pmatrix} A_j \\ -b_{i,j} \end{pmatrix}$ , it holds that  $t_i B_{i,j} = b_{i,i} - b_{i,j} \pmod{q}$  for all  $j$ .

In more detail, sample a random binary vector  $s_i \leftarrow \{0, 1\}^{m-1}$ , we set  $b_{i,j} = s_i A_j \pmod{q}$ . Denoting  $t_i = (s_i, 1)$ , we indeed have  $t_i B_{i,j} = b_{i,i} - b_{i,j} \pmod{q}$ .

**Encryption**  $C \leftarrow \text{MFHE.Encrypt}(pk_i, \mu)$ : To encrypt a bit  $\mu$  under the public-key  $pk_i$ , choose a random matrix  $R \in \mathbb{Z}_q^{n \times w}$  and a low-norm error matrix  $E \in \mathbb{Z}_q^{m \times w}$ , and set

$$C := B_{i,i}R + E + \mu G \pmod{q},$$

where  $G$  is a fixed  $m$ -by- $w$  “gadget matrix” (whose structure is not important for us here). Furthermore, as in [CM15, MW16], encrypt all bits of  $R$  in a similar manner. For our protocol, we use more error for the last row of the error matrix  $E$  than for the top  $m - 1$  rows. Namely, we choose  $\hat{E} \leftarrow \chi^{(m-1) \times w}$  and  $e' \leftarrow \chi'^w$  and set  $E = \begin{pmatrix} \hat{E} \\ e' \end{pmatrix}$ .

**Decryption**  $\mu := \text{MFHE.Dec}((sk_1, \dots, sk_N), C)$ : The invariant satisfied by ciphertexts in the scheme, similarly to GSW, is that an encryption of a bit  $\mu$  relative to secret key  $t$  is a matrix  $C$  that satisfies

$$tC = \mu \cdot tG + e \pmod{q}$$

for a low-norm error vector  $e$ , where  $G$  is the same “gadget matrix”. The vector  $t$  is the concatenation of all  $sk_i = t_i$  for all parties  $i$  participating in the evaluation.

This invariant holds for freshly encrypted ciphertexts since  $t_i B_{i,i} = 0 \pmod{q}$ , and so  $t_i(B_{i,i}R + E + \mu G) = \mu \cdot t_i G + t_i E \pmod{q}$ , where  $e = t_i E$  has low norm (as both  $t_i$  and  $E$  have low norm).

To decrypt, the secret-key holders compute  $u = t \cdot C \pmod{q}$ , outputting 1 if the result is closer to  $tG$  or 0 if the result is closer to 0.

---

<sup>7</sup>Parameters  $q, n, w$  are global and fixed once at the onset of the protocol.



**Evaluation**  $C := \text{MFHE.Eval}(\text{params}, C, (c_1, \dots, c_\ell))$ : Since ciphertexts satisfy the same invariant as in the original GSW scheme, then the homomorphic operations in GSW work just as well for this “dual” variant. Similarly the ciphertext-extension technique from [CM15, MW16] works also for this variant exactly as it does for the “primal” scheme (see below). Hence we get a multi-key FHE scheme.

**The ciphertext-expansion procedure.** The “gadget matrix”  $G$  used for these schemes has the property that there exists a low-norm vector  $u$  such that  $Gu = (0, 0, \dots, 0, 1)$ . Therefore, for every secret key  $t = (s|1)$ , we have  $tGu = 1 \pmod{q}$ . It follows that if  $C$  is an encryption of  $\mu$  wrt secret key  $t = (s|1)$ , then the vector  $v = Cu$  satisfies

$$\langle t, v \rangle = tCu = (\mu tG + e)u = \mu tGu + \langle e, u \rangle = \mu + \epsilon \pmod{q}$$

where  $\epsilon$  is a small integer. In other words, given an encryption of  $\mu$  wrt  $t$  we can construct a vector  $v$  such that  $\langle t, v \rangle \approx \mu \pmod{q}$ . Let  $A_1, A_2$  be public parameters for two users with secret keys  $t_1 = (s_1|1), t_2 = (s_2|1)$ , and recall that we denote  $b_{i,j} = s_i A_j$  and  $B_{i,i} = \begin{pmatrix} A_i \\ -s_i A_i \end{pmatrix} = \begin{pmatrix} A_i \\ -b_{i,i} \end{pmatrix}$ .

Let  $C = B_{1,1}R + E + \mu G$  be fresh encryption of  $\mu$  w.r.t.  $B_{1,1}$ , and suppose that we also have an encryption under  $t_1$  of the matrix  $R$ . We note that given any vector  $\delta$ , we can apply homomorphic operations to the encryption of  $R$  to get an encryption of the entries of the vector  $\rho = \rho(\delta) = \delta R$ . Then, using the technique above, we can compute for every entry  $\rho_i$  a vector  $x_i$  such that  $\langle t_1, x_i \rangle \approx \rho_i \pmod{q}$ . Concatenating all these vectors, we get a matrix  $X = X(\delta)$  such that  $t_1 X \approx \rho = \delta R \pmod{q}$ .

We consider the matrix  $C' = \begin{pmatrix} C & X \\ 0 & C \end{pmatrix}$ , where  $X = X(\delta)$  for a  $\delta$  to be determined later. We claim that for an appropriate  $\delta$  this is an encryption of the same plaintext  $\mu$  under the concatenated secret key  $t' = (t_1|t_2)$ . To see this, notice that

$$t_2 C = (s_2|1) \left( \begin{pmatrix} A_1 \\ -s_1 A_1 \end{pmatrix} R + E + \mu G \right) \approx (b_{2,1} - b_{1,1})R + \mu t_2 G \pmod{q},$$

and therefore setting  $\delta = b_{1,1} - b_{2,1}$ , which is value that can be computed from  $pk_1, pk_2$  we get

$$\begin{aligned} t' C' &= (t_1 C | t_1 X + t_2 C) \approx (\mu t_1 G | (b_{1,1} - b_{2,1})R + (b_{2,1} - b_{1,1})R + \mu t_2 G) \\ &= \mu(t_1 G | t_2 G) = \mu(t_1 | t_2) \begin{pmatrix} G & \\ & G \end{pmatrix}, \end{aligned}$$

as needed. As in the schemes from [CM15, MW16], this technique can be generalized to extend the ciphertext  $C$  into an encryption of the same plaintext  $\mu$  under the concatenation of any number of keys.