Secure MPC: Laziness Leads to GOD

Saikrishna Badrinarayanan[*] Aayush Jain[*] Nathan Manohar[*] Amit Sahai[*]

Abstract

Secure multi-party computation (MPC) [Yao82, Yao86, GMW87] is a problem of fundamental interest in cryptography. Traditional MPC protocols treat a "lazy" party (one that behaves honestly but aborts in the middle of the protocol execution) as a corrupt party that is colluding with the other corrupt parties. However, this outlook is unrealistic as there are various cases where an honest party may abort and turn "lazy" during the execution of a protocol without colluding with other parties. To address this gap, we initiate the study of what we call *secure lazy multi-party computation* with a formal definition that achieves meaningful correctness and security guarantees. We then construct a three-round lazy MPC protocol from standard cryptographic assumptions. Our protocol is malicious secure in the presence of a CRS and semi-malicious secure in the plain model without a CRS.

Using the techniques from above, 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 3-round GOD protocol that does not require any trusted setup, and it is optimal in terms of round complexity.

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

The key technical tool in our above protocols is a primitive called threshold multi-key fully homomorphic encryption which we define and construct assuming just learning with errors (LWE). We believe this primitive may be of independent interest.

[*]UCLA and Center for Encrypted Functionalities. {`saikrishna, aayushjain, nmanohar, sahai`}`@cs.ucla.edu`.
Research supported in part from a DARPA/ARL SAFEWARE award, NSF Fron- tier 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.

Contents

1 Introduction

Secure multi-party computation (MPC) [Yao82, Yao86, GMW87] has been a problem of fundamental interest in cryptography. In a secure multi-party computation 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 the traditional MPC setting, 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 1.3 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 corrupt and hence have its input discarded in the computation. 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.

As a result, we re-think the setting in a secure multi-party computation protocol to incorporate the more natural realistic scenario where a *lazy but honest* party is given the option to abort anytime during the protocol execution without being deemed to be corrupt and discarded. That is, the output computed by the remaining parties at the end of the protocol execution is still required to be a function of the joint inputs of all parties *including* those of the lazy parties. In this paper, we ask the following questions:

"What meaningful notion of correctness and security can we define in such a scenario?"
"Can we construct secure protocols for the above from standard assumptions?"
"If so, what about their efficiency?"

In this paper, we answer all the above questions. We initiate the study of *secure lazy multi-party computation* with a formal definition that achieves meaningful correctness and security guarantees. We then construct efficient (in terms of round complexity and communication complexity) malicious-secure protocols for the same from standard cryptographic assumptions. Using the techniques from above, we also achieve independently interesting consequences in the traditional MPC model: We construct the *first round-optimal* MPC protocol in the plain model (without a CRS) that achieves *guaranteed output delivery* (and hence, fairness) and is malicious-secure in the presence of an honest majority of parties.

We elaborate more on our results in Section 1.2. We first describe our new lazy MPC model and discuss the challenges encountered in defining such a model.

1.1 Lazy MPC Model

Consider a set of N parties $P_1, \ldots, P_N$ with inputs $x_1, \ldots, x_N$ respectively who wish to compute a function f. We define a lazy MPC protocol to have the following structure: the protocol consists of two phases - an "input commitment" phase and a "computation" phase.

- **Input Commitment:** All parties run the input commitment phase independently of the function being evaluated. Any party can abort and withdraw from the execution any time during this phase. Let's say some subset of the parties of size N_1 complete this phase.
- **Computation:** Any subset of the above N_1 parties that completed the input commitment phase can come together and run this phase of the protocol execution. Once again, parties can abort anytime during this phase as well. Let's say N_2 parties complete this phase. $(N_2 < N_1 < N)$.

Recall that our goal was that the final output computed must be a function of all parties including the lazy ones. Now, to define a meaningful notion of correctness, we obviously can't require that the output is based on the inputs of the lazy parties that aborted even before committing to their inputs. Therefore, informally, for correctness, we require that at the end of the protocol, the output computed is $f(x_1, \ldots, x_{N_1})$ where, without loss of generality, we assume the first N_1 parties complete the input commitment phase.

Indeed, one of the salient features of the above correctness requirement is that the next time the parties wish to run a different function on their same joint inputs, they only have to run the computation phase of the protocol. While this reusability is an appealing feature in itself, it also helps to illustrate the first hurdle in defining security.[1]

How do we define security? Let's say the adversary corrupts a set of K parties. Now, clearly, if these K parties alone can just run the computation phase by themselves, they can learn the outputs of arbitrary functions (including identity) evaluated on the inputs of all parties that completed the input commitment phase. Such a scenario can obviously not lead to any meaningful notion of security as the inputs of honest parties that completed the input commitment phase could be revealed completely to the adversary. Therefore, we need to restrict that these K corrupt parties can't run the computation phase by themselves. However, such a restriction then also places a constraint on the size N_2 of the number of parties that need to run the computation phase to get output, thereby affecting the correctness definition.

[1]We in fact achieve a stronger notion of reusability where even security holds in such a setting. That is, in the ideal world, to generate the messages on behalf of the honest parties in the input commitment phase, the simulator does not need to know the function being evaluated.

At this point, it is clear that we need to tie together the requirements for correctness and security in a meaningful manner. Let's consider a threshold T that parameterizes any protocol. For correctness, we will require that at least T parties have to complete the computation phase in order to learn the output - that is, $N_2 \geq T$. For security, we will require that the adversary can corrupt up to at most $(T-1)$ parties - that is, $K < T$.

Let's examine how this tradeoff applies to the scenarios of dishonest majority and honest majority. If we want to handle security against a dishonest majority of corrrupt parties (say $N-c$ for some constant c), then, for correctness, in order to recover output, only a very small fraction of parties, $c-1$, can be lazy and abort the protocol. That is, the protocol must be parameterized by the threshold $T = N - c + 1$. On the other hand, if, in the security game, we have an honest majority (say only $N/2 - 1$ parties are corrupt), then even if $N/2$ parties are lazy, we still get correctness. That is, the protocol can be parameterized by the threshold $T = N/2$.

Note that though the requirements for correctness and security seem tied together, as in standard MPC, the setting is completely different for both in the following sense: when we talk about correctness, there are no corrupt parties in the system and we just care about the correct output being computed even in the presence of honest but lazy parties. On the other hand, when we talk about security, we just care about privacy of honest parties' inputs in the presence of corrupt parties and not necessarily about honest parties getting any output. In the security game too, the honest parties could be lazy and this brings us to the next subtle issue.

Unlike standard MPC, our security definition is a bit more nuanced as honest parties can be lazy. Recall that in the standard MPC security definition, while a corrupt party might abort the protocol execution at any point, honest parties always stick to the protocol behavior. In order to deal with this, we allow the adversary to pick which honest parties abort and when. That is, at the start of each phase, the adversary lists a set of honest parties that will abort in that particular phase. Then, during the execution of that phase, in any round, the adversary can additionally specify which of the honest parties listed earlier should abort in the next round. Finally, observe that as in standard MPC, here too we can allow the adversary to be semi-honest/semi-malicious or fully malicious in its behavior during the protocol execution.

Here, to ease the exposition and present the main ideas used in defining the model, we considered a simplistic scenario using just a threshold T. In our formal definition, we describe the model with respect to arbitrary access structures that can be represented by any monotone boolean formula. We refer the reader to Section 5 for more details about this and other subtle issues in carefully defining the model. Finally, observe that our lazy MPC model is clearly stronger than the standard MPC model. That is, by disallowing any honest party from aborting in the above definition, we, in fact, achieve the standard MPC definition.

1.2 Our Results

1.2.1 Lazy MPC

In this paper, we first define a formal model for secure lazy MPC. Then, in that model, we show the following results:

Theorem 1 (Informal). *Assuming learning with errors (LWE), for any function f, there exists a three-round semi-malicious-secure lazy MPC protocol in the plain model with respect to any access structure $\mathbb{A}$ induced by a monotone boolean formula.*

We bootstrap the above protocol to be secure against malicious 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 2 (Informal). *Assuming LWE and Zaps, for any function f, there exists a three-round malicious-secure lazy MPC protocol in the CRS model with respect to any access structure $\mathbb{A}$ induced by a monotone boolean formula.*

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 $\mathsf{poly}(\lambda, d, N, \ell_{\in})$ where N is the number of parties, λ is the security parameter, $\ell_{\in}$ 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.

1.2.2 Standard MPC

In the standard MPC setting, we focus on the problem of constructing malicious-secure MPC protocols that achieve guaranteed output delivery (and hence fairness). 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 3 (Informal). *Assuming LWE, Zaps, and dense cryptosystems, 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 case of lazy MPC, our protocol 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.

1.2.3 Threshold Multi-Key FHE

The key technical tool we use in the above MPC protocols is a threshold multi-key fully homomorphic encryption scheme, which we define and build in this paper. Formally, we show the following theorem:

Theorem 4 (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.*

1.3 Related Work and Open Problems

We first mention a couple of related models and explain how they crucially differ from our setting. Then, we describe various prior works in the setting of dishonest majority MPC and MPC with guaranteed output delivery. Finally, we mention a concurrent work and conclude with a list of open problems.

Fail Stop Adversaries. The notion of fail-stop adversaries has been considered in several works [IKK+11, ALR13, MOR15]. The crucial difference from our setting is that, in the scenario of fail-stop adversaries, any party who aborts the protocol is considered to be corrupt while we also allow honest parties to abort the protocol.

Client-Server MPC. Secure computation in the client-server setting has been a widely studied problem [FKN94, IK97, NPS99, DI05, BCD+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+17, JKKR17, BGJ+17b, GKP17, BGJ+17a, HHPV17, BL18]).

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+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, Zaps, and dense cryptosystems. However, our construction also has depth-proportional communication and is reusable, properties not present in their scheme.

Future Directions. We initiate the study of "lazy" MPC - that is, MPC even in the presence of lazy parties who are honest but abort the protocol execution without colluding with the corrupted parties. We describe one formal model and construct secure protocols in this model. We stress that there could be several other interesting ways to model the above scenario, and we leave that as an interesting open problem. Another interesting direction would be to understand the round complexity of lazy MPC in the presence of various corruption patterns such as honest majority and dishonest majority. Additionally, one might be interested in a model where the access structure is not statically determined, but can be dynamically chosen by the parties during the protocol execution. Finally, we may consider another model where aborting parties are allowed to return in future rounds. We leave these directions to future work.

2 Technical Overview

The starting point of our new MPC protocols is the multi-key fully homomorphic encryption (multi-key FHE) paradigm, as in previous works such as [CM15, MW16, PS16, BHP17]. 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 $(pk_1, .., pk_n)$. 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^*}. It is conceivable that such a primitive could help to construct MPC protocols and this was indeed shown in the works of Mukherjee and Wichs [MW16] and Brakerski et al. [BHP17].

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). Indeed, looking ahead to our new notion of *lazy* MPC, it seems crucial that if we are to construct a *lazy* MPC protocol following the multi-key FHE recipe, we would require such an output reconstruction procedure. 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 threshold multi-key FHE[2] and this forms the technical heart of our MPC protocols.

2.1 Threshold Multi-Key FHE (TMFHE)

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 $(pk_1, ..., pk_n)$. Then, the evaluator can use secret keys $sk_1, .., sk_{n/2}$ to learn the value of m, as the set $\{1 \ldots, n/2\}$ satisfies the access structure. However, in doing so, an adversary can learn m without knowing sk_n, breaking 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 $ct \leftarrow \mathsf{Encrypt}(pk_1, .., pk_n, \mathbb{A}, m)$. 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 $S \notin \mathbb{A}$, $\mathsf{Encrypt}(pk_1, .., pk_n, \mathbb{A}, m_0) \approx_c \mathsf{Encrypt}(pk_1, .., pk_n, \mathbb{A}, m_1)$ 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[3] $S \notin \mathbb{A}$ (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.

Construction. 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 $(pk_1, .., pk_n)$, generate ciphertexts $ct_S \leftarrow \mathsf{Encrypt}(\{pk_i\}_{i \in S}, m)$ 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+17]. In a threshold FHE scheme, the setup algorithm samples a single public key fpk and n secret key shares $(\mathsf{fsk}_1, .., \mathsf{fsk}_n)$ for a secret key fsk that are shared according to the access

[2]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.

[3]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 $i \in [S]$, $(S \setminus \{i\}) \notin \mathbb{A}$.

structure $\mathbb{A}$. Using the public key 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 $S \notin \mathbb{A}$, $\mathsf{Encrypt}(pk, \mathbb{A}, m_0) \approx_c \mathsf{Encrypt}(pk, \mathbb{A}, m_1)$ 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+17] first samples $(pk, sk) \leftarrow \mathsf{FHE.Setup}(1^\lambda)$ and then secret shares sk according to the access structure using a "special purpose" secret sharing scheme to compute shares $(sk_1, .., sk_n)$ 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+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 fpk_1 and 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 $\{\mathsf{fpk}_1, \mathsf{fpk}_2\}$. If the secret keys corresponding to fpk_1 and fpk_2 are fsk_1 and 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 $[\mathsf{fsk}_1, \mathsf{fsk}_2]$. 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 fsk_1 and fsk_2 be denoted by $\mathsf{fsk}_{1,i}$ and $\mathsf{fsk}_{2,i}$, respectively. Since the decryption procedure of the multi-key FHE scheme is linear and the secret sharing of fsk_1 and 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 $[\mathsf{fsk}_{1,i}, \mathsf{fsk}_{2,i}]$. 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 fpk and secret key shares $\mathsf{fsk}_1, \ldots, \mathsf{fsk}_n$ corresponding to the access structure $\mathbb{A}$. The encryptor then encrypts 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 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 Constructing Lazy MPC

Our three-round lazy model MPC protocol 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. Let's first consider the setting of just semi-malicious adversaries. The overall structure of our lazy 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 (against a semi-malicious adversary) even in the lazy model, where honest parties could drop off from the protocol execution at any point.
One key difference from the previous works [MW16, BHP17] is the following: in the 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 lazy MPC protocol, the security guarantee given by the threshold multi-key FHE scheme allows us to prove our protocol secure with respect to corruption

of an arbitrary subset not satisfying the access pattern (given by any monotone boolean access structure). In particular, if we consider an N out of N threshold access structure, the protocol (without an additional transformation) is secure against an adversary that can 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 lazy MPC protocol secure even against malicious adversaries in the common reference string (CRS) model.

2.3 Standard MPC: Guaranteed Output Delivery

First, note that our three-round malicious-secure lazy MPC protocol with a CRS is also secure in the standard MPC model in the presence of an honest majority by setting the access structure to be an honest majority threshold access structure and preventing all honest parties from becoming "lazy" and aborting the protocol. Further, observe that in the honest majority setting, this protocol already achieves guaranteed output delivery! The reason being that since the access structure is an $(N/2+1)$ out of N threshold access structure and no honest party ever gets lazy during the protocol, we have that all the honest parties together are in the access structure 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.[4]
- 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

[4]This was a wonderful observation made in the work of Brakerski et al.[BHP17].

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[5]

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 λ. For an integer $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. 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 $\mathsf{negl}(\lambda)$ to denote a function that is negligible in λ. 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

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

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

satisfying the following specifications:

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

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

$ct \leftarrow \mathsf{Encrypt}(pk, m)$**:** *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.*

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

[5]As is the case with compiling semi-malicious protocols into malicious secure ones, we need the NIZK to be zero-knowledge and simulation-extractable.

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

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

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

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

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

Compactness of Ciphertexts. *There exists a polynomial,* poly*, such that $|ct| \le \mathsf{poly}(\lambda, d, N)$ 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 λ over the coins of all the algorithms in the following game:*

1. *On input the security parameter 1^λ, a circuit depth 1^d, and the number of parties 1^N, the adversary $\mathcal{A}$ outputs a non-corrupted party i.*
2. *Run $\mathsf{DistSetup}(1^\lambda, 1^d, 1^N, i) \to \mathsf{params}_i$. The adversary is given params_i.*
3. *The adversary outputs $\{\mathsf{params}_j\}_{j\in[N]\setminus\{i\}}$.*
4. params *is set to $\mathsf{params}_1 || \ldots || \mathsf{params}_N$. Run $\mathsf{KeyGen}(\mathsf{params}) \to (pk_i, sk_i)$. 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 $ct \leftarrow \mathsf{Encrypt}(pk_i, m_b)$ 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* Sim *such that for any PPT adversary $\mathcal{A}$, we have that the experiments $\mathsf{Expt}_{\mathcal{A},Real}(1^\lambda, 1^d, 1^N)$ and $\mathsf{Expt}_{\mathcal{A},\mathsf{Sim}}(1^\lambda, 1^d, 1^N)$ as defined below are statistically close as a function of λ over the coins of all the algorithms. The experiments are defined as follows:*

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

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

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

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

3.2 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 2 (Statistical Distance)**.** *Let* E *be a finite set,* Ω *a probability space, and* $X, Y : \Omega \to E$ *random variables. We define the* statistical distance *between* X *and* Y *to be the function* Δ *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,* Ω *a probability space, and let* $\{X_s^b\}_{s\in S, b\in\{0,1\}}$ *be a family of random variables* $X_s^b \colon \Omega \to E$ *indexed by an element* $s \in S$ *and a state* $b \in \{0,1\}$*. Further, assume that for every* $s \in S$ *we have* $\Delta(X_s^0, X_s^1) \le \delta$*. Now, for a stateful PPT algorithm* $\mathcal{A}$*, define the following experiment:*

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

- *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* $(s_1, x_1), \ldots, (s_m, x_m)$.

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

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 1 (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* $\Delta(E_1, E_2) < B_1/B_2$.

3.3 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].

3.3.1 Secret Sharing Basics.

We assume that the reader is familiar with the notion of a 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 3 (Monotone Access Structure)**.** *Let* $P = \{P_1, \ldots, P_N\}$ *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 4 (Restriction of Access Structure)**.** *Let* $P = \{P_1, \ldots, P_N\}$ *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* $P = \{P_1, \ldots, P_N\}$ *be a set of parties and let* S *be a subset of* P. *We denote by* $\mathbf{X}_S$ *the vector* $\mathbf{X}_S = (x_1, \ldots, x_N)$ *where* $x_i = 1$ *if* $P_i \in S$ *and* $x_i = 0$ *otherwise.*

Definition 5 (Efficient Access Structure)**.** *Let* $P = \{P_1, \ldots, P_N\}$ *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* $f_{\mathbb{A}} \colon \{0,1\}^N \to \{0,1\}$ *such that for all* $S \subseteq P$, $f_{\mathbb{A}}(\mathbf{X}_S) = 1$ *if and only if* $S \in \mathbb{A}$.

Definition 6 (Class of Monotone Access Structures)**.** *Let* $P = \{P_1, \ldots, P_N\}$ *be a set of parties. A* class of monotone access structures *is a collection* $\mathbb{S} = \{\mathbb{A}_1, \ldots, \mathbb{A}_t\} \subseteq \mathcal{P}(\mathcal{P}(P))$ *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 7 (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 8 (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 structure $\mathbb{A}$ such that there exists a secret sharing scheme Π 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 Π 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*

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

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

We denote by 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 $\vec{s} \in \mathbb{Z}_q^n$ 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.

3.3.2 $\{0,1\}$-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 9 ($\{0,1\}$-Linear Single Share Scheme ($\{0,1\}$-LSSS))**.** *Let $P = \{P_1, \ldots, P_N\}$ be a set of parties. The set $\{0,1\}$-$\mathsf{LSSS}_N \subseteq \mathsf{LSSS}_N$ is the collection of access structures $\mathbb{A} \in \mathsf{LSSS}_N$ such that there exists an efficient linear secret sharing scheme Π 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 10 (Maximal Invalid Share Set)**.** *Let $P = \{P_1, \ldots, P_N\}$ 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 11 (Minimal Valid Share Set)**.** *Let $P = \{P_1, \ldots, P_N\}$ 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 2 ([JRS17])**.** *Let $P = \{P_1, \ldots, P_N\}$ be a set of parties, $\mathbb{A} \in \{0,1\}$-LSSS, and Π 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 Pi 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 3 ([JRS17])**.** *Finding recovery coefficients $c_i \in \{0,1\}$ in a linear secret sharing scheme Π 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 12 (Derived $\{0,1\}$-LSSS ($\{0,1\}$-LSSSD)). *Let $P = \{P_1, \ldots, P_N\}$ be a set of parties. We denote by $\{0,1\}$-LSSSD_N the class of access structures $\mathbb{A} \in \mathsf{LSSS}_N$ such that there exists an $\ell \in \mathbb{N}$ polynomial in N and an access structure $\mathbb{B} \in \{0,1\}$-$\mathsf{LSSS}_{\ell n}$ for parties $P' = \{P'_1, \ldots, P'_{N\ell}\}$ such that we can associate the party $P_i \in P$ with the parties $P'_{\ell(i-1)}, P'_{\ell(i-1)+1}, \ldots, P'_{\ell i} \in P'$ 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 $S \subseteq [N]$,*

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

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 5 ([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.

3.4 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 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 13 (Multi-String NIZK). *A multi-string NIZK using N strings for a language $\mathcal{L}$ is a tuple of PPT algorithms*

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

satisfying the following specifications:

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

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

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

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

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

Soundness.

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

where Gen *is an oracle that on a query* q *outputs* $\mathsf{crs}_q \leftarrow \mathsf{Gen}(1^\lambda)$ *and sets* $S := S \cup \{\mathsf{crs}_q\}$*. 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* $\mathsf{SimGen}, \mathsf{SimProve}$ *such that*

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

and

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

$$\cong_c$$

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

where T *is the set containing all simulation trapdoors* τ *generated by* 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 14 (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* $\mathsf{ExtGen}, \mathsf{Ext}$ *such that* $\mathsf{ExtGen}(1^\lambda)$ *outputs* $(\mathsf{crs}, \tau, \xi)$*, a random string, a simulation trapdoor, and an extraction key, such that the output distribution* (crs, τ) *is identical to that of* SimGen *and*

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

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

[GO07] show the following.

Theorem 6 ([GO07])**.** *Assuming Zaps and dense cryptosystems, there exist simulation-extractable multi-string NIZKs.*

4 Threshold Multi-Key Fully Homomorphic Encryption

In this section, we present the definition of threshold multi-key fully homomorphic encryption (TMFHE) in the plain model with distributed setup[6]. TMFHE will be the main building block in our MPC protocols.

Definition 15 (TMFHE)**.** *Let $P = \{P_1, \ldots, P_N\}$ be a set of parties and let $\mathbb{S}$ 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*

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

satisfying the following specifications:

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

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

$ct \leftarrow \mathsf{Encrypt}(\mathsf{params}, pk_1, \ldots, pk_N, \mathbb{A}, m)$**:** *It takes as input the public parameters params, public keys $pk_1, \ldots, pk_N$, 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.*

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

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

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

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

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

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

[6]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].

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

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

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

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

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

1. *On input the security parameter* 1^λ*, 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* $\mathsf{DistSetup}(1^\lambda, 1^d, 1^N, i) \to \mathsf{params}_i$. *The adversary is given* $\{\mathsf{params}_i\}_{i \notin S}$. *Sample key pairs* $\mathsf{KeyGen}(1^\lambda) \to (pk_i, sk_i)$ *for* $i \notin S$. *The adversary is given* $\{pk_i\}_{i \notin S}$.
3. *For each* $i \in S$*, the adversary either outputs* params_i *and randomness* r_i^{KeyGen} *used to generate* (pk_i, sk_i) *or* $\perp$.

4. *Let $S_{\mathsf{params}} \subseteq [N]$ be the set of parties P_i for which params_i is defined and let $S_1 = S \cap S_{\mathsf{params}}$. The adversary then outputs messages $m_1, \ldots, m_\ell \in \{0,1\}^\lambda$ and a set $L \subseteq S_{\mathsf{params}} \backslash S_1$ of indices with $|L| = \ell$ for some $\ell \leq |S_{\mathsf{params}} \backslash S_1|$.*
5. *params is set to the concatenation of the params_i's for $i \in S_{\mathsf{params}}$. For $i \in S_1$, run $\mathsf{KeyGen}(1^\lambda; r_i^{\mathsf{KeyGen}})$ to obtain $(pk_i, sk_i)_{i \in S_1}$. Let $\mathcal{PK} = \{pk_i\}_{i \in S_{\mathsf{params}}}$. Let $\mathbb{A}'$ be the restriction of $\mathbb{A}$ to the parties in S_{params}. Run $(\{ct_i\}_{i \in L}, \mathsf{state}) \leftarrow \mathsf{Sim}_1(\mathsf{params}, \mathcal{PK}, \mathbb{A}', S_1, L)$ and give $\{ct_i\}_{i \in L}$ to the adversary.*
6. *For all $i \in S_1$, the adversary either outputs a pair $(m_i, r_i^{\mathsf{Encrypt}})$ for a message m_i and randomness used for encryption r_i^{Encrypt} or $\perp$. For the $i \in S_1$ for which $(m_i, r_i^{\mathsf{Encrypt}})$ is defined, set $ct_i = \mathsf{Enc}(\mathsf{params}, \mathcal{PK}, \mathbb{A}', m_i; r_i^{\mathsf{Encrypt}})$. Let $S_{ct} \subseteq S_{\mathsf{params}}$ be the set of indices for which ct_i is defined.*
7. *The adversary issues polynomially many queries of the form $(S'_k, S_{ct,k}, C_k : (\{0,1\}^\lambda)^{s_k} \to \{0,1\})$, where $S'_k \subseteq S_{\mathsf{params}} \backslash S_1$, $S_{ct,k} \subseteq S_{ct}$, $C_k \in \mathcal{C}$, 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 $\hat{ct}_k \leftarrow \mathsf{Eval}(C_k, \mathcal{CT}_k)$. After each query, the adversary receives $\{p_{i,k}\}_{i \in S'_k} \leftarrow \mathsf{Sim}_2(\mathsf{state}, \mu_k, \hat{ct}_k, S_1, S'_k, \{sk_i\}_{i \in S_1})$, 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. *$\mathcal{A}$ 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 Secure Lazy Multi-Party Computation

We now define our new MPC model called secure lazy multi-party computation. Recall that in our new model, the goal is to allow honest parties who are "lazy" for whatever reason to abort the protocol execution at any point without being penalized as a corrupt colluding party whose inputs are discarded from the output computation. In this section, we formally define our model of lazy MPC that incorporates such scenarios. We refer the reader to the introduction (Section 1.1) for a brief description and the challenges that arise when trying to formally define such a model.

We first present the syntax of a lazy MPC protocol and then the security definition.

Syntax. We now describe the syntax of a secure lazy multi-party computation protocol Π amongst N parties $P_1, \ldots, P_N$ for a functionality $f : (\{0,1\}^\lambda)^N \to \{0,1\}$ with respect to a monotone access structure $\mathbb{A}$ on these N parties. Let λ denote the security parameter as well as the length of each party's input. For simplicity, we consider functionalities with single bit outputs, but in general they could produce a polynomial number of output bits, potentially different for each party. In any given execution of the protocol, the i^{th} party uses some input $x_i \in \{0,1\}^\lambda$ and maintains a state

state_i. The protocol proceeds in rounds and in each round, every "remaining party" broadcasts a message to all other parties or aborts. Initially, all parties are remaining parties and once a party has aborted in a round, it cannot return to the protocol (it is no longer a part of the protocol). So, the "remaining parties" at any round $(t+1)$ are a subset of the remaining parties at round t. The protocol proceeds in two phases:

- **Input Commitment Phase:** In this phase, the parties exchange messages to commit to their respective inputs. The messages sent in this phase are independent of the functionality f. Parties that complete this phase successfully (without aborting) are said to have committed to their inputs.
- **Computation Phase:** In this phase, the parties that remain at the end of the previous phase exchange messages to compute f on all their joint (committed) inputs.

At the end of the protocol, all parties (including those that previously aborted) output out, which can be dependent on their input and the transcript of the protocol. Recall that a party can choose to abort at any stage in the above description and the rest of the parties (aka the "remaining parties") continue the protocol execution.

Correctness. Let $S \subseteq [N]$ be the set of "remaining parties" at the conclusion of the input commitment phase. Set $y_i = x_i$ for each $i \in S$. For each $i \notin S$, set $y_i = 0^\lambda$. Let $S' \subseteq S$ be the set of "remaining parties" at the conclusion of the computation phase of the protocol. The protocol Π is said to be correct if, for all inputs $x_1, \ldots, x_N \in \{0,1\}^\lambda$, all functionalities $f : (\{0,1\}^\lambda)^N \to \{0,1\}$ representable by a polynomial-sized boolean circuit, and all access structures $\mathbb{A}$ in an access structure class $\mathbb{S}$, out is statistically close to $f(y_1, \ldots, y_N)$ if $S' \in \mathbb{A}$, where the probability is taken over the coins of the parties.

Note that there is no guarantee on the output if the remaining parties S' at the conclusion of the protocol do not satisfy the access structure. However, if $S' \in \mathbb{A}$, then, crucially, even parties that aborted previously can learn the output just given the transcript of the protocol execution. In this work, we will deal with the case where $\mathbb{S}$ is the set of access structures representable by a $\{0,1\}$-LSSSD scheme. Note that in particular, this contains the set of access structures represented by a monotone boolean formula.

Security. Informally, the security requirement for lazy multi-party computation is similar to (and stronger than) that in standard multi-party computation. The difference here is that the adversary is additionally allowed to specify sets of honest parties apriori that will respectively abort during the input commitment and computation phases. That is, prior to the start of each phase, the adversary can specify a set of all honest parties that will abort during the phase and then, during the protocol execution in that phase, in each round, can adaptively decide which parties from this set abort in that particular round.

Formally, the security of a lazy multi-party computation protocol (with respect to a functionality f and an access structure $\mathbb{A}$) 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 $\mathcal{A}$, which attacks the real execution of the protocol, there exists an ideal world adversary Sim, which can *achieve the same effect* in the ideal-world. We now describe these in detail below.

The real execution. In the real world execution of the N-party protocol Π for computing f with respect to an access structure $\mathbb{A}$, Π is executed in the presence of an adversary Adv. The honest parties follow the instructions of Π, but may be forced to abort by the adversary. The real execution proceeds as follows.

- **Adversary picks corrupted set and aborting honest set:** Adv is given as input the security parameter λ and some auxiliary information z. Adv outputs a set $S \subseteq [N]$ of parties to corrupt as well as a set $S_{\mathsf{inp}} \subseteq [N] \backslash S$ of honest parties that will abort in the input commitment phase.
- **Input commitment phase:** The input commitment phase of the protocol is executed. Adv sends messages on behalf of the corrupted set of parties S. Further, prior to each round, Adv specifies a subset of parties from S_{inp} to abort in the current round subject to the restriction that at the end of the input commitment phase, all parties in S_{inp} have been instructed to abort by the adversary. An honest party that is instructed to abort does not send any message in the current (and subsequent) rounds and no longer participates in the protocol.
- **Adversary chooses an aborting honest set:** Adv outputs a set $S_{\mathsf{comp}} \subseteq [N] \backslash S \cup S_{\mathsf{inp}}$ of honest parties that will abort in the computation phase.
- **Computation phase:** The computation phase of the protocol is executed. As before, Adv sends messages on behalf of the corrupted set of parties S. Further, prior to each round, Adv specifies a subset of parties from S_{comp} to abort in the current round subject to the restriction that at the end of the computation phase, all parties in S_{comp} have been instructed to abort by the adversary. An honest party that is instructed to abort does not send any message in the current (and subsequent) rounds and no longer participates in the protocol.
- **Output:** All honest parties (including those that aborted at any stage in the protocol) produce an output. Adv also produces an output that is a function of its view.

The interaction of Adv in the protocol Π defines a random variable $\mathsf{REAL}_{\Pi,\mathsf{Adv}(z)}(\lambda, \vec{x})$, where $\vec{x} = (x_1, \ldots, x_N)$, 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) as well as the outputs of the honest parties. We let $\mathsf{REAL}_{\Pi,\mathsf{Adv}(z)}$ denote the distribution ensemble $\{\mathsf{REAL}_{\Pi,\mathsf{Adv}(z)}(\lambda, \vec{x})\}_{\lambda\in\mathbb{N},\vec{x}\in(\{0,1\}^\lambda)^N,z\in\{0,1\}^*}$.

The ideal execution. In the ideal execution of the N-party protocol Π for computing function f with respect to an access structure $\mathbb{A}$, an ideal world adversary Sim interacts with a trusted party. The ideal execution proceeds as follows.

- **Adversary picks corrupted set and aborting honest set:** Sim is given the security parameter λ and an auxiliary input z and outputs a set $S \subseteq [N]$ of parties to corrupt as well as a set $S_{\mathsf{inp}} \subseteq [N] \backslash S$ of honest parties that will abort in the input commitment phase.
- **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. The adversary controls the behavior of the parties in S.

- **Trusted party sends output to the adversary:** For $i \in S_{\mathsf{inp}}$, the trusted party sets y_i to 0^λ. For $i \notin S_{\mathsf{inp}}$, the trusted party sets $y_i = x'_i$. The trusted party outputs $f(y_1, \ldots, y_N)$ to Sim.

- **Adversary chooses to deliver output to other parties:** For each honest party P_i, Sim instructs the trusted party whether or not to deliver output to P_i.

- **Outputs:** 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 Sim with the trusted party defines a random variable $\mathsf{IDEAL}_{f,\mathsf{Sim}(z)}(\lambda, \vec{x})$, which we use to denote the distribution ensemble $\{\mathsf{IDEAL}_{f,\mathsf{Sim}(z)}(\lambda, \vec{x})\}_{\lambda\in\mathbb{N},\vec{x}\in(\{0,1\}^\lambda)^N,z\in\{0,1\}^*}$.

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

Definition 16. *Let λ be the security parameter. Let f be an N-party functionality and Π be an N-party protocol for $N \in \mathbb{N}$ for computing f with respect to a monotone access structure $\mathbb{A}$.*

- *We say that Π securely computes f with respect to $\mathbb{A}$ in the presence of malicious adversaries if for every PPT adversary Adv, there exists a PPT simulator Sim such that for every chosen corrupted set $S \subseteq [N]$, for every PPT distinguisher $\mathcal{D}$, the following quantity is negligible in λ if $S \notin \mathbb{A}$:*

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

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

5.1 Security against Semi-Malicious Adversaries

We can also consider a weaker adversarial model where the adversaries are said to be semi-malicious. We take this definition almost verbatim from [AJLA+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. Note that the adversary can continue to decide which honest party turns "lazy" and aborts, as in the case of malicious adversaries.

Definition 17. *Let λ be the security parameter. Let f be an N-party functionality and Π be an N-party protocol for $N \in \mathbb{N}$ for computing f with respect to a monotone access structure $\mathbb{A}$.*

- *We say that Π securely computes f with respect to $\mathbb{A}$ in the presence of semi-malicious adversaries if for every semi-malicious adversary* Adv*, there exists a PPT simulator* Sim *such that for every chosen corrupted set $S \subseteq [N]$, for every PPT distinguisher $\mathcal{D}$, the following quantity is negligible in λ if $S \notin \mathbb{A}$:*

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

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

6 Threshold Multi-Key FHE

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

Theorem 7 (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 Dec can be broken into two algorithms $\mathsf{Dec}_0, \mathsf{Dec}_1$ where $\mathsf{Dec}_0(sk, ct)$ 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 $E << q$, where ct is an encryption of μ. Dec_1 then takes this resulting value and rounds to recover μ.

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 μ with respect to public keys $pk_1, \ldots, pk_\ell$ is a matrix that satisfies
   $$\vec{s}\cdot\hat{ct} \approx \mu\vec{s}\cdot G$$
   for a gadget matrix G and $\vec{s} = (sk_1||\ldots||sk_\ell)$, 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 $G\vec{v} = (0, 0, \ldots, \lceil q/2\rceil)^T$. 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 8 ([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 $w = \mathsf{poly}(N)$.*

6.1 Construction

Let $\mathsf{MFHE} = (\mathsf{DistSetup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PartDec}, \mathsf{FinDec})$ be a multi-key FHE scheme instantiated with the construction in [BHP17]. Let $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme. Let χ^{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:

$\mathsf{DistSetup}(1^\lambda, 1^d, 1^N, i)$**:** Run $\mathsf{MFHE.DistSetup}(1^\lambda, 1^d, 1^N, i) \to \mathsf{params}_i$ and output params_i.

$\mathsf{KeyGen}(1^\lambda)$**:** Run $\mathsf{PKE.Setup}(1^\lambda) \to (pk, sk)$ and output (pk, sk).

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

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

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

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

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

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

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

$\mathsf{FinDec}(B)$**:** 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 $\approx \hat{\mu}\lceil q/2 \rfloor$. Then, round to recover $\hat{\mu}$.

6.2 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 $B = \{p_i\}_{i\in S} = \{\mathsf{PartDec}(i, sk_i, \hat{ct})\}_{i\in S}$ for some set of parties S as specified in the definition of correctness. If $S \notin \mathbb{A}$, then $\mathsf{FinDec}(B) = \perp$ 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

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

Note that this reconstruction procedure works even with the concatenation of secrets and multiplying by $\hat{ct'}$ because each of the fsk_i's is shared with respect to the same secret sharing scheme and the reconstruction procedure is linear. This gives

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

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

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

then rounding will correctly recover μ. Since e is sampled from an E-bounded distribution and each $e^{sm}_{i,t}$ 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.

6.3 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 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 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.

$\underline{\mathsf{Hyb}_0}$: This is the same as the "real" experiment. Namely,

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

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

$\underline{\mathsf{Hyb}_1}$: This is the same as Hyb_0 except we expand the TMFHE encryption and partial decryption procedures according to our construction.

$\mathsf{Hyb}_1(1^\lambda, 1^d, 1^n)$:

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

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

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

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

   Parse $\hat{ct}$ as $(\hat{ct}', \{ct_{i,j}\}_{(i,j)\in S'_{ct}\times S_{\mathsf{params}}})$. Define S_{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'_{shares} in an analogous manner for the set S'. For $(i,j) \in S'_{ct}\backslash L \times S_1$, decrypt $ct_{i,j}$ using sk_j to recover $\{\mathsf{fsk}_{i,j,k}\}_{i\in S'_{ct}\backslash L, j\in S_1, k\in[w]}$. For $(j,k) \in S'_{\mathsf{shares}}$, compute

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

   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 $(j,k) \in S'_{\mathsf{shares}}$, set

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

   where $e^{sm}_{j,k} \leftarrow \chi^{sm}$. 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.

<u>Hyb_2</u> : This is the same as Hyb_1 except that for all $i \in L, j \notin S_1$, we set the encrypted $\mathsf{fsk}_{i,j,k}$'s to 0. Note that these are the secret shares that the adversary is not able to recover.

$\mathsf{Hyb}_2(1^\lambda, 1^d, 1^n)$**:**

1. On input the security parameter 1^λ, 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 $\mathsf{DistSetup}(1^\lambda, 1^d, 1^N, i) \to \mathsf{params}_i$. The adversary is given $\{\mathsf{params}_i\}_{i\notin S}$. Sample key pairs $\mathsf{KeyGen}(1^\lambda) \to (pk_i, sk_i)$ for $i \notin S$. The adversary is given $\{pk_i\}_{i\notin S}$.
3. For each $i \in S$, the adversary either outputs params_i and randomness r_i^{KeyGen} used to generate (pk_i, sk_i) or $\perp$.

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

   For $i \in L$, run $\mathsf{MFHE.KeyGen}(\mathsf{params}) \to (\mathsf{fpk}_i, \mathsf{fsk}_i)$. Apply the secret sharing scheme associated with $\mathbb{A}'$ to fsk_i to arrive at $\{\mathsf{fsk}_{i,j,k}\}_{j \in S_{\mathsf{params}}, k \in [w]}$ for some $w = \mathsf{poly}(n)$. Set $ct'_i \leftarrow \mathsf{MFHE.Enc}(\mathsf{fpk}_i, m_i)$ and for $j \in S_{\mathsf{params}}$, set $ct_{i,j} = \mathsf{PKE.Enc}(pk_j, \{\mathsf{fsk}_{i,j,k}\}_{k \in [w]})$ if $j \in S_1$ and $ct_{i,j} = \mathsf{PKE.Enc}(pk_j, \vec{0})$ 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

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

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

   Parse $\hat{ct}$ as $(\hat{ct}', \{ct_{i,j}\}_{(i,j) \in S'_{ct} \times S_{\mathsf{params}}})$. Define S_{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'_{shares} in an analogous manner for the set S'. For $(i,j) \in S'_{ct} \backslash L \times S_1$, decrypt $ct_{i,j}$ using sk_j to recover $\{\mathsf{fsk}_{i,j,k}\}_{i \in S'_{ct} \backslash L, j \in S_1, k \in [w]}$. For $(j,k) \in S'_{\mathsf{shares}}$, compute

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

   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 $(j,k) \in S'_{\mathsf{shares}}$, set

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

   where $e^{sm}_{j,k} \leftarrow \chi^{sm}$. 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.

$\underline{\mathsf{Hyb}_3}$: This is the same as Hyb_2 except that for all $j \in S'$, the partial decryptions given to the adversary are simulated.

$\mathsf{Hyb}_3(1^\lambda, 1^d, 1^n)$:

1. On input the security parameter 1^λ, a circuit depth 1^d, and the maximal number of parties 1^n, the adversary $\mathcal{A}$ outputs a number of parties $N \le n$, 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 $\mathsf{DistSetup}(1^\lambda, 1^d, 1^N, i) \to \mathsf{params}_i$. The adversary is given $\{\mathsf{params}_i\}_{i \notin S}$. Sample key pairs $\mathsf{KeyGen}(1^\lambda) \to (pk_i, sk_i)$ for $i \notin S$. The adversary is given $\{pk_i\}_{i \notin S}$.
3. For each $i \in S$, the adversary either outputs params_i and randomness r_i^{KeyGen} used to generate (pk_i, sk_i) or $\perp$.
4. Let $S_{\mathsf{params}} \subseteq [N]$ be the set of parties P_i for which params_i is defined and let $S_1 = S \cap S_{\mathsf{params}}$. The adversary then outputs messages $m_1, \ldots, m_\ell \in \{0,1\}^\lambda$ and a set $L \subseteq S_{\mathsf{params}} \backslash S_1$ of indices with $|L| = \ell$ for some $\ell \le |S_{\mathsf{params}} \backslash S_1|$.
5. params is set to the concatenation of the params_i's for $i \in S_{\mathsf{params}}$. For $i \in S_1$, run $\mathsf{KeyGen}(1^\lambda; r_i^{\mathsf{KeyGen}})$ to obtain $(pk_i, sk_i)_{i \in S_1}$. Let $\mathcal{PK} = \{pk_i\}_{i \in S_{\mathsf{params}}}$. Let $\mathbb{A}'$ be the restriction of $\mathbb{A}$ to the parties in S_{params}.

   For $i \in L$, run $\mathsf{MFHE.KeyGen}(\mathsf{params}) \to (\mathsf{fpk}_i, \mathsf{fsk}_i)$. Apply the secret sharing scheme associated with $\mathbb{A}'$ to fsk_i to arrive at $\{\mathsf{fsk}_{i,j,k}\}_{j \in S_{\mathsf{params}}, k \in [w]}$ for some $w = \mathsf{poly}(n)$. Set $ct'_i \leftarrow \mathsf{MFHE.Enc}(\mathsf{fpk}_i, m_i)$ and for $j \in S_{\mathsf{params}}$, set $ct_{i,j} = \mathsf{PKE.Enc}(pk_j, \{\mathsf{fsk}_{i,j,k}\}_{k \in [w]})$ if $j \in S_1$ and $ct_{i,j} = \mathsf{PKE.Enc}(pk_j, \vec{0})$ 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

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

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

   Parse $\hat{ct}$ as $(\hat{ct}', \{ct_{i,j}\}_{(i,j) \in S'_{ct} \times S_{\mathsf{params}}})$. Define S_{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'_{shares} in an analogous manner for the set S'. If $S_1 \cup S' \notin \mathbb{A}'$, set $S_{\max} = S_{\mathsf{shares}} \cup S'_{\mathsf{shares}}$. Else, set $S_{\max}$ to be a maximally unqualified set of shares with $S_{\mathsf{shares}} \subseteq S_{\max} \subseteq S_{\mathsf{shares}} \cup S'_{\mathsf{shares}}$. If $S_1 \cup S' \in \mathbb{A}'$, set $\mu = C(\{m_i\}_{i \in S'_{ct}})$. Else, set $\mu = \perp$.

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

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

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

   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 $(j,k) \in S'_{\mathsf{shares}} \backslash S_{\max}$, let $T_{j,k} \subseteq S_{\max} \cup \{(j,k)\}$ 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 $(j,k) \in S'_{\mathsf{shares}}$, set

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

where $e^{sm}_{j,k} \leftarrow \chi^{sm}$. 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.

$\underline{\mathsf{Hyb}_4}$: This is the same as Hyb_3 except that for all $i \in L$, the secret key shares are generated with respect to 0 rather than fsk_i.

$\mathsf{Hyb}_4(1^\lambda, 1^d, 1^n)$**:**

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

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

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

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

   Parse $\hat{ct}$ as $(\hat{ct}', \{ct_{i,j}\}_{(i,j) \in S'_{ct} \times S_{\mathsf{params}}})$. Define S_{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'_{shares} in an analogous manner for the set S'. If $S_1 \cup S' \notin \mathbb{A}'$, set $S_{\max} = S_{\mathsf{shares}} \cup S'_{\mathsf{shares}}$. Else, set $S_{\max}$ to be a maximally unqualified set of shares with $S_{\mathsf{shares}} \subseteq S_{\max} \subseteq S_{\mathsf{shares}} \cup S'_{\mathsf{shares}}$. If $S_1 \cup S' \in \mathbb{A}'$, set $\mu = C(\{m_i\}_{i \in S'_{ct}})$. Else, set $\mu = \perp$.

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

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

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

   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 $(j,k) \in S'_{\mathsf{shares}} \backslash S_{\max}$, let $T_{j,k} \subseteq S_{\max} \cup \{(j,k)\}$ 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 $(j,k) \in S'_{\mathsf{shares}}$, set

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

   where $e^{sm}_{j,k} \leftarrow \chi^{sm}$. 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.

<u>Hyb_5</u> : This is the same as Hyb_4 except that for all $i \in L$, the ciphertexts given to the adversary contain MFHE encryptions of 0 rather than m_i.

$\mathsf{Hyb}_5(1^\lambda, 1^d, 1^n)$:

1. On input the security parameter 1^λ, 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 $\mathsf{DistSetup}(1^\lambda, 1^d, 1^N, i) \to \mathsf{params}_i$. The adversary is given $\{\mathsf{params}_i\}_{i \notin S}$. Sample key pairs $\mathsf{KeyGen}(1^\lambda) \to (pk_i, sk_i)$ for $i \notin S$. The adversary is given $\{pk_i\}_{i \notin S}$.
3. For each $i \in S$, the adversary either outputs params_i and randomness r_i^{KeyGen} used to generate (pk_i, sk_i) or $\perp$.
4. Let $S_{\mathsf{params}} \subseteq [N]$ be the set of parties P_i for which params_i is defined and let $S_1 = S \cap S_{\mathsf{params}}$. The adversary then outputs messages $m_1, \ldots, m_\ell \in \{0,1\}^\lambda$ and a set $L \subseteq S_{\mathsf{params}} \backslash S_1$ of indices with $|L| = \ell$ for some $\ell \leq |S_{\mathsf{params}} \backslash S_1|$.

5. params is set to the concatenation of the params_i's for $i \in S_{\mathsf{params}}$. For $i \in S_1$, run $\mathsf{KeyGen}(1^\lambda; r_i^{\mathsf{KeyGen}})$ to obtain $(pk_i, sk_i)_{i \in S_1}$. Let $\mathcal{PK} = \{pk_i\}_{i \in S_{\mathsf{params}}}$. Let $\mathbb{A}'$ be the restriction of $\mathbb{A}$ to the parties in S_{params}.
   For $i \in L$, run $\mathsf{MFHE.KeyGen}(\mathsf{params}) \to (\mathsf{fpk}_i, \mathsf{fsk}_i)$. Apply the secret sharing scheme associated with $\mathbb{A}'$ to 0 to arrive at $\{\mathsf{fsk}_{i,j,k}\}_{j \in S_{\mathsf{params}}, k \in [w]}$ for some $w = \mathsf{poly}(n)$. Set $ct'_i \leftarrow \mathsf{MFHE.Enc}(\mathsf{fpk}_i, 0^\lambda)$ and for $j \in S_{\mathsf{params}}$, set $ct_{i,j} = \mathsf{PKE.Enc}(pk_j, \{\mathsf{fsk}_{i,j,k}\}_{k \in [w]})$ if $j \in S_1$ and $ct_{i,j} = \mathsf{PKE.Enc}(pk_j, \vec{0})$ 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

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

6. For all $i \in S_1$, the adversary either outputs a pair $(m_i, r_i^{\mathsf{Encrypt}})$ for a message m_i and randomness used for encryption r_i^{Encrypt} or $\perp$. For the $i \in S_1$ for which $(m_i, r_i^{\mathsf{Encrypt}})$ is defined, set $ct_i = \mathsf{Enc}(\mathsf{params}, \mathcal{PK}, \mathbb{A}', m_i; r_i^{\mathsf{Encrypt}})$. Let $S_{ct} \subseteq S_{\mathsf{params}}$ be the set of indices for which ct_i is defined.
7. The adversary outputs a circuit $C : (\{0,1\}^\lambda)^s \to \{0,1\}$ 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 $\hat{ct} \leftarrow \mathsf{Eval}(C, \mathcal{CT})$.
8. The adversary outputs a set $S' \subseteq S_{\mathsf{params}} \backslash S_1$.
   Parse $\hat{ct}$ as $(\hat{ct}', \{ct_{i,j}\}_{(i,j) \in S'_{ct} \times S_{\mathsf{params}}})$. Define S_{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'_{shares} in an analogous manner for the set S'. If $S_1 \cup S' \notin \mathbb{A}'$, set $S_{\max} = S_{\mathsf{shares}} \cup S'_{\mathsf{shares}}$. Else, set $S_{\max}$ to be a maximally unqualified set of shares with $S_{\mathsf{shares}} \subseteq S_{\max} \subseteq S_{\mathsf{shares}} \cup S'_{\mathsf{shares}}$. If $S_1 \cup S' \in \mathbb{A}'$, set $\mu = C(\{m_i\}_{i \in S'_{ct}})$. Else, set $\mu = \perp$.
   For $(i,j) \in S'_{ct} \backslash L \times S_1$, decrypt $ct_{i,j}$ using sk_j to recover $\{\mathsf{fsk}_{i,j,k}\}_{i \in S'_{ct} \backslash L, j \in S_1, k \in [w]}$.
   For $(j,k) \in S_{\max}$, compute

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

   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 $(j,k) \in S'_{\mathsf{shares}} \backslash S_{\max}$, let $T_{j,k} \subseteq S_{\max} \cup \{(j,k)\}$ 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 $(j,k) \in S'_{\mathsf{shares}}$, set

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

   where $e^{sm}_{j,k} \leftarrow \chi^{sm}$. 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 Hyb_5. Namely, Sim_1 is the algorithm in Step 5 to generate the ciphertexts and Sim_2 is the algorithm in Step 8 used to generate the partial decryptions. The state passed from Sim_1 to Sim_2 is

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

the shares generated by Sim_1 for these indices when secret sharing 0.

Remark 1. *Note that although* Sim_2 *is given* $\{sk_i\}_{i \in S_1}$*, it only uses these secret keys to recover* $\{\mathsf{fsk}_{i,j,k}\}_{i \in S'_{ct} \setminus L, j \in S_1, k \in [w]}$*. If* Sim_2 *was instead given* $\{(m_i, r_i^{\mathsf{Encrypt}})\}_{i \in S'_{ct} \setminus L}$*, it could simulate in the same manner by using* $(m_i, r_i^{\mathsf{Encrypt}})$*'s to run the adversary's encryption computation and recover the secret key shares* $\{\mathsf{fsk}_{i,j,k}\}_{i \in S'_{ct} \setminus L, j \in S_1, k \in [w]}$*. This observation will be useful later when analyzing the security of our maliciously-secure MPC protocol in the plain model.*

Lemma 4. Hyb_0 *and* 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. □

Lemma 5. Hyb_1 *and* 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 $\mathsf{fsk}_{i,j,k}$ 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 $\mathsf{fsk}_{i,j,k}$ 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 $\mathsf{fsk}_{i,j,k}$ and an encryption of 0, contradicting the security of PKE. □

Lemma 6. *Assuming* $E/E_{sm} < \mathsf{negl}(\lambda)$*, then* Hyb_2 *and* Hyb_3 *are statistically indistinguishable.*

Proof. The only difference in the adversary's view between Hyb_2 and Hyb_3 is that in Hyb_2, all the partial decryptions for $(j,k) \in S'_{\mathsf{shares}}$ are generated using the real secret key shares, whereas in Hyb_3, the partial decryptions for $(j,k) \in S_{\max} \cap S'_{\mathsf{shares}}$ are generated using the real secret key shares, but the partial decryptions for $(j,k) \in S'_{\mathsf{shares}} \setminus (S_{\max} \cap S'_{\mathsf{shares}})$ are simulated using μ. Therefore, the distributions of $\tilde{p}_{j,k}$ and $p'_{j,k}$ for $(j,k) \in S_{\max} \cap S'_{\mathsf{shares}}$ in Hyb_2 and Hyb_3 are identical. For the remaining $(j,k) \in S'_{\mathsf{shares}}$, 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 $T_{j,k} \subseteq S_{\max} \cup \{(j,k)\}$ 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 Hyb_2, whereas in 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 Hyb_3, it follows that in Hyb_3, the adversary receives the value

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

and in Hyb_2, the adversary receives the value

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

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

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

it follows from Proposition 1 and Lemma 1 that the statistical distance between Hyb_2 and Hyb_3 is $\leq nwE/E_{sm} \leq \mathsf{poly}(n)E/E_{sm} = \mathsf{negl}(\lambda)$. 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.

□

Lemma 7. Hyb_3 *and* 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 Hyb_3 and 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. □

Lemma 8. Hyb_4 *and* 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 ℓ intermediate hybrids where in neighboring hybrids, we switch one of the encryptions of m_i to an encryption of 0^λ. 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. □

6.4 Instantiation

In order for correctness to hold, we required that $E + NwE_{sm} < q/4$. For security, we required that $NwE/E_{sm} = \mathsf{negl}(\lambda)$. Recall that $w = \mathsf{poly}(N)$. Let $W = \mathsf{poly}(N)$ 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.

7 Three-Round Lazy MPC

In this section, we construct a three round MPC protocol in the lazy MPC model, which was defined in Section 5. Our protocol supports all functionalities computable by polynomial-sized circuits and the class of access structures $\{0,1\}$-LSSSD. Formally, we show the following results.

Theorem 9. *Assuming LWE, there exists a three-round lazy MPC protocol in the plain model for any functionality f computable by a polynomial-sized circuit and any access structure $\mathbb{A} \in \{0,1\}$-LSSSD that is secure against semi-malicious adversaries. Furthermore, the protocol is reusable and has communication complexity $\mathsf{poly}(\lambda, d, N)$, where d is the depth of the circuit computing f.*

From Theorem 9, we can apply UC-secure NIZKs [DSDCO+01, GOS12] to obtain the following corollary.

Corollary 1. *Assuming LWE and DLIN, there exists a three-round lazy MPC protocol in the CRS model for any functionality f computable by a polynomial-sized circuit and any access structure $\mathbb{A} \in \{0,1\}$-LSSSD that is secure against malicious adversaries. Furthermore, the protocol is reusable and has communication complexity $\mathsf{poly}(\lambda, d, N)$, where d is the depth of the circuit computing f.*

Instantiating Corollary 1 with the N out of N access structure, we arrive at the following.

Corollary 2. *Assuming LWE and DLIN, there exists a three-round lazy 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 $\mathsf{poly}(\lambda, d, N)$, where d is the depth of the circuit computing f.*

In our three-round lazy MPC protocol, the first two rounds are the input commitment phase and the third round is the computation phase. The first round functions as a setup round and the second round functions as the input commitment round. The first two rounds can be reused to evaluate a new functionality on the same inputs in the remaining round. The main building block used in the construction is the TMFHE scheme from Section 6.

7.1 Semi-Malicious Security in the Plain Model

We first describe a three-round lazy MPC protocol that is secure against semi-malicious adversaries in the plain model.

7.1.1 Construction

Notation:

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

Protocol: We now describe the construction of our three-round lazy MPC protocol Π that is secure against semi-malicious adversaries.

- **Input Commitment Phase:**
  - **Round 1:** Each party P_i does the following:
    1. Run $\mathsf{TMFHE.DistSetup}(1^\lambda, 1^d, 1^N, i)$ to obtain params_i.
    2. Run $\mathsf{TMFHE.KeyGen}(1^\lambda)$ to compute (pk_i, sk_i).
    3. Output $(\mathsf{params}_i, pk_i)$.
  - **Round 2:** Each party P_i does the following:
    1. Parse the message (if one was sent) from P_j as $(\mathsf{params}_j, pk_j)$. Let $S_1 \subseteq [N]$ be the set of parties that sent a message in round 1.
    2. Truncate each params_j for $j \in S_1$ to the appropriate size given $|S_1|$[7]. Set params as the concatenation of the truncated 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 $\mathsf{TMFHE.Encrypt}(\mathsf{params}, \mathcal{PK}, \mathbb{A}', x_i)$ 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^λ.
    2. Run $\mathsf{TMFHE.Eval}(C', \mathcal{CT})$ to obtain $\hat{ct}$.
    3. Run $\mathsf{TMFHE.PartDec}(i, sk_i, \hat{ct})$ to obtain p_i.
    4. Output p_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. 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 $\mathsf{TMFHE.FinDec}(B)$ 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 $C'(\{x_i\}_{i \in S}) = f(y_1, \ldots, y_N)$ where $y_i = x_i$ if $i \in S$ and 0^λ otherwise. Furthermore, if $S' \in \mathbb{A}$, then $S' \in \mathbb{A}'$ and therefore running $\mathsf{TMFHE.FinDec}$ will correctly recover $f(y_1, \ldots, y_N)$ as desired.

[7]Note that the 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.

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

7.1.2 Security

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

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

- **Before Protocol Execution:** Sim receives a set $S \subseteq [N]$ of corrupted parties and a set $S_{\mathsf{inp}} \subseteq [N] \backslash S$ of honest parties that will abort in the input commitment phase.
- **Input Commitment Phase (Round 1):** For each honest party P_i that has not been told to abort in round 1 by Adv, Sim does the following:
  1. Run $\mathsf{TMFHE.DistSetup}(1^\lambda, 1^d, 1^N, i)$ to compute params_i.
  2. Run $\mathsf{TMFHE.KeyGen}(1^\lambda)$ to compute (pk_i, sk_i).
  3. Give $(\mathsf{params}_i, pk_i)$ as P_i's round 1 message to Adv.

  Sim then receives round 1 messages from Adv.
- **Input Commitment Phase (Round 2):** Sim parses the message (if one was sent) from party P_j as $(\mathsf{params}_j, pk_j)$. Let $S_1 \subseteq [N]$ be the set of parties that sent a message in round 1. It truncates each params_j to the appropriate size for $|S_1|$ parties and sets params as the concatenation of the truncated 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^2_{hon} be the set of honest parties that send a message in round 2. Let S^1_{corr} be the set of corrupted parties that sent a message in round 1. Sim does the following:
  1. Run $\mathsf{Sim}_1(\mathsf{params}, \mathcal{PK}, \mathbb{A}', S^1_{\mathsf{corr}}, S^2_{\mathsf{hon}})$ to compute $(\{ct_i\}_{i \in S^2_{\mathsf{hon}}}, \mathsf{state})$, where Sim_1 is the first algorithm of the TMFHE simulator.
  2. Give ct_i as P_i's round 2 message to Adv for $i \in S^2_{\mathsf{hon}}$.

  Let $S_2 \subseteq [N]$ be the set of parties that sent a round 2 message. For corrupted parties P_i in S_2, Sim receives the input x_i used by Adv and sends it to the trusted party. For corrupted parties P_i that are not in S_2, Sim sends 0^λ to the trusted party.
- **Query to Ideal Functionality:** Sim receives the output b from the trusted party.
- **Computation Phase (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^λ. Adv outputs a set $S_{\mathsf{comp}} \subseteq N \backslash S \cup S_{\mathsf{inp}}$ of honest parties that will abort in the computation phase. Let S^3_{hon} be the set of honest parties that have not been told to abort in round 3 by Adv. For corrupted parties P_i in S^1_{corr}, Sim extracts the secret keys sk_i that they generated. Sim does the following

1. Run $\mathsf{Sim}_2(\mathsf{state}, b, \hat{ct}, S^1_{\mathsf{corr}}, S^2_{\mathsf{hon}}, \{sk_i\}_{i \in S^1_{\mathsf{corr}}})$ to compute $\{p_j\}_{j \in S^2_{\mathsf{hon}}}$, where 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 $j \in S^3_{\mathsf{hon}}$, give p_j to 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 9. *For any semi-malicious adversary* Adv*, for the above simulator* Sim*, for any chosen access structure* $S \notin \mathbb{A}$*,*

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

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

Proof. Suppose there was some semi-malicious adversary Adv for which there existed a distinguisher $\mathcal{D}$ that could distinguish between the real and ideal world experiments. Then, there exists an adversary Adv' that could break the security of the underlying TMFHE scheme. Adv' proceeds as follows.

1. Adv' runs Adv, which outputs a set $S \subseteq [N]$ of parties to corrupt and a set $S_{\mathsf{inp}} \subseteq [N] \backslash S$ of honest parties that will abort in the input commitment phase.

2. Adv outputs a set of parties $S^1_{\mathsf{inp}} \subseteq S_{\mathsf{inp}}$ that will abort in round 1 (they will never send a message). Let $S_{\mathsf{parties}} = [N] \backslash S^1_{\mathsf{inp}}$ and let $N' = |S_{\mathsf{parties}}|$. Adv' outputs $N' \leq N$ as its number of parties, the corrupted set $S \subseteq S_{\mathsf{parties}}$, and the access structure $\mathbb{A}'$ induced by restricting $\mathbb{A}$ to the parties in S_{parties}.

3. For $i \in S_{\mathsf{parties}} \backslash S$, Adv' receives $(\mathsf{params}_i, pk_i)$ and gives this to Adv as P_i's round 1 message.

4. For some of the $j \in S$, Adv will output $(\mathsf{params}_j, pk_j)$. By running Adv, Adv' is able to determine the randomness r_j^{KeyGen} used by Adv to generate pk_j and outputs $(\mathsf{params}_j, r_j^{\mathsf{KeyGen}})$.

5. Let S^2_{hon} be the set of honest parties that will send a round 2 message. Adv' outputs this set along with the inputs $x_i \in \{0,1\}^\lambda$ for $i \in S^2_{\mathsf{hon}}$. Adv' is given ct_i for $i \in S^2_{\mathsf{hon}}$ and gives this to Adv as P_i's round 2 message.

6. Let S^2_{corr} be the set of corrupted parties that send a round 2 message. By running Adv, Adv' is able to extract the input x_i and randomness r_i^{Encrypt} used by Adv for each $i \in S^2_{\mathsf{corr}}$. Adv' outputs $(x_i, r_i^{\mathsf{Encrypt}})$ for all $i \in S^2_{\mathsf{corr}}$.

7. Let S_2 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^λ. Adv' outputs C' along with S_2.

8. Let S^3_{hon} be the set of honest parties told by Adv to send a round 3 message. Adv' outputs S^3_{hon} and receives partial decryptions p_i for $i \in S^3_{\mathsf{hon}}$. Adv' gives these to Adv as P_i's round 3 message. Adv outputs some function of its view and Adv' outputs the same value along with $\{x_i\}_{i \notin S}$.

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

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 Π is function-independent.
2. Our TMFHE simulator can simulate partial decryptions for a polynomial number of adaptively chosen circuit queries.

7.2 Malicious Security in the CRS Model

The lazy MPC protocol described above in the plain model is only secure against semi-malicious adversaries. However, if we work in the common reference string (CRS) model, we can use UC-secure non-interactive zero knowledge (NIZK) arguments [DSDCO+01, GOS12] to convert the protocol to one that is secure against malicious adversaries. A generic transformation was shown in [AJLA+12], and we refer the reader to their paper for full details.

8 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. 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 lazy MPC protocol of Section 7 and has depth-proportional communication as well as reusability of the first two rounds. Formally, we show the following.

Theorem 10. *Assuming LWE, Zaps and dense cryptosystems, 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* $\mathsf{poly}(\lambda, d, N)$*, where d is the depth of the circuit computing f, N is the number of parties and λ denotes the security parameter and input length.*

In order to prove Theorem 10, we take the semi-malicious secure lazy MPC protocol of Section 7 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 never ask an honest party to abort. From Theorem 9, it follows immediately that such a protocol is semi-malicious secure. 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 crs in round 1 and then require each party to also provide a NIZK argument in rounds 2 and 3 using these crs's to ensure that they submitted a valid message in that round.

8.1 Semi-Maliciously Secure MPC with Honest Majority

We 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.

8.1.1 Construction

The construction is the same as the lazy MPC protocol described in Section 7, except we now view it in the standard model where honest parties never abort. 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 6 with the underlying PKE scheme instantiated with one where any string is a valid public key (a dense cryptosystem).

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 7.

8.1.2 Security

Semi-malicious security is immediate as the protocol of Section 7 was secure against a stronger semi-malicious adversary that was also allowed to force honest parties to abort and abort its own corrupted parties. 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. Furthermore, since 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^{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 6.3, the simulator does not need to know sk_i. Rather, it suffices to know $(x_i, r_i^{\mathsf{Encrypt}})$, 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.

8.2 Malicious Security

In this section, we show Theorem 10. 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.

8.2.1 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 λ denote the security parameter and without loss of generality, assume $|x_i| = \lambda$ for all $i \in [N]$.

- Let $\mathsf{TMFHE} = (\mathsf{DistSetup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PartDec}, \mathsf{FinDec})$ be the previously constructed threshold multi-key FHE scheme from Section 6 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 $\mathsf{NIZK} = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Verify})$ be a simulation-extractable multi-string NIZK.

Protocol: We now describe the construction of our three-round MPC protocol Π with guaranteed output delivery that is secure against malicious adversaries. To compare against our previous protocol in Section 7, we highlight the changes in red.

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

- **<u>Round 2:</u>** Each party P_i does the following:

1. Parse the message (if one was sent) from P_j as $(\mathsf{params}_j, pk_j, \mathsf{crs}_j)$ 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 params_j for $j \in S_1$ to the appropriate size given $|S_1|$. Set params as the concatenation of the truncated params_j's for $j \in S_1$. Set $\mathcal{PK} = \{pk_j\}_{j \in S_1}$. Let $\mathcal{CRS} = \{\mathsf{crs}_j\}_{j \in S_1}$. 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 $\mathsf{TMFHE.Encrypt}(\mathsf{params}, \mathcal{PK}, \mathbb{A}', x_i; r_i)$ to compute ct_i.
4. Run $\mathsf{NIZK.Prove}(\mathcal{CRS}, y_i, (x_i, r_i))$ to compute π_i, where y_i is the statement that there exists some input x and randomness r such that $\mathsf{TMFHE.Encrypt}(\mathsf{params}, \mathcal{PK}, \mathbb{A}', x; r) = ct_i$.
5. Output (ct_i, π_i).

- **<u>Round 3:</u>** Each party P_i does the following:

1. Parse the previous message (if one was sent) from P_j as (ct_j, π_j) and check that $\mathsf{NIZK.Verify}(\mathcal{CRS}, y_j, \pi_j) = 1$. 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^λ.
2. Run $\mathsf{TMFHE.Eval}(C', \mathcal{CT})$ to compute $\hat{ct}$.
3. Sample randomness r'_i and run $\mathsf{TMFHE.PartDec}(i, sk_i, \hat{ct}; r'_i)$ to compute p_i.
4. Run $\mathsf{NIZK.Prove}(\mathcal{CRS}, z_i, (sk_i, r'_i))$ to compute π'_i, where z_i is the statement that there exists some randomness r, r' such that $\mathsf{TMFHE.KeyGen}(1^\lambda; r) = (pk_i, sk)$ and $\mathsf{TMFHE.PartDec}(i, sk, \hat{ct}; r') = p_i$.
5. Output (p_i, π'_i).

- **<u>Output Computation:</u>** Each party P_i does the following:

1. Parse the previous message (if one was sent) from P_j as (p_j, π'_j) and check that $\mathsf{NIZK.Verify}(\mathcal{CRS}, z_j, \pi'_j) = 1$. 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 $\mathsf{TMFHE.FinDec}(B)$ 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 7 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 7 was $\mathsf{poly}(\lambda, d, N)$ and the size of the NIZK reference strings and proofs are $\mathsf{poly}(\lambda, d, N)$ 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 lazy protocol of Section 7. Namely, they compute $C(y_1, \ldots, y_N)$ where $y_i = x_i$ if P_i sent valid messages in rounds 1 and 2 and $y_i = 0^\lambda$ otherwise.

8.2.2 Security

We provide a description of the simulator.

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

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

  Sim then receives round 1 messages from Adv.
- **Round 2:** Sim parses the message (if one was sent) from party P_j as $(\mathsf{params}_j, pk_j, \mathsf{crs}_j)$. Let $S_1 \subseteq [N]$ be the set of parties that sent a message in round 1. It truncates each params_j to the appropriate size for $|S_1|$ parties and sets params as the concatenation of the truncated params_j's for all $j \in S_1$. Let $\mathcal{PK}$ denote $\{pk_j\}_{j \in S_1}$. Let $\mathcal{CRS}$ denote $\{\mathsf{crs}_j\}_{j \in S_1}$. Let $\mathbb{A}'$ be the access structure induced by restricting $\mathbb{A}$ to the parties in S_1. Let S_{hon} be the set of honest parties. Let $T = \{\tau_j\}_{j \in S_{\mathsf{hon}}}$. Let $E = \{\xi_j\}_{j \in S_{\mathsf{hon}}}$. Let S^1_{corr} be the set of corrupted parties that sent a message in round 1. Sim does the following:
  1. Run $\mathsf{Sim}_1(\mathsf{params}, \mathcal{PK}, \mathbb{A}', S^1_{\mathsf{corr}}, S_{\mathsf{hon}})$ to obtain $(\{ct_i\}_{i \in S_{\mathsf{hon}}}, \mathsf{state})$, where Sim_1 is the first algorithm of the TMFHE simulator.
  2. For each honest party P_i, run $\mathsf{SimProve}(\mathcal{CRS}, T, y_i)$ to compute π_i where y_i is the statement that there exists some input x and randomness r such that $\mathsf{TMFHE.Encrypt}(\mathsf{params}, \mathcal{PK}, \mathbb{A}', x; r) = ct_i$.
  3. Give (ct_i, π_i) as P_i's round 2 message to Adv for $i \in S_{\mathsf{hon}}$.
- **Query to Ideal Functionality:**
  1. Parse the round 2 message (if one was sent) from P_j as (ct_j, π_j) and check that $\mathsf{NIZK.Verify}(\mathcal{CRS}, y_j, \pi_j) = 1$. 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 $\mathsf{Ext}(\mathcal{CRS}, E, y_j, \pi_j)$ 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^λ 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^λ. Let S^2_{corr} be the set of corrupted parties that sent a round 2 message that passed verification. Sim does the following

  1. Run $\mathsf{Sim}_2(\mathsf{state}, b, \hat{ct}, S^1_{\mathsf{corr}}, S_{\mathsf{hon}}, \{(x_i, r_i)\}_{i \in S^2_{\mathsf{corr}}})$ to obtain $\{p_j\}_{j \in S_{\mathsf{hon}}}$, where 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 $j \in S_{\mathsf{hon}}$, run $\mathsf{SimProve}(\mathcal{CRS}, T, z_j)$ to compute π'_j where z_j is the statement that there exists some randomness r, r' such that $\mathsf{TMFHE.KeyGen}(1^\lambda; r) = (pk_j, sk)$ and $\mathsf{TMFHE.PartDec}(j, sk, \hat{ct}; r') = p_j$.
  3. For $j \in S_{\mathsf{hon}}$, give (p_j, π'_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 $\mathsf{Sim}_1, \mathsf{Sim}_2$.

Reusability. Reusability follows from the following:

1. The reusability of the protocol in Section 7.
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$^+$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.

[ALR13] Gilad Asharov, Yehuda Lindell, and Tal Rabin. A full characterization of functions that imply fair coin tossing and ramifications to fairness. In Amit Sahai, editor, *Theory of Cryptography*, pages 243–262, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[BCD$^+$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 Roger Dingledine and Philippe Golle, editors, *Financial Cryptography and Data Security*, pages 325–343, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[Bei96] Amos Beimel. Phd thesis. *Israel Institute of Technology, Technion, Haifa, Israel,*, 1996.

[BGG$^+$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$^+$17] 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 *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part III*, pages 275–303, 2017.

[BGJ$^+$17a] 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, 2017. In CRYPTO 2018.

[BGJ$^+$17b] Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Dakshita Khurana, and Amit Sahai. Round optimal concurrent MPC via strong simulation. In *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*, pages 743–775, 2017.

[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 *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 1–10, New York, NY, USA, 1988. ACM.

[CKKC13] Seung Geol Choi, Jonathan Katz, Ranjit Kumaresan, and Carlos Cid. Multi-client non-interactive verifiable computation. In Amit Sahai, editor, *Theory of Cryptography*, pages 499–518, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[CL14] Ran Cohen and Yehuda Lindell. Fairness versus guaranteed output delivery in secure multiparty computation. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, pages 466–485, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[Cle86] R Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, pages 364–369, New York, NY, USA, 1986. ACM.

[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.

[DI05] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *Proceedings of the 25th Annual International Conference on Advances in Cryptology*, CRYPTO'05, pages 378–394, Berlin, Heidelberg, 2005. Springer-Verlag.

[DSDCO$^+$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.

[FKN94] Uri Feige, Joe Killian, and Moni Naor. A minimal model for secure computation (extended abstract). In *Proceedings of the Twenty-sixth Annual ACM Symposium on Theory of Computing*, STOC '94, pages 554–563, New York, NY, USA, 1994. ACM.

[GIKR02] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. On 2-round secure multiparty computation. In Moti Yung, editor, *Advances in Cryptology — CRYPTO 2002*, pages 178–193, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[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.

[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*, pages 695–704, 2011.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, 2013.

[HHPV17] Shai Halevi, Carmit Hazay, Antigoni Polychroniadou, and Muthuramakrishnan Venkitasubramaniam. Round-optimal secure multi-party computation. 2017. In CRYPTO 2018.

[IK97] Yuval Ishai and Eyal Kushilevitz. Private simultaneous messages protocols with applications. In *Proceedings of the Fifth Israel Symposium on the Theory of Computing Systems (ISTCS '97)*, ISTCS '97, pages 174–, Washington, DC, USA, 1997. IEEE Computer Society.

[IKK$^+$11] Yuval Ishai, Jonathan Katz, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. On achieving the &ldquo;best of both worlds&rdquo; in secure multiparty computation. *SIAM J. Comput.*, 40(1):122–141, February 2011.

[IKP10] Yuval Ishai, Eyal Kushilevitz, and Anat Paskin. Secure multiparty computation with minimal interaction. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, pages 577–594, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[JKKR17] Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Ron Rothblum. Distinguisher-dependent simulation in two rounds and its applications. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, 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 multi-party 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.

[LW11] Allison B. Lewko and Brent Waters. Decentralizing attribute-based encryption. In *EUROCRYPT*, 2011.

[MOR15] Tal Moran, Ilan Orlov, and Silas Richelson. Topology-hiding computation. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I*, pages 159–181, 2015.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key fhe. In *EUROCRYPT*, 2016.

[NPS99] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, EC '99, pages 129–139, New York, NY, USA, 1999. ACM.

[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 *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, pages 217–238, 2016.

[PW10] Rafael Pass and Hoeteck Wee. Constant-round non-malleable commitments from sub-exponential one-way functions. In *EUROCRYPT*, pages 638–655, 2010.

[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.

A 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 χ 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 χ, χ' with χ' producing much larger errors than χ (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 χ' and χ'' is negligible (in the security parameter n). This condition holds, for example, if χ, χ' are discrete Gaussian distributions around zero with parameters p, p', respectively, such that p'/p is super-polynomial (in n).

Distributed Setup $\mathsf{params}_i \leftarrow \mathsf{MFHE.DistSetup}(1^\kappa, 1^N, i)$**:** Set the parameters $q = \mathsf{poly}(N) n^{\omega(1)}$ (as needed for FHE correctness), $m > (Nn+1)\log q + 2\kappa$, and $w = m \log q$.[8] Sample and output a random matrix $A_i \in \mathbb{Z}_q^{(m-1)\times n}$.

Key Generation $(pk_i, sk_i) \leftarrow \mathsf{MFHE.KeyGen}(\mathsf{params}, i)$**:** Recall that $\mathsf{params} = \{\mathsf{params}_i\}_{i\in[N]} = \{A_i\}_{i\in[N]}$. 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 $b_{i,j} = s_i A_j \mod q$. Denoting $t_i = (s_i, 1)$, we indeed have $t_i B_{i,j} = b_{i,i} - b_{i,j} \pmod q$.

Encryption $C \leftarrow \mathsf{MFHE.Encrypt}(pk_i, \mu)$**:** To encrypt a bit μ 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

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

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 $\mu := \mathsf{MFHE.Dec}((sk_1, \ldots, sk_N), C)$**:** The invariant satisfied by ciphertexts in the scheme, similarly to GSW, is that an encryption of a bit μ 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 $u = t \cdot C \mod q$, outputting 1 if the result is closer to tG or 0 if the result is closer to 0.

[8]Parameters q, n, w are global and fixed once at the onset of the protocol.

Evaluation $C := \mathsf{MFHE.Eval}(\mathsf{params}, C, (c_1, \ldots, c_\ell))$: 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 $Gu = (0, 0, \ldots, 0, 1)$. Therefore, for every secret key $t = (s|1)$, we have $tGu = 1 \pmod{q}$. It follows that if C is an encryption of μ 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 ϵ is a small integer. In other words, given an encryption of μ 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 μ 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 δ, 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 ρ_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 δ to be determined later. We claim that for an appropriate δ this is an encryption of the same plaintext μ under the concatenated secret key $t' = (t_1|t_2)$. To see this, notice that

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

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

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

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 μ under the concatenation of any number of keys.