Secure MPC: Laziness Leads to GOD

Saikrishna Badrinarayanan
Visa Research
sabadrin@visa.com

Aayush Jain
UCLA
aayushjain@cs.ucla.edu

Nathan Manohar
UCLA
nmanohar@cs.ucla.edu

Amit Sahai
UCLA
sahai@cs.ucla.edu

Abstract

Motivated by what we call "honest but lazy" parties in the context of secure multi party computation, we revisit the notion of multi-key FHE schemes (MFHE). In MFHE, 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. However, this definition of MFHE works only for n-out-of-n access structures and, thus, each node in the system is a point of failure. In the context of "honest but lazy" parties, it is necessary to be able to decrypt even when only given a subset of partial decryptions (say t out of n). In order to solve this problem, we introduce a new notion of multi-key FHE designed to handle arbitrary access patterns that can reconstruct the output. We call it a threshold multi-key FHE scheme (TMFHE).

Our main contributions are the following:

- We formally define and construct TMFHE for any access structure given by a monotone boolean formula, assuming LWE.
- We construct the first simulation-extractable multi-string NIZK from polynomially hard LWE.
- We use TMFHE and our multi-string NIZK to obtain the first round-optimal (three round) MPC protocol in the plain model with *guaranteed output delivery* secure against malicious adversaries or, more generally, mixed adversaries (which supports "honest but lazy" parties), assuming LWE.
- Our MPC protocols simultaneously achieve security against the *maximum* number of corruptions under which guaranteed output delivery is achievable, depth-proportional communication complexity, and reusability.

Contents

1 Introduction

Starting with the breakthrough work of Gentry [Gen09], fully homomorphic encryption (FHE) has been extensively studied over a long sequence of works (see e.g. [Gen09, BV11b, BV11a, BGV12, GSW13]). In an FHE scheme, given a public key pk and a ciphertext of a message m encrypted using this public key, a user can homomorphically evaluate this ciphertext with respect to any circuit C to generate a new ciphertext ct that is an encryption of $C(m)$ without learning anything about the message. Then, the decryptor, using the secret key sk can decrypt this message to recover the output $C(m)$. However, traditionally, FHE schemes are single-key in nature: that is, they can be used to perform arbitrary computation on data encrypted using the same public key.

In this work, we build a new multi-party generalization of FHE that we call Threshold Multi-Key FHE, which we build from the LWE assumption. We then use this new primitive to achieve efficient secure multi-party protocols (MPC) in a model that allows for some honest parties to be "lazy," as we discuss below. Subsequent to our work, our Threshold Multi-Key FHE was used in [GPS19], which explicitly extends our MPC model with honest but lazy parties to also allow lazy parties to return in future rounds and builds upon our MPC protocol to achieve their results. We believe both our notion of Threshold Multi-Key FHE and our MPC model and protocol will continue to find other applications, as well (see e.g. [CCLS20], for another subsequent result that builds upon ours). We now elaborate on our contributions.

Multi-Key FHE. Lopez-Alt et al. [LTV12] introduced the notion of multi-key fully homomorphic encryption. Informally, in a multi-key FHE scheme, any message encrypted using a public key pk_i can be "expanded" so that the resulting ciphertext is encrypted with respect to a set of public keys $(pk_1, .., pk_n)$. 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 the semantic security of encryption, a multi-key FHE scheme also requires that given any expanded (and possibly evaluated) ciphertext ct encrypting a message m, any set of $(n-1)$ secret keys $\{sk_i\}_{i\neq i^*}$ for any i^*, and the message m, it is possible to statistically simulate the partial decryption p_{i^*}. Multi-key FHE has been extensively studied [CM15, MW16, PS16, BHP17] and has proven particularly useful in the context of building round-efficient secure multiparty computation protocols for protocols achieving security with abort. Recall that in security with abort, a single party that aborts could potentially prevent all honest parties from receiving the output.

1.1 A New Primitive: Threshold Multi-Key FHE

However, none of the existing multi-key FHE schemes enable the output to be reconstructed unless all the n partial decryptions are given out and hence they only "work" for n-out-of-n access structures. Unfortunately, this leads to situations where every secret key owner in the system represents a single point of failure, since if their partial decryption is not given out, it is not possible to recover the output. This is sufficient for protocols only achieving security with abort, as this security notion allows the functionality to fail if even a single party misbehaves. If we want to create schemes that are capable of handling failures, we would necessarily want one to be able to decrypt even when one only possesses a subset of partial decryptions (say t out of n).

At first glance, it seems that our goal is simply incompatible with the notion of multi-key FHE. For instance, suppose that a ciphertext encrypting m under a public key pk can be combined with two public keys pk' and pk'', and "expanded" into a ciphertext encrypting m under a 2-out-of-3 threshold under the triple of public keys $\{pk, pk', pk''\}$. Such a feature would imply the insecurity of the original encryption, since an adversary could sample the public keys $\{pk', pk''\}$ together with their secret keys $\{sk', sk''\}$, and then use the two secret keys $\{sk', sk''\}$ to obtain m using the expanded ciphertext.

In order to solve this problem, we introduce a new notion of *threshold* multi-key FHE[1] where ciphertexts cannot be "expanded." Instead, in our notion, given a collection of public keys $\{pk_1, \dots, pk_n\}$, it is possible for an encryptor to encrypt a message m with respect to an access pattern such as t-out-of-n. Then this ciphertext would only be decryptable by combining partial decryptions obtained from holders of at least t corresponding secret keys. As we show in this work, it turns out that this functionality is sufficient for obtaining new applications to MPC (see below for details).

In this work, we first formally define threshold multi-key FHE in a general way, and then we show to construct this new primitive from the learning with errors (LWE) assumption. Formally, we show the following theorem:

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

In Section 2, we describe the challenges and techniques involved in our construction. Our next contribution is an application of threshold multi-key FHE in the context of round-optimal secure MPC protocols with guaranteed output delivery (GOD).

1.2 Application to Round-Optimal MPC

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

MPC Supporting "Honest but Lazy" Parties. In traditional MPC, every party is required to remain online and participate completely in the protocol execution. This applies not only to "classical" MPC protocols where every party has to participate and send a message in every round of the protocol, but also to other interesting variants such as protocols in the client-server setting where all the servers are required to remain active until the end of the protocol execution. We refer the reader to Section 1.4 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

[1] We remark that in fact, some existing standard multi-key FHE schemes [MW16] also sometimes used the term threshold multi-key FHE to refer to their primitive, which requires an n-out-of-n threshold. We will use threshold multi-key FHE to denote only our stronger notion supporting general thresholds.

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 become "lazy" during the execution of a protocol without having to be deemed as colluding with the corrupt parties. A few potential reasons include:

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

Motivated by the above realistic scenarios, we would like to construct MPC protocols that can handle "honest but lazy" parties without simply lumping them in with the other corrupted parties (since treating all aborting parties as "malicious" will unrealistically enhance the power of the adversary and limit our protocol's capabilities). Furthermore, we would like our protocol to be robust to aborting parties (that is, have guaranteed output delivery). Informally, this means that at the end of the protocol execution, regardless of the behavior of the adversary, the honest parties can still compute the output of the function on all their joint inputs (with either a default or the actual input for each of the corrupted parties). Ideally, we would like to achieve a stronger form of guaranteed output delivery, where, when possible, the output of the protocol is with respect to the actual input of all the "honest but lazy" parties, rather than some default input. This is akin to stating that provided an "honest but lazy" party actually sent a message dependent on its input, the protocol will compute the functionality with respect to this party's input, regardless of whether or not the party aborted during the rest of the protocol. We call this property *input fidelity.* In this work, we ask

Can we construct round-optimal protocols in the plain model that achieve the above desiderata?

If such protocols are achievable, then

Can these protocols handle the maximum number of possible corruptions?

What can we say about the assumptions, communication complexity, and reusability of such protocols?

Using our new primitive, threshold multi-key FHE, we are able to answer all the above satisfactorily. We construct the first round-optimal (three-round) MPC protocol in the plain model that achieves our desired properties. Moreover, our protocol is capable on handling the *maximum* number of corruptions that a protocol can possibly support while achieving the desired properties. Our protocol relies only on the learning with errors (LWE) assumption. Furthermore, our protocol has depth-proportional communication complexity and is reusable.

Formalizing Our Desired Properties. Formally, we study MPC with guaranteed output delivery in the presence of threshold mixed adversaries, introduced by Fitzi et al. [FHM98,FHM99]. In this setting, a threshold mixed adversary $\mathcal{A}$ is allowed to corrupt three sets of parties $(\mathcal{A}_{\mathbf{Mal}}, \mathcal{A}_{\mathbf{Sh}}, \mathcal{A}_{\mathbf{Fc}})$ such that the following holds: (i) $|\mathcal{A}_{\mathbf{Mal}}| \leq t_{\mathbf{Mal}}$, $|\mathcal{A}_{\mathbf{Sh}}| \leq t_{\mathbf{Sh}}$, and $|\mathcal{A}_{\mathbf{Fc}}| \leq t_{\mathbf{Fc}}$, for a tuple of thresholds $(t_{\mathbf{Mal}}, t_{\mathbf{Sh}}, t_{\mathbf{Fc}})$. (ii) The set of parties in $\mathcal{A}_{\mathbf{Mal}}$ are maliciously corrupted meaning that the adversary can choose to behave using any arbitrary polynomial time algorithm on behalf of each of them. (iii) The set of parties in $\mathcal{A}_{\mathbf{Sh}}$ are corrupted in a semi-honest manner and so the adversary is required to follow the protocol execution honestly on behalf of each of them. (iv) The set of parties in $\mathcal{A}_{\mathbf{Fc}}$ are corrupted in a fail-corrupt manner meaning that for each party in this set, the adversary can specify when that party is required to abort the protocol execution. Until then, these parties follow the protocol execution honestly. Note that the adversary never gets to see the inputs or internal state of any of the fail-corrupt parties and hence these parties capture our motivation of "honest but lazy" parties - where their laziness is enforced by the adversary in the security game. In this work, our goal is to build a round-optimal MPC protocol with guaranteed output delivery in this model that also simultaneously satisfies the following desirable properties:

- **Security Against the Maximum Number of Corruptions:** Security should hold against a threshold mixed adversary that can corrupt the maximum number of parties under which guaranteed output delivery is achievable.

- **Input Fidelity:** In line with our motivation, we want our protocol to satisfy not only guaranteed output delivery, but also the stronger property that the output of the computation is a function of the joint inputs of all parties, including those that aborted after a "certain point". Intuitively, we would like our protocol to be divided into two phases - an input commitment phase and a computation phase. We refer to the end of the input commitment phase as this "point." That is, in the scenario where the adversary corrupts a set of parties in a fail-corrupt manner, for every fail-corrupt party P_i that aborts after the input commitment phase, its input y_i that is used to compute the final output $C(y_1, \ldots, y_n)$ is set to be its actual input x_i used in the protocol so far and not a default input $\perp$. Recall that this aligns with our original motivation where we wish to not discard honest but lazy parties and deem them to be corrupt.

- **Depth-Proportional Communication Complexity:** For any function f, the communication complexity of the protocol should be $\mathsf{poly}(\lambda, d, N, \ell_{\mathsf{inp}})$ where N is the number of parties, λ is the security parameter, ℓ_{inp} is the input length for each party, d is the depth of the circuit computing f.

- **Reusability:** Given the transcript of the input commitment phase of the protocol, the computation phase of the protocol should be able to be reused across an unbounded polynomial number of executions to compute different functions on the same fixed joint inputs of all the parties.

Prior to our work, much of the focus in this model was on obtaining feasibility results, understanding under what corruption patterns is secure computation even possible, and improving the communication complexity. We refer to Section 1.4 for a more detailed discussion on the prior work in this model. In particular, Hirt et al. [HMZ08] showed that in the setting of a threshold mixed adversary, MPC with guaranteed output delivery is possible if and only if $2t_{\mathbf{Mal}} + t_{\mathbf{Sh}} + t_{\mathbf{Fc}} < N$, where N is the total number of parties. Since we are interested in guaranteed output delivery, we focus on constructing MPC protocols that are secure against $(t_{\mathbf{Mal}}, t_{\mathbf{Sh}}, t_{\mathbf{Fc}})$-threshold mixed adversaries, for any $(t_{\mathbf{Mal}}, t_{\mathbf{Sh}}, t_{\mathbf{Fc}})$ satisfying the above inequality. Furthermore, in light of the result of Gordon et al. [GLS15] showing that three rounds are required for MPC with guaranteed output delivery in the traditional model (this can be viewed as a special case of the threshold mixed adversary model, where $t_{\mathbf{Sh}}$ and $t_{\mathbf{Fc}}$ are both 0), we observe that a three round protocol will be round-optimal in this setting.

Utilizing our new primitive, threshold multi-key FHE, given any tuple of thresholds $(t_{\mathbf{Mal}}, t_{\mathbf{Sh}}, t_{\mathbf{Fc}})$ satisfying the Hirt et al. [HMZ08] inequality, we construct the first round-optimal (three-round) MPC protocol with guaranteed output delivery that is secure against such a threshold mixed adversary. Since guaranteed output delivery is possible if and only if the Hirt et al. [HMZ08] inequality holds, our resulting protocol is *optimal* in terms of the best possible corruption we can tolerate. The first two rounds of our protocol form the input commitment phase, and round 3 is the computation phase. Our protocol has input fidelity, in the sense that the functionality is computed with respect to the inputs of all parties that did not abort in the first two rounds, *even* if that party aborts in round three. Additionally, given the transcript of the input commitment phase (the first two rounds of the protocol), the third round can be reused across an unbounded polynomial number of executions to compute different functions on the same fixed joint inputs of all parties. Our protocol also has depth-proportional communication complexity. Formally, we show the following result:

Theorem 2 (Informal)**.** *Assuming learning with errors (LWE), for any function f on N inputs, for any tuple of thresholds $(t_{\mathbf{Mal}}, t_{\mathbf{Sh}}, t_{\mathbf{Fc}})$ satisfying $2t_{\mathbf{Mal}} + t_{\mathbf{Sh}} + t_{\mathbf{Fc}} < N$, there exists a three-round MPC protocol with guaranteed output delivery in the plain model that is secure against a $(t_{\mathbf{Mal}}, t_{\mathbf{Sh}}, t_{\mathbf{Fc}})$-mixed adversary. The protocol has input fidelity, depth-proportional communication complexity, and is reusable.*

By instantiating Theorem 2 with the $(\lceil N/2-1\rceil, 0, 0)$-mixed adversary we achieve an interesting result in the traditional MPC world in the plain model: in particular, notice that this setting corresponds to an honest majority of parties and as a result, we get a three round MPC protocol in the plain model with guaranteed output delivery. As mentioned previously, our protocol is round optimal for this setting as well due to the lower bound of Gordon et al. [GLS15]. Formally, we achieve the following corollary, matching the round complexity of the recent independent work [ACGJ18], but for the first time, also achieving input fidelity, reusability, and depth-proportional communication complexity, assuming only LWE.

Corollary 1 (Informal)**.** *Assuming LWE, for any function f, there exists a three-round MPC protocol with guaranteed output delivery in the plain model in the presence of an honest majority.*

1.3 Multi-String NIZK from LWE

As a stepping stone to achieving Theorem 2, we first consider the weaker setting of a $(t_{\mathbf{Sm}}, t_{\mathbf{Sh}}, t_{\mathbf{Fc}})$-semi-malicious mixed adversary that corrupts the sets $(\mathcal{A}_{\mathbf{Sm}}, \mathcal{A}_{\mathbf{Sh}}, \mathcal{A}_{\mathbf{Fc}})$ of parties such that the first set of parties $\mathcal{A}_{\mathbf{Sm}}$, with $|\mathcal{A}_{\mathbf{Sm}}| \leq t_{\mathbf{Sm}}$, is only corrupted in a semi-malicious manner - that is, on behalf of each party in this set, the adversary can pick any arbitrary randomness of its choice but using this randomness, the party is required to execute the protocol honestly. We define this formally in the technical sections. Once we have constructed a protocol that is secure against a semi-malicious mixed adversary, we are able to bootstrap it to one that is secure against a (malicious) mixed adversary in the plain model using a multi-string non-interactive zero knowledge (NIZK) argument.

In a multi-string NIZK argument system, introduced in the work of Groth and Ostrovsky [GO07], a set of parties can each generate one CRS that can then be combined to compute one unified CRS which is used to compute NIZKs. The guarantee is that as long as a majority of the individual CRS strings are honestly generated, the argument system is correct and secure. Unfortunately, one of the tools in the construction of multi-string NIZKs in [GO07] was a Zap [DN07], which is not known from polynomially hard LWE. In order to obtain Theorem 2 assuming only polynomially hard LWE, we construct a (simulation-extractable) multi-string NIZK directly from LWE, which may be of independent interest. Formally, we show the following.

Theorem 3 (Informal)**.** *Assuming polynomially hard LWE, there exists a simulation-extractable multi-string NIZK for NP.*

1.4 Related Work

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+17, GKP17, BGJ+, HHPV17, BL18]). We stress that while the exact round complexity of MPC in the dishonest majority setting has been extensively studied, it is not clear or analyzed whether any of these protocols are also secure in the more general framework of a general mixed adversary.

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

MPC with Guaranteed Output Delivery. There have been a variety of prior works regarding MPC with guaranteed output delivery and/or fairness in the broadcast model. Cleve [Cle86] showed that we cannot construct fair MPC protocols unless there are an honest majority of parties. [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. There have also been a couple of very nice recent works that take on the challenging task of constructing MPC protocols with guaranteed output delivery with information-theoretic security. [ABT19] construct a three-round protocol for NC1 circuits that can handle a malicious adversary that corrupts up to a quarter of the parties. [GLS19] construct a protocol for poly-sized circuits over point-to-point channels with round-complexity the number of multiplication gates in the circuit that can handle a malicious adversary that corrupts up to a third of the parties. Recently, Halevi et al [HIK+19] studied guaranteed output delivery in the setting of functionalities where only one party gets output. In a recent work, Patra and Ravi [PR19] studied round optimal MPC protocols with GOD in the presence of dynamic corruption while the focus of our work is only static corruption.

Independent Work. Recently, in an independent work, Ananth et. al [ACGJ18] also constructed a three-round honest majority MPC protocol with guaranteed output delivery in the plain model, assuming PKE and ZAPs. Their techniques are substantially different from ours, and we note that if we instantiate our protocol with the $(\lceil N/2-1\rceil, 0, 0)$ tuple of thresholds, we are able to match their result, assuming LWE, as shown in Corollary 1. Moreover, our protocol simultaneously achieves depth-proportional communication complexity and reusability, properties not achievable by their protocol. Furthermore, we note that our general protocol can handle threshold mixed adversaries, whereas their protocol is only secure against malicious adversaries in the honest majority setting.

Subsequent works. The work of [CSW19] (which cites us as prior work) can use a threshold PKI model, which is a very strong form of certified PKI model, to achieve some of our results (guaranteed output delivery, depth proportional communication) in 2 rounds. In this work, we do not make any trust assumptions. However, we observe that our protocol already gives a 2-round protocol with a much weaker form of PKI where the public keys can be any arbitrary string. Thus, our work also implies results in a "plain" PKI setting. Last-round reusability, which we achieve, was also not studied in [CSW19]. However, we note that the focus of [CSW19] was to understand adaptive security in the context of communication efficient protocols, which we do not study.
A recent series of works [KRR17, CCRR18, HL18, CCH+19, PS19] have developed a framework for instantiating the Fiat-Shamir transform [FS87] using a hash function that satisfies a property called correlation-intractability [CGH04]. This culminated in the work of Peikert and Shiehian [PS19], who were able to obtain the first NIZK from LWE by constructing a correlation-intractable hash function family for (bounded) circuits from LWE. Following this, there have been two works [BFJ+20, JJ19], subsequent to ours, that construct two message statistically witness indistinguishable ZAP arguments from quasipolynomial LWE. From this, using the work of [GO07] one can construct a multi-string NIZK from quasipolynomial LWE. We obtain a multi-string NIZK argument system assuming only the polynomial hardness of LWE.

2 Technical Overview

We first describe the challenges involved in defining and constructing our new primitive of threshold multi-key FHE in the next subsection. This is followed by the techniques involved in constructing our round-optimal MPC protocol with guaranteed output delivery. Finally, we discuss the techniques used to construct a multi-string NIZK from LWE.

2.1 Threshold Multi-Key FHE (TMFHE)

2.1.1 Definitional Challenges.

Recall that we would like to construct a version of multi-key FHE that only requires some (say t out of n) of the partial decryption shares in order to reconstruct the output as opposed to all n partial decryptions, as is required in all existing multi-key FHE schemes.

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, if we follow the standard procedure used by known multi-key FHE schemes, 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 $S \notin \mathbb{A}$,

$$\mathsf{Encrypt}(pk_1, \ldots, pk_n, \mathbb{A}, m_0) \approx_c \mathsf{Encrypt}(pk_1, .., pk_n, \mathbb{A}, m_1)$$

   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[2] $S \notin \mathbb{A}$ (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.

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

2.1.2 Construction Overview.

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+18]. 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 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+18] 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+18] 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 MPC with Guaranteed Output Delivery

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

Our three-round MPC protocol secure against a threshold mixed adversary follows the same recipe as in the works of Mukherjee and Wichs [MW16] and Brakerski et al. [BHP17] who construct MPC protocols from multi-key FHE. We adapt it to instead use the underlying system as a threshold multi-key FHE scheme. Further, we will parametrize our protocol using an access structure $\mathbb{A}$ which will be used to run the setup of the threshold multi-key FHE scheme. Recall that since we are interested in the setting where guaranteed output delivery is possible, we require that $(t_{\mathbf{Mal}}, t_{\mathbf{Sh}}, t_{\mathbf{Fc}})$ respect the Hirt et al. [HMZ08] inequality. That is, $2t_{\mathbf{Mal}} + t_{\mathbf{Sh}} + t_{\mathbf{Fc}} < N$. In our protocol, given a threshold tuple $(t_{\mathbf{Mal}}, t_{\mathbf{Sh}}, t_{\mathbf{Fc}})$, $\mathbb{A}$ will be set as the $(N - t_{\mathbf{Mal}} - t_{\mathbf{Fc}})$-out-of-$N$ access structure. This ensures that $t_{\mathbf{Mal}} + t_{\mathbf{Sh}}$, the maximum number of parties for which the adversary can view the internal state is less than the required threshold to satisfy the access structure.

Security Against Semi-Malicious Mixed Adversaries. Let's first consider the simpler setting where the first set of corrupted parties $\mathcal{A}_{\mathbf{Mal}}$ can only be semi-malicious. That is, on behalf of each of them, the adversary can pick randomness of its choice but the parties are required to follow the protocol behavior honestly using this randomness. The adversary may also choose to have these parties abort at any time. A more formal definition is given in Appendix B. The overall structure of our MPC protocol with respect to any access structure is the following:

- In round 1, each party generates its parameters and public key for the threshold multi-key FHE scheme.
- In round 2, each party individually encrypts its input with respect to the combined set of public keys and access structure and broadcasts the ciphertext.
- All parties can now homomorphically compute a threshold multi-key FHE encryption of the output, with respect to the functionality under consideration. Then, each party broadcasts a partial decryption of the output using its secret key. The partial decryptions can be combined to recover the output in plaintext.

It can be readily observed from the definition of threshold multi-key FHE that this protocol satisfies correctness and security even in the presence of a threshold mixed adversary (with semi-malicious corruptions), where some lazy honest parties could drop off from the protocol execution at any point as determined by the fail-corrupt corruption. Furthermore, the fact that the protocol has guaranteed output delivery can be observed by noting that at most $t_{\mathbf{Mal}} + t_{\mathbf{Fc}}$ parties will abort. So, at least $N - t_{\mathbf{Mal}} - t_{\mathbf{Fc}}$ parties will remain, which is sufficient to recover the output. Note that since we have restricted the adversary to behave semi-maliciously instead of maliciously on the set $\mathcal{A}_{\mathbf{Mal}}$, every message sent will be "valid."
One key difference from the previous works [MW16,BHP17] is the following: in the standard model MPC protocols of [MW16,BHP17], due to the design of the multi-key FHE primitive, the protocol is secure only against a semi-malicious adversary that corrupts all but one party. They then need to transform it to a protocol that is secure against an adversary that can corrupt any arbitrary number of parties up to all but one of them. In our MPC protocol, the security guarantee given by the threshold multi-key FHE scheme allows us to prove a more general statement that our protocol is in fact secure even if the adversary chooses to corrupt fewer parties than it is capable of (it chooses to corrupt less than the threshold number of parties).

Handling Malicious Adversaries. The final step in achieving our MPC protocol is to allow the set $\mathcal{A}_{\mathbf{Mal}}$ to be maliciously corrupted. One way to do this would be to use a NIZK and have each party send a proof in each round that they computed their message properly; if the NIZK proof does not verify, the party would be treated as malicious and ignored. Unfortunately, using a NIZK would require us to introduce a CRS, and we want our protocol to be in the plain model.

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 needed the following two properties:

- The honest parties' matrices are generated uniformly at random.[3]
- The simulator, before the beginning of round three, 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. As discussed previously, in a multi-string NIZK argument system, a set of parties can each generate one CRS that can then be combined to compute one unified CRS which is used to compute NIZKs. The guarantee is that as long as a majority of the individual CRS strings are honestly generated, the argument system is correct and secure[4].

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. In rounds 2 and 3, each party also sends a NIZK along with their message, and the other parties make sure the NIZK verifies. If the NIZK does not verify, the party that submitted an invalid message is ignored for the rest of the protocol and treated as if it had aborted instead.

There is one additional hurdle to ensuring that a multi-string NIZK suffices for our setting. The multi-string NIZK is only secure if a *majority* of the CRSs are honestly generated. However, we want our protocol to be secure against any $(t_{\mathbf{Mal}}, t_{\mathbf{Sh}}, t_{\mathbf{Fc}})$- mixed adversary, where $2t_{\mathbf{Mal}} + t_{\mathbf{Sh}} + t_{\mathbf{Fc}} < N$. In particular, we need the multi-string NIZK to be secure in settings without an honest majority! Fortunately, the multi-string NIZK is still secure in our setting, provided that the CRSs are *uniformly random* strings. To see why this is the case, we first observe that $t_{\mathbf{Fc}}$, the number of fail-corrupt parties does not present any difficulties. This is because these parties fall under the "honest but lazy" parties in our motivation, and so while the adversary can force them to abort, the adversary can never learn any internal state information of these parties or cause them to behave dishonestly. Therefore, any CRS output by these parties will be an honest CRS, and so

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

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

choosing to not have these parties abort prior to round 1 only increases the number of honest CRSs that are output. The second observation is that any semi-honest corruptions also do not cause any difficulties. This is because the honest procedure for generating a CRS is to simply sample a random string. Therefore, even if an adversary semi-honestly corrupts a party, it can neither prevent it from outputting an honestly generated random string nor learn any state information that could compromise the random string. Therefore, all the CRSs output by the semi-honest corrupt and fail-corrupt parties are honest, and since $2t_{\mathbf{Mal}} + t_{\mathbf{Sh}} + t_{\mathbf{Fc}} < N$, it follows that a majority of the CRSs are honestly generated. Therefore, security of the multi-string NIZK system holds and we obtain a plain model construction. In this work, we construct a multi-string NIZK from LWE that satisfies this additional property required of the CRS and we elaborate more on this construction now.

2.3 Multi-String NIZK from LWE

The above demonstrated that a simulation-extractable multi-string NIZK would allow us to obtain our round-optimal MPC protocol. However, a multi-string NIZK is not known to exist from LWE. Previously it was known from statistically sound ZAPS as shown in the work of [GO07]. However, ZAPs are not known to exist from polynomially hard LWE. One might think that we could use the recent result of Peikert and Shiehian [PS19], which constructs either a statistically-sound NIZK in the common reference string model or a computationally-sound NIZK in the common random string model. One might think that we could use the transformation of Dwork and Naor [DN07] to obtain a ZAP from LWE and then apply the transformation of [GO07]. However, this does not work, since their transformation crucially requires a *statistically-sound* NIZK in the common *random* string model, which is not known from polynomially hard LWE (the recent works of [BFJ+20,JJ19] construct such ZAPs from quasipolynomial LWE). Therefore, we require a different approach. We construct the first multi-string NIZK from LWE and use it as a tool in obtaining our round-optimal MPC result.

Our construction proceeds in two main steps. We first build a multi-string non-interactive witness indistinguishable (NIWI) argument from LWE and then show how to bootstrap it to obtain a simulation-extractable multi-string NIZK.

A recent series of works [KRR17,CCRR18,HL18,CCH+19,PS19] have developed a framework for instantiating the Fiat-Shamir transform [FS87] using a hash function that satisfies a property called correlation-intractability [CGH04]. This culminated in the work of Peikert and Shiehian [PS19], who were able to obtain the first NIZK from LWE by constructing a correlation-intractable hash function family for (bounded) circuits from LWE. The notion of a correlation-intractable hash function family is defined formally in Appendix A.5. Informally, a hash function family $\mathcal{H}$ is correlation-intractable for a relation $\mathcal{R}$ if given a sampled key K, it is hard to find an x such that $(x, \mathcal{H}_K(x)) \in \mathcal{R}$. Following the formula introduced in the above works, we will apply the Fiat-Shamir transform to the Σ protocol for Graph Hamiltonicity by Blum [Blu86] in order to obtain our multi-string NIZK.

Multi-String NIWI from LWE. The first step is to construct a multi-string NIWI from LWE. A multi-string NIWI is defined analogously to a multi-string NIZK. That is, in a multi-string NIWI, a set of parties can each generate one CRS that can then be combined to compute one unified CRS which is used to compute NIWIs. The guarantee is that as long as a majority of the individual CRS strings are honestly generated, the argument system is correct and secure.

To construct the multi-string NIWI, we first construct a non-interactive commitment scheme in the multi-string model with the property that the scheme remains hiding and binding provided that a majority of the CRSs are honestly generated. At a high level, this is done by having each CRS be a public key pk_i of a public key encryption (PKE) scheme. To commit to a message m, one simply secret shares m using a $\lfloor n/2 \rfloor + 1$-out-of-n secret sharing scheme to obtain shares $(m_1, \ldots, m_n)$, then encrypts m_i under pk_i, and outputs these n ciphertexts as the commitment. Since a majority of the public keys were generated honestly, a majority of the shares are hidden by the encryption, so the commitment scheme satisfies hiding. By the correctness of the PKE scheme, the resulting commitment scheme must also be binding. Furthermore, we observe that this commitment scheme also has an associated trapdoor that facilitates extraction of the message committed. In particular, any majority of the secret keys sk_i can be used as a trapdoor as they can recover a majority of message shares from the commitment and, therefore, the message.

The multi-string NIWI is built by having each party generate its CRS in the setup phase as a public key pk_i of a PKE scheme and a hash key K_i from the correlation hash function family $\mathcal{H}$. To prove a statement $x \in L$ using a witness w, we run λ parallel repetitions of the Σ protocol using the above commitment scheme as the underlying commitment scheme and making it non-interactive via the Fiat-Shamir transformation, with the hash function instantiated using $\mathcal{H}_{K_i}$. A proof is the transcript of all the parallel executions of the Σ protocol. Soundness follows from the correlation-intractability of the hash function family $\mathcal{H}$, the binding property of the commitment scheme and the soundness of the underlying Σ protocol. Witness indistinguishability follows from the witness indistinguishability of the underlying Σ protocol and the fact that the commitment scheme is hiding even if a minority of shares are learned. We refer the reader to Section 7.2 for more details.

Obtaining a Multi-String NIZK. In order to obtain a multi-string NIZK from our multi-string NIWI, we use the standard trick found in [FLS99, GO07] each party also generates a random string r_i as part of their CRS and the statement that is proven using the multi-string NIWI now is that $x \in L$ OR a majority of the r_i's are actually the output of a pseudorandom generator G. Soundness and zero knowledge then follow via standard arguments, and we refer the reader to Section 7.3 for more details. We then observe that we can also prove simulation-extractability of our multi-string NIZK if we additionally use the commitment scheme from before once again and require the prover to commit to its witness using this scheme. The statement being proved using the multi-string NIWI would now be that either $x \in L$ using a witness w that was committed OR a majority of the r_i's are actually the output of a pseudorandom generator G. Further, in order to prove that the scheme is simulation extractable, here, we will instantiate all the underlying PKE schemes inside the extra commitment scheme (for the witness) with CCA-secure PKE schemes. As a result, our extractor for the simulation-extractable NIZK can use the secret keys of all the honest parties for this extra commitment scheme as a trapdoor to learn the witness associated with the adversary's proof. We refer the reader to Section 7.3 for more details about the proof.

Finally, recall that in order to use the multi-string NIZK in our MPC protocol, we require that the CRS generated by each party is a uniformly random string. However, in our construction, in addition to the random string r, the CRS consists of two public keys (one for committing to the witness and one for the commitment used in the Σ protocol) and a hash key K for a correlation-intractable hash function family $\mathcal{H}$. We will use an encryption scheme whose public keys are statistically-close to uniform and we also observe that the hash key is statistically-close to uniform. This ensures that the CRS is also statistically-close to uniform. We then prove that this is in fact sufficient for the MPC application and we don't require the CRS to be a uniformly random string. We refer to Section 7.4 for more details.

Roadmap. We define some preliminaries in Section 3. Then, we formally define threshold multi-key FHE in Section 4 and give our construction in Section 5. In Section 6, we describe our round optimal MPC protocol with guaranteed output delivery against threshold mixed adversaries. Finally, in Section 7, we construct multi-string NIZKs.

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 S. When we say an error distribution is E-bounded, we mean that the errors are in $[-E, E]$.

Cryptographic Primitives. We formally define multi-key FHE, secret sharing, correlation intractable hash functions, simulation-extractable multi-string NIZKs, and Sigma protocols in Appendix A. We define MPC against a threshold mixed adversary with guaranteed output delivery following the works of [FHM99, FHM98] in Appendix B.

Guaranteed Output Delivery (GOD) Consider an MPC protocol π amongst N parties. Informally, π is said to possess guaranteed output delivery (GOD) if for every PPT malicious adversary, for all possible sets of inputs $\{x_1, \ldots, x_N\}$, for any function f, the following holds: At the end of the execution of π, every honest party outputs $f(y_1, \ldots, y_n)$ where $y_i = x_i$ for every honest party P_i and $y_j = x_j/\bot$ for every corrupt party P_j.

4 Threshold Multi-Key FHE: Definition

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

Definition 1 (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:

[5]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].

$\mathsf{params}_i \leftarrow \mathsf{DistSetup}(1^\lambda, 1^d, 1^N, i)$**:** *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* i*th 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, the public keys, and the 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* $C\colon (\{0,1\}^\lambda)^\ell \rightarrow \{0,1\} \in \mathcal{C}_\lambda$ *of depth* $\leq d$ *and ciphertexts* $ct_1, \ldots, ct_\ell$ *for* $\ell = \mathsf{poly}(N)$*. It outputs an evaluated ciphertext* $\widehat{ct}$*.*

$p_i \leftarrow \mathsf{PartDec}(i, sk, \hat{ct})$**:** *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* $\ell = \mathsf{poly}(N)$*, and any boolean circuit* $C\colon (\{0,1\}^\lambda)^\ell \rightarrow \{0,1\} \in \mathcal{C}_\lambda$ *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* DistSetup*,* KeyGen*,* Encrypt*, and* PartDec*,*

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

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 exist 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* $\mathsf{Expt}_{\mathcal{A},\mathsf{Sim}}(1^\lambda, 1^d, 1^N)$ *are computationally indistinguishable.*

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

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 an access structure* $\mathbb{A} \in \mathbb{S}$ *over* N *parties and a maximal set* $S \subseteq [N]$ *such that* $S \notin \mathbb{A}$*.*
2. *For* $i \in [N]$*, run* $\mathsf{DistSetup}(1^\lambda, 1^d, 1^N, i) \rightarrow \mathsf{params}_i$*. The adversary is given* $\{\mathsf{params}_i\}_{i \in [N]}$*. Sample key pairs* $\mathsf{KeyGen}(1^\lambda) \rightarrow (pk_i, sk_i)$ *for* $i \in [N]$*. The adversary is given* $\{pk_i\}_{i \in [N]}$ *and* $\{sk_i\}_{i \in S}$*.*
3. *The adversary then outputs messages* $m_1, \ldots, m_\ell \in \{0,1\}^\lambda$ *for* $\ell = \mathsf{poly}(N)$*.*

4. params *is set to the concatenation of the* params_i*'s for* $i \in [N]$*. Let* $\mathcal{PK} = \{pk_i\}_{i\in[N]}$*. The adversary is given* $ct_i \leftarrow \mathsf{Enc}(\mathsf{params}, \mathcal{PK}, \mathbb{A}, m_i)$ *for* $i \in [\ell]$*.*
5. *The adversary issues polynomially many queries of the form* $(C_k : (\{0,1\}^\lambda)^\ell \rightarrow \{0,1\})$*, where* $C_k \in \mathcal{C}$*. Let the evaluated ciphertext be* $\hat{ct}_k \leftarrow \mathsf{Eval}(C_k, ct_1, \ldots, ct_\ell)$*. After each query, the adversary receives* $p_{i,k} \leftarrow \mathsf{PartDec}(i, sk_i, \hat{ct}_k)$ *for all* $i \in [N]\backslash S$*.*
6. $\mathcal{A}$ *outputs* out*. The output of the experiment is* out*.*

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

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 an access structure* $\mathbb{A} \in \mathbb{S}$ *over* N *parties and a maximal set* $S \subseteq [N]$ *such that* $S \notin \mathbb{A}$*.*
2. *For* $i \in [N]$*, run* $\mathsf{DistSetup}(1^\lambda, 1^d, 1^N, i) \rightarrow \mathsf{params}_i$*. The adversary is given* $\{\mathsf{params}_i\}_{i\in[N]}$*. Sample key pairs* $\mathsf{KeyGen}(1^\lambda) \rightarrow (pk_i, sk_i)$ *for* $i \in [N]$*. The adversary is given* $\{pk_i\}_{i\in[N]}$ *and* $\{sk_i\}_{i\in S}$*.*
3. *The adversary then outputs messages* $m_1, \ldots, m_\ell \in \{0,1\}^\lambda$ *for* $\ell = \mathsf{poly}(N)$*.*
4. params *is set to the concatenation of the* params_i*'s for* $i \in [N]$*. Let* $\mathcal{PK} = \{pk_i\}_{i\in[N]}$*. The adversary is given* $\{ct_i\}_{i\in[\ell]} \leftarrow \mathsf{Sim}_1(\mathsf{params}, \mathcal{PK}, \mathbb{A})$*.*
5. *The adversary issues polynomially many queries of the form* $(C_k : (\{0,1\}^\lambda)^\ell \rightarrow \{0,1\})$*, where* $C_k \in \mathcal{C}$*. Let the evaluated ciphertext be* $\hat{ct}_k \leftarrow \mathsf{Eval}(C_k, ct_1, \ldots, ct_\ell)$*. After each query, the adversary receives* $\{p_{i,k}\}_{i\notin S} \leftarrow \mathsf{Sim}_2(\mu_k, \hat{ct}_k, S, \{sk_i\}_{i\in S})$*, where* $\mu_k = C_k(\{m_i\}_{i\in[\ell]})$*.*
6. $\mathcal{A}$ *outputs* out*. The output of the experiment is* out*.*

The security notion is inspired by the security definitions of multi-key FHE [MW16, BHP17] suitably adapted to the context of general access structures. Observe that the above definition captures both the semantic security of ciphertexts and the simulation security of partial decryptions.

Looking ahead to our MPC protocol, we will actually need some stronger guarantees from the TMFHE scheme, which adds complexity to the security definition. 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.

The (more general) formal definition is given below.

Definition 2 (Simulation Security of TMFHE)**.** *There exist stateful PPT algorithms* $\mathsf{Sim}_1, \mathsf{Sim}_2$ *such that for any PPT adversary* $\mathcal{A}$*, the experiments* $\mathsf{Expt}_{\mathcal{A},Real}(1^\lambda, 1^d, 1^n)$ *and* $\mathsf{Expt}_{\mathcal{A},\mathsf{Sim}}(1^\lambda, 1^d, 1^n)$ *defined below are computationally indistinguishable.*

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

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* $\mu_k = \bot$ *otherwise.*
8. $\mathcal{A}$ *outputs* out*. The output of the experiment is* out*.*

Remark 1. *Due to the notational complexity of the above security definition, we have restricted the definition to the case where there is at most a single message associated with each party* P_i*. This definition is sufficient for our applications of TMFHE to constructing MPC. However, the definition can be generalized to allow a polynomial number of messages associated with each party* P_i*, and the security proofs naturally extend to this setting.*

5 Threshold Multi-Key FHE: Construction

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

Theorem 4 (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 use several ingredients. First, we initialize a multi-key FHE scheme using the construction in [BHP17]. Then, we utilize the techniques in the construction of threshold FHE in [JRS17][6], 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 use a public key encryption scheme to tie everything together.

In more detail, 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
$$\vec{s} \cdot \hat{ct} \approx \mu \vec{s} \cdot G$$

[6] We note that the work of Boneh et al. [BGG+18] is a merge of [JRS17] and [BGGK17].

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 5 ([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 $w = \mathsf{poly}(N)$.*

5.1 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 $\mathsf{MFHE.KeyGen}(\mathsf{params}) \to (\mathsf{fpk}, \mathsf{fsk})$. Compute $\{\mathsf{fsk}_{i,j}\}_{i\in[N], j\in[w]}$ for some $w = \mathsf{poly}(N)$ by applying the $\{0,1\}$-LSSSD scheme associated with $\mathbb{A}$ to fsk to . 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 $\approx \hat{\mu}\lceil q/2 \rceil$. Then, round to recover $\hat{\mu}$.

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

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

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.

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.

5.2 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 $m_1, \ldots, m_\ell \in \{0,1\}^\lambda$ 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 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 $m_1, \ldots, m_\ell \in \{0,1\}^\lambda$ 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 $ct_{i,j} = \mathsf{PKE.Enc}(pk_j, \{\mathsf{fsk}_{i,j,k}\}_{k \in [w]})$. 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

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

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

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

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

$\underline{\mathsf{Hyb}_2}$: 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 $\mu = \bot$.
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 $N \leq n$, 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 $\bot$.
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 $ct'_i \leftarrow \mathsf{MFHE.Enc}(\mathsf{fpk}_i, m_i)$ 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 $\bot$. 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 $\mu = \bot$.
   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.

$\underline{\mathsf{Hyb}_5}$: 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 $\mathsf{MFHE.KeyGen}(\mathsf{params}) \rightarrow (\mathsf{fpk}_i, \mathsf{fsk}_i)$. 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 $C : (\{0,1\}^\lambda)^s \rightarrow \{0,1\}$ along with a subset $S'_{ct} \subseteq S_{ct}$ with $C \in \mathcal{C}$ and $s = |S'_{ct}| \leq |S_{ct}|$. Let $\mathcal{CT} = \{ct_i\}_{i \in S'_{ct}}$ and let the evaluated ciphertext be $\hat{ct} \leftarrow \mathsf{Eval}(C, \mathcal{CT})$.
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 2. *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 showing our MPC protocol in the plain model is secure against threshold mixed adversaries.*

Lemma 1. Hyb_0 *and* Hyb_1 *are 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 2. 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 3. *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

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

and in Hyb_2, the adversary receives the value

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

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

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

it follows from Proposition 1 and Lemma 19 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 4. Hyb_3 *and* Hyb_4 *are 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 5. 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 Round-Optimal MPC with Guaranteed Output Delivery Secure Against Threshold Mixed Adversaries

In this section, we use threshold multi-key FHE to construct a round-optimal (three-round) MPC protocol in the plain model with guaranteed output delivery that is secure against a threshold mixed adversary (defined in Appendix B), assuming LWE. Our protocol supports all functionalities computable by polynomial-sized circuits and is parameterized by a tuple of thresholds $(t_{\mathbf{Mal}}, t_{\mathbf{Sh}}, t_{\mathbf{Fc}})$ that represent the number of malicious, semi-honest, and fail-corrupt corruptions that the adversary is allowed to make, respectively. Our protocol has guaranteed output delivery and is secure provided that $2t_{\mathbf{Mal}} + t_{\mathbf{Sh}} + t_{\mathbf{Fc}} < N$, the Hirt et al. [HMZ08] inequality that characterizes the threshold values under with guaranteed output delivery is possible to achieve.

Thus, our resulting protocol is both *optimal* in terms of the best possible corruption we can tolerate and also *round-optimal* (since at least three rounds are required for a protocol to have guaranteed output delivery, as shown by Gordon et al. [GLS15]). Moreover, our protocol has depth-proportional communication complexity, is reusable, and has input fidelity for "honest but lazy" parties. Formally, we show the following.

Theorem 6. *Assuming LWE, for any function f, for any tuple of thresholds $(t_{\mathbf{Mal}}, t_{\mathbf{Sh}}, t_{\mathbf{Fc}})$ satisfying $2t_{\mathbf{Mal}} + t_{\mathbf{Sh}} + t_{\mathbf{Fc}} < N$, there exists a three-round MPC protocol with guaranteed output delivery in the plain model that is secure against a $(t_{\mathbf{Mal}}, t_{\mathbf{Sh}}, t_{\mathbf{Fc}})$-mixed adversary. Furthermore, the protocol is reusable, has communication complexity $\mathsf{poly}(\lambda, d, N)$, where d is the depth of the circuit computing f and the functionality is computed with respect to the inputs of all parties that send valid messages in the first two rounds.*

Note that our result in the mixed adversary setting is in fact broader and more general than the traditional MPC setting. By instantiating Theorem 6 with the $(\lceil N/2 - 1\rceil, 0, 0)$-mixed adversary (this corresponds to the honest-majority setting against a malicious adversary), we immediately obtain the following corollary.

Corollary 2. *Assuming LWE, for any function f, there exists a three-round MPC protocol with guaranteed output delivery in the plain model that is secure against a malicious adversary in the honest majority setting. Furthermore, the protocol is reusable and has communication complexity $\mathsf{poly}(\lambda, d, N)$, where d is the depth of the circuit computing f.*

Like Theorem 6, this result is round-optimal and supports the maximum possible number of corruptions.

6.1 Security Against a Semi-Malicious Mixed Adversary

As a stepping stone to showing Theorem 6, we first construct a protocol that satisfies all the properties of Theorem 6, except that it is only secure against a semi-malicious mixed adversary (defined in Appendix B), which is simply a mixed adversary that corrupts some parties *semi-maliciously*, rather than maliciously. We describe below our three-round MPC protocol that is secure against a $(t_{\mathbf{Sm}}, t_{\mathbf{Sh}}, t_{\mathbf{Fc}})$-semi-malicious mixed adversary $\mathcal{A} = (\mathcal{A}_{\mathbf{Sm}}, \mathcal{A}_{\mathbf{Sh}}, \mathcal{A}_{\mathbf{Fc}})$ for $2t_{\mathbf{Sm}} + t_{\mathbf{Sh}} + t_{\mathbf{Fc}} < N$.

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 $\leq d$. Without loss of generality, assume $|x_i| = \lambda\ \forall i \in [N]$. Let $(\mathsf{DistSetup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PartDec}, \mathsf{FinDec})$ be the previously constructed threshold multi-key FHE scheme. Fix $(t_{\mathbf{Sm}}, t_{\mathbf{Sh}}, t_{\mathbf{Fc}})$ satisfying $2t_{\mathbf{Sm}} + t_{\mathbf{Sh}} + t_{\mathbf{Fc}} < N$. Let $\mathbb{A}$ be the $(N - t_{\mathbf{Sm}} - t_{\mathbf{Fc}})$-out-of-$N$ threshold access structure.

Protocol: We now describe our construction.

- **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 (that is, the $(N - t_{\mathbf{Sm}} - t_{\mathbf{Fc}})$-out-of-$|S_1|$ access structure).
    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 $\perp$.

[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.

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.

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 $\leq d$. Let $(t_{\mathbf{Sm}}, t_{\mathbf{Sh}}, t_{\mathbf{Fc}})$ be the corruption thresholds of the adversary, where $2t_{\mathbf{Sm}} + t_{\mathbf{Sh}} + t_{\mathbf{Fc}} < N$. Let $\mathbb{A}$ be the $(N - t_{\mathbf{Sm}} - t_{\mathbf{Fc}})$-out-of-$N$ access structure. Sim proceeds as follows:

- **Before Protocol Execution:** From the semi-malicious mixed adversary Adv, Sim receives a tuple of sets $(\mathcal{A}_{\mathbf{Sm}}, \mathcal{A}_{\mathbf{Sh}}, \mathcal{A}_{\mathbf{Fc}})$ of corrupted parties, with $|\mathcal{A}_{\mathbf{Sm}}| \leq t_{\mathbf{Sm}}$, $|\mathcal{A}_{\mathbf{Sh}}| \leq t_{\mathbf{Sh}}$, and $|\mathcal{A}_{\mathbf{Fc}}| \leq t_{\mathbf{Fc}}$.

- **Input Commitment Phase (Round 1):** For every fail-corrupt party that Adv wishes to abort in this round, Sim instructs the corresponding party. For each honest and each fail-corrupt party not yet instructed to abort, 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. Give $(\mathsf{params}_i, pk_i)$ as P_i's round 1 message to Adv.

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

- **Input Commitment Phase (Round 2):** For every fail-corrupt party that Adv wishes to abort in this round, Sim instructs the corresponding party. Then, 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 and fail-corrupt parties that send a message in round 2. Let S^1_{corr} be the set of corrupted (semi-malicious and semi-honest) 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 semi-maliciously and semi-honestly corrupted parties P_i in S_2, Sim receives the input x_i used by Adv and sends it to the trusted party. For the fail-corrupt parties that already aborted, Sim sends 0^λ to the trusted party.

- **Query to Ideal Functionality:** Sim receives the output b from the trusted party.
- **Computation Phase (Round 3):** For every fail-corrupt party that Adv wishes to abort in this round, Sim instructs the corresponding party. Let $\mathcal{CT} = \{ct_j\}_{j \in S_2}$. Let C' be the circuit induced by hardcoding the inputs to C corresponding to aborted fail-corrupt parties as 0^λ. Let S^3_{hon} be the set of honest and fail-corrupt parties that have not yet been told to abort in round 3 by Adv. For corrupted (semi-honest and semi-malicious) parties P_i in S^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 as P_j's round 3 message to Adv.

  Sim tells the trusted party to send b to all honest parties.

Lemma 6. *For any tuple of thresholds $(t_{\boldsymbol{Sm}}, t_{\boldsymbol{Sh}}, t_{\boldsymbol{Fc}})$ with $2t_{\boldsymbol{Sm}}+t_{\boldsymbol{Sh}}+t_{\boldsymbol{Fc}} < N$, for any $(t_{\boldsymbol{Sm}}, t_{\boldsymbol{Sh}}, t_{\boldsymbol{Fc}})$-semi-malicious mixed adversary $\mathsf{Adv} = (\mathcal{A}_{\boldsymbol{Sm}}, \mathcal{A}_{\boldsymbol{Sh}}, \mathcal{A}_{\boldsymbol{Fc}})$, for the above simulator Sim,*

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

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

Proof. Suppose there was some $(t_{\mathbf{Sm}}, t_{\mathbf{Sh}}, t_{\mathbf{Fc}})$-semi-malicious mixed adversary $\mathsf{Adv} = (\mathcal{A}_{\mathbf{Sm}}, \mathcal{A}_{\mathbf{Sh}}, \mathcal{A}_{\mathbf{Fc}})$ 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. Recall that $\mathbb{A}$ is the $N - t_{\mathbf{Sm}} - t_{\mathbf{Fc}}$-out-of-$N$ access structure. Adv' proceeds as follows.

1. Adv' runs Adv, which outputs a tuple of sets $(\mathcal{A}_{\mathbf{Sm}}, \mathcal{A}_{\mathbf{Sh}}, \mathcal{A}_{\mathbf{Fc}})$ of corrupted parties.
2. Adv outputs a set of fail-corrupt parties $S^1_{\mathsf{inp}} \subseteq \mathcal{A}_{\mathbf{Fc}}$ 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 $S = (\mathcal{A}_{\mathbf{Sm}} \cup \mathcal{A}_{\mathbf{Sh}}) \subseteq S_{\mathsf{parties}}$, 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 each $j \in S$, Adv will output $(\mathsf{params}_j, pk_j)$. By running Adv, Adv' is able to determine the randomness r^{KeyGen}_j used by Adv to generate pk_j and outputs $(\mathsf{params}_j, r^{\mathsf{KeyGen}}_j)$.
5. Let S^2_{hon} be the set of honest and fail-corrupt 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. By running Adv, Adv' is able to extract the input x_i and randomness r^{Encrypt}_i used by Adv for each $i \in S$. Adv' outputs $(x_i, r^{\mathsf{Encrypt}}_i)$ for all $i \in S$.

7. Let $S_2 = (S^2_{\mathsf{hon}} \cup \mathcal{A}_{\mathbf{Sm}} \cup \mathcal{A}_{\mathbf{Sh}})$ be the set of parties that sent a round 2 message. Let C' be the circuit induced by C by setting the input of all parties that did not send a round 2 message to 0^λ. Adv' outputs C' along with S_2.

8. Let S^3_{hon} be the set of honest and fail-corrupt parties that 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 $\{x_i\}_{i \notin S}$.

Since $\mathbb{A}$ is the $N - t_{\mathbf{Sm}} - t_{\mathbf{Fc}}$-out-of-$N$ access structure and $2t_{\mathbf{Sm}} + t_{\mathbf{Sh}} + t_{\mathbf{Fc}} < N$, it follows that $|\mathcal{A}_{\mathbf{Sm}} \cup \mathcal{A}_{\mathbf{Sh}}| \leq t_{\mathbf{Sm}} + t_{\mathbf{Sh}} < N - t_{\mathbf{Sm}} - t_{\mathbf{Fc}}$, and therefore, $\mathcal{A}_{\mathbf{Sm}} \cup \mathcal{A}_{\mathbf{Sh}} \notin \mathbb{A}'$ (the $N - t_{\mathbf{Sm}} - t_{\mathbf{Fc}}$-out-of-$N'$ access structure), so Adv' is a valid adversary for the TMFHE security game. 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.

□

Guaranteed Output Delivery and Input Fidelity. Observe that since all the honest parties and all the semi-honestly corrupted parties will never abort, there are at least $N - t_{\mathbf{Sm}} - t_{\mathbf{Fc}}$ partial decryption shares given out at the end of the protocol. Therefore, the output can be recovered and our protocol has guaranteed output delivery. Moreover, our protocol satisfies the property that the output of the computation is a function of the joint inputs of all parties, including those that aborted after the input commitment phase was completed. That is, in the scenario where the adversary corrupts a set of parties in a fail-corrupt manner, for every fail-corrupt party P_i that aborts after the input commitment phase, its input y_i that is used to compute the final output $C(y_1, \ldots, y_n)$ is set to be its actual input x_i used in the protocol so far and not a default input $\perp$. Recall that this is in line with our motivation for studying this setting where an honest but lazy party is not entirely discarded and its input is still considered in the computation if it aborted after the input commitment phase.

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.

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.

6.2 Handling a Malicious Mixed Adversary

In the above protocol, the adversary can only corrupt some subset $\mathcal{A}_{\mathbf{Sm}}$ of the parties semi-maliciously, some subset $\mathcal{A}_{\mathbf{Sh}}$ in a semi-honest manner and another subset $\mathcal{A}_{\mathbf{Fc}}$ in a fail-corrupt manner. In order to show Theorem 6, we need to allow the adversary to corrupt the first subset $\mathcal{A}_{\mathbf{Sm}}$ maliciously.

Our first observation is that the protocol is secure even against mixed adversaries that are allowed make parties in $\mathcal{A}_{\mathbf{Sm}}$ behave *maliciously* in round 1, but only semi-maliciously in rounds 2 and 3. After noting this, we further observe that if we had a simulation-extractable multi-string NIZK [GO07] in the plain model where the honest party's behavior when generating a CRS is to simply sample a uniformly random string[8], then we could upgrade to security against malicious mixed adversaries. 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 CRSs to ensure that they submitted a valid message in that round. As mentioned previously, the multi-string NIZK is only secure if a *majority* of the CRSs are honestly generated. However, we want our protocol to be secure against any $(t_{\mathbf{Mal}}, t_{\mathbf{Sh}}, t_{\mathbf{Fc}})$- mixed adversary, where $2t_{\mathbf{Mal}} + t_{\mathbf{Sh}} + t_{\mathbf{Fc}} < N$. In particular, we are no longer in the honest majority setting. As discussed earlier, this is not an issue because only the CRSs corresponding to a maliciously-corrupted party could be dishonestly generated and since the honest-generation behavior is to simply output a uniformly random string, a party that is semi-honestly corrupted will also output a perfectly good CRS. Furthermore, since the number of maliciously-corrupted parties is a minority of the total number of parties that send a CRS, a majority of the CRSs will be honestly generated and security of the multi-string NIZK holds.

Security Against a Round 1 Malicious Mixed Adversary. We begin by showing security of the protocol in Section 6.1 against a semi-malicious mixed adversary that can behave maliciously in round 1. Since params_i in the MFHE construction in [BHP17] is simply a matrix A_i of random entries, it follows that every A_i 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). However, a malicious adversary may send a pk_i that does not correspond to any possible public key output by the $\mathsf{TMFHE.KeyGen}$ algorithm. So, in the proof, 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 5, the simulator does not need to know sk_i or r_i^{KeyGen}. 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. Thus, an analogous simulator and proof can be used to show security against this adversary.

Upgrading to Malicious Security via Multi-String NIZKs. We now show how to use a simulation-extractable multi-string NIZK with uniformly random CRSs to upgrade the protocol in Section 6.1 to one that achieves Theorem 6. The final step is to show that such a multi-string NIZK can be built from LWE. This was not previously known, and we show this in Section 7.

Construction. 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 5 with the underlying PKE scheme in-

[8]For ease of exposition, we assume here that the honest CRS is a uniformly random string. However, there is a subtle technical issue, which we handle in Section 7 where we construct the multi-string NIZK.

stantiated with one where any string is a valid public key (a dense cryptosystem). Fix $(t_{\mathbf{Mal}}, t_{\mathbf{Sh}}, t_{\mathbf{Fc}})$ satisfying $2t_{\mathbf{Mal}} + t_{\mathbf{Sh}} + t_{\mathbf{Fc}} < N$. Let $\mathbb{A}$ be the $N - t_{\mathbf{Mal}} - t_{\mathbf{Fc}}$-out-of-$N$ threshold access structure. Let $\mathsf{NIZK} = (\mathsf{Gen}, \mathsf{Prove}, \mathsf{Verify})$ be a simulation-extractable multi-string NIZK. To compare against our previous protocol in Section 6.1, we highlight the changes in red.

- **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. Run $\mathsf{NIZK.Gen}(1^{\lambda'})$ to compute crs_i, where $\lambda' = \mathsf{poly}(\lambda, d, N)$ is the size of statements that will be proven.
  4. Output $(\mathsf{params}_i, pk_i, \mathsf{crs}_i)$.

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

  1. Parse the message (if one was sent) from P_j as $(\mathsf{params}_j, pk_j, \mathsf{crs}_j)$ 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 (that is, the $(N - t_{\mathbf{Sm}} - t_{\mathbf{Fc}})$-out-of-$|S_1|$ access structure).
  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).

- **Round 3:** 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 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).

- **Output Computation:** 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 6.1 and perfect completeness of the multi-string NIZK. Depth-proportional communication complexity follows from the fact that the communication complexity of the protocol in Section 6.1 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.

Guaranteed Output Delivery and Input Fidelity. Guaranteed output delivery and input fidelity follow the fact that these properties held for the protocol in Section 6.1 and that since honestly generated CRSs are always a majority (since honest strings are simply uniformly random and the number of malicious corruptions is a minority), by soundness of the multi-string NIZK, an adversary cannot cheat and submit an invalid ciphertext as its round 2 message since this message will be discarded with overwhelming probability. The output recovered is the same as that in the protocol of Section 6.1. 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.

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 $(t_{\mathbf{Mal}}, t_{\mathbf{Sh}}, t_{\mathbf{Fc}})$ be the corruption thresholds of the adversary, where $2t_{\mathbf{Mal}} + t_{\mathbf{Sh}} + t_{\mathbf{Fc}} < N$. Let $\mathbb{A}$ be the $(N - t_{\mathbf{Mal}} - t_{\mathbf{Fc}})$-out-of-$N$ access structure. 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:

- **<u>Before Protocol Execution:</u>** Sim receives a tuple of sets $(\mathcal{A}_{\mathbf{Mal}}, \mathcal{A}_{\mathbf{Sh}}, \mathcal{A}_{\mathbf{Fc}})$ of corrupted parties, with $|\mathcal{A}_{\mathbf{Mal}}| \leq t_{\mathbf{Mal}}$, $|\mathcal{A}_{\mathbf{Sh}}| \leq t_{\mathbf{Sh}}$, and $|\mathcal{A}_{\mathbf{Fc}}| \leq t_{\mathbf{Fc}}$.
- **<u>Round 1:</u>** For every fail-corrupt party that Adv wishes to abort in this round, Sim instructs the corresponding party. For each honest and each fail-corrupt party not yet instructed to abort, 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.

  For each semi-honest corrupt party $P_i \in \mathcal{A}_{\mathbf{Sh}}$, Sim does the following:

1. Sample randomness $r_i^{\mathsf{DistSetup}}$ and r_i^{KeyGen} to be used by the TMFHE.DistSetup and TMFHE.KeyGen algorithms, respectively.
2. Run $\mathsf{ExtGen}(1^\lambda)$ to compute $(\mathsf{crs}_i, \tau_i, \xi_i)$.
3. Give $(r_i^{\mathsf{DistSetup}}, r_i^{\mathsf{KeyGen}}, \mathsf{crs}_i)$ as P_i's round 1 randomness (note that this forces P_i to output crs_i as its CRS, as the CRS is uniform).

Sim then receives round 1 messages from Adv on behalf of every party in the sets $\mathcal{A}_{\mathbf{Mal}}$ and $\mathcal{A}_{\mathbf{Sh}}$. Let S_{crs} denote the set of honest parties, semi-honest parties, and fail-corrupt parties that sent a message in round 1.

- **Round 2:** For every fail-corrupt party that Adv wishes to abort in this round, Sim instructs the corresponding party. Then, 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^2_{hon} be the set of honest and fail-corrupt parties that send a message in round 2. Let $T = \{\tau_j\}_{j \in S_{\mathsf{crs}}}$. Let $E = \{\xi_j\}_{j \in S_{\mathsf{crs}}}$. Let S^1_{corr} be the set of corrupted (malicious or semi-honest) 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 obtain $(\{ct_i\}_{i \in S^2_{\mathsf{hon}}}, \mathsf{state})$, where Sim_1 is the first algorithm of the TMFHE simulator.
  2. For each honest and fail-corrupt party not yet instructed to abort, 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 $i \in S^2_{\mathsf{hon}}$.

  Sim then receives round 2 messages from Adv on behalf of every party in the sets $\mathcal{A}_{\mathbf{Mal}}$ and $\mathcal{A}_{\mathbf{Sh}}$.

- **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 semi-honest parties P_j in S_2, Sim receives the input x_j used by Adv and sends it to the trusted party. For the fail-corrupt and malicious parties that already aborted, Sim sends 0^λ to the trusted party. For malicious 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.
  2. Sim receives the output b from the trusted party.

- **Round 3:** For every fail-corrupt party that Adv wishes to abort in this round, Sim instructs the corresponding party. Let $\mathcal{CT} = \{ct_j\}_{j \in S_2}$. Let C' be the circuit induced by hardcoding the inputs to C corresponding to 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. Let S^3_{hon} be the set of honest and fail-corrupt parties that have not yet been told to abort in round 3 by Adv. Sim does the following

1. Run $\mathsf{Sim}_2(\mathsf{state}, b, \hat{ct}, S^1_{\mathsf{corr}}, S^2_{\mathsf{hon}}, \{(x_i, r_i)\}_{i \in S^2_{\mathsf{corr}}})$ to obtain $\{p_j\}_{j \in S^2_{\mathsf{hon}}}$, 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 $j \in S^3_{\mathsf{hon}}$, 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 $j \in S^3_{\mathsf{hon}}$, give (p_j, π'_j) as P_j's round 3 message to Adv.

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 6.1.
2. The NIZK in round 3 can be generated afresh for different invocations of the protocol while preserving security.

7 Multi-String NIZKs

In this section, we build a simulation-extractable multi-string NIZK argument system (Appendix A.4) for NP based on the learning with errors (LWE) assumption. We first show how to build a multi-string non-interactive witness indistinguishable argument system (NIWI) from LWE. We then give a transformation from multi-string NIWI to multi-string simulation-extractable NIZK that follows along the lines of the work of Groth and Ostrovsky [GO07]. Formally, we show the following results:

Theorem 7. *Assuming LWE, there exists a multi-string non-interactive witness indistinguishable argument system for NP.*

Theorem 8. *Assuming LWE, there exists a multi-string simulation-extractable NIZK argument system for NP.*

One of the key tools in our constructions is a Sigma protocol (Appendix A.6). Before we describe the construction of our multi-string NIWI protocol, in the next subsection, we describe a specific trapdoor commitment scheme that we will use to instantiate the Sigma protocol with, in the multi-string NIWI protocol. In the following subsection, we give the construction and security proof of our multi-string NIWI protocol and in the final subsection, we describe the generic transformation from multi-string NIWI to multi-string NIZK.

7.1 Commitment Scheme

In this section, we construct a new non-interactive commitment scheme $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Decom})$ in the CRS model assuming LWE. In addition to the standard properties of a commitment scheme, we require that the scheme has a trapdoor td such that given the commitment string, the trapdoor can be used to efficiently generate the decommitment information with overwhelming probability. Furthermore, we additionally have the feature that even if the adversary generates some portion of the CRS, the scheme still remains hiding and binding as long as a majority of the components are honestly generated. We elaborate more on this after the construction.

The construction and properties of the scheme are below. Let λ be the security parameter. Let $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a semantically secure public key encryption scheme based on LWE. Let $(\mathsf{Share}, \mathsf{Recon})$ be a $(\lfloor n/2 \rfloor + 1)-$out-of-$n$ threshold secret sharing scheme [Sha79].

1. $\mathsf{Setup}(1^\lambda, 1^n)$: For each $i \in [n]$, compute $(pk_i, sk_i) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$. Set $\mathsf{crs} = (pk_1, \ldots, pk_n)$.
2. $\mathsf{Commit}(\mathsf{crs} = pk_1, ..., pk_n, \mathsf{msg})$: The commitment algorithm does the following:
   - Compute $m_1, ..., m_n \leftarrow \mathsf{Share}(\mathsf{msg})$ - that is, they are the shares upon secret sharing the input msg.
   - For each $i \in [n]$, compute $ct_i \leftarrow \mathsf{PKE.Enc}(pk_i, m_i; r_i)$ where r_i is uniformly generated.
   - Output $ct = (ct_1, ..., ct_n)$.
3. $\mathsf{Decom}(ct)$: The decommitment algorithm outputs the tuples of values $\{(m_i, r_i)\}_{i \in [n]}$ where m_i is the share of the message and r_i is the randomness used to encrypt m_i. The verifier outputs 1 if:
   - For each $i \in [n]$, $ct_i = \mathsf{PKE.Enc}(m_i, pk_i; r_i)$.
   - $\mathsf{Recon}(m_1, \ldots, m_n) \neq \perp$.

We now list some properties of the commitment scheme. For both hiding and binding, we consider the stronger scenario where there exists a set $S \subset [n]$ of size $(\lfloor n/2 \rfloor + 1)$, where $\{pk_i\}_{i \in S}$ are generated honestly using $\mathsf{PKE.Setup}$ and pk_i for $i \in [n] \setminus S$ are chosen by a PPT adversary on seeing $\{pk_i\}_{i \in S}$. That is, the adversary gets to pick part of the CRS. This will be crucial in the application of our commitment scheme to the Multi-String NIWI protocol.

Hiding: Since an honest majority of the public key-secret key pairs in the CRS were honestly generated, from the security of the public key encryption scheme and the threshold secret sharing scheme, it is easy to see that the commitment scheme satisfies hiding.

Binding: Now, for any commitment string $ct = (ct_1, ..., ct_n)$, with overwhelming probability over the choice of the randomness used to honestly generate pk_i for $i \in S$, there exists at most one message m such that there exists m_i, r_i for $i \in [n]$ satisfying:

1. $m_1, ..., m_n$ forms a secret sharing of m.
2. $ct_i = \mathsf{PKE.Enc}(pk_i, m_i; r_i)$ for $i \in [n]$.

Thus, the scheme satisfies binding.

Trapdoor: Note that given a set of secret keys $\{sk_i\}_{i \in \mathcal{S}}$ where $|\mathcal{S}| > \frac{n}{2}$ and a commitment string $ct = (ct_1, \ldots, ct_n)$, the message committed can be recovered efficiently as follows: for each $i \in \mathcal{S}$, compute $m_i = \mathsf{PKE.Dec}(sk_i, ct_i)$. Then, recover the message committed as $\mathsf{msg} = \mathsf{Recon}(\{m_i\}_{i \in \mathcal{S}})$. Thus, given a CRS $(pk_1, \ldots, pk_n)$, the associated trapdoor $\mathsf{td} = (\{sk_i\}_{i \in \mathcal{S}})$ for any set $\mathcal{S}$ with $|\mathcal{S}| > \frac{n}{2}$ where sk_i is the secret key corresponding to the public key pk_i.

7.2 Multi-String NIWI

We now describe our construction of a multi-string non-interactive witness indistinguishable argument system below. Let λ de the security parameter which also denotes the size of the input instances x. Let L be the NP language under consideration. Let Σ be a Sigma protocol as defined in Appendix A.6 which can be based on LWE (due to the commitment scheme). Let m be the number of parallel repetitions used inside the protocol Σ. Let n denote the maximum number of parties in the system. Consider a relation family $\mathcal{R} = \{\mathcal{R}_\lambda\}_{\lambda \in Z}$ defined as follows: $\mathcal{R}_\lambda$ consists of tuples $((x, a), y)$ where: $|x| = \lambda$, $|a|$ = size of the first message of protocol Σ, $|y| = m$ = size of the second message of protocol Σ and given (x, a), y can be efficiently computed by a circuit of size equal to the size of the circuit computing the second message of the Sigma protocol. Let ℓ denote the size of representing any relation in $\mathcal{R}_\lambda$. Let $\mathcal{H}$ be a correlation intractable function (Appendix A.5) for the relation family $\mathcal{R}$. Peikert and Shiehian [PS19] recently constructed such a hash function based on LWE. Let $\mathsf{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a semantically secure public key encryption scheme based on LWE.

1. $\mathsf{Setup}(1^\lambda, 1^n)$: The setup algorithm takes as input the security parameter λ (which also fixes the length of the instances) and the maximum number of parties n and does the following.
   - Sample $(pk, sk) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$
   - Sample $K \leftarrow \mathcal{H}.\mathsf{Setup}(1^\lambda, 0^\ell)$ where ℓ is defined before the construction.
   - Output $\mathsf{crs} = (pk, K)$.
2. $\mathsf{Prove}(\mathsf{CRS}, x, w)$: The prove algorithm takes as input $\mathsf{CRS} = (\mathsf{crs}_1, ..., \mathsf{crs}_n)$ where each $\mathsf{crs}_i = (pk_i, K_i)$ and does the following:
   - For each index $i \in [n]$, compute $a_i = (c_{i,1}, c_{i,2})$ where $c_{i,1}$ and $c_{i,2}$ are commitments computed according to the first message of the Σ protocol for the statement $x \in L$ by running the algorithm Commit from the previous section with the input crs being $(pk_1, \ldots, pk_n)$.
   - Compute $\mathcal{H}.\mathsf{Eval}(K_i, x, a_i) \rightarrow e_i$.
   - For each $i \in [n]$, use a_i, e_i and the witness w to compute the third message z_i of the Σ protocol for the statement $x \in L$.
   - Output $(\{a_i, e_i, z_i\}_{i \in [n]})$ as the proof.
3. $\mathsf{Verify}(\mathsf{CRS}, x, \sigma)$: Parse $\sigma = (\{a_i, e_i, z_i\}_{i \in [n]})$, $\mathsf{CRS} = (\mathsf{crs}_1, ..., \mathsf{crs}_n)$ where each $\mathsf{crs}_i = (pk_i, K_i)$. For each $i \in [n]$, do:
   - Check if $\mathcal{H}.\mathsf{Eval}(K_i, x, a_i) = e_i$.
   - Check if a_i, e_i, z_i verifies according to the Σ protocol.

   Output 1 if all the above verifications pass.

Completeness. Completeness of the protocol can be easily observed from the correctness of the underlying primitives: the protocol Σ and the hash function H.

7.2.1 Security Proof

Soundness. Consider an adversary $\mathcal{A}$ and a challenger Ch. We now prove computational soundness of the protocol above. We do so via a pair of hybrids where the first hybrid corresponds to the real soundness experiment.

- Hyb_0 : This hybrid corresponds to the honest soundness experiment.
  - First, the adversary $\mathcal{A}$ declares a set $\mathcal{S} \subset [n]$ of size $(\lfloor n/2 \rfloor + 1)$.
  - For each $i \in \mathcal{S}$, Ch generates a string crs_i as follows.
    * Compute $(pk_i, sk_i) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$.
    * Compute $K_i \leftarrow \mathcal{H}.\mathsf{Setup}(1^\lambda, 0^\ell)$.
    * Set $\mathsf{crs}_i = (K_i, pk_i)$.
  - On input $\{\mathsf{crs}_i\}_{i \in \mathcal{S}}$, adversary computes crs_j for each $j \notin \mathcal{S}$.
  - Finally, $\mathcal{A}$ outputs the remaining part of the CRS $\{\mathsf{crs}_j\}_{j \notin \mathcal{S}}$ together with the statement x^* and proof $(\{a_i^*, e_i^*, z_i^*\}_{i \in [n]})$.
  - The adversary wins if $x^* \notin L$ and the proof verifies.
- Hyb_1 : This hybrid is the same as the previous hybrid except that for each $i \in S$, K_i is generated as follows. $K_i \leftarrow \mathcal{H}.\mathsf{Setup}(1^\lambda, R_i^*)$ where the relation R_i^* consists of tuples of the form $((x^*, a_i^*), y_i^*)$ where y_i^* is as follows: Consider function $f_{bad,\lambda,m,\{sk_i\}_{i \in \mathcal{S}}}$ that takes as input (x^*, a_i^*) and computes the string $e_{bad,i}$ such that there exists $z_{bad,i}$ and $(a_i^*, e_{bad,i}, z_{bad,i})$ verifies according to the Sigma protocol. $y_i^* = (e_{bad,i})$. Recall that if $x^* \notin L$, then there exists at most one such string $e_{bad,i}$ for any a_i^*.

We now complete the proof of soundness with the following claims.

Lemma 7. *Assuming the statistical indistinguishability of hash keys property of the correlation intractable hash function, for any polynomial time adversary $\mathcal{A}$,* $\Pr[\mathcal{A} \text{ wins in } \mathsf{Hyb}_1] \geq \Pr[\mathcal{A} \text{ wins in } \mathsf{Hyb}_0] - \mathsf{negl}$

Proof. The only difference between the two hybrids is that for each $i \in S$, K_i is generated differently. It is generated as $\mathcal{H}.\mathsf{Setup}(1^\lambda, 0^\ell)$ in Hyb_0, whereas it is generated as $\mathcal{H}.\mathsf{Setup}(1^\lambda, f_{bad,\lambda,m,\{sk_i\}_{i \in \mathcal{S}}})$ in Hyb_1. From the statistical indistinguishability of hash keys property of the correlation intractable hash function, the two hybrids are statistically indistinguishable and this proves the claim. □

Lemma 8. *Assuming the computational correlation intractable property of the hash function, for any polynomial time adversary $\mathcal{A}$,* $\Pr[\mathcal{A} \text{ wins in } \mathsf{Hyb}_1] \leq \mathsf{negl}(\lambda)$.

Proof. This claim is true due to the computational correlation intractable property of the hash function $\mathcal{H}$. If the adversary breaks soundness then with non-negligible probability, by the soundness property of the underlying Sigma protocol, it must hold that $f_{bad,\lambda,m,\{sk_i\}_{i \in \mathcal{S}}}(x^*, a_i) = e_i^*$ for each $i \in \mathcal{S}$ where e_i^* is the message output by $\mathcal{A}$ as the second round message of the Σ protocol. Therefore, we can build a reduction that uses the adversary $\mathcal{A}$ to compute $y_i^* = e_i^*$ for each $i \in \mathcal{S}$ such that $((x^*, a_i^*), y_i^*) \in R_i^*$, thus breaking the correlation intractable property of the hash function $\mathcal{H}$, which is a contradiction. □

This completes the proof.

Witness Indistinguishability. Let $\mathcal{A}$ denote the adversary and Ch denote the challenger. Let x be the challenge instance of length λ and w_0 and w_1 be the corresponding witness. We prove witness indistinguishability via a sequence of computationally indistinguishable hybrids where the first hybrid corresponds to the witness w_0 being used and the last hybrid correspond to witness w_1 being used.

- Hyb_0 : This hybrid is described as follows:
  1. $\mathcal{A}$ declares a set $\mathcal{S}$ of size $\lfloor n/2 \rfloor + 1$.
  2. Ch generates crs_i for $i \in \mathcal{S}$ as follows.
     - Generate $(pk_i, sk_i) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$.
     - Generate $K_i \leftarrow \mathcal{H}.\mathsf{Setup}(1^\lambda, 0^\ell)$.
     - Set $\mathsf{crs}_i = (K_i, pk_i)$.
  3. On input crs_i for $i \in \mathcal{S}$, $\mathcal{A}$ computes crs_i for $i \in [n] \setminus \mathcal{S}$. Set $\mathsf{CRS} = (\mathsf{crs}_1, ..., \mathsf{crs}_n)$
  4. Then, the challenger Ch uses w_b to generate proof honestly $(x, a_1, ..., a_n, , e_1, ..., e_n, z_1, ..., z_n)$.
- Hyb_j for each $(j \in [n])$: This hybrid is the same as Hyb_{j-1} except that now, for index $i = j$, the tuple (a_i, e_i, z_i) in the proof is generated using witness w_1. Note that Hyb_n corresponds to the experiment where the challenger runs the honest prover algorithm using witness w_1.

We now complete the proof by arguing that every pair of hybrids are computationally indistinguishable.

Lemma 9. *For all $j \in [n]$, Hyb_j is computationally indistinguishable from Hyb_{j-1} assuming the witness indistinguishability property of Blum's Sigma protocol.*

Proof. The only difference between the two hybrids is in how the tuple (a_j, e_j, z_j) is generated in the proof. In Hyb_{j-1}, it is generated using witness w_0 whereas in Hyb_j, it is generated using witness w_1. Before proceeding to the proof, we first set up some notation. Recall from the description of the Sigma protocol that we in fact have m parallel repetitions of Blum's protocol. Therefore, let's denote $a_j = (a_{j,1}, \dots, a_{j,m}), e_j = (e_{j,1}, \dots, e_{j,m}), z_j = (z_{j,1}, \dots, z_{j,m})$. We now prove this lemma via a sequence of computationally indistinguishable sub-hybrids below where $\mathsf{Sub.Hyb}_0$ corresponds to Hyb_{j-1} and $\mathsf{Sub.Hyb}_m$ corresponds to Hyb_j.

- $\mathsf{Sub.Hyb}_0$ corresponds to Hyb_{j-1}.
- $\mathsf{Sub.Hyb}_k$ for $k \in [m]$: Is identical to the previous sub-hybrid $\mathsf{Sub.Hyb}_{k-1}$ except that the tuple $(a_{j,k}, e_{j,k}, z_{j,k})$ is now computed using witness w_1.

From the witness indistinguishability property of the underlying Blum's Sigma protocol, it is easy to observe that $\mathsf{Sub.Hyb}_k$ is indistinguishable from $\mathsf{Sub.Hyb}_{k-1}$ for all $k \in [m]$. Thus, this completes the proof. □

7.3 Multi-String NIZK from Multi-String NIWI

We now describe the transformation from a multi-string NIWI argument system to a multi-string simulation-extractable NIZK argument system. Let λ be the security parameter which also denotes the size of the input instances x. Let L be the NP language under consideration and R be the corresponding relation. Let n denote the maximum number of parties in the system. Let $\mathsf{MSNIWI} = (\mathsf{MSNIWI.Setup}, \mathsf{MSNIWI.Prove}, \mathsf{MSNIWI.Verify})$ be a multi-string NIWI argument system based on LWE from the previous section. Let G be a length doubling pseudorandom generator that takes a seed of length λ as input. Let $\mathsf{PKE} = \mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a CCA secure encryption scheme. Let $(\mathsf{Share}, \mathsf{Recon})$ be a $(\lfloor n/2 \rfloor + 1)-$out-of-$n$ threshold secret sharing scheme. The construction of the multi-string simulation-extractable NIZK is described below.

1. $\mathsf{Setup}(1^\lambda, 1^n)$: The setup algorithm takes as input the security parameter λ (which also fixes the length of the instances) and the maximum number of parties n and does the following.
   - Compute $\mathsf{crs}' \leftarrow \mathsf{MSNIWI.Setup}(1^\lambda, 1^n)$.
   - Pick a string r of length $2 \cdot \lambda$ uniformly at random.
   - Compute $(pk, sk) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$.
   - Output $\mathsf{crs} = (\mathsf{crs}', r, pk)$.
2. $\mathsf{Prove}(\mathsf{CRS}, x, w)$: The prove algorithm takes as input $\mathsf{CRS} = (\mathsf{crs}_1, ..., \mathsf{crs}_n)$ where each $\mathsf{crs}_i = (\mathsf{crs}'_i, r_i, pk_i)$ and does the following:
   - Compute $w_1, ..., w_n \leftarrow \mathsf{Share}(w)$.
   - Compute $ct = (ct_1, \ldots, ct_n)$ where for each $i \in [n]$, $ct_i = \mathsf{PKE.Enc}(pk_i, w_i; rw_i)$.
   - Compute $\pi \leftarrow \mathsf{MSNIWI.Prove}(\mathsf{CRS}' = (\mathsf{crs}'_1, \ldots, \mathsf{crs}'_n), y = (x, ct, r_1, \ldots, r_n), w')$ for the statement $y \in L_1$ using witness $w' = (w, rw_1, \ldots, rw_n, \perp)$ where the NP language L_1 is defined by the following relation R_1:
     statement: $y = (x, ct, r_1, \ldots, r_n)$
     witness: $w' = (w, rw_1, \ldots, rw_n, s_1, \ldots, s_n)$
     $R_1(y, w') = 1$ iff
     - $ct_i = \mathsf{PKE.Enc}(pk_i, w_i; rw_i)$ for each $i \in [n]$ such that $\mathsf{Recon}(w_1, \ldots, w_n) = w$ and $(x, w) \in R$ (OR)
     - $\exists$ a set $\mathcal{S}$ of size $(\lfloor \frac{n}{2} \rfloor + 1)$ such that for each $i \in \mathcal{S}$, $\mathsf{G}(s_i) = r_i$.
   - Output (x, π, ct).
3. $\mathsf{Verify}(\mathsf{CRS}, x, (\pi, ct))$: Output $\mathsf{MSNIWI.Verify}(\mathsf{CRS}' = (\mathsf{crs}'_1, \ldots, \mathsf{crs}'_n), y = (x, ct, r_1, \ldots, r_n), \pi)$ for the language L_1.

Completeness. Completeness of the protocol can be easily observed from the correctness of the underlying primitives: the multi-string NIWI protocol MSNIWI, the encryption scheme PKE and the pseudorandom generator G.

7.3.1 Security Proof

Soundness. Consider an adversary $\mathcal{A}$ that breaks the soundness property - that is, $\mathcal{A}$ outputs a statement $x \notin L$ and a proof (π, ct) such that $\mathsf{Verify}(\mathsf{CRS}, x, (\pi, ct)) = 1$ with non-negligible probability. First, observe that from the soundness of the underlying multi-string NIWI argument system, since the proof verifies successfully, the statement $y = (x, ct, r_1, \ldots, r_n) \in L_1$. Hence, one of the two statements in the relation R_1 must be true. However, since at least $(\frac{n}{2}+1)$ of the strings r_i were chosen uniformly at random by the Challenger, the probability that any of them would be the output of the pseudorandom generator is negligible. Thus, the probability that the second statement in relation R_1 is true is negligible. Therefore, the first statement in R_1 must be true which implies that $x \in L$ which is a contradiction. This proves that the multi-string NIZK system is sound.

Zero Knowledge We now prove the zero knowledge property for our construction. The description of the simulator Sim is given below.

1. $\mathsf{Setup}(1^\lambda, 1^n)$: For each honest party, the simulator does the following:
   - Compute $\mathsf{crs}' \leftarrow \mathsf{MSNIWI.Setup}(1^\lambda, 1^n)$.
   - Pick a string s of length λ uniformly at random and compute $r = \mathsf{G}(s)$.
   - Compute $(pk, sk) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$.
   - Output $\mathsf{crs} = (\mathsf{crs}', r, pk)$.
2. $\mathsf{Prove}(\mathsf{CRS}, x)$: The simulator's prove algorithm takes as input $\mathsf{CRS} = (\mathsf{crs}_1, ..., \mathsf{crs}_n)$ where each $\mathsf{crs}_i = (\mathsf{crs}'_i, r_i)$ and does the following:
   - Denote set $\mathcal{S} = \{s_i\}$ of size at least $(\frac{n}{2} + 1)$ where for each $i \in \mathcal{S}$, $\mathsf{G}(s_i) = r_i$. These are the PRG seeds chosen by the simulator in the setup phase on behalf of the honest parties.
   - Compute $ct = (ct_1, \ldots, ct_n)$ where for each $i \in \mathcal{S}$, $ct_i = \mathsf{PKE.Enc}(pk_i, 0; rw_i)$ and for each $i \notin \mathcal{S}$, $ct_i = \mathsf{PKE.Enc}(pk_i, w_i; rw_i)$ where w_i is picked uniformly at at random.
   - Compute $\pi \leftarrow \mathsf{MSNIWI.Prove}(\mathsf{CRS}' = (\mathsf{crs}'_1, \ldots, \mathsf{crs}'_n), y = (x, ct, r_1, \ldots, r_n), w')$ for the statement $y \in L_1$ using witness $w' = (\perp, \{s_i\}_{i \in \mathcal{S}})$ for the trapdoor statement.
   - Output (x, π, ct).

We now prove that the real and ideal worlds are computationally indistinguishable via a sequence of hybrids. Consider a simulator SimHyb. The first hybrid Hyb_0 corresponds to the real world where SimHyb behaves as an honest prover who has both (x, w) in its interaction with the adversary and the last hybrid corresponds to the ideal world where SimHyb behaves as the simulator Sim who has access only to the statement x in its interaction with the adversary.

- Hyb_0 : This hybrid corresponds to the real world where the adversary interacts with an honest prover.
- Hyb_1: In this hybrid, in the setup phase, on behalf of each honest party, the simulator SimHyb picks r as done in the ideal world as follows: pick a string s of length λ uniformly at random and compute $r = \mathsf{G}(s)$.

- Hyb_2: In this hybrid, the simulator SimHyb computes the proof using the trapdoor statement of the multi-string NIWI by relying on the knowledge of the pre-images to the pseudorandom generator $\{s_i\}_{i \in \mathcal{S}}$ where $\mathcal{S}$ denotes the set of honest parties. This is identical to how the proof is computed in the ideal world.

- Hyb_3: In this hybrid, SimHyb computes $ct = (ct_1, \ldots, ct_n)$ where for each $i \in \mathcal{S}$, $ct_i = \mathsf{PKE.Enc}(pk_i, 0; rw_i)$.

- Hyb_4: In this hybrid, SimHyb computes $ct = (ct_1, \ldots, ct_n)$ where for each $i \in [n]/\mathcal{S}$, $ct_i = \mathsf{PKE.Enc}(pk_i, w_i; rw_i)$ where w_i is picked uniformly at at random and not as secret shares of the witness w. This hybrid is identical to the ideal world.

We now prove that every pair of consecutive hybrids is computationally indistinguishable and this completes the proof of zero knowledge.

Lemma 10. *Assuming the security of the pseudorandom generator* G, Hyb_0 *is computationally indistinguishable from* Hyb_1.

Proof. The only difference between the two hybrids is that in Hyb_0, the values r in the CRS are generated uniformly at random while in Hyb_1, they are generated as output of the pseudorandom generator G. Thus, if there exists an adversary $\mathcal{A}$ that can distinguish these two hybrids with non-negligible probability, we can use $\mathcal{A}$ to break the security of the pseudorandom generator which is a contradiction. □

Lemma 11. *Assuming the witness indistinguishability property of the multi-string NIWI argument system,* Hyb_1 *is computationally indistinguishable from* Hyb_2.

Proof. The only difference between the two hybrids is that in Hyb_1, the proof π is generated using the first statement in the multi-string NIWI while in Hyb_2, π is generated using the trapdoor statement that requires knowledge of the pseudorandom generator pre-images. Thus, if there exists an adversary $\mathcal{A}$ that can distinguish these two hybrids with non-negligible probability, we can use $\mathcal{A}$ to break the witness indistinguishability property of the multi-string NIWI which is a contradiction. □

Lemma 12. *Assuming the semantic security of the public key encryption scheme,* Hyb_2 *is computationally indistinguishable from* Hyb_3.

Proof. The only difference between the two hybrids is that for each $i \in \mathcal{S}$, in Hyb_2, the values ct_i are computed as encryptions of the shares of the witness w while in Hyb_3, they are computed as encryption of 0. Observe that only the public key is given to the adversary. Thus, if there exists an adversary $\mathcal{A}$ that can distinguish these two hybrids with non-negligible probability, we can use $\mathcal{A}$ to break the semantic security of the encryption scheme which is a contradiction. □

Lemma 13. *Assuming the security of the secret sharing scheme,* Hyb_3 *is computationally indistinguishable from* Hyb_4.

Proof. The only difference between the two hybrids is that for each $i \notin \mathcal{S}$, $ct_i = \mathsf{PKE.Enc}(pk_i, w_i; rw_i)$ where in Hyb_3, w_i is a secret share of the witness w while in Hyb_4, w_i is picked uniformly at random.

Since the size of the set $\mathcal{S}$ is at least $(\frac{n}{2}+1)$, the number of these w_i values are lesser than the threshold for the secret sharing scheme. Thus, if there exists an adversary $\mathcal{A}$ that can distinguish these two hybrids with non-negligible probability, we can use $\mathcal{A}$ to break the security of the secret sharing scheme which is a contradiction.

□

Simulation Extractability We now prove that the above scheme is simulation extractable - that is, there exists an extractor Ext that, on input a successful proof produced by the adversary $\mathcal{A}$ for any statement x can extract a corresponding witness w for $x \in L$, even when $\mathcal{A}$ has access to an oracle that produces simulated proofs (as shown in the zero knowledge proof). We first describe the extractor Ext below before proving the above property.

1. $\mathsf{ExtGen}(1^\lambda, 1^n)$: For each honest party, the extractor's setup algorithm generates the CRS and the associated trapdoors as done by the simulator in the ideal world. That is, it does the following:
   - Compute $\mathsf{crs}' \leftarrow \mathsf{MSNIWI.Setup}(1^\lambda, 1^n)$.
   - Pick a string s of length λ uniformly at random and compute $r = \mathsf{G}(s)$.
   - Compute $(pk, sk) \leftarrow \mathsf{PKE.Setup}(1^\lambda)$.
   - Output $\mathsf{crs} = (\mathsf{crs}', r, pk)$.
2. $\mathsf{Ext}(x, (\pi, ct))$: On input a statement x and a proof (π, ct) from the adversary $\mathcal{A}$, the extractor does the following:
   - Denote set $\mathcal{S} = \{s_i\}$ of size at least $(\frac{n}{2}+1)$ where for each $i \in \mathcal{S}$, Ext knows sk_i generated as part of the setup phase.
   - For each $i \in \mathcal{S}$, compute $w_i = \mathsf{PKE.Dec}(ct_i, sk_i)$.
   - Compute and output $w = \mathsf{Recon}(\{w_i\}_{i \in \mathcal{S}})$.

We now prove the simulation-extraction property by a series of hybrid arguments. For ease of notation, lets denote the output of the hybrid to be 1 if in that hybrid, with non-negligible probability, the extractor algorithm Ext fails to output a valid witness w but the adversary's proof verifies successfully. Very briefly, the proof follows the same structure as in the case of the zero knowledge argument - we first go from the simulated world to the real world where the oracle provides honestly generated proofs. We argue that the adversary's advantage doesn't change in this transition. Finally, we argue that in the real world, the adversary's advantage is negligible by the same argument as in the soundness of the protocol.

- Hyb_0: This corresponds to the ideal world experiment where the adversary has access to an oracle that produces simulated proofs.
- Hyb_1: In this hybrid, the simulator computes $ct = (ct_1, \ldots, ct_n)$ where for each $i \in [n]/\mathcal{S}$, $ct_i = \mathsf{PKE.Enc}(pk_i, w_i; rw_i)$ where w_i are secret shares of the witness w.
- Hyb_2: In this hybrid, the simulator computes $ct = (ct_1, \ldots, ct_n)$ where for each $i \in \mathcal{S}$, $ct_i = \mathsf{PKE.Enc}(pk_i, w_i; rw_i)$.

- Hyb_3: In this hybrid, the simulator SimHyb computes the NIWI using the witness w as done by the honest prover algorithm.
- Hyb_4: In this hybrid, in the setup phase, on behalf of each honest party, algorithm ExtGen picks r uniformly at random as done in the real world.

Lemma 14. *Assuming the security of the secret sharing scheme,* $|Pr[\mathsf{Hyb}_0 = 1] - Pr[\mathsf{Hyb}_1 = 1]| \leq \mathsf{negl}(\lambda)$.

Proof. This proof is identical to the proof of Lemma 13 in the zero knowledge proof. In particular, if the adversary's advantage changes between the two hybrids, we can use that to break the security of the secret sharing scheme. □

Lemma 15. *Assuming the CCA security of the encryption scheme,* $|Pr[\mathsf{Hyb}_1 = 1] - Pr[\mathsf{Hyb}_2 = 1]| \leq \mathsf{negl}(\lambda)$.

Proof. This proof is somewhat similar to the proof of Lemma 12 in the zero knowledge proof. In particular, if the adversary's advantage changes between the two hybrids, we can use that to break the CCA security of the public key encryption scheme. The only difference from that proof here is that in the reduction to the CCA secure encryption scheme, we now need access to the decryption oracle to run the extractor algorithm Ext which was not needed in the proof of Lemma 12. □

Lemma 16. *Assuming the witness indistinguishability property of the multi-string NIWI system,* $|Pr[\mathsf{Hyb}_2 = 1] - Pr[\mathsf{Hyb}_3 = 1]| \leq \mathsf{negl}(\lambda)$.

Proof. This proof is identical to the proof of Lemma 11 in the zero knowledge proof. In particular, if the adversary's advantage changes between the two hybrids, we can use that to break the witness indistinguishability property of the multi-string NIWI argument system. □

Lemma 17. *Assuming the security of the pseudorandom generator,* $|Pr[\mathsf{Hyb}_3 = 1] - Pr[\mathsf{Hyb}_4 = 1]| \leq \mathsf{negl}(\lambda)$.

Proof. This proof is identical to the proof of Lemma 10 in the zero knowledge proof. In particular, if the adversary's advantage changes between the two hybrids, we can use that to break the security of the pseudorandom generator. □

Finally, we will argue that the probability that Hyb_4 outputs 1 is negligible and this completes the proof.

Lemma 18. *Assuming the soundness of the multi-string NIWI argument, correctness of the CCA secure encryption scheme and correctness of the secret sharing scheme,* $Pr[\mathsf{Hyb}_4 = 1] \leq \mathsf{negl}(\lambda)$.

Proof. As in the proof of soundness, we can observe from the soundness of the multi-string NIWI argument system that if the adversary produces a statement x and a proof (π, ct) that verifies successfully, then it must be the case that $x \in L$. Further, $(x, w) \in R$ where R is the NP relation for language L and $ct = (ct_1, \ldots, ct_n)$ where for each $i \in [n]$, $ct_i = \mathsf{PKE.Enc}(w_i, pk_i)$ and $\{w_i\}_{i \in [n]}$ is a secret sharing of the witness w. Therefore, by the correctness of the reconstruction algorithm of the secret sharing scheme and the decryption algorithm of the encryption scheme, the extractor outputs this witness w with overwhelming probability. Thus, the probability that the adversary $\mathcal{A}$ outputs a statement x and a proof (π, ct) that successfully verifies but the extractor doesn't output a corresponding witness w for the statement $x \in L$ is negligible and this completes the proof. □

Common Random String. Observe that if the CCA secure encryption scheme used in our construction and the one underlying the multi-string NIWI has the property that the public keys are statistically-close to uniform, then the CRS generated in the setup by each party is statistically-close to uniform. We note that CCA secure encryption schemes with public keys statistically-close to uniform exist from the LWE assumption [BCHK07, BGG+14]. To see that the CRS is statistically-close to uniform, note that the CRS consists of the following components:

- Two public keys of a CCA-secure encryption scheme.
- A uniformly random string r.
- A hash key K for a correlation-intractable hash function family $\mathcal{H}$, which is known from LWE with hash keys statistically-close to uniform [PS19].

7.4 Semi-Honest Corruptions

We now observe another interesting property of our multi-string NIZK argument that is crucial in its application the MPC protocol in the presence of a threshold mixed adversary. In particular, we note that our multi-string simulation-extractable NIZK remains secure not only in the presence of an honest majority but even in the following scenario: the adversary corrupts two sets of parties $(\mathcal{A}_1, \mathcal{A}_2)$ such that $\mathcal{A}_1$ consists of all parties maliciously corrupted with $|\mathcal{A}_1| < \frac{n}{2}$ as before, $\mathcal{A}_2$ consists of parties that are semi-honestly corrupted and in particular, follow the protocol behaviour correctly, and $|\mathcal{A}_1 \cup \mathcal{A}_2| < n$.

Protocol Description. First, we describe a slight modification to the above protocol. Observe that if we ran the multi-string NIZK Setup algorithm, it would output CRSs that are statistically-close to uniform. Instead, in our protocol, we have honest parties instead run a Setup' algorithm, which simply outputs a uniformly random string of the appropriate length. With overwhelming probability, this will correspond to a CRS that could have been output by the "real" setup algorithm, and all the required properties of the multi-string NIZK hold. It is only in the ideal world that we will run the honest setup algorithm Setup as part of the simulated setup on behalf of every honest party since the extractor Ext needs the secret keys sk to extract the adversary's witness.

Proof. Now, in order to prove that our scheme is still secure, we are faced with the following challenge: we now have a dishonest majority of corrupt parties unlike before which could break zero knowledge or soundness. Lets focus on the set of parties in $\mathcal{A}_2$ that were corrupted in a semi-honest manner. For each of these parties, the simulator will set its randomness appropriately to ensure the following two things:

- The public key pk and the randomness r generated as part of the CRS by that party in round 1 are honestly generated.
- Furthermore, the simulator knows the corresponding secret key sk associated with that public key and that r is the output of the pseudorandom generator G for which the simulator knows the pre-image s.

Thus, as long as the number of maliciously corrupt parties $\mathcal{A}$ is of size less than $\frac{n}{2}$, the simulator will be able to both produce fake proofs (via the PRG preimages) and extract the witness from the adversary's proofs (by running the Ext algorithm using knowledge of majority of the secret keys). Additionally, the adversary will also not be able to cheat since it knows only less than half of the simulation trapdoors that were generated by the maliciously corrupt parties. Therefore, our proofs from before would work as is and this scheme still remains secure.

8 Acknowledgements

Saikrishna Badrinarayanan, Aayush Jain, Nathan Manohar, and Amit Sahai were supported in part from DARPA SAFEWARE and SIEVE awards, NTT Research, NSF Frontier Award 1413955, and NSF grant 1619348, 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 Award HR00112020024 and 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, NTT Research, or the U.S. Government. Saikrishna Badrinarayanan was also partially supported by an IBM PhD fellowship. Aayush Jain was also partially supported by a Google PhD fellowship.

References

[ABT19] Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. Degree 2 is complete for the round-complexity of malicious mpc. Cryptology ePrint Archive, Report 2019/200, 2019. https://eprint.iacr.org/2019/200.

[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.

[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 *Financial Cryptography and Data Security*, Berlin, Heidelberg, 2009.

[BCHK07] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.*, 36(5):1301–1328, 2007.

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

[BFH+08] Zuzana Beerliová-Trubíniová, Matthias Fitzi, Martin Hirt, Ueli M. Maurer, and Vassilis Zikas. MPC vs. SFE: perfect security in a unified corruption model. In *TCC*, pages 231–250, 2008.

[BFJ+20] Saikrishna Badrinarayanan, Rex Fernando, Aayush Jain, Dakshita Khurana, and Amit Sahai. Statistical ZAP arguments. *EUROCRYPT*, 2020.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. 2014.

[BGG+18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In *CRYPTO*, 2018.

[BGGK17] Dan Boneh, Rosario Gennaro, Steven Goldfeder, and Sam Kim. A lattice-based universal thresholdizer for cryptographic systems. *IACR Cryptol. ePrint Arch.*, 2017:251, 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 *ASIACRYPT*, pages 275–303, 2017.

[BGJ+] Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Amit Sahai. Promise zero knowledge and its applications to round optimal MPC. *IACR Cryptology ePrint Archive*, 2017:1088. In CRYPTO 2018.

[BGJ+17] Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Dakshita Khurana, and Amit Sahai. Round optimal concurrent MPC via strong simulation. In *TCC*, pages 743–775, 2017.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325, 2012.

[BHP17] Zvika Brakerski, Shai Halevi, and Antigoni Polychroniadou. Four round secure computation without setup. In *Theory of Cryptography*, 2017.

[BL18] Fabrice Benhamouda and Huijia Lin. k-round mpc from k-round ot via garbled interactive circuits. *EUROCRYPT*, 2018.

[Blu86] Manuel Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians*, volume 1, page 2. Citeseer, 1986.

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC*, 1988.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106, 2011.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *CRYPTO*, pages 505–524, 2011.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-shamir: From practice to theory. 2019.

[CCLS20] T.-H. Hubert Chan, Kai-Min Chung, Wei-Kai Lin, and Elaine Shi. MPC for MPC: secure computation on a massively parallel computing architecture. In *ITCS*, 2020.

[CCRR18] Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D. Rothblum. Fiat-shamir and correlation intractability from strong kdm-secure encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 91–122. Springer International Publishing, 2018.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, July 2004.

[CKKC13] Seung Geol Choi, Jonathan Katz, Ranjit Kumaresan, and Carlos Cid. Multi-client non-interactive verifiable computation. In *Theory of Cryptography*, 2013.

[CL14] Ran Cohen and Yehuda Lindell. Fairness versus guaranteed output delivery in secure multiparty computation. In *ASIACRYPT*, 2014.

[Cle86] R Cleve. Limits on the security of coin flips when half the processors are faulty. In *STOC*, New York, NY, USA, 1986.

[CM15] Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled fhe from learning with errors. In *CRYPTO*, 2015.

[COSV17a] Michele Ciampi, Rafail Ostrovsky, Siniscalchi, and Ivan Visconti. Delayed-input non-malleable zero knowledge and multi-party coin tossing in four rounds. In *TCC*, 2017.

[COSV17b] Michele Ciampi, Rafail Ostrovsky, Siniscalchi, and Ivan Visconti. Round-optimal secure two-party computation from trapdoor permutations. In *TCC*, 2017.

[CPA+08] Ashish Choudhary, Arpita Patra, B. V. Ashwinkumar, K. Srinathan, and C. Pandu Rangan. Perfectly reliable and secure communication tolerating static and mobile mixed adversary. In *ICITS*, pages 137–155, 2008.

[CSW19] Ran Cohen, Abhi Shelat, and Daniel Wichs. Adaptively secure MPC with sublinear communication complexity. In *CRYPTO*, 2019.

[DI05] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *CRYPTO*, Berlin, Heidelberg, 2005.

[DN07] Cynthia Dwork and Moni Naor. Zaps and their applications. *SIAM J. Comput.*, 36(6):1513–1543, 2007.

[FHM98] Matthias Fitzi, Martin Hirt, and Ueli M. Maurer. Trading correctness for privacy in unconditional multi-party computation (extended abstract). In *CRYPTO*, pages 121–136, 1998.

[FHM99] Matthias Fitzi, Martin Hirt, and Ueli M. Maurer. General adversaries in unconditional multi-party computation. In *ASIACRYPT*, pages 232–246, 1999.

[FKN94] Uri Feige, Joe Killian, and Moni Naor. A minimal model for secure computation (extended abstract). In *STOC*, New York, NY, USA, 1994.

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM J. Comput.*, 1999.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, 1987.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[GIKR02] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. On 2-round secure multiparty computation. In *CRYPTO*, 2002.

[GKP17] Sanjam Garg, Susumu Kiyoshima, and Omkant Pandey. On the exact round complexity of self-composable two-party computation. In *EUROCRYPT*, pages 194–224, 2017.

[GLS15] S. Dov Gordon, Feng-Hao Liu, and Elaine Shi. Constant-round MPC with fairness and guarantee of output delivery. In *CRYPTO*, 2015.

[GLS19] Vipul Goyal, Yanyi Liu, and Yifan Song. Communication-efficient unconditional mpc with guaranteed output delivery. In *CRYPTO*, 2019.

[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 *CRYPTO*, 2007.

[Goy11] Vipul Goyal. Constant round non-malleable protocols using one way functions. In *STOC*, 2011.

[GPS19] Yue Guo, Rafael Pass, and Elaine Shi. Synchronous, with a chance of partition tolerance. In *CRYPTO*, 2019.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, pages 75–92, 2013.

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

[HIK+19] Shai Halevi, Yuval Ishai, Eyal Kushilevitz, Nikolaos Makriyannis, and Tal Rabin. On fully secure mpc with solitary output. 2019.

[Hir01] Martin Hirt. *Multi party computation: efficient protocols, general adversaries, and voting.* PhD thesis, ETH Zurich, Zürich, Switzerland, 2001.

[HL18] J. Holmgren and A. Lombardi. Cryptographic hashing from strong one-way functions (or: One-way product functions and their applications). In *FOCS*, 2018.

[HMZ08] Martin Hirt, Ueli M. Maurer, and Vassilis Zikas. MPC vs. SFE : Unconditional and computational security. In *ASIACRYPT*, pages 1–18, 2008.

[HT13] Martin Hirt and Daniel Tschudi. Efficient general-adversary multi-party computation. In *ASIACRYPT*, pages 181–200, 2013.

[IK97] Yuval Ishai and Eyal Kushilevitz. Private simultaneous messages protocols with applications. In *ISTCS*, Washington, DC, USA, 1997.

[IKLP06] Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. On combining privacy with guaranteed output delivery in secure multiparty computation. In *CRYPTO*, pages 483–500, 2006.

[IKP10] Yuval Ishai, Eyal Kushilevitz, and Anat Paskin. Secure multiparty computation with minimal interaction. In *CRYPTO*, Berlin, Heidelberg, 2010.

[JJ19] Abhishek Jain and Zhengzhong Jin. Statistical zap arguments from quasi-polynomial LWE. *IACR Cryptology ePrint Archive*, 2019:839, 2019.

[JKKR17] Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Ron Rothblum. Distinguisher-dependent simulation in two rounds and its applications. In *CRYPTO*, pages 158–189, 2017.

[JRS17] Aayush Jain, Peter M. R. Rasmussen, and Amit Sahai. Threshold fully homomorphic encryption. ePrint, 2017. https://eprint.iacr.org/2017/257.

[KMR11] Seny Kamara, Payman Mohassel, and Mariana Raykova. Outsourcing multi-party computation. ePrint, 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.

[KRR17] Yael Tauman Kalai, Guy N. Rothblum, and Ron D. Rothblum. From obfuscation to the security of fiat-shamir for proofs. In *CRYPTO*, 2017.

[LO14] Joshua Lampkins and Rafail Ostrovsky. Communication-efficient MPC for general adversary structures. In *SCN*, pages 155–174, 2014.

[LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *STOC*, pages 1219–1234, 2012.

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

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

[NPS99] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *EC*, 1999.

[Pas04] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *STOC*, 2004.

[PR19] Arpita Patra and Divya Ravi. Beyond honest majority: The round complexity of fair and robust multi-party computation. 2019.

[PS16] Chris Peikert and Sina Shiehian. Multi-key FHE from lwe, revisited. In *TCC Part II*, 2016.

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In *CRYPTO*, 2019.

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

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[SPCR09] Kannan Srinathan, Arpita Patra, Ashish Choudhary, and C. Pandu Rangan. Unconditionally secure message transmission in arbitrary directed synchronous networks tolerating generalized mixed adversary. In *ASIACCS*, pages 171–182, 2009.

[Wee10] Hoeteck Wee. Black-box, round-efficient secure computation via non-malleability amplification. In *FOCS*, pages 531–540, 2010.

[Yao82] Andrew C. Yao. Protocols for secure computations. In *SFCS*, 1982.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.

[ZHM09] Vassilis Zikas, Sarah Hauser, and Ueli M. Maurer. Realistic failures in secure multiparty computation. In *TCC*, pages 274–293, 2009.

[Zik10] Vassilis Zikas. *Generalized corruption models in secure multi-party computation.* PhD thesis, ETH Zurich, 2010.

A Deferred Preliminaries

A.1 Multi-Key FHE

We recall the definition of multi-key FHE in the plain model with distributed setup as found in [BHP17].

Definition 3 (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* $\mathsf{params} = \mathsf{params}_1 || \dots || \mathsf{params}_N$.

$(pk, sk) \leftarrow \mathsf{KeyGen}(\mathsf{params})$**:** *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.*

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

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

$\widehat{\mu} \leftarrow \mathsf{FinDec}(p_1, \dots, p_\ell)$**:** *It takes as input partial decryptions $p_1, \dots, p_\ell$ 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 $m_1, \dots, m_\ell \in \{0,1\}^\lambda$ for $\ell \le N$, any sequence $I_1, \dots, I_\ell \in [N]$ 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* $1 \le i \le \ell$, $\hat{ct} = \mathsf{Eval}(C, ct_1, \dots, ct_\ell)$, *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,

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

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* $\mathsf{params}_1 || \dots || \mathsf{params}_N$. *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* $\mathsf{params}_1 || \dots || \mathsf{params}_N$*. 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)*,* $m_1, \dots, m_\ell \in \{0,1\}^\lambda$*,* $I_1, \dots, I_\ell \in [N]$*, and a set of circuits* $\{C_k : (\{0,1\}^\lambda)^\ell \to \{0,1\}\}_{k\in[t]}$ *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* $\hat{ct}_k \leftarrow \mathsf{Eval}(C_k, ct_1, \dots, ct_\ell)$ *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* $\mathsf{params}_1 || \dots || \mathsf{params}_N$*. 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)*,* $m_1, \dots, m_\ell \in \{0,1\}^\lambda$*,* $I_1, \dots, I_\ell \in [N]$*, and a set of circuits* $\{C_k : (\{0,1\}^\lambda)^\ell \to \{0,1\}\}_{k\in[t]}$ *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* $\hat{ct}_k \leftarrow \mathsf{Eval}(C_k, ct_1, \dots, ct_\ell)$ *for all* $k \in [t]$*.*
7. *Define* $\mu_k = C_k(m_1, \dots, m_\ell)$*. For all* $k \in [t]$*, the adversary is given* $p_{i,k} \leftarrow \mathsf{Sim}(\mu_k, \hat{ct}, i, \{sk_j\}_{j\in[N]\setminus\{i\}})$*.*
8. $\mathcal{A}$ *outputs* out*. The output of the experiment is* out*.*

A.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 4 (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 $\Delta(X_s^0, X_s^1) \leq \delta$. 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 19 (Smudging Lemma [MW16]). *Let $B_1, B_2 \in \mathbb{N}$. For any $e_1 \in [-B_1, B_1]$ let E_1 and E_2 be independent random variables uniformly distributed on $[-B_2, B_2]$ and define the two stochastic variables $X_1 = E_1 + e_1$ and $X_2 = E_2$. Then $\Delta(E_1, E_2) < B_1/B_2$.*

A.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].

A.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 5 (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 6 (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. 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 7 (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 8 (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 9 (t-out-of-n secret sharing)**.** *Let $P = \{P_1, \ldots, P_N\}$ 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 10 (Linear Secret Sharing Scheme (LSSS))**.** *Let $P = \{P_1, \ldots, P_N\}$ 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 $r_2, \ldots, r_n \in \mathbb{Z}_p$ are chosen at random and the vector $\vec{v} = M \cdot (s, r_2, \ldots, r_n)^T$ 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*
$$(1, 0, \ldots, 0) \in \mathsf{span}(\{M[i]\}_{i \in \bigcup_{j \in S} T_j})$$
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
$$\sum_{i \in \bigcup_{j \in S} T_j} c_i M[i] = (1, 0, \ldots, 0).$$
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 $\vec{s}_1, \ldots, \vec{s}_\ell \in \mathbb{Z}_p^n$. 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.

A.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 11 ($\{0,1\}$-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 12 (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 13 (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 20 ([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 $\mathbb{A} \in \{0,1\}\text{-}\mathsf{LSSS}$, we can find recovery coefficients efficiently. However, it is worth noting that this does not mean that deciding whether an access structure belongs to $\{0,1\}\text{-}\mathsf{LSSS}$ is feasible. In our applications we will instead specifically construct secret sharing schemes that belong to $\{0,1\}\text{-}\mathsf{LSSS}$.

Lemma 21 ([JRS17]). *Finding recovery coefficients $c_i \in \{0,1\}$ in a linear secret sharing scheme Π for an access structure $\mathbb{A} \in \{0,1\}\text{-}\mathsf{LSSS}$ 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 14 (Derived $\{0,1\}\text{-}\mathsf{LSSS}$ ($\{0,1\}\text{-}\mathsf{LSSSD}$)). *Let $P = \{P_1, \ldots, P_N\}$ be a set of parties. We denote by $\{0,1\}\text{-}\mathsf{LSSSD}_N$ 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 $\mathbb{B} \in \{0,1\}\text{-}\mathsf{LSSS}_{\ell n}$ 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 $\{0,1\}\text{-}\mathsf{LSSSD}$ scheme is a secret sharing scheme where the shares satisfy a $\{0,1\}\text{-}\mathsf{LSSS}$ scheme, but each party receives multiple shares.

Theorem 9 ([JRS17]). *The class of access structures $\{0,1\}\text{-}\mathsf{LSSSD}_N$ 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 $\{0,1\}\text{-}\mathsf{LSSSD}$.

A.4 Multi-String NIZK

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

Definition 15 (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.

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

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* SimGen, 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 16 (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* ExtGen, 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*.*

A.5 Correlation Intractable Hash Functions

We adapt definitions of a correlation intractable hash function family from [PS19, CCH+19].

Definition 17. *We say that a relation* $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$ *is searchable in size* S *if there exists a function* $f : \mathcal{X} \to \mathcal{Y}$ *that is implementable in a boolean circuit of size* S*, such that if* $(x, y) \in \mathcal{R}$ *then* $y = f(x)$*.*

Having defined efficiently searchable relation, we define correlation intractability for a class of relations $\mathcal{R}$.

Definition 18. *Let* $\mathcal{R} = \{\mathcal{R}_\lambda\}_{\lambda \in \mathbb{Z}}$ *be a relation family. A hash function family* $\mathcal{H} = (\mathsf{Setup}, \mathsf{Eval})$ *is correlation intractable (CI) if for every non-uniform polynomial-size adversary* $\mathcal{A}$*, there exists a negligible function such that for every* $R \in \mathcal{R}_\lambda$

$$\Pr_{K \leftarrow \mathcal{H}.\mathsf{Setup}(1^\lambda, R)}[\mathcal{A}(K) = (x, \mathcal{H}.\mathsf{Eval}(K, x)) \in R] \leq \mathsf{negl}$$

We also require additional property which we refer to as statistical indistinguishability of hash keys. This property states that for all large enough λ and $R_1, R_2 \in \mathcal{R}_\lambda$, for any adversary $\mathcal{A}$ (even unbounded),

$$|\Pr_{K \leftarrow \mathcal{H}.\mathsf{Setup}(1^\lambda, R_1)}[\mathcal{A}(K) = 1] - \Pr_{K \leftarrow \mathcal{H}.\mathsf{Setup}(1^\lambda, R_2)}[\mathcal{A}(K) = 1]| \leq 2^{-\lambda^{O(1)}}$$

The work of [PS19] showed how to construct correlation intractable for the family of circuits given by all polynomial sized circuits of depth λ from LWE with subexponential approximation factors.

A.6 Sigma-Protocol

In this section we recall the Σ protocol for Graph Hamiltonicity (which is an NP complete language) by Blum [Blu86] that can be based on commitment schemes and one way functions. The Graph Hamiltonicity language has as instance a graph G, which can be represented as an adjacency matrix in $\{0,1\}^{\binom{\lambda}{2}}$ where λ is the number of nodes. Its witness is a subgraph H which forms a cycle in G. The Σ protocol consists of three messages. For a complete description refer to [Blu86]. The protocol is a parallel repetition of the following basic protocol between a prover P and verifier V.

- Prover send $c_1 = \mathsf{Com}(\pi(G))$ and $c_2 = \mathsf{Com}(\pi)$ where π is a random permutation. Here Com is a perfectly binding bit commitment scheme.
- Verifier chooses $e \leftarrow \{0,1\}$ and sends it over to P.
- If $e = 0$, prover opens up both commitments c_1, c_2 to reveal $\pi(G)$ and π. Otherwise it opens up a cycle in c_1.

Properties:

- This protocol is a honest verifier zero-knowledge protocol with constant soundness error.
- We can consider a parallel repetition of the basic protocol to amplify the soundness guarantee and reduce the error[9]. Such a protocol satisfies statistical soundness with soundness error bounded by 2^{-m} where m is the number of parallel repetitions. Thus, for every instance G not admitting a hamiltonian cycle and first message $\{a_j = (c_{j,1}, c_{j,2})\}_{j\in[m]}$, there exists at most one string $e \in \{0,1\}^m$ for which there exists a third message $\{z_j\}_{j\in[m]}$ such that (a_j, e_j, z_j) verifies with respect to the basic protocol for all $j \in [m]$. Also, string e can be computed by an efficient function if the commitment scheme used to compute the first message has a trapdoor sk[10]. Let this function be called $f_{bad,\lambda,m,sk}$ and it is parameterized by the number of nodes in the graph λ, the number of parallel repetitions m, and the trapdoor sk for the commitment scheme. The size of the circuit representing $f_{bad,\lambda,m,sk}$ is polynomial in $(\lambda, m, |sk|)$.
- Note that the protocol satisfies standard witness indistinguishability against malicious verifiers. In particular, parallel repetition of the constant soundness error protocol also retains witness indistinguishability against malicious verifiers while reducing the soundness error. That is, witness indistinguishability composes under parallel repetition.

B MPC with Threshold Mixed Adversaries: Definition

In this section, we formally define the notion of secure multiparty computation against a threshold mixed adversary as defined in the works of Fitzi et al. [FHM98, FHM99]. Additionally, we consider guaranteed output delivery. Recall that a $(t_{\mathbf{Mal}}, t_{\mathbf{Sh}}, t_{\mathbf{Fc}})$-threshold mixed adversary $\mathcal{A} = (\mathcal{A}_{\mathbf{Mal}}, \mathcal{A}_{\mathbf{Sh}}, \mathcal{A}_{\mathbf{Fc}})$ is one that corrupts a set of parties $\mathcal{A}_{\mathbf{Mal}}$ maliciously, a set of parties $\mathcal{A}_{\mathbf{Sh}}$ in a semi-honest manner and a set of parties $\mathcal{A}_{\mathbf{Fc}}$ in a fail-corrupt manner. It is required to satisfy the threshold constraints: $|\mathcal{A}_{\mathbf{Mal}}| \le t_{\mathbf{Mal}}$, $|\mathcal{A}_{\mathbf{Sh}}| \le t_{\mathbf{Sh}}$, $|\mathcal{A}_{\mathbf{Fc}}| \le t_{\mathbf{Fc}}$. While the former two notions are quite standard, we recall that in a fail-corrupt corruption, the adversary can instruct the corrupted party to stop its protocol execution at any point. Note that in the case of fail-corrupt corruption, the adversary does not get to learn the internal state of the corrupted parties at any point. For simplicity, we will omit the threshold constraints on the adversary for the rest of this section and assume they are implicit. We now present the formal definition of an MPC protocol secure against a $(t_{\mathbf{Mal}}, t_{\mathbf{Sh}}, t_{\mathbf{Fc}})$-threshold mixed adversary $\mathcal{A} = (\mathcal{A}_{\mathbf{Mal}}, \mathcal{A}_{\mathbf{Sh}}, \mathcal{A}_{\mathbf{Fc}})$ with static corruption and guaranteed output delivery.

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

[9] Note that such an amplification would not be possible against malicious verifiers as the zero knowledge property doesn't compose in that case.

[10] Recall that given the commitment a and the trapdoor sk, the decommitment can be efficiently generated.

B.1 Defining Security

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

Formally, the security of a multi-party computation protocol with guaranteed output delivery against a threshold mixed adversary with respect to a functionality f is defined by comparing the real-world execution of the protocol with an ideal-world evaluation of f by a trusted party. More concretely, it is required that for every adversary $\mathsf{Adv} = (\mathcal{A}_{\mathbf{Mal}}, \mathcal{A}_{\mathbf{Sh}}, \mathcal{A}_{\mathbf{Fc}})$, 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. Let's denote $\vec{x} = (x_1, \ldots, x_n)$.

The real execution. In the real world execution of the n-party protocol Π for computing f, Π is executed in the presence of an adversary Adv. The honest parties follow the instructions of Π. Initially, the Adv is given as input the security parameter λ and some auxiliary information z. Then, Adv outputs a tuple of sets $\mathcal{A}_{\mathbf{Mal}}, \mathcal{A}_{\mathbf{Sh}}, \mathcal{A}_{\mathbf{Fc}} \subseteq [N]$ of parties to corrupt and gets as input the inputs of all the parties in the sets $\mathcal{A}_{\mathbf{Mal}}$ and $\mathcal{A}_{\mathbf{Sh}}$. Adv sends all messages in place of corrupted parties in the sets $\mathcal{A}_{\mathbf{Mal}}$ and $\mathcal{A}_{\mathbf{Sh}}$. For each party in the set $\mathcal{A}_{\mathbf{Mal}}$, it may follow an arbitrary polynomial-time strategy. For each party in the set $\mathcal{A}_{\mathbf{Sh}}$, the adversary is required to execute the protocol honestly. For each party in the set $\mathcal{A}_{\mathbf{Fc}}$, the adversary can choose to instruct that party to abort the execution at any point in the protocol. Once again, note that the adversary does not learn the internal state of any fail-corrupt party.

The interaction of Adv in 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 subject to the restriction on the semi-honest parties' behaviour) 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, an ideal world adversary Sim interacts with a trusted party. The ideal execution proceeds as follows.

- **Adversary picks corrupted sets:** Sim is given the security parameter λ and an auxiliary input z and outputs a tuple of sets $\mathcal{A}_{\mathbf{Mal}}, \mathcal{A}_{\mathbf{Sh}}, \mathcal{A}_{\mathbf{Fc}} \subseteq [N]$ of parties to corrupt.
- **Parties send inputs to the trusted party:** The parties send their inputs to the trusted party, and we let x'_i denote the value sent by P_i. Note that for each party P_i in $\mathcal{A}_{\mathbf{Sh}}$, the adversary is required to send its actual input x_i. For each party P_k in $\mathcal{A}_{\mathbf{Fc}}$, the adversary can decide whether P_k should send its input or not but the adversary can't change the input. For each party in $\mathcal{A}_{\mathbf{Mal}}$, the adversary is free to interact as it wishes.

- **Trusted party sends output:** For every party P_i whose input it did not receive, the trusted party sets y_i to 0^λ. For other parties that did send their inputs, the trusted party sets $y_i = x'_i$. The trusted party outputs $f(y_1, \ldots, y_N)$ to Sim and every honest party.
- **Outputs:** Sim outputs an arbitrary function of its view, and the honest parties output the value obtained from the trusted party.

The interaction of Sim with the trusted party defines a random variable $\mathsf{IDEAL}_{f,\mathsf{Sim}(z)}(\lambda, \vec{x})$.

Definition 19. *Let λ be the security parameter. Let f be an N-party functionality and Π be an N-party protocol for computing f.*

- *We say that Π securely computes f with guaranteed output delivery in the presence of threshold mixed adversaries if for every PPT threshold mixed adversary Adv , there exists a PPT simulator Sim such that 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 $z \in \{0,1\}^*$.

B.2 Security against Semi-Malicious Mixed Adversaries

Semi-Malicious Adversary. We take the definition of a semi-malicious adversary 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.

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

Definition 20. *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.*

- *We say that* Π *securely computes* f *with guaranteed output delivery in the presence of semi-malicious mixed adversaries if for every PPT semi-malicious mixed adversary* Adv *, there exists a PPT simulator* Sim *such that 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* $z \in \{0,1\}^*$.

C Multi-Key FHE Construction in [BHP17]

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

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

Recall that the LWE problem is parametrized by integers n, m, q (with $m > n \log q$) and a distribution 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 $w = m \log q$.[11] 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 $t_i B_{i,j} = b_{i,i} - b_{i,j} \pmod q$.

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

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.

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

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

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.