Strongly Secure Universal Thresholdizer

Ehsan Ebrahimi[1] and Anshu Yadav[2]

[1] University of Luxembourg, Esch-sur-Alzette, Luxembourg
ehsan.ebrahimi@uni.lu
[2] Institute of Science and Technology Austria, Klosterneuburg, Austria
anshu.yadav06@gmail.com

Abstract

A *universal thresholdizer* (UT), constructed from a threshold fully homomorphic encryption by Boneh *et. al* , Crypto 2018, is a general framework for universally thresholdizing many cryptographic schemes. However, their framework is insufficient to construct strongly secure threshold schemes, such as threshold signatures and threshold public-key encryption, etc.

In this paper, we strengthen the security definition for a *universal thresholdizer* and propose a scheme which satisfies our stronger security notion. Our UT scheme is an improvement of Boneh *et. al* 's construction at the level of threshold fully homomorphic encryption using a key homomorphic pseudorandom function. We apply our strongly secure UT scheme to construct strongly secure threshold signatures and threshold public-key encryption.

Keywords: Universal Thresholdizer · Threshold Signature · Threshold FHE · Threshold PKE

Table of Contents

1 Introduction

Threshold cryptography [DF89] refers to distributing a privileged operation that requires a secret key, like signing in a signature scheme and decryption in an encryption key, among n parties so that any t of them can collaborate to perform the final computation. Distributing shares of the secret key between multiple parties make the system fault-tolerant. The reasons are: 1) A corrupted number of parties below the threshold value are not able to evaluate the cryptographic primitive. 2) The system would perform correctly, even in the presence of few corrupt parties, if honest parties beyond the threshold value participate in the evaluation. Due to the importance of protecting the secret key, almost all the cryptographic primitives have their corresponding threshold system. Examples are threshold public-key encryption schemes [CG99], threshold signature schemes [Sho00], threshold pseudorandom functions [NPR99], threshold symmetric encryption [AMMR18], threshold message authentication codes [MPS+02], etc.

In a non-interactive threshold scheme, each party computes a partial output completely independently of others and then any t partial outputs can be publicly combined to get the final output. A natural notion of security then says that no polynomial time adversary must be able to compute the final output even if it is given up to $t-1$ partial outputs. Most of the papers on threshold cryptography model this by allowing the adversary to get the partial secret keys of $t-1$ parties. Ideally, the adversary should be allowed to choose the $t-1$ parties adaptively. However, achieving adaptive security is hard, and hence, generally the security definitions restrict the adversary to output the $t-1$ parties in the beginning. In addition, they also disallow the adversary to ask the partial outputs on challenge inputs (for e.g., partial signature on challenge message in threshold signature and partial decryption of challenge ciphertext in CCA security of threshold

PKE scheme) [GWW+13,dPKM+24,Bol02,BGG+18]. This notion of security was strengthened by Bellare, *et. al* [BCK+22] in the context of threshold signatures, where the adversary is now allowed to output a forgery on one of the messages for which it has already asked a partial signature. In more detail, let c be the number of parties whose keys are given to the adversary. Then the adversary is allowed to output a forgery on any message for which it has received at most $t-1-c$ partial signatures[3]. They further showed that some of the well-known signature schemes, BLS [BLS04,Bol03] and FROST [KG20] can, in fact, be proven secure in their stronger security definition. BLS and FROST are based on classical assumptions based on discrete logarithm problem and is thus not secure against a quantum adversary.

Motivated by Bellare *et. al* 's work, we started with the question - is the only lattice based non-interactive threshold signature scheme of Boneh *et. al* [BGG+18] (referred to as BGGJKRS from mow on) secure in the stronger definition given by Bellare *et. al* ? We found that it is hard to prove the stronger security for BGGJKRS in its current form. We discuss the issues in the technical overview. We observed that the stronger definition can in fact be viewed as an adaptive version of the current definition, which explains the difficulty in proving it. We provide a detailed discussion on adaptive nature of the Bellare *et. al* 's stronger security notion in technical overview.

We then improved BGGJKRS construction using key homomorphic pseudorandom functions (KHPRF), which are standard PRF, with additional property that for all K_1, K_2, for all inputs x, $\mathsf{PRF}(K_1, x) + \mathsf{PRF}(K_2, x) = \mathsf{PRF}(K_1 + K_2, x)$. We then show that our improved construction satisfies the stronger security notion of Bellare *et. al* .

Boneh *et. al* constructed the threshold signature from a tool called universal thresholdizer (UT), which they defined and built from lattice based assumptions. The universal thresholdizer can be used as a compiler to thresholdize any cryptographic primitive and is itself built from threshold FHE, which again, the authors define and construct from any "special" FHE in the same paper. We realized that our improvement in threshold signature scheme can actually be implemented at the level of TFHE so that it improves its security, which in turn, improves the security of universal thresholdizer built from TFHE in a way that helps in achieving the stronger security for any threshold cryptographic primitive that uses UT as the thresholdizer. In the main body of this paper, we define the desired stronger notion of security for TFHE and UT and provide a construction for TFHE that achieves this notion. We then show that the universal thresholdizer built from our TFHE achieves the desired security. We finally show applications of stronger universal thresholdizer to construct threshold signatures and CCA secure PKE with the stronger security. However, in the technical overview, we describe the challenges and our ideas using the specific case of threshold signatures, because some of the ideas, especially the adaptivity perspective of the stronger security definition is more clear when viewed at the applications level, instead of TFHE or UT.

Our contributions:

- We strengthen the security definition of universal thresholdizer (UT) in [BGG+18], which is needed to prove the stronger security of threshold cryptoprimitives built using the UT. In turn, we define stronger security property for threshold FHE (TFHE) needed to prove the stronger security of UT.
- We improve the construction of TFHE in [BGG+18] to achieve the stronger security property, we define.
- Using our TFHE, we get the first lattice based construction of non-interactive threshold signature with the stronger security as defined in [BCK+22,BTZ22].
- Along the lines of [BCK+22], we define a stronger security notion for CCA-PKE, and show that the constructions in [BGG+18] satisfy this security if the universal thresholdizer UT satisfies stronger security.
- In [ASY22], Agrawal *et. al* constructed partially adaptive threshold signatures in random oracle model, that allows the adversary to issue key queries (all at once) in the middle of the game. We combine our technique with Agrawal *et. al* 's to construct TFHE (and its applications) with stronger security while also allowing the adversary to issue key queries (all at once) in the middle of the game.

[3] Bellare *et. al* defined different levels of security improvements. However, in this work, we focus only on their TS-UF-1 definition.

1.1 Stronger Security from Adaptivity Perspective

A natural notion of security for t-out-of-n threshold signatures says that any PPT adversary $\mathcal{A}$ should not be able to generate a signature on any message m^* even if it gets (i) partial signatures on m^* from up to any $t-1$ parties and (ii) complete (or any number of partial) signatures on messages $m \neq m^*$. Item (ii) is easy to provide and considered by all versions of unforgeability definition in the literature. For item (i), $\mathcal{A}$ can obtain partial signatures on m^* in two ways - (i) get the partial signing keys from the $t-1$ parties (ii) only get the partial signatures (but not the keys) on m^* from these parties. Clearly, the adversary gets more power in the first case. In an ideal situation, the adversary can adaptively decide which $t-1$ parties to corrupt. However, this natural notion of security (mostly referred to as adaptive unforgeability) is hard to achieve and hence, most of the constructions in this area consider a selective notion of unforgeability where the adversary must decide the set of corrupted parties in the beginning. There is also a notion of *partial adaptivity*, where the adversary can choose the set of corrupted parties in the middle of the unforgeability game (i.e. after seeing some (partial) signatures on other messages), but the entire set of corrupted parties must be declared together [ASY22]. Since once the adversary has decided the set of corrupted parties, it is always beneficial for it to ask the signing keys rather than just the partial signatures, these definitions assume that the adversary gets the partial signing keys from $t-1$ parties, and is not allowed to issue any partial signing query on the challenge message m^*. However, we can also consider partial adaptivity in an orthogonal direction, where the adversary divides the set of corrupted parties as *fully corrupted*, for which it asks the signing keys and *semi-corrupted* from which it only asks the partial signature on m^*. The adversary must output the set of fully corrupted parties selectively, i.e., in the beginning of the game, but can decide the semi-corrupted parties adaptively throughout the game. This is the notion considered in [BCK+22,BTZ22] under stronger security definition which they call as $\mathsf{TS\text{-}UF\text{-}1}$. Indeed we can define security properties that combine adaptivity in both the directions. In Fig. 1, we present different security properties that can be thus obtained, and relation between them.

Between P_2 and P_1^*, we could not show either of them implied by the other. However, this is not surprising since both P_1^* and P_2 improve P_1 in different directions.

What is interesting is that even P_3 does not seem to imply P_1^* except with a loss of factor Q, where Q is the number of signing queries. This can be understood as follows. Let TS be any threshold signature scheme and $\mathcal{A}_{\mathsf{P}_1^*}$ be an adversary against P_1^* security of TS. We want to show that if $\mathcal{A}_{\mathsf{P}_1^*}$ exists then there exists an adversary $\mathcal{A}_{\mathsf{P}_3}$ against P_3 security of TS. The difficulty in the reduction comes when $\mathcal{A}_{\mathsf{P}_1^*}$ issues a signing query (m, i) such that $m = m^*$. In this situation, $\mathcal{A}_{\mathsf{P}_3}$ must not ask the partial signature from P_3 challenger. Instead it should ask for fsk_i (the partial signing key of party i) and compute $\mathsf{PartSign}(i, \mathsf{fsk}_i, m)$ itself. But the problem is that $\mathcal{A}_{\mathsf{P}_3}$ does not know when $m = m^*$. So, $\mathcal{A}_{\mathsf{P}_3}$ guesses m^* as follows: let Q_s be the number of different messages queried for partial signatures. Then $Q_s \leq Q$, where Q is the total number of signing queries. Guess an index $k \in [Q_s + 1]$ and assume the k-th unique message as m^*. Then, for each signing query (m, i) with $m \neq m^*$ from $\mathcal{A}_{\mathsf{P}_1^*}$, the reduction forwards the query to P_3 challenger. For $m = m^*$, the reduction queries for fsk_i and computes the partial signature itself. Thus, P_3 implies P_1^*, but with a loss of factor $Q_s \leq Q$.

We summarize all the discussed notions in Figure 1. An arrow in the picture indicates a path to strengthen a notion, i.e., it is not an implication.

1.2 Technical Overview

Our construction uses the $\mathsf{BGGJKRS}$ construction, which does not satisfy the stronger security notion, as the base and builds upon it to get the stronger security. We first recall the $\mathsf{BGGJKRS}$ construction for threshold signature.

Recap of BGGJKRS threshold signature. The $\mathsf{BGGJKRS}$ threshold signature scheme is a round optimal (non-interactive) scheme built from a "universal thresholdizer" that can thresholdize a number of cryptographic primitives. The thresholdizer is built from a thresholdized version of any "special" fully homomorphic encryption (FHE) where the decryption of any ciphertext ct, using a key fsk, is a linear operation as $\langle \mathsf{ct}, \mathsf{fsk} \rangle$,

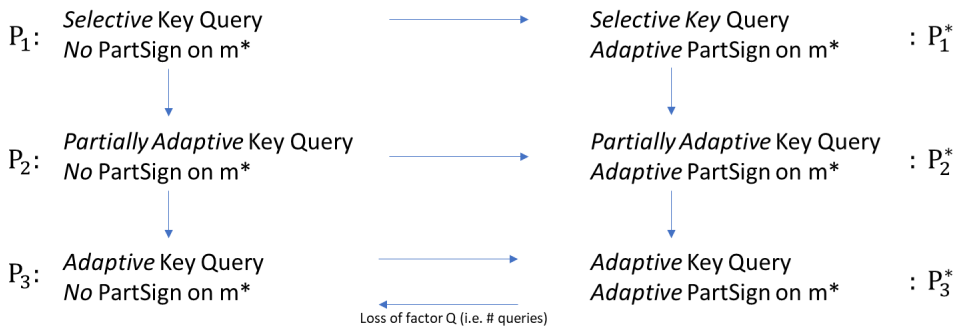

Fig. 1. Threshold Signature security with different level of adaptivity. For any two properties, P_i, P_j, $\mathsf{P}_i \rightarrow \mathsf{P}_j$ means that P_j is stronger than (at least as strong as) P_i.

which gives $\lfloor q/2 \rceil m + e$, followed with rounding which gives m. Here, q is the working modulus and e is a "small" error and depends on the LWE assumption on which FHE is built. In the thresholdized version the decryption key fsk is t-out-of-n secret shared between n parties. To decrypt a ciphertext ct, each party can compute a partial decryption of ct using its own key share fsk_i and then any t partial decryptions can be (publicly) combined to get a complete decryption. The BGGJKRS construction for threshold signature $\mathsf{TS} = (\mathsf{TS.Setup}, \mathsf{TS.PartSign}, \mathsf{TS.Combine})$ uses (i) a linear secret sharing scheme SS, where the individual shares are combined through a linear process (SS.Combine) to get the full secret, (ii) a fully homomorphic encryption scheme FHE, where the decryption algorithm is a linear operation[4], and (iii) a signature scheme Sig. Then the construction is:

- $\mathsf{TS.Setup}()$: generates FHE keys $(\mathsf{fpk}, \mathsf{fsk})$, signing keys $(\mathsf{svk}, \mathsf{ssk})$, and encrypts ssk as $\mathsf{ct} = \mathsf{FHE.Enc}(\mathsf{ssk})$. It then uses SS to secret share fsk among the n parties as $\{\mathsf{fsk}_i\}_{i\in[n]}$ for t-out-of-n threshold access structure. Finally, it sets $\mathsf{pp} = (\mathsf{fpk}, \mathsf{svk}, \mathsf{ct})$, $\{\mathsf{sk}_i = \mathsf{fsk}_i\}_{i\in[n]}$, $\mathsf{vk} = \mathsf{svk}$.
- $\mathsf{TS.PartSign}(i, \mathsf{sk}_i, m)$: firstly computes encryption of signature, σ_m, on m using $\mathsf{FHE.Eval}$ as $\mathsf{ct}_{\sigma_m} = \mathsf{FHE.Eval}(\mathsf{Sig.Sign}_m, \mathsf{ct})$ and returns partial decryption of ct_{σ_m} as $\sigma_{m,i} = \langle \mathsf{ct}_{\sigma_m}, \mathsf{fsk}_i \rangle + \mathsf{noise}$. Here, $\mathsf{Sig.Sign}_m$ is the signing circuit of Sig with message m being hardwired.
- $\mathsf{TS.Combine}(S, \{\sigma_{m,i}\}_{i\in S})$: For $|S| \geq t$, the TS combine algorithm first runs the SS.Combine algorithm on the partial signatures and then performs rounding to gets the signature as $\sigma_m = round(\mathsf{SS.Combine}(\{\sigma_{m,i}\}_{i\in S}))$.

Security sketch for BGGJKRS construction. We first briefly recall the security game. In the beginning of the game, the adversary is given the partial signing keys from up to $t-1$ parties of its choice. In addition, the adversary can adaptively issue partial signing queries of the form (m, i) to receive partial signature $\sigma_{m,i}$ on m from party i. In the end, to win the game, the adversary must output a forgery (m^*, σ^*) on a message m^* for which no partial signature was requested.

For the ease of presentation, let us consider a simple case of 2-out-of-2 access structure. Thus, the FHE decryption key fsk is secret shared between the two parties as : pick a random fsk_1 and set $\mathsf{fsk}_2 = \mathsf{fsk} - \mathsf{fsk}_1$. Wlog, we assume that the adversary asks the signing key for the first party[5].

The security is through a sequence of hybrids, where as usual, the initial hybrid (H_0) is the real game. Then in the next hybrid (H_1), for any queried message m, the challenger computes the partial signatures corresponding to the honest party P_2 without using its signing key fsk_2. Instead, $\sigma_{m,2}$ is computed as $\lfloor q/2 \rceil \sigma_m - \langle \mathsf{ct}_{\sigma_m}, \mathsf{fsk}_1 \rangle + \mathsf{noise}$, while $\sigma_{m,1}$ is computed honestly as $\langle \mathsf{ct}_{\sigma_m}, \mathsf{fsk}_1 \rangle + \mathsf{noise}$[6]. This does not change

[4] For ciphertext $\mathsf{ct} \in \mathbb{Z}_q^\lambda$ and decryption key $\mathsf{fsk} \in \mathbb{Z}_q^\lambda$, $\mathsf{FHE.Decrypt}(\mathsf{fsk}, \mathsf{ct})$ first computes $\langle \mathsf{ct}, \mathsf{fsk} \rangle$, which gives $\lfloor q/2 \rceil m + e$, followed by rounding which outputs m.

[5] Since it is in the best interest of the adversary to get keys from the maximum possible number of parties, the security definition assumes that the adversary gets keys from $t-1$ parties.

[6] noise is added to hide the FHE error e, but is not the main focus of current discussion.

the adversary's view because of linearity of $\mathsf{FHE.Decrypt}$. The purpose behind this step is to use the shares of FHE key fsk only from an "invalid" set of parties (i.e. a set having at most $t-1$ parties) so that, then, in the next hybrid (H_2), the setup algorithm can use zero vector instead of fsk to generate the partial signing keys $\mathsf{fsk}_1, \mathsf{fsk}_2$, using the security of the secret sharing scheme. At this stage, fsk is not used at all, and hence in the next hybrid (H_3), using FHE security, ct in the setup is changed from $\mathsf{ct} = \mathsf{FHE.Encrypt}(\mathsf{ssk})$ to $\mathsf{ct} = \mathsf{FHE.Encrypt}(0)$. Finally, at this stage ssk is not used and we can use the security of underlying signature scheme to argue that the adversary cannot output a forgery in this hybrid, which implies that the adversary cannot generate a forgery in the real world as well.

The reduction to Sig security goes as follows. After getting svk from the Sig challenger the reduction algorithm samples FHE keys $(\mathsf{fpk}, \mathsf{fsk})$ and discards fsk. It secret shares 0 into fsk_1 and fsk_2, encrypts 0 as $\mathsf{ct} = \mathsf{FHE.Encrypt}(0)$ and sends $\mathsf{pp} = (\mathsf{fpk}, \mathsf{ct})$ and fsk_1 as signing key for party 1. Whenever the adversary issue signing query on any message m from party 2, the reduction algorithm computes ct_{σ_m} and sends a signing query on m to Sig challenger and gets σ_m. It then computes $\sigma_{2,m} = \sigma_m - \langle \mathsf{ct}_{\sigma_m}, \mathsf{fsk}_1 \rangle + \mathsf{noise}$.

In the end when the adversary outputs a forgery (m^*, σ_{m^*}), the reduction forwards it to the Sig challenger. Since the adversary is not allowed to query partial signatures on m^*, the reduction never asks signature on m^* from the Sig challenger and hence, (m^*, σ_{m^*}) is a valid forgery against the Sig challenger.

Problem in proving stronger security. In the stronger security, the adversary is allowed to query partial signature on m^* from up to g parties, where $g = t - 1 - c$ and c is the number of parties for which the signing key is obtained. To better understand the difficulty in proving the stronger security in $\mathsf{BGGJKRS}$, let us consider a scenario where the adversary does not issue any key query. Following the proof strategy as before, while answering partial signing queries, the challenger would want to make sure that it uses only either fsk_1 or fsk_2 but not both. The challenger needs to decide, which one? But, since now partial signature queries on m^* are also allowed, when a signing query (m, i) for $i \in \{1, 2\}$ is received, the challenger needs to be careful in using σ_m because in the end, if $m^* = m$, the reduction against the Sig challenger in the last hybrid fails. This is so because whenever the challenger needs σ_m to reply a partial signing query on m in (H_3 in the proof of $\mathsf{BGGJKRS}$ scheme above), the reduction against Sig gets it from the Sig challenger, and in this case, the reduction cannot output a forgery on m. Let us understand the challenger's dilemma with the following example:

Suppose the adversary issues the first signing query as $(m_1, 1)$. Now the challenger needs to decide whether to use fsk_1 to compute $\sigma_{m_1,1}$ or not. Let us analyze both the options and show that both the choices can go wrong:
Option 1:

- The challenger uses fsk_1 to compute $\sigma_{m_1,1}$.
- The adversary issues next query as $(m_2, 2)$. Now since fsk_1 is already used, the challenger cannot use fsk_2 to generate $\sigma_{m_2,2}$. Hence, it computes $\sigma_{m_2,2}$ as $\lfloor q/2 \rceil \sigma_{m_2} - \langle \mathsf{ct}_{\sigma_{m_2}}, \mathsf{fsk}_1 \rangle + \mathsf{noise}$.
- The adversary outputs a forgery (m_2, σ_{m_2}). Observe that this is a valid forgery from the TS adversary since it has queried partial signature on m_2 from party 2 only. But since the challenger used σ_{m_2} to compute the partial signature, (m_2, σ_{m_2}) cannot be returned as a forgery to the Sig challenger.

Option 2:

- The challenger uses fsk_2 to compute $\sigma_{m_1,1}$ as $\sigma_{m_1,1}$ as $\lfloor q/2 \rceil \sigma_{m_1} - \langle \mathsf{ct}_{\sigma_{m_1}}, \mathsf{fsk}_2 \rangle + \mathsf{noise}$.
- The adversary outputs a forgery (m_1, σ_{m_1}). Observe that this is a valid forgery from the TS adversary since it has queried partial signature on m_1 from party 1 only. But since the challenger used σ_{m_1} to compute the partial signature, (m_1, σ_{m_1}) cannot be returned as a forgery to the Sig challenger.

Remark 1. In the toy example above, one may be tempted to use guessing, and that would indeed work here. But guessing becomes hard for general case of t-out-of-n access structure with neither $n - t$ nor $t - c$ being a constant, where c is the number of parties for which the adversary asks the signing keys.

Our Solution: From the above arguments we can derive the following wishful strategy for the challenger (i) use the secret shares of fsk from at most $t-1$ parties (as before), (ii) for any signing query (m, i) do not use σ_m till $|S_m \cup S^*| \leq t-1$, where S^* is the set of parties for which the adversary asks the signing keys and $S_m = \{j \ : \ (m, j)$ has been queried so far (including the query $(m, i))\}$.

Attempt 1: For each query (m, i) simply return a random value + noise till $|S^* \cup S_m| \leq t-1$ and argue that since fsk_i is random, $\langle \mathsf{ct}_{\sigma_m}, \mathsf{fsk}_i \rangle$ is also random. But we can observe that this strategy immediately fails if the adversary issues partial signing query from party i for Q different messages for $Q > |\mathsf{fsk}_i|$.

Solution: We use the same strategy as before, where for each query (m, i), the challenger simply returns a random value + noise till $|S^* \cup S_m| \leq t-1$. But now, we use a different argument to argue indistinguishability from the honestly computed value using fsk_i. In more detail, let us consider the previous toy example of 2-out-of-2 access structure. Let the adversary issues first signing query as $(m_1, 1)$. The challenger returns

$$\sigma_{m_1,1} = r_{m_1,1} + \mathsf{noise},$$

where $r_{m_1,1}$ is uniformly random. Then if and whenever the adversary issues a signing query $(m_1, 2)$, the challenger returns

$$\sigma_{m_1,2} = \lfloor q/2 \rceil \sigma_{m_1} - r_{m_1,1} + \mathsf{noise}.$$

Notice that this time it is safe to use σ_{m_1} because party 1 and party 2 together form a valid party set and hence the challenger knows that m_1 cannot be m^*. To argue indistinguishability in the adversary's view, we implicitly view $r_{m_1,1} = \langle \mathsf{ct}_{\sigma_{m_1}}, \mathsf{fsk}_1 \rangle + r'_{m_1,1}$, and write

$$\begin{aligned}
\sigma_{m_1,2} &= \lfloor q/2 \rceil \sigma_{m_1} - r_{m_1,1} + \mathsf{noise} \\
&= \lfloor q/2 \rceil \sigma_{m_1} - \langle \mathsf{ct}_{\sigma_{m_1}}, \mathsf{fsk}_1 \rangle - r'_{m_1,1} + \mathsf{noise} \\
&= \lfloor q/2 \rceil \sigma_{m_1} - \langle \mathsf{ct}_{\sigma_{m_1}}, \mathsf{fsk}_1 \rangle - \langle \mathsf{ct}_{\sigma_{m_1}}, \mathsf{fsk}_2 \rangle + \langle \mathsf{ct}_{\sigma_{m_1}}, \mathsf{fsk}_2 \rangle - r'_{m_1,1} + \mathsf{noise} \\
&= \lfloor q/2 \rceil \sigma_{m_1} - \langle \mathsf{ct}_{\sigma_{m_1}}, \mathsf{fsk} \rangle + \langle \mathsf{ct}_{\sigma_{m_1}}, \mathsf{fsk}_2 \rangle - r'_{m_1,1} + \mathsf{noise} \\
&= \lfloor q/2 \rceil \sigma_{m_1} - \lfloor q/2 \rceil \sigma_{m_1} + e + \langle \mathsf{ct}_{\sigma_{m_1}}, \mathsf{fsk}_2 \rangle - r'_{m_1,1} + \mathsf{noise} \\
&= \langle \mathsf{ct}_{\sigma_{m_1}}, \mathsf{fsk}_2 \rangle + (-r'_{m_1,1}) + e + \mathsf{noise}.
\end{aligned}$$

This view leaves the extra terms $r'_{m_1,1}$ and $(-r'_{m_1,1})$ in $\sigma_{m_1,1}$ and $\sigma_{m_1,2}$, respectively. We can think of $r'_{m_1,1}$ and $(-r'_{m_1,1})$ as 2-out-of-2 random secret shares of 0. To address these extra random values, we modify the original construction by adding pseudorandom components to partial signatures as following. Let $\mathsf{PRF} : \mathcal{K} \times \{0,1\}^\lambda \to \mathcal{Y}$ be a PRF. Then

$$\begin{aligned}
\sigma^{\mathsf{real}}_{m_1,1} &= \langle \mathsf{ct}_{\sigma_{m_1}}, \mathsf{fsk}_1 \rangle + \mathsf{PRF}(K_1, m_1) + \mathsf{noise}, \text{ and} \\
\sigma^{\mathsf{real}}_{m_1,2} &= \langle \mathsf{ct}_{\sigma_{m_1}}, \mathsf{fsk}_2 \rangle + \mathsf{PRF}(K_2, m_1) + \mathsf{noise}.
\end{aligned}$$

It is easy to see that for correctness, $\mathsf{PRF}(K_1, m_1) + \mathsf{PRF}(K_2, m_1)$ must be zero[7]. However, for a general PRF for any two keys K and K', $\mathsf{PRF}(K, m_1) + \mathsf{PRF}(K', m_1) \neq 0$ with high probability. So, we use a key homomorphic PRF and K_1, K_2 are generated as 2-out-of-2 secret shares of $0 \in \mathcal{K}$. We recall that if PRF is key homomorphic, then for all $K, K' \in \mathcal{K}$ and all input x, $\mathsf{PRF}(K, x) + \mathsf{PRF}(K', x) = \mathsf{PRF}(K + K', x)$. Thus, for $K_2 = -K_1$, $\mathsf{PRF}(K_2, m_1) = \mathsf{PRF}(-K_1, m_1) = -\mathsf{PRF}(K_1, m_1)$[8]. Now, we can argue indistinguishability of real partial signatures from simulated ones from PRF security that allows to replace $\mathsf{PRF}(K_1, m_1)$ with random $r'_{m_1,1}$, and also from the security of secret sharing that ensures that K_1 is uniformly random.

In general, our modified construction is:

[7] Indeed, it suffices that $\mathsf{PRF}(K_1, m_1) + \mathsf{PRF}(K_2, m_1)$ can be computed publicly. In that case the $\mathsf{TS.Combine}$ algorithm can first compute and subtract the extra term $\mathsf{PRF}(K_1, m_1) + \mathsf{PRF}(K_2, m_1)$ before rounding.

[8] For $0 \in \mathcal{K}$, any $K \in \mathcal{K}$ and any input x, $\mathsf{PRF}(K, x) = \mathsf{PRF}(K + 0, x) = \mathsf{PRF}(0, x) + \mathsf{PRF}(K, x)$. Hence, $\mathsf{PRF}(0, x) = 0$. This further implies $\mathsf{PRF}(-K, x) = -\mathsf{PRF}(K, x)$.

– $\mathsf{sk}_i = (K_i, \mathsf{fsk}_i)$, where $\{K_1, \ldots, K_n\}$ are t-out-of-n secret shares of $0 \in \mathcal{K}$.

– $\mathsf{TS.PartSign}(i, \mathsf{sk}_i, m)$ computes $\sigma_{m,i} = \langle \mathsf{ct}_{\sigma_m}, \mathsf{fsk}_i \rangle + \mathsf{PRF}(K_i, m) + \mathsf{noise}_i$

A simple calculation shows the correctness. For security, we need that for $\{K_1, \ldots, K_n\}$ generated as secret shares of $0 \in \mathcal{K}$ and for all input $x \in \mathcal{X}$, the following two distributions are indistinguishable (i) $\{\mathsf{PRF}(K_1, x), \ldots, \mathsf{PRF}(K_n, x)\}$ versus (ii) $(r_{x,1}, \ldots, r_{x,n})$, where $\{r_{x,1}, \ldots, r_{x,n}\}$ are generated as secret shares of $0 \in \mathcal{Y}$. This should hold even if the adversary is given such samples for polynomially many different x of the adversary's choice. Here, $\mathcal{K}$ is the key space, $\mathcal{X}$, the input space and $\mathcal{Y}$ is the output space of the PRF. We prove this based on the security of PRF and the secret sharing scheme. Please see section 4 for details.

Lattice based key homomorphic PRF? Known lattice based constructions of key homomorphic PRF are only *almost* key homomorphic, that is $\mathsf{PRF}(K_1, x) + \mathsf{PRF}(K_2, x) = \mathsf{PRF}(K_1 + K_2, x) \pm \delta$, where δ is a small constant, mostly in $\{0, 1, 2\}$. Clearly, the above security property does not hold for almost key homomorphic PRF, because the adversary can combine the secrets using $\mathsf{SS.Combine}$ and if the result is zero, then it is the second case, else the first case, with good probability. We need extra care to work with almost key homomorphic PRF, where we also add some flooding noise to hide the error incurred by homomorphic evaluation of almost KHPRF. Please refer to section 4 for details.

Comparing with [ASY22]. In [ASY22], Agrawal *et. al* improve the $\mathsf{BGGJKRS}$ construction by providing adaptivity in the other direction as we discussed previously. They construct a threshold signature scheme that allows the adversary to output S^*, the set of parties for which it queries the signing keys, in the middle of the game, but does not allow partial signing queries on m^*. In that case also, to answer the partial signature queries before key query, the challenger has similar dilemma - to decide which of the FHE key shares to use. However the nature of the problem is different - again considering the previous toy example of 2-out-of-2 access structure, suppose the challenger decides to use fsk_1 and simulate the partial signatures from Party 2. That is, for any message m, $\sigma_{m,1} = \langle \mathsf{ct}_{\sigma_m}, \mathsf{fsk}_1 \rangle + \mathsf{noise}$ and $\sigma_{m,2} = \lfloor q/2 \rceil \sigma_m - \langle \mathsf{ct}_{\sigma_m}, \mathsf{fsk}_1 \rangle + \mathsf{noise}$. This time there is no issue related to deciding whether to use σ_m or not. Instead, the issue is that if the adversary asks fsk_2, then the challenger will be in trouble, as it will end up using both the key shares of fsk. To address this situation, [ASY22] also use a similar idea - to simulate the partial signatures till S^* is received without using either fsk_1 or fsk_2. That is, for a signing query on message m, the challenger computes $\lfloor q/2 \rceil \sigma_m$ and secret shares it as $r_{m,1}$ and $r_{m,2}$. Then it sets $\sigma_{m,1} = r_{m,1} + \mathsf{noise}$ and $\sigma_{m,2} = r_{m,2} + \mathsf{noise}$, and implicitly views $r_{m,i} = \langle \mathsf{ct}_{\sigma_m}, \mathsf{fsk}_i \rangle + r'_{m,i}$, for $i = 1, 2$. To account for extra $r'_{m,1}$ and $r'_{m,2}$, Agrawal *et. al* also modify the $\mathsf{BGGJKRS}$ construction, but instead of PRF they use random oracle. In particular, $\mathsf{sk}_1 = (\mathsf{fsk}_1, R_1, K)$, $\mathsf{sk}_2 = (\mathsf{fsk}_2, R_2, K)$. Here R_1 and R_2 are random vectors of length n (here 2) such that $R_1 + R_2 = 0$ and H is a hash function modeled as random oracle. $\mathsf{PartSign}(\mathsf{sk}_i, m) = \langle \mathsf{ct}_{\sigma_m}, \mathsf{fsk}_i \rangle + \langle H(K, m), R_i \rangle + \mathsf{noise}$, for $i = 1, 2$. When the adversary outputs S^*, the challenger does the following: for each message m for which a partial signature has been queried, it solves for h_m such that $\langle h_m, R_i \rangle = r'_{m,i}$ for all $i \in S^*$. It then programs $H(K, m) = h_m$ and returns $\{\mathsf{sk}_i\}_{i \in S^*}$. In our toy example, let $S^* = \{1\}$, then the challenger solves for $\langle h_m, R_1 \rangle = r'_{m,1}$, programs $H(K, m) = h_m$ and returns $\mathsf{sk}_1 = (\mathsf{fsk}_1, R_1, K)$. After this point, the challenger has no uncertainty and it behaves in the same way as in the proof of $\mathsf{BGGJKRS}$. Observe that it is crucial that K has entropy and is hidden from the adversary until it outputs S^*. In fact, that is the reason this improvement gives only partial adaptivity and not the full adaptivity.

Looking back to our problem, we cannot use the ROM based trick from [ASY22] because, in our case, as soon as the adversary gets a signing key, it learns K, but the uncertainty for the challenger regarding when to use fsk_i or σ_m remains.

On the other hand, interestingly, the PRF based trick does not work for [ASY22]. This is because, if $r'_{m,i}$ for $i \in [n]$ replaces $\mathsf{PRF}(K_i, m)$, then when the adversary outputs S^*, the challenger will need to provide a K'_i such that $\mathsf{PRF}(K'_i, m) = r'_{m,i}$, which we don't know how to do. For us, the PRF based improvement works, because the challenger never needs to output such a key.

Combining our approach with [ASY22] to get partially adaptive key queries with partial signatures on m^*. We can combine our PRF based technique with the ROM based technique in [ASY22] to get a construction that has the advantage of both the constructions. That is, the adversary can ask key queries in the middle of the game (but all at once) and also issue partial signing queries on m^*. We refer the readers to Appendix B for more details.

Applying the techniques at the TFHE level. In [BGG+18], Boneh *et. al* construct a threshold FHE (TFHE), using which they construct a universal thresholdizer (UT), which, in turn, is used to thresholdize various cryptographic primitives - threshold signature (TS), threshold CCA-PKE (TPKE).

We observed that all ideas and improvements that we discussed above in the context of threshold signatures, can in fact be implemented at the TFHE level itself, which then will have the obvious advantage of improving the security of all the threshold primitives that are thresholdized from TFHE.

A TFHE is the same as FHE except that the decryption algorithm is thresholdized in TFHE. In more detail,

- Setup algorithm outputs a public key fpk and n secret keys $\{\mathsf{fsk}_i\}_{i\in[n]}$.
- The Eval algorithm is the same as the one in FHE. It takes as input, the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$, where $\mathsf{ct}_i = \mathsf{Encrypt}(\mu_i)$, $\mu_i \in \{0,1\}$, and a circuit $C : \{0,1\}^k \to \{0,1\}$ and outputs a ciphertext ct that encrypts $C(\mu_1, \ldots, \mu_k)$.
- The PartDec algorithm takes as input a partial decryption key fsk_i and a ciphertext ct and outputs a partial decryption p_i.
- The FinDec algorithm takes as input a set of partial decryptions of a ciphertext from a valid set of parties (i.e. the number of parties in the set is at least t) and outputs the encrypted message.

Boneh *et. al* define semantic security and simulation security for TFHE. The semantic security is the same as the semantic security of any PKE. Roughly speaking simulation security is defined as follows. There exists an efficient simulator $\mathcal{S}$, which can simulate the secret key shares $\{\mathsf{fsk}_i\}_{i\in[n]}$, without using fsk in the Setup. Later on, given a set of ciphertexts $\{\mathsf{ct}_1, \ldots, \mathsf{ct}_k\}$ which encrypts adversarially chosen message bits $\{\mu_1, \ldots, \mu_k\}$, and any circuit $C : \{0,1\}^k \to \{0,1\}$ along with $C(\mu_1, \ldots, \mu_k)$ and a set S, $\mathcal{S}$ simulates partial decryptions of ct_C on behalf of parties in the set S. Here $\mathsf{ct}_C = \mathsf{TFHE}.\mathsf{Eval}(\{\mathsf{ct}_1, \ldots, \mathsf{ct}_k\}, C)$. The simulator does not need to know the message bits $\mu_1, \ldots, \mu_k$. The simulation security says that for any PPT adversary who gets partial decryption keys $\{\mathsf{fsk}_i\}_{i\in S^*}$ corresponding to any invalid set of parties of its choice in the beginning, the simulated view is indistinguishable from the real view. For precise definition, we refer to Definition 24.For the current discussion, we only need to observe that in this definition, the simulator $\mathcal{S}$ takes $C(\mu_1, \ldots, \mu_k)$ as input irrespective of whether S is a valid (i.e. $|S| \geq t$) or an invalid set ($|S| < t$). We observed that this is the main hurdle in proving stronger security of threshold signature (and also the other primitives) built from the universal thresholdizer (UT) in [BGG+18]. We propose stronger definition of simulation security for TFHE, where the simulator $\mathcal{S}$ needs $C(\mu_1, \ldots, \mu_k)$ only when it generates partial decryption for a valid set of parties. We then propose the same improvement in UT security, which in turn allows us to prove the stronger security for thresholdized primitives. We apply the ideas discussed above in the context of TSig at the level of TFHE to achieve the stronger simulation security.

Stronger security for threshold CCA-PKE. We define a stronger CCA security definition for a threshold public-key encryption scheme. In a real life attack scenario, an adversary capable of corrupting a number of parties below the threshold value can participate in the decryption of a targeted ciphertext ct^* with the help of a corrupt party. Let $g = t - |S^*|$ be the gap between the threshold value and the number of corrupt parties. A natural notion of CCA security should allow up to $g - 1$ number of partial decryption queries on the challenge ciphertext ct^*. We define a CCA security definition in which the adversary is allowed to query partial decryption queries on ct^* from up to $g - 1$ honest parties. Furthermore, we show that the threshold public-key encryption scheme in [BGG+18] constructed from a UT scheme and a public-key encryption satisfies our stronger notion of CCA security if UT is strongly secure (which we define) and the underlying public-key encryption is CCA secure.

Other related work. Significant research has been done in building lattice based threshold cryptography with efficient parameters. Uunfortunately efficient constructions are achieved at the cost of increasing the number of communication rounds between the parties [dPKM$^+$24] [GKS24] or restricting to n-out-of-n access structure [DOTT21,MS23]. In [dPKM$^+$24] del Pino *et. al* construct an efficient threshold signature that involves 3 rounds. In [GKS24], Gur, Katz and Silde improve [ASY22] but at the cost of increasing the number of rounds to 2. Moreover both the constructions achieve weaker notion of unforgeability (P_1 in Fig 1). In [CCK23] improve the efficiency of [BGG$^+$18] UT using iterative Shamir secret sharing.
Organization of the paper: We provide the notations and preliminaries in Sections 2. In section 3 we define simulators for secret sharing which we use in our construction. In section 4, we present key homomorphic PRF. In section 5, we define stronger definition of TFHE and provide our construction that satisfies the stronger security. In section 6, we define and prove stronger security definition of UT. In section 7, we prove stronger security of threshold signatures and threshold PKE from strongly secure UT.

2 Preliminaries

Notations: We represent the t-out-of-n threshold access structure as $\mathbb{A}_{t,n}$. At many places we refer to a party P_i as i only (i.e. discarding the letter P). For secret sharing scheme where each party's shares consist of multiple shares, we refer to each share index by the term 'party-share' or only 'share'. For a matrix $\mathbf{M}$ of dimensions $\ell \times N$, we represent the i-th row as $\mathbf{M}[i]$ or $\mathbf{M}_i$. Let $S \subseteq [\ell]$, then we use $\mathbf{M}[S]$ or $\mathbf{M}_S$ to represent the matrix formed by rows in set S. For a vector $\mathbf{x}$, unless otherwise stated, x_i represents the i-th element in $\mathbf{x}$.

2.1 Fully Homomorphic Encryption (FHE).

A fully homomorphic encryption scheme is an encryption scheme that allows computations on encrypted data.

Definition 1 (Fully Homomorphic Encryption). *A fully homomorphic encryption scheme* FHE *is a tuple of* PPT *algorithms* $\mathsf{FHE} = (\mathsf{FHE.KeyGen}, \mathsf{FHE.\ Encrypt}, \mathsf{FHE.\ Eval}, \mathsf{FHE.\ Decrypt})$ *defined as follows:*

- $\mathsf{FHE.KeyGen}(1^\lambda, 1^d) \to (\mathsf{pk}, \mathsf{sk})$*: On input the security parameter* λ *and a depth bound* d*, the* KeyGen *algorithm outputs a key pair* $(\mathsf{pk}, \mathsf{sk})$*.*
- $\mathsf{FHE.\ Encrypt}(\mathsf{pk}, \mu) \to \mathsf{ct}$*: On input a public key* pk *and a message* $\mu \in \{0,1\}$*, the encryption algorithm outputs a ciphertext* ct*.*
- $\mathsf{FHE.\ Eval}(\mathsf{pk}, \mathcal{C}, \mathsf{ct}_1, \ldots, \mathsf{ct}_k) \to \hat{\mathsf{ct}}$*: On input a public key* pk*, a circuit* $\mathcal{C} : \{0,1\}^k \to \{0,1\}$ *of depth at most* d*, and a tuple of ciphertexts* $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$*, the evaluation algorithm outputs an evaluated ciphertext* $\hat{\mathsf{ct}}$*.*
- $\mathsf{FHE.\ Decrypt}(\mathsf{pk}, \mathsf{sk}, \hat{\mathsf{ct}}) \to \hat{\mu}$*: On input a public key* pk*, a secret key* sk *and a ciphertext* $\hat{\mathsf{ct}}$*, the decryption algorithm outputs a message* $\hat{\mu} \in \{0, 1, \perp\}$*.*

The definition above can be adapted to handle plaintexts over larger sets than $\{0,1\}$. Note that the evaluation algorithm takes as input a (deterministic) circuit rather than a possibly randomized algorithm. An FHE should satisfy compactness, correctness and security properties defined below.

Definition 2 (Compactness). *An* FHE *scheme is compact if there exists a polynomial function* $f(\cdot,\cdot)$ *such that for all* λ*, depth bound* d*, circuit* $\mathcal{C} : \{0,1\}^k \to \{0,1\}$ *of depth at most* d*, and* $\mu_i \in \{0,1\}$ *for* $i \in [k]$*, the following holds: for* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda, 1^d)$*,* $\mathsf{ct}_i \leftarrow \mathsf{FHE.\ Encrypt}(\mathsf{pk}, \mu_i)$ *for* $i \in [k]$*,* $\hat{\mathsf{ct}} \leftarrow \mathsf{FHE.\ Eval}(\mathsf{pk}, \mathcal{C}, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$*, the bit-length of* $\hat{ct}$ *is at most* $f(\lambda, d)$*.*

Definition 3 (Correctness). *An* FHE *scheme is correct if for all* λ*, depth bound* d*, circuit* $\mathcal{C} : \{0,1\}^k \to \{0,1\}$ *of depth at most* d*, and* $\mu_i \in \{0,1\}$ *for* $i \in [k]$*, the following holds: for* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda, 1^d)$*,* $\mathsf{ct}_i \leftarrow \mathsf{FHE.\ Encrypt}(\mathsf{pk}, \mu_i)$ *for* $i \in [k]$*,* $\hat{\mathsf{ct}} \leftarrow \mathsf{FHE.\ Eval}(\mathsf{pk}, \mathcal{C}, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$*, we have*

$$\Pr[\mathsf{FHE.\ Decrypt}(\mathsf{pk}, \mathsf{sk}, \hat{\mathsf{ct}}) = \mathcal{C}(\mu_1, \ldots, \mu_k)] = 1 - \lambda^{-\omega(1)}.$$

Definition 4 (Security). *An* FHE *scheme is secure if for all λ and depth bound d, the following holds: for any adversary $\mathcal{A}$ with run-time $2^{o(\lambda)}$, the following experiment outputs 1 with probability $2^{-\Omega(\lambda)}$:*

1. *On input the security parameter λ and a depth bound d, the challenger runs* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda, 1^d)$ *and* $\mathsf{ct} \leftarrow \mathsf{FHE.Encrypt}(\mathsf{pk}, b)$ *for* $b \leftarrow \{0,1\}$. *It provides* $(\mathsf{pk}, \mathsf{ct})$ *to* $\mathcal{A}$.
2. $\mathcal{A}$ *outputs a guess* b'. *The experiment outputs* 1 *if* $b = b'$.

Similar to BGGJKRS , our constructions also use a *special* FHE having some additional properties as described in [BGG+18]. These properties are satisfied by direct adaptations of typical FHE schemes such as [BV11,GSW13] (see, e.g., [BGG+18, Appendix B]).

Definition 5 (Special FHE). *An* FHE *scheme is a special* FHE *scheme if it satisfies the following properties:*

1. *On input* $(1^\lambda, 1^d)$, *the key generation algorithm* $\mathsf{FHE.KeyGen}$ *outputs* $(\mathsf{pk}, \mathsf{sk})$, *where the public key contains a prime q and the secret key is a vector* $\mathsf{sk} \in \mathbb{Z}_q^m$ *for some* $m = \mathsf{poly}(\lambda, d)$.
2. *The decryption algorithm* $\mathsf{FHE.Decrypt}$ *consists of two functions* $(\mathsf{FHE.decode}_0, \mathsf{FHE.decode}_1)$ *defined as follows:*
   - $\mathsf{FHE.decode}_0(\mathsf{sk}, \mathsf{ct})$*: On input an encryption of a message* $\mu \in \{0,1\}$ *and a secret key vector* sk, *it outputs* $p = \mu\lfloor q/2 \rceil + e \in \mathbb{Z}_q$ *for* $e \in [-cB, cB]$ *with* $B = B(\lambda, d, q)$ *and e is an integer multiple of c. This algorithm must be a linear operation over* $\mathbb{Z}_q$ *in the secret key* sk.
   - $\mathsf{FHE.decode}_1(p)$*: On input* $p \in \mathbb{Z}$, *it outputs* 1 *if* $p \in [-\lfloor q/4 \rceil, \lfloor q/4 \rceil]$, *and* 0 *otherwise.*

   The bound $B = B(\lambda, d, q)$ *is referred to as the associated noise bound parameter of the construction and c as the associated multiplicative constant.*

2.2 Zero Knowledge Proofs with Pre-processing

These are specific type of zero knowledge proof system where all but the last communication round between a prover and a verifier can be pre-processed offline. This is captured by a pre-processing step that generates a common reference string for the verifier σ_V and a separate common reference string for the prover σ_P. Since only the last step of the communication is required for the online phase, the proof system can be viewed as a weaker variant of a non-interactive zero knowledge (NIZK) proof system. Such proof system, termed as zero knowledge proof system with pre-processing (PZK) in [BGG+18], can be constructed from a much weaker assumption of one-way functions [DSMP88,LS91] than a standard NIZK proof system and is sufficient for our purpose.

Definition 6. *Let L be a language with relation R. A tuple of* PPT *algorithms* $\mathsf{PZK} = (\mathsf{PZK.Pre}, \mathsf{PZK.Prove}, \mathsf{PZK.Verify})$ *is a zero knowledge proof system with pre-processing if the following conditions are true. For* $(\sigma_V, \sigma_P) \leftarrow \mathsf{PZK.Pre}(1^\lambda)$*:*

Completeness: *For every* $(x, w) \in R$, *we have that:* $\Pr[\mathsf{PZK.Verify}(\sigma_V, x, \pi) = 1 \ : \ \pi \leftarrow \mathsf{PZK.Prove}(\sigma_P, x, w)] = 1$ *where the probability is over the internal randomness of all the* PZK *algorithms.*

Soundness: *For every* $x \notin L$, *we have that:* $\Pr[\exists\ \pi \ : \ \mathsf{PZK.Verify}(\sigma_V, x, \pi) = 1] = \mathit{neg}(\lambda)$, *where the probability is over* PZK.Pre.

Zero-Knowledge: *There exists a* PPT *algorithm $\mathcal{S}$ such that for any x, w where $V(x, w) = 1$, the following two distributions are computationally indistinguishable:*

$$\{\sigma_V, \mathsf{PZK.Prove}(\sigma_P, x, w)\} \approx_c \{\mathcal{S}(x)\}$$

2.3 Some Useful Definitions and Lemmas

Definition 7 (Statistical Distance). *Let E be a finite set, Ω a probability space, and $X, Y : \Omega \to E$ random variables. The statistical distance between X and Y is defined as a function Δ defined by*

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

.

Lemma 1 (Smudging Lemma [AJLA+12, MW16). *] Let $B_1, B_2 \in \mathbb{N}$. For any $e_1 \in [-B_1, B_1]$, let E_1 and E_2 be independent random variables uniformly distributed on $[-B_2, B_2]$ and define the two stochastic variables $X_1 = E_1 + e_1$ and $X_2 = E_2$. Then $\Delta(E_1, E_2) < B_1/B_2$.*

3 Secret Sharing

We first recall the definitions related to secret sharing. Then we define two simulators that can simulate secret shares for parties one by one as per the need. More importantly, the simulators do not need the actual secret till the shares are generated for less than the threshold number of parties.

Access Structures

Definition 8 (Monotone Access Structure). *Let $P = \{P_i\}_{i\in[n]}$ be a set of parties. A collection $\mathbb{A} \subseteq \mathcal{P}(P)$ is monotone if for any two sets $B, C \subseteq P$, if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$. A monotone access structure on P is a monotone collection $\mathbb{A} \subseteq \mathcal{P}(P) \setminus \emptyset$.*

Definition 9 (Efficient Access Structure). *Let $P = \{P_i\}_{i\in[n]}$ be a set of parties. For any set of parties $S \subseteq P$, let $\mathbf{x}_S = (x_1, \ldots, x_n) \in \{0,1\}^n$ with $x_i = 1$ iff $P_i \in S$. An access structure $\mathbb{A}$ on P is called an efficient access structure if there exists a polynomial size circuit $f_{\mathbb{A}} : \{0,1\}^n \to \{0,1\}$, such that for all $S \subseteq P$, $f_{\mathbb{A}}(\mathbf{x}_S) = 1$ iff $S \in \mathbb{A}$.*

Definition 10 ((In)Valid Party Sets).

1. *The sets in $\mathbb{A}$ are called valid party sets and the sets in $\mathcal{P}(P) \setminus \mathbb{A}$ are called invalid party sets.*
2. *Let $S \subseteq P$ be a subset of parties in P. S is called maximal invalid party set if $S \notin \mathbb{A}$, but for any $P_i \in P \setminus S$, we have $S \cup \{P_i\} \in \mathbb{A}$.*
3. *S is called minimal valid party set if $S \in \mathbb{A}$, but for any $S' \subsetneq S$, we have $S' \notin \mathbb{A}$.*

Notation: In this work, we sometimes drop the word monotone. When it is clear from the context, we use either i or P_i to represent party P_i.

Definition 11 (Secret sharing). *Let $P = \{P_1, \ldots, P_n\}$ be a set of parties and $\mathbb{S}$ be a class of efficient access structures on P. A secret sharing scheme SS for a secret space $\mathcal{K}$ is a tuple of PPT algorithms $\mathsf{SS} = (\mathsf{SS.Share}, \mathsf{SS.Combine})$ defined as follows:*

- $\mathsf{SS.Share}(\mathsf{k}, \mathbb{A}) \to (s_1, \ldots, s_n)$*: On input a secret $\mathsf{k} \in \mathcal{K}$ and an access structure $\mathbb{A}$, the sharing algorithm returns shares $s_1, \ldots, s_n$ for all parties.*
- $\mathsf{SS.Combine}(B) \to \mathsf{k}$*: On input a set of shares $B = \{s_i\}_{i\in S}$, where $S \in \mathbb{A}$, the combining algorithm outputs a secret $\mathsf{k} \in \mathcal{K}$.*

A secret sharing algorithm must satisfy the following correctness and privacy properties.

Definition 12 (Correctness). *For all $S \in \mathbb{A}$ and $\mathsf{k} \in \mathcal{K}$, if $(s_1, \ldots, s_n) \leftarrow \mathsf{SS.Share}(\mathsf{k}, \mathbb{A})$, then*

$$\mathsf{SS.Combine}(\{s_i\}_{i\in S}) = \mathsf{k}.$$

Definition 13 (Privacy). *For all $S \notin \mathbb{A}$ and $\mathsf{k}_0, \mathsf{k}_1 \in \mathcal{K}$, if $(s_{b,1}, \ldots, s_{b,n}) \leftarrow \mathsf{SS.Share}(\mathsf{k}_b, \mathbb{A})$ for $b \in \{0,1\}$, then the distributions $\{s_{0,i}\}_{i\in S}$ and $\{s_{1,i}\}_{i\in S}$ are identical.*

Definition 14 (Linear Secret Sharing (LSSS)). *Let $P = \{P_i\}_{i\in[n]}$ be a set of parties and $\mathbb{S}$ be a class of efficient access structures. A secret sharing scheme SS with secret space $\mathcal{K} = \mathbb{Z}_p$ for some prime p is called a linear secret sharing scheme if it satisfies the following properties:*

- $\mathsf{SS.Share}(\mathsf{k}, \mathbb{A})$*: There exists a matrix $\mathbf{M} \in \mathbb{Z}_p^{\ell \times N}$ called the share matrix, and each party P_i is associated with a partition $T_i \subseteq [\ell]$. We assume a $\mathsf{SS.Setup}$ algorithm that outputs the share matrix and the partitions as pp which is implicitly assumed to be an input to the $\mathsf{SS.Share}$ and the $\mathsf{SS.Combine}$ algorithm.*
  To create the shares on a secret k, the sharing algorithm first samples uniform values $r_1, \ldots, r_{N-1} \leftarrow \mathbb{Z}_p$ and defines a vector $\mathbf{w} = \mathbf{M} \cdot (\mathsf{k}, r_1, \ldots, r_{N-1})^\intercal$. The share k_i for P_i consists of the entries $k_i = \{w_j\}_{j\in T_i}$.
  At many places in the paper, when it is clear from the context, we directly write $(w_1, \ldots, w_\ell) \leftarrow \mathsf{SS.Share}(\mathsf{k}, \mathbb{A})$ instead of $(k_1, \ldots, k_n) \leftarrow \mathsf{SS.Share}(\mathsf{k}, \mathbb{A})$ with $k_i = \{w_j\}_{j\in T_i}$.

– $\mathsf{SS.Combine}(B)$*: For any valid set* $S \in \mathbb{A}$*, we have*

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

over $\mathbb{Z}_p$ *where* $\mathbf{M}[j]$ *denotes the* j*th row of* $\mathbf{M}$*. Any valid set of parties* $S \in \mathbb{A}$ *can efficiently find the coefficients* $\{c_j\}_{j \in \bigcup_{i \in S} T_i}$ *satisfying*

$$\sum_{j \in \bigcup_{i \in S} T_i} c_j \cdot \mathbf{M}[j] = (1, 0, \ldots, 0)$$

and recover the secret by computing $\mathsf{k} = \sum_{j \in \bigcup_{i \in S} T_i} c_j \cdot w_j$*. The coefficients* $\{c_j\}$ *are called recovery coefficients.*

To secret-share a vector $\mathbf{s} = \{s_1, \ldots, s_m\} \in \mathbb{Z}_p^m$, we can simply secret-share each entry s_i using fresh randomness. This gives secret share vectors $\mathbf{s}_1, \ldots, \mathbf{s}_\ell \in \mathbb{Z}_p^m$. Using these secret shares, the secret vector $\mathbf{s}$ can be recovered using the same coefficients as that for a single field element.

Definition 15 ((In)Valid Share Sets). *Let* $P = \{P_1, \ldots, P_n\}$ *be a set of parties,* $\mathbb{S}$ *a class of efficient structures on* P*, and* SS *a linear secret sharing scheme with share matrix* $\mathbf{M} \in \mathbb{Z}_q^{\ell \times N}$*. For a set of indices* $T \subseteq [\ell]$*,* T *is said to be a valid share set if* $(1, 0, \ldots, 0) \in \mathsf{span}(\{\mathbf{M}[j]\}_{j \in T})$*, and an invalid share set otherwise. We also use following definitions:*

- *A set of indices* $T \subseteq [\ell]$ *is a maximal invalid share set if* T *is an invalid share set, but for any* $i \in [\ell] \setminus T$*, the set* $T \cup \{i\}$ *is a valid share set.*
- *A set of indices* $T \subseteq [\ell]$ *is a minimal valid share set if* T *is a valid share set, but for any* $T' \subsetneq T$*,* T' *is an invalid share set.*

Definition 16 (Threshold Access Structure (TAS)). *Let* $P = \{P_1, \ldots, P_n\}$ *be a set of parties. An access structure* $\mathbb{A}_{t,n}$ *is called a threshold access structure if for every set of parties* $S \subseteq P$*,* $S \in \mathbb{A}_{t,n}$ *if and only if* $|S| \geq t$*. We let* TAS *to be the class of all access structures* $\mathbb{A}_{t,n}$ *for all* $t, n \in \mathbb{N}$*.*

Below we describe the properties of two kinds of linear secret sharing schemes for threshold access structure - (i) Shamir secret sharing and (ii)$\{0,1\}$- linear secret sharing.

Theorem 1 (Shamir Secret Sharing). *Let* $P = \{P_1, \ldots, P_n\}$ *be a set of parties, and let* TAS *be the class of threshold access structures on* P*. Then, there exists a linear secret sharing scheme (Definition 14)* $\mathsf{SS} = (\mathsf{SS.Share}, \mathsf{SS.Combine})$ *with secret space* $\mathcal{K} = \mathbb{Z}_q$ *for some prime* q *satisfying the following properties:*

- *For any secret* $s \in \mathbb{Z}_q$ *and* $\mathbb{A}_{t,n} \in \mathsf{TAS}$*, each share for party* P_i *consists of a single element* $w_i \in \mathbb{Z}_q$*. Let us denote* $w_0 = s$*.*
- *For every* $i, j \in [n] \cup \{0\}$ *and set* $S \subset [n] \cup \{0\}$ *of size* t*, there exists an efficiently computable Lagrange coefficients* $\gamma_{i,j}^S \in \mathbb{Z}_q$ *such that* $w_j = \sum_{i \in S} \gamma_{i,j}^S \cdot w_i$*.*

We will use $\mathsf{ShSS} = (\mathsf{ShSS.Share}, \mathsf{ShSS.Combine})$ to refer to a Shamir secret sharing scheme in this work.

Lemma 2 ([ABV+12]). *Let* $P = \{P_1, \ldots, P_n\}$ *be a set of parties,* TAS *the class of threshold access structures on* P*, and* ShSS *a Shamir secret sharing scheme with secret space* $\mathbb{Z}_p$ *for some prime* q *with* $(n!)^3 \leq q$*. Then, for any set* $S \subset [n] \cup \{0\}$ *of size* t*, and for any* $i, j \in [n]$*, the product* $(n!)^2 \cdot \gamma_{i,j}^S$ *has bound*

$$|(n!)^2 \cdot \gamma_{i,j}^S| \leq (n!)^3.$$

We use the following lemma about TAS from [BGG+18].

Lemma 3 ($\{0,1\}$-LSSS for TAS [BGG+18]). *Let* $P = \{P_1, \ldots, P_n\}$ *be a set of parties. Let* $\mathbb{A}_{t,n}$ *be a* t*-out-of-*n *threshold access structure. There exists an efficient linear secret sharing scheme* $\mathsf{bSS} = (\mathsf{bSS.Share}, \mathsf{bSS.Combine})$ *over the secret space* $\mathcal{K} = \mathbb{Z}_q$ *satisfying the following property:*

- *Let s be a shared secret and $\{w_j\}_{j\in T_i}$ be the share of party P_i for $i \in [n]$. Then, for every set $S \in \mathbb{A}_{t,n}$, there exists a subset $T \subseteq \bigcup_{i\in S} T_i$ such that $s = \sum_{j\in T} w_j$. Moreover the set T can be computed efficiently for all $S \in \mathbb{A}_{t,n}$.*

We call a linear secret sharing scheme that satisfy the properties above as a $\{0,1\}$-linear secret sharing scheme[9].

Note that for any $\{0,1\}$-linear secret sharing scheme bSS, and for any minimal valid share set $T \subseteq [\ell]$, we have that $\sum_{j\in T} w_j = s$. We will use $\mathsf{bSS} = (\mathsf{bSS.Share}, \mathsf{bSS.Combine})$ to refer to a $\{0,1\}$-LSSS in this work.

3.1 LSSS Simulators

In the above definition of linear secret sharing (Definition 14), the $\mathsf{SS.Share}$ algorithm takes the secret as input and outputs the secret shares for all the parties in one step. However, the secret shares can indeed be generated in a sequential manner - one party (or even one share) at a time - and the secret is needed only when the set of parties (or shares) become valid. This is straightforward, for example, for n-out-of-n secret sharing, where $n-1$ secret shares are chosen independently randomly as $r_1, \ldots, r_{n-1}$ and then the last share is computed as $s - \sum_{i\in[n-1]} r_i$. However, for more general case of t-out-of-n sharing, where each party can have more than one secret share, it is little more involved. We formalize this by defining two simulators - SSSimI and SSSimV as follows:

- $\mathsf{SSSimI}(\mathsf{pp}, S, \{w_i\}_{i\in S}, R, \mathsf{st}) \to (\{w_j\}_{j\in R}, \mathsf{st}')$: takes as input a set $S \subseteq [\ell]$, for which the secret shares are already assigned, along with the assigned secret shares $\{w_i\}_{i\in S}$, a subset $R \subseteq [\ell]$, for which the secret shares are to be generated and a state st, and outputs the secret shares for indices in R and a new state st'. Wlog, we assume $R \cap S = \emptyset$, otherwise for such $j \in R \cap S$, the simulator simply returns the w_j from the input secret shares and works with $R \setminus S$. We note that the simulator does not take the secret s being shared and that $S \cup R$ must be an invalid share set.

  1. If $S \cup R$ is a valid share set, then return $\perp$.
  2. Initialize $I = \emptyset$ and $W = S$.
  3. For each $j \in R$,
     - If $\mathbf{M}[j] \notin \mathsf{Span}(\mathbf{M}_W)$ (i.e., is independent of rows in $\mathbf{M}_W$), then $I = I \cup \{j\}$ and $W = W \cup \{j\}$; and $w_j \leftarrow \mathbb{Z}_q$.
       Else, $\exists\ \{\gamma_\alpha\}_{\alpha\in W}$ such that $\mathbf{M}[j] = \sum_{\alpha\in W} \gamma_\alpha \mathbf{M}[\alpha]$. Set $w_j = \sum_{\alpha\in W} \gamma_\alpha w_\alpha$.
  4. Return $\{w_\alpha\}_{\alpha\in R}$ and $\mathsf{st}' = \mathsf{st}$.

- $\mathsf{SSSimV}(\mathsf{pp}, S, \{w_i\}_{i\in S}, s, R, \mathsf{st}) \to (\{w_j\}_{j\in R}, \mathsf{st}')$: takes as input a set $S \subseteq [\ell]$, for which the secret shares are already assigned, along with the assigned secret shares $\{w_i\}_{i\in S}$, the secret s, a subset $R \subseteq [\ell]$, for which the secret shares are to be generated and a state st, and outputs the secret shares for indices in R and a new state st'. Wlog, we assume $R \cap S = \emptyset$, otherwise for such $j \in R \cap S$, the simulator simply returns the w_j from the input secret shares and work with $R \setminus S$. We note that this simulator takes the secret s being shared and that $S \cup R$ is a valid share set.

  1. If $\mathsf{st} = \emptyset$
     (a) If S is a valid share set, then return $\perp$.
     (b) Find a maximal invalid share set $X \subseteq [\ell]$ such that $S \subseteq X$.
     (c) Initialize $I = \emptyset$ and $W = S$.
     (d) For $j \in X \setminus S$,
        i. If $\mathbf{M}[j] \notin \mathsf{Span}(\mathbf{M}_W)$, sample $w_j \leftarrow \mathbb{Z}_q$ and update $I = I \cup \{j\}$ and $W = W \cup \{j\}$.
        ii. Else compute w_j from $\{w_\alpha\}_{\alpha\in W}$ in the same ways as in Step 3 in SSSimI above.
     (e) $\mathsf{st}' = (X, \{w_\alpha\}_{\alpha\in X})$
     (f) For $j \in R \setminus X$

[9] We are using a slightly different naming from [BGG+18]. They call such a scheme a *special* linear secret sharing scheme and the class of access structures that supports special LSS as $\{0,1\}$-LSSS.

i. Since X is a *maximally* invalid share set, $X \cup \{j\}$ is a valid share set. Hence, there exists $\{c_\alpha\}_{\alpha \in X \cup \{j\}}$[10] such that $(1, 0, \ldots, 0) = \sum_{\alpha \in X \cup \{j\}} c_\alpha \mathbf{M}[\alpha]$, which implies $s = \sum_{\alpha \in X \cup \{j\}} c_\alpha w_\alpha$. Hence, compute $w_j = \frac{s - \sum_{\alpha \in X} c_\alpha w_\alpha}{c_j}$

(g) Return $\{w_\alpha\}_{\alpha \in R}$ and st'.

2. Else
   (a) Parse st as $(X, \{w_\alpha\}_{\alpha \in X})$ where X is a maximally invalid share set - if not, abort.
   (b) Again, wlog we assume $R \cap X = \emptyset$, because otherwise for all $j \in X \setminus R$, w_j is already set and is returned as it is.
   (c) For $j \in R \setminus X$
      i. Since X is a *maximally* invalid share set, $X \cup \{j\}$ is a valid share set. Hence, there exists $\{c_\alpha\}_{\alpha \in X \cup \{j\}}$ such that $s = \sum_{\alpha \in X \cup \{j\}} c_\alpha w_\alpha$. Hence, compute $w_j = \frac{s - \sum_{\alpha \in X} c_\alpha w_\alpha}{c_j^{X \cup \{j\}}}$
3. Return $\{w_\alpha\}_{\alpha \in R}$ and $\mathsf{st}' = \mathsf{st}$.

Below we show that the secret shares generated by the simulators SSSimI and SSSimV have the same distribution as those generated by the $\mathsf{SS.Share}$ algorithm. We formalize this via the following claim.

Claim 2 *For all adversary $\mathcal{A}$, $\mathsf{Expt}^0_{\mathsf{SSSim}}$ and $\mathsf{Expt}^1_{\mathsf{SSSim}}$, as described below are identical in $\mathcal{A}$'s view.*

$\mathsf{Expt}^b_{\mathsf{SSSim}}(1^\lambda)$:

1. *Upon input the security parameter 1^λ and a threshold access structure $\mathbb{A}_{t,n}$, the challenger runs $\mathsf{SS.Setup}(1^\lambda, \mathbb{A}_{t,n})$ to generate a share matrix $\mathbf{M}$ and the partitions $\{T_i\}_{i \in [n]}$, and sends them to $\mathcal{A}$.*
2. *$\mathcal{A}$ outputs a secret $s \in \mathbb{Z}_q$ (for which secret shares are to be generated).*
3. *The challenger does the following:*
   - *If $b = 0$, generates $\{w_j\}_{j \in [\ell]} \leftarrow \mathsf{SS.Share}(s, \mathbb{A}_{t,n})$*
     Else if $b = 1$, initializes $W = \emptyset$ and $\mathsf{st} = \emptyset$.
4. *For $\kappa = 1$ to ℓ,*
   (a) $\mathcal{A}$ outputs $j_\kappa \in [\ell]$ (wlog we assume that j_κ was not queried before).
   (b) If $b = 0$, the challenger returns w_{j_κ} (generated in Step 3)
   (c) If $b = 1$ and $W \cup \{j_\kappa\}$ is an invalid share set, generate $(w_{j_\kappa}, \mathsf{st}') \leftarrow \mathsf{SSSimI}(W, \{w_\alpha\}_{\alpha \in W}, \{j_\kappa\}, \mathsf{st})$; stores w_{j_κ}, updates $W = W \cup \{j_\kappa\}, \mathsf{st} = \mathsf{st}'$ and returns w_{j_κ} to $\mathcal{A}$.
   (d) If $b = 1$ and $W \cup \{j_\kappa\}$ is a valid share set, generate $(w_{j_\kappa}, \mathsf{st}') \leftarrow \mathsf{SSSimV}(W, \{w_\alpha\}_{\alpha \in W}, s, \{j_\kappa\}, \mathsf{st})$; stores w_{j_κ}, updates $W = W \cup \{j_\kappa\}, \mathsf{st} = \mathsf{st}'$ and returns w_{j_κ} to $\mathcal{A}$.
5. *At the end, $\mathcal{A}$ outputs its guess bit b' and wins if $b' = b$.*

Proof. We first recall the $\mathsf{SS.Share}$ algorithm. The secret shares of s, $\{w_j\}_{j \in [\ell]}$ are computed as $\mathbf{M} \cdot (s, r_1, \ldots, r_{N-1})^\mathsf{T} = (w_1, \ldots, w_\ell)^\mathsf{T}$, where $r_1, \ldots, r_{N-1}$ are chosen uniformly randomly from $\mathbb{Z}_q$.

To prove the claim, we show that the secret shares $\{w_j\}_{j \in [\ell]}$ generated by the simulators can also be expressed as $\mathbf{M} \cdot (s, r_1, \ldots, r_{N-1})^\mathsf{T} = (w_1, \ldots, w_\ell)^\mathsf{T}$, for uniformly random $r_1, \ldots, r_{N-1}$.

Let $k \in [\ell]$ be the index where the transition from invalid to valid happens - i.e., $\{j_1, \ldots, j_k\}$ is an invalid share set and $\{j_1, \ldots, j_k, j_{k+1}\}$ is a valid share set. Let I be the maximal invalid share set chosen by SSSimV when called to generate $w_{j_{k+1}}$. Note that for all subsequent queries also, the same invalid share set I is used. Further, let $E \subseteq I$ be the set of indices for which the secret shares are chosen uniformly randomly. That is, $E = \{j \mid w_j \leftarrow \mathbb{Z}_q, j \in I\}$. From the definition of the simulators, we note that the rows in $\mathbf{M}_E$ are independent. Let $|E| = c$. Now we make the following observations: since I is a maximally invalid share set and $E \subseteq I$, E is also an invalid share set. Hence, for all $j \in [\ell] \setminus I$, $\mathbf{M}[j] \notin \mathsf{Span}(\mathbf{M}_E)$, otherwise the adversary could recover the shares for a valid share set $I \cup \{j\}$ from the shares of an invalid share set I, where the j-th share w_j could be generated from $\{w_\alpha\}_{\alpha \in E}$, thus breaking the security of secret sharing scheme. Thus,

[10] These coefficients can be different for different sets $X \cup \{j\}$, but we don't explicitly specify the set $X \cup \{j\}$ in c_α to keep the notation simple.

$rank(\mathbf{M}) \geq |E| + 1$ and hence, $N \geq |E| + 1$ or $|E| \leq N - 1$. Thus, there are more unknowns $(r_1, \ldots, r_{N-1})$ than the number of independent equations. So, we can solve for them. Moreover, since for each $j \in E$, w_j is uniformly random in $\mathbb{Z}_q$, $r_1, \ldots, r_{N-1}$ are also uniformly random in $\mathbb{Z}_q$. In more details,

- Case 1: $|E| = N - 1$. Then given $\mathbf{M}_E$ and $\{w_j\}_{j \in E}$, we can uniquely solve for $r_1, \ldots, r_{N-1}$ such that $\mathbf{M}_E \cdot (s, r_1, \ldots, r_{N-1})^\mathsf{T} = (w_j)_{j \in E}$, and since $\{w_j\}_{j \in E}$ are also chosen uniformly randomly from the same space as r's, $\{r_j\}_{j \in [N-1]}$ are also uniformly random.
- Case 2: $|E| < N - 1$. In this case, we first choose $N - 1 - |E|$ r's uniformly randomly from $\mathbb{Z}_q$ and then uniquely solve for the rest. In more detail, let $E_C \subseteq [2, N]$ be the set of columns in $\mathbf{M}$ such that the matrix formed by rows in E and columns in E_C is a square full rank matrix. Thus $|E| = |E_C|$. Then, for all $i \in [2, N] \setminus E_C$, $r_{i-1} \leftarrow \mathbb{Z}_q$, and the remaining r's: $\{r_{i-1}\}_{i \in E_C}$ are solved uniquely as in case 1 and by the same argument, these r's are also uniformly random.

Finally, it is straightforward to verify that for any $(s, r_1, \ldots, r_{N-1})$, such that $\forall\, j \in E$, $\mathbf{M}[j] \cdot (s, r_1, \ldots, r_{N-1})^\mathsf{T} = w_j$, $\mathbf{M}[\kappa] \cdot (s, r_1, \ldots, r_{N-1})^\mathsf{T} = w_\kappa$, for all $\kappa \in [\ell] \setminus E$.

4 Key Homomorphic PRF

We recall the definitions related to pseudorandom functions.

Definition 17 (Pseudorandom Function (PRF)). *A function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ with key space $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$ is a secure pseudorandom function if for all* PPT *adversary $\mathcal{A}$,*

$$|\Pr[k \leftarrow \mathcal{K} \ : \ \mathcal{A}^{F(k,\cdot)}(1^\lambda) = 1] - \Pr[f \leftarrow \mathsf{Funcs}(\mathcal{X}, \mathcal{Y}) \ : \ \mathcal{A}^{f(\cdot)}(1^\lambda) = 1]| = \mathsf{neg}(\lambda),$$

where $\mathsf{Funcs}(\mathcal{X}, \mathcal{Y})$ *denotes the set of all functions with domain $\mathcal{X}$ and range $\mathcal{Y}$.*

Definition 18 (Key Homomorphic PRF). *Any function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a key homomorphic* PRF *(*KHPRF*) [NPR99,BLMR13] if it satisfies the following two properties:*

1. *It must be a PRF.*
2. *It satisfies key homomorphism: for any $k_1, k_2 \in \mathcal{K}$, $F(k_1, x) + F(k_2, x) = F(k_1 + k_2, x)$ for all $x \in \mathcal{X}$*[11]*.*

Definition 19 (Almost Key Homomorphic PRF).
A δ-almost KHPRF *[BLMR13] is the same as the standard KHPRF (Definition 18) except that the second condition is different as:*

2. *It satisfies (almost) key homomorphism: for any $k_1, k_2 \in \mathcal{K}$, $F(k_1, x) + F(k_2, x) = F(k_1 + k_2, x) + e$, where $|e| \leq \delta$, for all $x \in \mathcal{X}$.*

Almost KHPRF are constructed from LWE in [BLMR13,BP14,Kim20], where $\delta = 1$ or 2, depending upon the choice of the parameters.

We prove the following lemma which says that: for a secure KHPRF F and linear secret sharing scheme SS, let K is a KHPRF key and is secret shared as $(K_1, \ldots, K_n) \leftarrow \mathsf{SS.Share}(K, \mathbb{A}_{t,n})$. Then for all PPT adversary $\mathcal{A}$, who outputs polynomially many queries of the form (i, x), the following two views are indistinguishable - in the first (real) world, $\mathcal{A}$ receives $F(K_i, x)$, while in the other (ideal) world, $\mathcal{A}$ receives $R_{x,i}$, where $(R_{x,1}, \ldots, R_{x,n}) \leftarrow \mathsf{SS.Share}(F(K, x), \mathbb{A}_{t,n})$[12]. This is true even if the adversary can corrupt up to $t-1$ parties (now outputs for only uncorrupted parties are random in the ideal world) and also knows K. Intuitively, this holds because, from the security of SS, the key shares $K_1, \ldots, K_n$ are "random" with the constraint that any valid combination of these keys gives K. Hence, from KHPRF security we can replace $F(K_i, x)$ in the

[11] In general, if $\mathcal{K}$ is a group with operation '+' and $\mathcal{Y}$ is a group with operation '$*$', then $F(k_1, x) * F(k_2, x) = F(k_1 + k_2, x)$.

[12] Each K_i may indeed consist of multiple keys as $K_i = \{k_j\}_{j \in T_i}$. In that case, $F(K_i, x) = \{F(k_j, x)\}_{j \in T_i}$ and $R_{x,i} = \{r_{x,j}\}_{j \in T_i}$.

real world with random $R_{x,i}$ in the ideal world under the constraint that any valid combination of $R_{x,\cdot}$ gives $F(K,x)$. These $\{R_{x,i}\}_{i\in[n]}$ can indeed be generated as secret shares of $F(K,x)$.

However, observe that in the case of almost KHPRF, the adversary can distinguish between the two worlds as follows. Let us consider a simple case of 2-out-of-2 sharing of an almost KHPRF key k (known to the adversary) as $k = k_1 + k_2$ without any corruption. On any input x, the adversary is given $F(k_1, x)$ and $F(k_2, x)$ in the real world. In the ideal world, the adversary is given a random secret shares of $F(k,x) : r_1, r_2$ such that $r_1 + r_2 = F(k,x)$. The adversary can distinguish the two worlds by adding the received values - if they add up exactly to $F(k,x)$, then it is the ideal world, else the real world with "good" probability. Hence, in almost KHPRF, we must add noise to hide the error introduced due to homomorphic evaluation.

We further observe that since each addition may add a δ error, the total error introduced due to the homomorphic evaluation of partially evaluated PRF values (let us call it e_p) may actually depend on the secret sharing schemes - in particular, the recovery coefficients. In $\{0,1\}$-LSSS, the recovery co-efficients are binary and hence it is easier to bound the error as $|e_p| \leq \ell$, where ℓ is the number of rows in the share matrix $\mathbf{M}$. In Shamir secret sharing the recovery co-efficients can be arbitrary in $\mathbb{Z}_q$. Hence, in that case one would need the technique of 'clearing the denominators' as in [BGG+18] and modify the game accordingly. In this paper, we work with $\{0,1\}$-LSSS. However, the same ideas work for Shamir secret sharing as well.

Below we state and prove the lemma directly for the general case of almost KHPRF for $\{0,1\}$-LSSS.

Before formally defining the lemma, let us define an intermediate algorithm SS.ShareInt for generating secret shares when some of the secret shares are already set. Thus, $\mathsf{SS.ShareInt}(\mathsf{pp}, T, \{w_j\}_{j\in T}, s)$ takes as input the public parameters, an invalid share set $T \subset [\ell]$ and the corresponding secret shares, $\{w_j\}_{j\in T}$ and the secret s and outputs the secret shares for $[\ell] \setminus T$. We assume that the input shares are consistent in the following sense: for all $j \in T$ such that the row $\mathbf{M}[j] \in \mathsf{Span}(\mathbf{M}_{T\setminus\{j\}})$, that is, $\mathbf{M}[j] = \sum_{\kappa\in T\setminus\{j\}} c_\kappa \mathbf{M}[\kappa]$, where $\{c_\kappa\}_{\kappa\in T\setminus\{j\}}$ are constants, $w_j = \sum_{\kappa\in T\setminus\{j\}} c_\kappa w_\kappa$. The algorithm is defined as follows:

$\mathsf{SS.ShareInt}(T, \{w_j\}_{j\in T}, s, \mathbb{A}_{t,n})$:

- Find a set of *random* values $r_1, \ldots, r_{N-1}$ such that for each $\alpha \in T$, $\langle \mathbf{M}[\alpha], (s, r_1, \ldots, r_{N-1})\rangle = w_\alpha$. These r's are chosen as follows:
  - Let $T_I \subseteq T$ such that $\mathbf{M}_{T_I}$ is maximally independent set of rows within $\mathbf{M}_T$. Let $|T_I| = c$. Further, let $T_{\mathsf{col}} \subseteq [2, N]$ be a set of columns in $\mathbf{M}$ such that the matrix formed by the rows and columns of $\mathbf{M}$ in T_I and T_{col}, respectively is a square full-rank matrix.
  - Sample $r_{j-1} \leftarrow \mathbb{Z}_q$ for all $j \in [2, N] \setminus T_{\mathsf{col}}$.
  - Solve (deterministically) $\mathbf{M}_{T_I}(s, r_1, \ldots, r_{N-1})^\mathsf{T} = w_{T_I}$ to compute $\{r_{j-1}\}_{j\in T_{\mathsf{col}}}$.
  - For each $\alpha \in [\ell] \setminus T$, $w_\alpha = \langle \mathbf{M}_\alpha, (s, r_1, \ldots, r_{N-1})\rangle$

Now we are ready to define the lemma.

Lemma 4. *Let* F *be any secure δ-almost* KHPRF *and* SS *is a secure binary linear secret sharing scheme (Definition 14 and Lemma 3). Then for all* PPT *adversary $\mathcal{A}$, $\Pr[\mathcal{A}$ wins$] \leq 1/2 + \mathsf{neg}(\lambda)$ in the following experiments if $\delta\ell/E_{sm} \leq \mathsf{neg}(\lambda)$, where E_{sm} is flooding noise used to hide the error in homomorphic evaluation of* KHPRF*.*

$\mathsf{Expt}_{\mathcal{A},\mathsf{almost\text{-}KHPRF}}(1^\lambda, \mathbb{A}_{t,n})$*:*

1. *Upon input the security parameter λ and a threshold access structure $\mathbb{A}_{t,n}$, the challenger $\mathcal{C}$ finds a share matrix $\mathbf{M}$ of dimensions $\ell \times N$ along with the n partitions as $(\mathbf{M}, \{T_i\}_{i\in[n]}) \leftarrow \mathsf{bSS.Setup}(1^\lambda, \mathbb{A}_{t,n})$. It sends $\mathsf{bSS.pp} = (\mathbf{M}, \{T_i\}_{i\in[n]})$ to $\mathcal{A}$.*
2. *$\mathcal{C}$ samples a challenge bit, $b \leftarrow \{0,1\}$.*
3. *$\mathcal{A}$ outputs a PRF key K and an invalid share set $T^* \subseteq [\ell]$.*
4. *$\mathcal{C}$ runs $\{k_1, \ldots, k_\ell\} \leftarrow \mathsf{bSS.Share}(K, \mathbb{A}_{t,n})$ and returns $\{k_j\}_{j\in T^*}$ to $\mathcal{A}$.*
5. *Then $\mathcal{A}$ issues polynomial number of evaluation queries of the form (x, j) adaptively, where $j \in [\ell]$ and x is an input to PRF F.*
6. *For each evaluation query (x, j), $\mathcal{C}$ does the following.*

- *Samples* $\eta_{x,j} \leftarrow [-E_{sm}, E_{sm}]$*, where* $\delta\ell/E_{sm} \leq \mathsf{neg}(\lambda)$.
- *If* $b = 0$*, it returns* $y_{x,j} = F(k_j, x) + \eta_{x,j}$.
  Else,
  - *If* $j \in T^*$*, return* $y_{x,j} = F(k_j, x) + \eta_{x,j}$.
  - *Else,* $\mathcal{C}$ *does the following:*
    If x *is queried for the first time (irrespective of any* $j \in [\ell]$*), then it first computes* $y_{x,\alpha} = F(k_\alpha, x)$ *for all* $\alpha \in T^*$ *and runs* $\{y_{x,\alpha}\}_{\alpha\in[\ell]\setminus S^*} \leftarrow \mathsf{bSS.ShareInt}(\mathsf{bSS.pp}, T^*, \{y_{x,\alpha}\}_{\alpha\in T^*}, F(K,x), \mathbb{A}_{t,n})$ *and returns* $y_{x,j} + \eta_{x,j}$*. It saves* $\{y_{x,\alpha}\}_{\alpha\in[\ell]\setminus T^*}$ *for future queries.*
    Else $\mathcal{C}$ *returns the saved* $y_{x,j} + \eta_{x,j}$.

7. $\mathcal{A}$ *outputs* b' *and wins if* $b' = b$.

Note 1.
- Here, we are using a slight abuse of notation - we work directly at the level of party shares in $[\ell]$ without identifying the parties to which they belong. For example, we write $(k_1, \ldots, k_\ell) \leftarrow \mathsf{bSS.Share}(K, \mathbb{A}_{t,n})$ instead of $\{\{k_j\}_{j\in T_i}\}_{i\in[n]} \leftarrow \mathsf{bSS.Share}(K, \mathbb{A}_{t,n})$.
- In the above game, the adversary can issue evaluation queries for multiple party shares together as (T, x), where $T \subseteq [\ell]$.

Proof. We prove the lemma via the following hybrids:

Hybrid_0: This is the real game with $b = 0$.

Hybrid_1: In this hybrid, the challenger does the following:
- After receiving the set T^*, it first fixes a maximal invalid share set $I \subseteq [\ell]$ such that $T^* \subseteq I$.
- For any evaluation query (x, j) such that $j \in I$, the challenger computes and returns $y_{x,j} = y'_{x,j} + \eta_{x,j}$, where $y'_{x,j} = F(k_j, x)$ and $\eta_{x,j}$ is sampled as described in the experiment.
- For any evaluation query (x, j) such that $j \notin I$, the challenger computes $y'_{x,j}$ from $\{y'_{x,\alpha}\}_{\alpha\in I}$ and $F(K, x)$, without using k_j. In more detail, since I is a *maximally* invalid share set, $I \cup \{j\}$ is a valid share set and hence, there exist binary constants $\{b_\alpha\}_{\alpha\in I\cup\{j\}}$[13] such that $K = \sum_{\alpha\in I\cup\{j\}} b_\alpha k_\alpha$. The challenger computes $y'_{x,j} = F(K, x) - \sum_{\alpha\in I} y'_{x,\alpha}$, where $y'_{x,\alpha} = F(k_\alpha, x)$, and returns $y_{x,j} = y'_{x,j} + \eta_{x,j}$, $\forall\ \alpha \in I$,.
  The two hybrids are statistically indistinguishable due to key homomorphism of F and because $\delta\ell/E_{sm} \leq \mathsf{neg}(\lambda)$.
  In more detail, for all input x, for all $j \in I$, $y_{x,j}$ is same in both the hybrids. For $j \in [\ell] \setminus I$,

$$
\begin{aligned}
y_{x,j}^{\mathsf{Hybrid}_1} &= F(K,x) - \sum_{\alpha\in I} b_\alpha F(k_\alpha, x) + \eta_{x,j} \\
&= F((K - \sum_{\alpha\in I} b_\alpha k_\alpha), x) + e_p + \eta_{x,j} \\
&\quad \text{from almost key homomorphism of } F \\
&\quad \text{here } |e_p| \leq \delta(1 + \sum_{\alpha\in I} b_\alpha) \leq \delta(|I| + 1) \leq \delta\ell \\
&= F(k_j, x) + e_p + \eta_{x,j} \\
&\approx_s F(k_j, x) + \eta_{x,j} \quad \text{because } \delta\ell/E_{sm} \leq \mathsf{neg}(\lambda) \\
&= y_{x,j}^{\mathsf{Hybrid}_0}
\end{aligned}
$$

Hybrid_2: This is the same as the previous hybrid, except the following changes: after receiving set T^*, the challenger fixes the set I as in the previous hybrid, and then computes a set $E \subseteq I \setminus T^*$ as follows:
1. Initialize $E = \emptyset$ and $W = T^*$
2. For $j \in I \setminus T^*$,

[13] These binary constants can be different for different valid sets, but we do not indicate that explicitly to keep the notations simple. Also, since I is an invalid set, b_j must be 1.

(a) if $\mathbf{M}[j] \notin \mathsf{Span}(\mathbf{M}_W)$, $E = E \cup \{j\}$, $W = W \cup \{j\}$

We note that all the rows in $\mathbf{M}_E$ are mutually independent and also independent of rows in $\mathbf{M}_{T^*}$. Then for all queried input x, for all $j \in T^* \cup E$, $y'_{x,j} = F(k_j, x)$ (as in the previous hybrid), but for all $j \in I \setminus (E \cup T^*)$, $y'_{x,j}$ is computed indirectly from $\{y'_{x,\alpha}\}_{\alpha \in E \cup T^*}$ using key homomorphism as follows: let $\mathbf{M}[j] = \sum_{\alpha \in E \cup T^*} c_\alpha \mathbf{M}[\alpha]$, for some constants $\{c_\alpha\}_{\alpha \in E \cup T^*}$. Then $y'_{x,j} = \sum_{\alpha \in E \cup T^*} c_\alpha y'_{x,\alpha}$ and $y_{x,j} = y'_{x,j} + \eta_{x,j}$. Since we are working with binary linear secret sharing scheme, $c_\alpha \in \{-1, 0, 1\}$. For all $j \in [\ell] \setminus I$, the queries are answered as in the previous hybrid.

The two hybrids Hybrid_2 and Hybrid_1 are statistically indistinguishable by almost key homomorphism of F and becasue $\delta\ell/E_{sm} \leq \mathsf{neg}(\lambda)$. In more detail, for $j \in I \setminus (E \cup T^*)$, since $\mathbf{M}[j] = \sum_{\alpha \in E \cup S^*} \mathbf{M}[\alpha]$, $k_j = \sum_{\alpha \in E \cup T^*} k_\alpha$ because of the way linear secret sharing is defined. Thus, for $j \in I \setminus (E \cup T^*)$, we get

$$
\begin{aligned}
y_{x,j}^{\mathsf{Hybrid}_2} &= \sum_{\alpha \in E \cup T^*} c_\alpha F(k_\alpha, x) + \eta_{x,j} \\
&= F\Big(\sum_{\alpha \in E \cup T^*} c_\alpha k_\alpha, x\Big) + e_p + \eta_{x,j} \\
&\quad \text{from almost key homomorphism of } F. \\
&\quad \text{here } |e_p| \leq \sum_{\alpha \in E \cup T^*} \delta c_\alpha \leq \delta |E \cup T^*| \leq \delta\ell \\
&= F(k_j, x) + e_p + \eta_{x,j} \\
&\approx_s F(k_j, x) + \eta_{x,j} \quad \text{because } \delta\ell/E_{sm} \leq \mathsf{neg}(\lambda) \\
&= y_{x,j}^{\mathsf{Hybrid}_1}
\end{aligned}
$$

Hybrid_3: In this hybrid the key shares $\{k_\alpha\}_{\alpha \in E \cup T^*}$ are generated using SSSimI (note that by now, we do not need any other PRF key shares) as:

1. $(\{k_j\}_{j \in T^*}, \mathsf{st}') \leftarrow \mathsf{SSSimI}(\mathsf{pp}, \emptyset, \emptyset, T^*, \mathsf{st} = \emptyset)$
2. $(\{k_j\}_{j \in E}, \mathsf{st}'') \leftarrow \mathsf{SSSimI}(\mathsf{pp}, T^*, \{k_\alpha\}_{\alpha \in T^*}, E, \mathsf{st}')$

From the description of the SSSimI and the set E, and the fact that $E \cup T^*$ is an invalid share set, we note that for all $\alpha \in E$, k_α is simply $k_\alpha \leftarrow \mathbb{Z}_q$.

Hybrid_2 and Hybrid_3 are equivalent in the adversary's view from Claim 2.

Hybrid_{3+a}, where $1 \leq a \leq |E|$: Let $E = \{j_1, j_2, \ldots, j_{|E|}\}$. Then in Hybrid_{3+a}, for $1 \leq \kappa \leq a$, y'_{x,j_κ} is chosen uniformly randomly from the range of the PRF F. For $a < \kappa \leq |E|$, y_{x,j_κ} is computed as in the previous hybrid as $y'_{x,j_\kappa} = F(k_{j_\kappa}, x)$.

For $0 \leq a < |E|$, Hybrid_{3+a} and Hybrid_{3+a+1} are computationally indistinguishable from the PRF security of F.

Hybrid_{Last}: This is the honestly played game with $b = 1$. In Claim 3, we show that $\mathsf{Hybrid}_{3+|E|}$ is the same as Hybrid_{Last} in the adversary's view.

Claim 3 $\mathsf{Hybrid}_{3+|E|}$ *is identical to* Hybrid_{Last} *in the adversary's view.*

Proof. The proof is same as the proof of claim 2. That is, again we can explain the distribution of the PRF values generated in $\mathsf{Hybrid}_{3+|E|}$ as those generated by $\mathsf{bSS.ShareInt}$ algorithm (in Hybrid_{Last}). In particular, let $E_{S^*} \subseteq T^*$ be the set of maximally independent rows in $\mathbf{M}_{T^*}$ and $E \subseteq I \setminus T^*$ be as defined in $\mathsf{Hybrid}_{3+|E|}$. Then for each x, $(y'_{x,j})_{j \in [\ell]}$ computed in $\mathsf{Hybrid}_{3+|E|}$ can be written as $\mathbf{M} \cdot (F(K, x), r_{x,1}, \ldots, r_{x,N-1})^\mathsf{T}$, where $r_{x,1}, \ldots, r_{x,N-1}$ have the same distribution as those chosen by the $\mathsf{bSS.ShareInt}$ algorithm in Hybrid_4 - we sample $N - 1 - |E \cup E_{T^*}|$ of the r's unifromly randomly from $\mathbb{Z}_q$. The choice of r's to sample randomly is done in the same way as in the case 2 of the proof of Claim 2. The remaining $|E \cup E_{T^*}|$ r's are then solved deterministically. Since $\{y'_{x,j}\}_{j \in E}$ are uniformly random, the joint distribution of $r_{x,\cdot}$ values will be same as those chosen by the $\mathsf{bSS.ShareInt}$ algorithm.

We also provide a generalization of Claim 2 in appendix A.

5 Threshold Fully Homomorphic Encryption

We first recall the definitions of TFHE from [BGG+18] along with its correctness and security (semantic and simulation) properties. We then define our stronger notion of simulation security needed for constructing UT with stronger security, which in turn is needed for building thresholdized primitives with stronger notion of security.

Definition 20 (Threshold Fully Homomorphic Encryption (TFHE)). *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 fully homomorphic encryption scheme for $\mathbb{S}$ is a tuple of* PPT *algorithms*
$\mathsf{TFHE} = (\mathsf{TFHE.Setup}, \mathsf{TFHE.Encrypt}, \mathsf{TFHE.Eval}, \mathsf{TFHE.PartDec}, \mathsf{TFHE.FinDec})$ *with the following properties:*

- $\mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}) \to (\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n)$*: On input the security parameter λ, a depth bound d, and an access structure $\mathbb{A}$, the setup algorithm outputs a public key* pk*, and a set of secret key shares $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$.*
- $\mathsf{TFHE.Encrypt}(\mathsf{pk}, \mu) \to \mathsf{ct}$*: On input a public key* pk*, and a plaintext $\mu \in \{0,1\}$, the encryption algorithm outputs a ciphertext* ct*.*
- $\mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k) \to \mathsf{ct}$*: On input a public key* pk*, circuit $C : \{0,1\}^k \to \{0,1\}$ of depth at most d, and a set of ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$, the evaluation algorithm outputs a cipehrtext* ct*.*
- $\mathsf{TFHE.PartDec}(\mathsf{pk}, \mathsf{sk}_i, \mathsf{ct}) \to p_i$*: On input a public key* pk*, a secret key share sk_i, and a ciphertext* ct*, the partial decryption algorithm outputs a partial decryption p_i related to the party P_i.*
- $\mathsf{TFHE.FinDec}(\mathsf{pk}, B) \to \mu$*: On input a public key* pk*, and a set $B = \{p_i\}_{i \in S}$ for some $S \subseteq \{P_1, \ldots, P_n\}$, the final decryption algorithm outputs a plaintext $\mu \in \{0, 1, \perp\}$.*

As in a standard FHE *scheme, we require that a* TFHE *scheme satisfies compactness, correctness, and security.*

Definition 21 (Compactness). *A* TFHE *scheme is compact if there exists polynomials $\mathsf{poly}_1(\cdot)$ and $\mathsf{poly}_2(\cdot)$ such that for all λ, depth bound d, circuit $C : \{0,1\}^k \to \{0,1\}$ of depth at most d, and $\mu \in \{0,1\}$, the following holds. For $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}), \{\mathsf{ct}_i \leftarrow \mathsf{TFHE.Encrypt}(\mathsf{pk}, \mu_i)\}_{i \in [k]}$, $\mathsf{ct} \leftarrow \mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k), p_j \leftarrow \mathsf{TFHE.PartDec}(\mathsf{pk}, \mathsf{sk}_j, \mathsf{ct})$ for any $j \in [n]$, $|\mathsf{ct}| \leq \mathsf{poly}(\lambda, d)$ and $|p_j| \leq \mathsf{poly}(\lambda, d, n)$.*

Definition 22 (Evaluation Correctness). *A* TFHE *scheme satisfies evaluation correctness if for all λ, depth bound d, access structure $\mathbb{A} \in \mathbb{S}$, circuit $C : \{0,1\}^k \to \{0,1\}$ of depth at most $d, S \in \mathbb{A}$, and $\mu_i \in \{0,1\}$ for $i \in [k]$, the following condition holds. For $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A})$, $\mathsf{ct}_i \leftarrow \mathsf{TFHE.Encrypt}(\mathsf{pk}, \mu_i)$ for $i \in [k]$, $\mathsf{ct} \leftarrow \mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$,*

$$\Pr[\mathsf{TFHE.FinDec}(\mathsf{pk}, \{\mathsf{TFHE.PartDec}(\mathsf{pk}, \mathsf{sk}_i, \mathsf{ct})\}_{i \in S}) = C(\mu_1, \ldots, \mu_k)] = 1 - \mathit{neg}(\lambda).$$

Definition 23 (Semantic Security). *A* TFHE *scheme satisfies semantic security if for all λ, and depth bound d, the following holds. For any* PPT *adversary $\mathcal{A}$, the following experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{TFHE},\mathsf{sem}}(1^\lambda, 1^d)$ outputs 1 with negligible probability:*
$\mathsf{Expt}_{\mathcal{A},\mathsf{TFHE},\mathsf{sem}}(1^\lambda, 1^d)$*:*

1. *On input the security parameter 1^λ and a circuit depth 1^d, the adversary $\mathcal{A}$ outputs $\mathbb{A} \in \mathbb{S}$.*
2. *The challenger runs $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A})$ and provides* pk *to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs a set $S \subseteq \{P_1, \ldots, P_n\}$ such that $S \notin \mathbb{A}$.*
4. *The challenger provides $\{\mathsf{sk}_i\}_{i \in S}$ along with $\mathsf{TFHE.Encrypt}(\mathsf{pk}, b)$ for $b \leftarrow \{0,1\}$ to $\mathcal{A}$.*
5. *$\mathcal{A}$ outputs a guess b'. The experiment outputs 1 if $b' = b$.*

We now describe the notion of simulation security for TFHE*. Intuitively, simulation security definition says that no information about the key shares or the messages $\mu_1, \ldots, \mu_k$ should be leaked by the partial or final decryption other than what is already implied by the result of the homomorphic operation $C(\mu_1, \ldots, \mu_k)$.*

Definition 24 (Simulation Security [BGG+18]). *A* TFHE *scheme satisfies simulation security if for all λ, depth bound d, and access structure $\mathbb{A}$, the following holds. There exists a stateful* PPT *algorithm $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for any* PPT *adversary $\mathcal{A}$, the following experiments $\mathsf{Expt}^{\mathsf{Real}}_{\mathcal{A},\mathsf{TFHE\text{-}Sim}}(1^\lambda, 1^d)$ and $\mathsf{Expt}^{\mathsf{Ideal}}_{\mathcal{A},\mathsf{TFHE\text{-}Sim}}(1^\lambda, 1^d)$ are indistinguishable:*
$\mathsf{Expt}^{\mathsf{Real}}_{\mathcal{A},\mathsf{TFHE\text{-}Sim}}(1^\lambda, 1^d)$

1. *On input the security parameter 1^λ and a circuit depth 1^d, the adversary $\mathcal{A}$ outputs $\mathbb{A} \in \mathbb{S}$.*
2. *The challenger runs $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A})$ and provides pk to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs a maximal invalid party set $S^* \subseteq \{P_1, \ldots, P_n\}$ and messages $\mu_1, \ldots, \mu_k \in \{0,1\}$.*
4. *The challenger provides the keys $\{\mathsf{sk}_i\}_{i \in S^*}$ and $\{\mathsf{ct}_i \leftarrow \mathsf{TFHE.Encrypt}(\mathsf{pk}, \mu_i)\}_{i \in [k]}$ to $\mathcal{A}$.*
5. *$\mathcal{A}$ issues a polynomial number of adaptive queries of the form $(S \subseteq \{P_1, \ldots, P_n\}, C)$ for circuits $C : \{0,1\}^k \to \{0,1\}$ of depth at most d. For each query, the challenger computes $\mathsf{ct} \leftarrow \mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ and provides $\{\mathsf{TFHE.PartDec}(\mathsf{pk}, \mathsf{sk}_i, \mathsf{ct})\}_{i \in S}$ to $\mathcal{A}$.*
6. *At the end of the experiment, $\mathcal{A}$ outputs a distinguishing bit b.*

$\mathsf{Expt}^{\mathsf{Ideal}}_{\mathcal{A},\mathsf{TFHE\text{-}Sim}}(1^\lambda, 1^d)$

1. *On input the security parameter 1^λ and a circuit depth 1^d, the adversary $\mathcal{A}$ outputs $\mathbb{A} \in \mathbb{S}$.*
2. *The challenger runs $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n, \mathsf{st}) \leftarrow \mathcal{S}_1(1^\lambda, 1^d, \mathbb{A})$ and provides pk to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs a maximal invalid party set $S^* \subseteq \{P_1, \ldots, P_n\}$ and messages $\mu_1, \ldots, \mu_k \in \{0,1\}$.*
4. *The challenger provides the keys $\{\mathsf{sk}_i\}_{i \in S^*}$ and $\{\mathsf{ct}_i \leftarrow \mathsf{TFHE.Encrypt}(\mathsf{pk}, \mu_i)\}_{i \in [k]}$ to $\mathcal{A}$.*
5. *$\mathcal{A}$ issues a polynomial number of adaptive queries of the form $(S \subseteq \{P_1, \ldots, P_n\}, C)$ for circuits $C : \{0,1\}^k \to \{0,1\}$ of depth at most d. For each query, the challenger runs the simulator $\{p_i\}_{i \in S} \leftarrow \mathcal{S}_2(C, \{\mathsf{ct}_1, \ldots, \mathsf{ct}_k\}, C(\mu_1, \ldots, \mu_k), S, \mathsf{st})$ and sends $\{p_i\}_{i \in S}$ to $\mathcal{A}$.*
6. *At the end of the experiment, $\mathcal{A}$ outputs a distinguishing bit b.*

Definition 25 (Stronger Simulation Security). *A* TFHE *scheme satisfies stronger simulation security if for all λ, depth bound d, and access structure $\mathbb{A}$, the following holds. There exists a stateful* PPT *algorithm $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_I, \mathcal{S}_V)$ such that for any* PPT *adversary $\mathcal{A}$, the following experiments, $\mathsf{Expt}^{\mathsf{Real}}_{\mathcal{A},\mathsf{TFHE\text{-}strSim}}(1^\lambda, 1^d)$ and $\mathsf{Expt}^{\mathsf{Ideal}}_{\mathcal{A},\mathsf{TFHE\text{-}strSim}}(1^\lambda, 1^d))$ are indistinguishable:*

$\mathsf{Expt}^{\mathsf{Real}}_{\mathcal{A},\mathsf{TFHE\text{-}strSim}}(1^\lambda, 1^d)$*:*

1. *On input the security parameter 1^λ and a circuit depth 1^d, the adversary $\mathcal{A}$ outputs $\mathbb{A} \in \mathbb{S}$.*
2. *The challenger runs $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A})$ and provides pk to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs an invalid (not necessarily maximal) party set $S^* \subseteq \{P_1, \ldots, P_n\}$ and messages $\mu_1, \ldots, \mu_k \in \{0,1\}$.*
4. *The challenger sends the keys $\{\mathsf{sk}_i\}_{i \in S^*}$ and $\{\mathsf{ct}_i \leftarrow \mathsf{TFHE.Encrypt}(\mathsf{pk}, \mu_i)\}_{i \in [k]}$ to $\mathcal{A}$.*
5. *$\mathcal{A}$ issues a polynomial number of adaptive queries of the form $(S \subseteq \{P_1, \ldots, P_n\}, C)$ for circuits $C : \{0,1\}^k \to \{0,1\}$ of depth at most d. For each query, the challenger computes $\mathsf{ct} \leftarrow \mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ and provides $p_i \leftarrow \{\mathsf{TFHE.PartDec}(\mathsf{pk}, \mathsf{sk}_i, \mathsf{ct})\}_{i \in S}$ to $\mathcal{A}$.*
6. *At the end of the experiment, $\mathcal{A}$ outputs a distinguishing bit b.*

$\mathsf{Expt}^{\mathsf{Ideal}}_{\mathcal{A},\mathsf{TFHE\text{-}strSim}}(1^\lambda, 1^d)$*:*

1. *On input the security parameter 1^λ and a circuit depth 1^d, $\mathcal{A}$ outputs $\mathbb{A} \in \mathbb{S}$.*
2. *The challenger runs $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n, \mathsf{st}) \leftarrow \mathcal{S}_1(1^\lambda, 1^d, \mathbb{A})$ and sends pk to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs an invalid (not necessarily maximal) party set $S^* \subseteq \{P_1, \ldots, P_n\}$ and messages $\mu_1, \ldots, \mu_k \in \{0,1\}$.*
4. *The challenger sends the keys $\{\mathsf{sk}_i\}_{i \in S^*}$ and $\{\mathsf{ct}_i \leftarrow \mathsf{TFHE.Encrypt}(\mathsf{pk}, \mu_i)\}_{i \in [k]}$ to $\mathcal{A}$.*
5. *$\mathcal{A}$ issues a polynomial number of adaptive queries of the form $(S \subseteq \{P_1, \ldots, P_n\}, C)$ for circuits $C : \{0,1\}^k \to \{0,1\}$ of depth at most d. For each query, the challenger provides $\{p_i\}_{i \in S}$ computed as follows:*

– *If C is queried for the first time, then initialize $S_C = \emptyset$ and $\mathsf{st}_C = \mathsf{st}$.*
– *If $S \cup S_C \cup S^*$ is an invalid party set, then*

$$(\mathsf{st}'_C, \{p_i\}_{i\in S}) \leftarrow \mathcal{S}_I(C, \{\mathsf{ct}_1, \ldots, \mathsf{ct}_k\}, S, \mathsf{st}_C)$$

Else, if $S \cup S_C \cup S^*$ is a valid party set, then

$$(\mathsf{st}'_C, \{p_i\}_{i\in S}) \leftarrow \mathcal{S}_V(C, \{\mathsf{ct}_1, \ldots, \mathsf{ct}_k\}, C(\mu_1, \ldots, \mu_k), S, \mathsf{st}_C).$$

– *Update $S_C = S_C \cup S$ and $\mathsf{st}_C = \mathsf{st}'_C$.*

6. *At the end of the experiment, $\mathcal{A}$ outputs a distinguishing bit b.*

Our definition differs from [BGG+18] mainly in the ideal experiment where we define two simulators $\mathcal{S}_I$ and $\mathcal{S}_V$ instead of $\mathcal{S}_2$ in [BGG+18]. Note that $\mathcal{S}_I$ simulates the response for partial evaluation queries (S, C) if S forms an invalid set of parties along with S^* and the sets corresponding to previous queries for the circuit C. The key property of $\mathcal{S}_I$ is that it does not need $C(\mu_1, \ldots, \mu_k)$ as its input. $\mathcal{S}_V$ simulates the response for partial evaluation queries (S, C) when S forms a valid party set (along with S^* and the sets corresponding to previous queries for the circuit C) and takes $C(\mu_1, \ldots, \mu_k)$ as one of its input.

5.1 Construction of Threshold FHE from {0,1}-LSSS

Construction 1 (TFHE) *Let $P = \{P_1, ..., P_n\}$. We use the following building blocks to construct a* TFHE *for P:*

– *A* special *fully homomorphic encryption scheme,* $\mathsf{FHE} = (\mathsf{FHE.Setup}, \mathsf{FHE.Encrypt}, \mathsf{FHE}, \mathsf{Eval}, \mathsf{FHE.Decrypt})$ *with noise bound $B = B(\lambda, d, q)$ and multiplicative constant 1 (Definition 5).*
– *A (δ-almost)* KHPRF*, $F : \mathcal{K} \times \{0,1\}^\lambda \to \mathbb{Z}_q$.*
– *A* $\{0,1\}$-LSSS*, $\mathsf{bSS} = (\mathsf{bSS.Share}, \mathsf{bSS.Combine})$. We use T_i to denote the i-th partition of the share matrix $\mathbf{M}$. We use $= \ell = \ell(\lambda, n)$ to denote a fixed polynomial bound on the size of the share: $|T_i| \le \ell$ for all $i \in [n]$. We let B_{sm} be the bound on the smudging noise to hide the* LWE *error $e \in [-B, B]$ and E_{sm} the bound on the noise added to smudge the error in homomorphic evaluation of* KHPRF*.*
– *A collision resistant hash function $H : \{0,1\}^* \to \{0,1\}^\lambda$.*

$\mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}_{t,n})$**:** *On input the security parameter λ, depth bound d, and threshold access structure $\mathbb{A}_{t,n}$, the setup algorithm does the following:*

1. *Samples* $(\mathsf{fpk}, \mathsf{fsk}) \leftarrow \mathsf{FHE.Setup}(1^\lambda, 1^d)$.
2. *Secret shares* $0 \in \mathcal{K}$ *as* $(K_1, \ldots, K_n) \leftarrow \mathsf{bSS.Share}(0, \mathbb{A}_{t,n})$ *and* fsk *as* $(\mathsf{fsk}_1, \ldots, \mathsf{fsk}_n) \leftarrow \mathsf{bSS.Share}(\mathsf{fsk}, \mathbb{A}_{t,n})$. *We let* $\mathsf{fsk}_i = \{\mathsf{fhesk}_j\}_{j\in T_i}$ *and* $K_i = \{k_j\}_{j\in T_i}$.
3. *Outputs* $\mathsf{tfpk} = \mathsf{fpk}$ *and* $\mathsf{tfsk}_i = (\mathsf{fsk}_i, K_i)$ *for all* $i \in [n]$.

$\mathsf{TFHE.Encrypt}(\mathsf{tfpk}, \mu)$**:** *On input the public key* tfpk *and input* $\mu \in \{0,1\}$*, the encryption algorithm computes and returns* $\mathsf{ct} = \mathsf{FHE.Encrypt}(\mathsf{tfpk}, \mu)$.

$\mathsf{TFHE.Eval}(\mathsf{tfpk}, C, \{\mathsf{ct}_1, \ldots, \mathsf{ct}_k\})$**:** *On input the public key* tfpk*, a circuit* $C : \{0,1\}^k \to \{0,1\}$ *of depth at most d and ciphertexts* $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$*, the evaluation algorithm computes and returns* $\mathsf{ct}_C = \mathsf{FHE.Eval}(\mathsf{tfpk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$.

$\mathsf{TFHE.PartDec}(\mathsf{tfpk}, \mathsf{tfsk}_i, \mathsf{ct})$**:** *On input the public key* tfpk*, a ciphertext* ct *and partial decryption key* tfsk_i*, the partial decryption algorithm does the following.*

1. *Parse* $\mathsf{tfsk}_i = (\{\mathsf{fhesk}_j\}_{j\in T_i}, \{k_j\}_{j\in T_i})$ *and sample* $\{\xi_j\}_{j\in T_i} \leftarrow [-B_{sm}, B_{sm}]$ *and* $\{\eta_j\}_{j\in T_i} \leftarrow [-E_{sm}, E_{sm}]$.
2. *Compute and return* $p_i = \{y_j = \mathsf{FHE.decode}_0(\mathsf{fhesk}_j, \mathsf{ct}) + \xi_j + F(k_j, H(\mathsf{ct})) + \eta_j\}_{j\in T_i}$[14].

$\mathsf{TFHE.FinDec}(\mathsf{tfpk}, S, \{p_i\}_{i\in S})$**:** *On input a public key* tfpk*, a set $S \subseteq [n]$, and a set of partial decryption shares $\{p_i\}_{i\in S}$, it first checks if $S \in \mathbb{A}_{t,n}$. If no, then it outputs $\perp$. Else, it computes and returns*

$$\mu = \mathsf{FHE.decode}_1(\mathsf{bSS.Combine}(\{p_i\}_{i\in S})),$$

which involves following steps: parse $p_i = \{y_j\}_{j\in T_i}$ for each $i \in S$ and compute a minimal valid shares set $T \subseteq \bigcup_{i\in S} T_i$. Then compute $\mathsf{FHE.decode}_1(\sum_{j\in T} y_j)$.

[14] The two smudging noises, ξ_j and η_j can in fact be merged together by appropriately setting the parameters.

Correctness

Theorem 4. *Assume* FHE *is a special FHE (Definition 5) that satisfies correctness with noise bound* B*,* bSS *is a* $\{0,1\}$*-*LSSS *that satisfies correctness and* F *is a* δ*-almost* KHPRF*. Then the above construction of* TFHE *(Construction 1) with parameters* B, B_{sm}, E_{sm} *such that* $B + \ell(\delta + B_{sm} + E_{sm}) \leq q/4$ *(where* $\ell = \mathsf{poly}(\lambda, n)$ *is the number of rows in the share matrix* $\mathbf{M}$*) satisfies evaluation correctness.*

Proof. Let $\mathsf{tfpk}, \mathsf{fpk}, \mathsf{fsk}, \{\mathsf{tfsk}_i\}_{i\in[n]}, \{\mathsf{fhesk}_j, k_j\}_{j\in[\ell]}$ be as defined in the construction. Let $C : \{0,1\}^k \to \{0,1\}$ be a circuit of depth atmost d, and $\mathsf{ct}_i = \mathsf{TFHE}.\mathsf{Encrypt}(\mathsf{fpk}, \mu_i)$, for $\mu_i \in \{0,1\}$, for all $i \in [k]$. Let $\mathsf{ct} = \mathsf{TFHE}.\mathsf{Eval}(\mathsf{fpk}, C, \{\mathsf{ct}_i\}_{i\in[k]})$. Then we need to show that for all $S \in \mathbb{A}_{t,n}$, $\{p_i = \mathsf{TFHE}.\mathsf{PartDec}(\mathsf{fpk}, \mathsf{tfsk}_i, \mathsf{ct})\}_{i\in S}$,

$$\Pr[\mathsf{TFHE}.\mathsf{FinDec}(\mathsf{fpk}, \{p_i\}_{i\in S}) = C(\mu_1, \ldots, \mu_k)] \geq 1 - \mathsf{neg}(\lambda).$$

We have, from the construction, $\mathsf{ct}_i = \mathsf{FHE}.\mathsf{Encrypt}(\mathsf{fpk}, \mu_i)$,

$$\mathsf{ct} = \mathsf{FHE}.\mathsf{Eval}(\mathsf{fpk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k),$$

$$p_i = \{y_j\}_{j\in T_i}, \text{where } y_j = \mathsf{FHE.decode}_0(\mathsf{fhesk}_j, \mathsf{ct}) + \xi_j + F(k_j, H(\mathsf{ct})) + \eta_j,$$

for all $i \in [n]$. For a minimally valid share set $T \subseteq \bigcup_{i\in S} T_i$, we have, from the definition of $\mathsf{bSS.Combine}$,

$$\mathsf{bSS.Combine}(\{p_i\}_{i\in S}) = \sum_{j\in T} y_j.$$

Thus, we have

$$\begin{aligned}
&\mathsf{TFHE}.\mathsf{FinDec}(\mathsf{fpk}, \{p_i\}_{i\in S}) \\
&= \mathsf{FHE.decode}_1(\sum_{j\in T} y_j) \\
&= \mathsf{FHE.decode}_1(\sum_{j\in T}(\mathsf{FHE.decode}_0(\mathsf{fhesk}_j, \mathsf{ct}) + F(k_j, H(\mathsf{ct})) + \xi_j + \eta_j)) \\
&= \mathsf{FHE.decode}_1(\mathsf{FHE.decode}_0(\sum_{j\in T} \mathsf{fhesk}_j, \mathsf{ct}) + F(\sum_{j\in T} k_j, H(\mathsf{ct})) + e_p + \sum_{j\in T} \xi_j + \sum_{j\in T} \eta_j) \\
&= \mathsf{FHE.decode}_1(\mathsf{FHE.decode}_0(\mathsf{fsk}, \mathsf{ct}) + F(0, H(\mathsf{ct})) + e_p + \sum_{j\in T} \xi_j + \sum_{j\in T} \eta_j), \\
&\quad \text{where } |e_p| \leq |T|\delta \leq \ell\delta \\
&= \mathsf{FHE.decode}_1(\lfloor q/2 \rceil C(\mu_1, \ldots, \mu_k) + e + e_p + \sum_{j\in T} \xi_j + \sum_{j\in T} \eta_j) \\
&= C(\mu_1, \ldots, \mu_k)
\end{aligned}$$

Here, the third equation follows from the linearity of $\mathsf{FHE.decode}_0$ and PRF F, e_p is the error due to homomorphic evaluation of *almost* KHPRF and is atmost $\delta\ell$. The fourth equality follows from the correctness of bSS and the sixth equality follows from the correctness of $\mathsf{FHE.decode}_0$. Finally, the correctness follows from the correctness of $\mathsf{FHE.decode}_1$ since $|e + e_p + \sum_{j\in T} \xi_j + \sum_{j\in T} \eta_j| \leq B + |T|(\delta + B_{sm} + E_{sm}) \leq B + \ell(\delta + B_{sm} + E_{sm}) \leq q/4$.

Parameters For security and correctness, we require

- $B + \ell B_{sm} + \delta\ell + \ell E_{sm} \leq q/4$ (for correctness)
- $B/B_{sm} \leq \mathsf{neg}(\lambda)$ and $\delta\ell/E_{sm} \leq \mathsf{neg}(\lambda)$ (for security).

We observe that since ℓ is $\mathsf{poly}(\lambda, n)$ and δ is a constant, our parameters are similar to those in [BGG+18]. As noted there, FHE satisfying these parameters is known from subexponential LWE assumption [GSW13,BV11], which is as hard as approximating the shortest vector with subexponential approximation factors.

5.2 Security and Compactness

Theorem 5. *Assume that* FHE *is a fully homomorphic encryption scheme that satisfies security (Definition 4) and* bSS *is a secure linear secret sharing scheme. Then the above construction of* TFHE *satisfies semantic security.*

Theorem 6. *Assume that* FHE *is a compact fully homomorphic encryption scheme (Definition 3). Then the above construction of* TFHE *satisfies compactness.*

Since the TFHE ciphertext is same as the FHE ciphertext, the above two theorems follow directly from the security of FHE and bSS, and compactness of FHE, respectively.

Simulation security

Theorem 7. *Assume that* FHE *is secure(Definition 4), F is a secure δ-almost* KHPRF *and* bSS *is a secure* $\{0,1\}$-LSSS *and H is collision resistant. Then the above construction of* TFHE *satisfies stronger simulation security (Definition 25).*

Proof. Let $\mathcal{A}$ be any PPT adversary against the stronger simulation security of TFHE. Then we prove the above theorem via the following sequence of hybrid experiments with $\mathcal{A}$.

Hybrid_0 : This is the real experiment $\mathsf{Expt}^{\mathsf{real}}_{\mathcal{A},\mathsf{TFHE\text{-}strSim}}(1^\lambda, 1^d)$. On input the access structure, $\mathbb{A}_{t,n}$, the challenger runs $(\mathsf{tfpk}, \mathsf{tfsk}_1, \ldots, \mathsf{tfsk}_n) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}_{t,n})$ and sends tfpk to $\mathcal{A}$. Then $\mathcal{A}$ outputs a set $S^* \subseteq [n]$ such that $|S^*| < t$ and a set of messages $\mu_1, \ldots, \mu_k \in \{0,1\}$. The challenger then computes $\mathsf{ct}_i = \mathsf{TFHE.Encrypt}(\mathsf{tfpk}, \mu_i)$ for $i \in [k]$ and returns $\{\mathsf{tfsk}_i\}_{i\in S^*}$ and $\{\mathsf{ct}_i\}_{i\in[k]}$ to $\mathcal{A}$. For each evaluation query (S, C) from $\mathcal{A}$, the challenger computes $\mathsf{ct}_C = \mathsf{TFHE.Eval}(\mathsf{tfpk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ and returns the following to $\mathcal{A}$:

$$\{p_{C,i} = \mathsf{TFHE.PartDec}(\mathsf{tfpk}, \mathsf{tfsk}_i, \mathsf{ct}_C)\}_{i\in S}.$$

Each $p_{C,i} = \{y_{C,j}\}_{j\in T_i}$, where $y_{C,j} = \mathsf{FHE.decode}_0(\mathsf{fhesk}_j, \mathsf{ct}_C) + F(k_j, H(\mathsf{ct}_C)) + \xi_{C,j} + \eta_{C,j}$. In the following, for any $X \subseteq [n]$, we let $T_X = \bigcup_{i\in X} T_i$.

Hybrid_1**:** This is the same as the previous hybrid except that for each query (S, C), the PRF component in TFHE.PartDec is replaced with random values. In more detail, for $i \in S \cap S^*$, $p_{C,i}$ is computed as in the real world. For $i \in S \setminus S^*$, $p_{C,i}$ is computed as follows:

- If C is queried for the first time, generate $\{r_{C,j}\}_{j\in T_{[n]\setminus S^*}} \leftarrow \mathsf{bSS.ShareInt}(T_{S^*}, \{F(k_j, C)\}_{j\in T_{S^*}}, 0)$, and save them in a list $\mathcal{L}_C$ for future iterations and queries. Else, lookup for previously saved values of $\{r_{C,j}\}_{j\in T_i}$ in $\mathcal{L}_C$.
- Compute $y_{C,j} = \mathsf{FHE.decode}_0(\mathsf{fhesk}_j, \mathsf{ct}_C) + r_{C,j} + \xi_{C,j} + \eta_{C,j}$ for all $j \in T_i$.

Hybrid_2**:** This is the same as the previous hybrid, except that for each query (S, C), for all $i \in S \setminus S^*$, the PRF components in $p_{C,i}$ are generated from bSS simulators as follows.

- If C is queried for the first time then initialize $S_C = \emptyset$, $\mathsf{st}_C = \emptyset$, $\mathcal{L}_C = \emptyset$.
- If $S^* \cup S_C \cup S$ is an invalid party set then generate $(\{r_{C,j}\}_{j\in T_{S\setminus S^*}}, \mathsf{st}'_C) \leftarrow \mathsf{bSS.SSSimI}(T_{S^*\cup S_C}, \{r_{C,j}\}_{j\in T_{S^*\cup S_C}}, T_{S\setminus S^*}, \mathsf{st}_C$
  Else, if $S^* \cup S_C \cup S$ is a valid party set then generate $(\{r_{C,j}\}_{j\in T_{S\setminus S^*}}, \mathsf{st}'_C) \leftarrow \mathsf{bSS.SSSimV}(T_{S^*\cup S_C}, \{r_{C,j}\}_{j\in T_{S^*\cup S_C}}, 0, T_{S\setminus S}$

  Here, for $j \in T_{S^*}$, $r_{C,j} = F(k_j, H(\mathsf{ct}_C))$ and for $j \in T_{S_C\setminus S^*}$, $r_{C,j}$ is computed during previous queries for C and is stored in $\mathcal{L}_C$.
- Update $S_C = S_C \cup S$ and $\mathsf{st}_C = \mathsf{st}'_C$ and add $\{r_{C,j}\}_{j\in T_{S\setminus S^*}}$ to $\mathcal{L}_C$.

Then compute and return $y_{C,j} = \mathsf{FHE.decode}_0(\mathsf{fhesk}_j, \mathsf{ct}) + r_{C,j} + \xi_{C,j} + \eta_{C,j}$ for all $j \in T_{S\setminus S^*}$.

Hybrid_3**:** This is the same as the previous hybrid, except that for $i \in S \setminus S^*$, $p_{C,i} = \{y_{C,j}\}_{j\in T_i}$ is computed differently as $y_{C,j} = \tilde{r}_{C,j} + \xi_{C,j} + \eta_{C,j}$, where $\tilde{r}_{C,j}$ are generated as follows:

- If C is queried for the first time then initialize $S_C = \emptyset$, $\mathsf{st}_C = \emptyset$, $\mathcal{L}_C = \emptyset$.

– If $S^* \cup S_C \cup S$ is an invalid party set then generate

$$(\{\tilde{r}_{C,j}\}_{j\in T_{S\setminus S^*}}, \mathsf{st}'_C) \leftarrow \mathsf{bSS.SSSimI}(T_{S^*\cup S_C}, \{\tilde{r}_{C,j}\}_{j\in T_{S^*\cup S_C}}, T_{S\setminus S^*}, \mathsf{st}_C).$$

Else, if $S^* \cup S_C \cup S$ is a valid party set then generate

$$(\{\tilde{r}_{C,j}\}_{j\in T_{S\setminus S^*}}, \mathsf{st}'_C) \leftarrow \mathsf{bSS.SSSimV}(T_{S^*\cup S_C}, \{\tilde{r}_{C,j}\}_{j\in T_{S^*\cup S_C}}, \lfloor q/2 \rceil C(\mu_1,\ldots,\mu_k), T_{S\setminus S^*}, \mathsf{st}_C).$$

Here, for $j \in T_{S^*}$, $\tilde{r}_{C,j} = \mathsf{FHE.decode}_0(\mathsf{fhesk}_j, \mathsf{ct}_C) + F(k_j, H(\mathsf{ct}_C))$ and for $j \in T_{S_C\setminus S^*}$, $\tilde{r}_{C,j}$ is computed during previous queries for C and is saved in $\mathcal{L}_C$.

– Update $S_C = S_C \cup S$ and $\mathsf{st}_C = \mathsf{st}'_C$ and save $\{\tilde{r}_{C,j}\}_{j\in T_{S\setminus S^*}}$ in $\mathcal{L}_C$.

Hybrid_4**:** In this hybrid, the challenger secret shares $0^{|\mathsf{fsk}|}$ in place of fsk in the setup phase.

Hybrid_5**:** In this hybrid, ct_i encrypts 0 instead of μ_i.

We observe that the challenger does not use fsk or $\mu_1,\ldots,\mu_k$ to generate the secret key shares $\{\mathsf{fsk}_i\}_{i\in[n]}$ or to reply $\mathsf{PartDec}$ queries. Furthermore, for any queried circuit C, the challenger does not use $C(\mu_1,\ldots,\mu_k)$ as long as the partial decryptions (of ct_C) are generated for an invalid set of parties. Thus the challenger in Hybrid_4 corresponds to the simulator in the ideal experiment, as desired.

Indistinguishability of Hybrids.

$\mathsf{Hybrid}_0 \approx_c \mathsf{Hybrid}_1$**:** We observe that the only difference between the two hybrids is that in Hybrid_0, the PRF component in each partial evaluation is computed honestly using the PRF key share, while in Hybrid_1, for each circuit C, the PRF components are generated using the $\mathsf{ShareInt}$ algorithm. Hence, the indistinguishability between Hybrid_0 and Hybrid_1 follows directly from Lemma 4.

Hybrid_1 **and** Hybrid_2 **are identical in** $\mathcal{A}$**'s view:** The only difference between Hybrid_1 and Hybrid_2 is that in Hybrid_1 the PRF components are genertaed from $\mathsf{ShareInt}$ algorithm, while in Hybrid_2 they are generated using the simulators $\mathsf{bSS.SSSimI}$ and $\mathsf{bSS.SSSimV}$ depending on whether the set S in the query (S, C) forms an invalid or a valid set (along with S^* and the subsets S' in the previous queries of the form (S', C)). The indistinguishability between the two hybrids follows directly from the claim 14.

Claim 8 *Given that* $B/B_{sm} \in \mathsf{neg}(\lambda)$, Hybrid_2 *and* Hybrid_3 *are statistically indistinguishable.*

Proof. We observe that the two hybrids differ only in the way query (S, C) is answered for the honest parties. For any queried circuit C, firstly, let

– S_C be the set of parties for which partial evaluations were queried during the entire experiment.
– I_C be the maximally invalid share set chosen by the $\mathsf{bSS.SSSimV}$ simulator. If $S_C \cup S^*$ is an invalid party set then let $I_C = T_{S_C\cup S^*}$.
– E_C be the set of party shares for which $r_{C,j}$ in Hybrid_2 (resp. $\tilde{r}_{C,j}$ in Hybrid_3) are chosen uniformly randomly from $\mathbb{Z}_q$[15]. From the description of the hybrids, it can be observed that $E_C \subseteq I_C$ and $E_C \cap T_{S^*} = \emptyset$.

Then we make the following observations:

1. For all $j \in T_{S^*}$, $y_{C,j}$ is computed in the same way in both the hybrids.
2. For all $j \in E_C$, $r_{C,j} \leftarrow \mathbb{Z}_q$ in Hybrid_2 (resp. $\tilde{r}_{C,j} \leftarrow \mathbb{Z}_q$ in Hybrid_3). Thus,

$$\begin{aligned} y_{C,j}^{\mathsf{Hybrid}_2} &= \mathsf{FHE.decode}_0(\mathsf{fhesk}_j, \mathsf{ct}_C) + r_{C,j} + \xi_{C,j} + \eta_{C,j} \\ &= \tilde{r}_{C,j} + \xi_{C,j} + \eta_{C,j} \\ &= y_{C,j}^{\mathsf{Hybrid}_3} \end{aligned}$$

In the above, since $r_{C,j}$ is uniformly random, we can replace $\mathsf{FHE.decode}_0(\mathsf{fhesk}_j, \mathsf{ct}_C) + r_{C,j}$ with uniformly random $\tilde{r}_{C,j}$.

[15] Since the choice of E_C and I_C depends only on the order of queries and the share matrix M, they are same in both the hybrids.

3. For all $j \in I_C \setminus (T_{S^*} \cup E_C)$, $r_{C,j} = \sum_{\alpha \in E_C \cup T_{S^*}} c_\alpha r_{C,\alpha}$ in Hybrid_2 (resp. $\tilde{r}_{C,j} = \sum_{\alpha \in E_C \cup T_{S^*}} c_\alpha \tilde{r}_{C,\alpha}$ in Hybrid_3), where $\{c_\alpha\}$ are such that $M[j] = \sum_{\alpha \in E_C \cup T_{S^*}} c_\alpha M[\alpha]$. This also gives us:

$$\mathsf{fhesk}_j = \sum_{\alpha \in E_C \cup T_{S^*}} c_\alpha \mathsf{fhesk}_\alpha.$$

Thus,

$$\begin{aligned}
y_{C,j}^{\mathsf{Hybrid}_2} &= \mathsf{FHE.decode}_0(\mathsf{fhesk}_j, \mathsf{ct}_C) + r_{C,j} + \xi_{C,j} + \eta_{C,j} \\
&= \mathsf{FHE.decode}_0(\mathsf{fhesk}_j, \mathsf{ct}_C) + \sum_{\alpha \in E_C \cup T_{S^*}} c_\alpha r_{C,\alpha} + \xi_{C,j} + \eta_{C,j} \\
&= \mathsf{FHE.decode}_0(\sum_{\alpha \in E_C \cup T_{S^*}} c_\alpha \mathsf{fhesk}_\alpha, \mathsf{ct}_C) + \sum_{\alpha \in E_C \cup T_{S^*}} c_\alpha r_{C,\alpha} + \xi_{C,j} + \eta_{C,j} \\
&= \sum_{\alpha \in E_C \cup T_{S^*}} c_\alpha \mathsf{FHE.decode}_0(\mathsf{fhesk}_\alpha, \mathsf{ct}_C) + \sum_{\alpha \in E_C \cup T_{S^*}} c_\alpha r_{C,\alpha} + \xi_{C,j} + \eta_{C,j} \\
&= \sum_{\alpha \in E_C} c_\alpha (\mathsf{FHE.decode}_0(\mathsf{fhesk}_\alpha, \mathsf{ct}_C) + r_{C,\alpha}) \\
&\quad + \sum_{\alpha \in T_{S^*}} c_\alpha (\mathsf{FHE.decode}_0(\mathsf{fhesk}_\alpha, \mathsf{ct}_C) + r_{C,\alpha}) + \xi_{C,j} + \eta_{C,j} \\
&= \sum_{\alpha \in E_C} c_\alpha \tilde{r}_{C,\alpha} + \sum_{\alpha \in T_{S^*}} c_\alpha \tilde{r}_{C,\alpha} + \xi_{C,j} + \eta_{C,j} \\
&= \sum_{\alpha \in E_C \cup T_{S^*}} c_\alpha \tilde{r}_{C,\alpha} + \xi_{C,j} + \eta_{C,j} \\
&= \tilde{r}_{C,j} + \xi_{C,j} + \eta_{C,j} = y_{C,j}^{\mathsf{Hybrid}_3}
\end{aligned}$$

Here, the third equation follows from the linearity of $\mathsf{FHE.decode}_0$. In the fifth equation, $\mathsf{FHE.decode}_0(\mathsf{fhesk}_\alpha, \mathsf{ct}_C) + r_{C,\alpha} \equiv \tilde{r}_{C,\alpha}$ in E_C, because $r_{C,\alpha}$ is uniformly random for $\alpha \in E_C$ and thus hides $\mathsf{FHE.decode}_0(\mathsf{fhesk}_\alpha, \mathsf{ct}_C)$; for $\alpha \in T_{S^*}$, $\mathsf{FHE.decode}_0(\mathsf{fhesk}_\alpha, \mathsf{ct}_C) + r_{C,\alpha} \equiv \tilde{r}_{C,\alpha}$ by definition.

4. Finally, for $j \in T_{S_C} \setminus I_C$, $r_{C,j}$ in Hybrid_2 (resp. $\tilde{r}_{C,j}$ in Hybrid_3) are computed as $0 - \sum_{\alpha \in I_C} c_\alpha r_{C,\alpha}$[16] (resp. $\tilde{r}_{C,j} = \lfloor q/2 \rceil C(\mu_1, \ldots, \mu_k) - \sum_{\alpha \in I_C} c_\alpha \tilde{r}_{C,\alpha}$ in Hybrid_3), where $\{c_\alpha\}_{\alpha \in I_C}$ are such that $M[j] + \sum_{\alpha \in I_C} c_\alpha M[\alpha] = (1, 0, \ldots, 0)$[17]. Thus, we also have

$$\mathsf{fhesk}_j = \mathsf{fsk} - \sum_{\alpha \in I_C} c_\alpha \mathsf{fhesk}_\alpha.$$

[16] since $F(0, H(\mathsf{ct}_C)) = 0$.

[17] Since I_C is a maximally *invalid* share set and $I_C \cup \{j\}$ is a valid share set, $c_j = 1$

Thus,

$$\begin{aligned}
y_{C,j}^{\mathsf{Hybrid}_2} &= \mathsf{FHE.decode}_0(\mathsf{fhesk}_j, \mathsf{ct}_C) + r_{C,j} + \xi_{C,j} + \eta_{C,j} \\
&= \mathsf{FHE.decode}_0((\mathsf{fsk} - \sum_{\alpha \in I_C} c_\alpha \mathsf{fhesk}_\alpha), \mathsf{ct}_C) - \sum_{\alpha \in I_C} c_\alpha r_{C,\alpha} + \xi_{C,j} + \eta_{C,j} \\
&= \mathsf{FHE.decode}_0(\mathsf{fsk}, \mathsf{ct}_C) - \sum_{\alpha \in I_C} c_\alpha \mathsf{FHE.decode}_0(\mathsf{fhesk}_\alpha), \mathsf{ct}_C) - \sum_{\alpha \in I_C} c_\alpha r_{C,\alpha} + \xi_{C,j} + \eta_{C,j} \\
&= \lfloor q/2 \rceil C(\mu_1, \ldots, \mu_k) + e - \sum_{\alpha \in I_C} c_\alpha (\mathsf{FHE.decode}_0(\mathsf{fhesk}_\alpha), \mathsf{ct}_C) + r_{C,\alpha}) + \xi_{C,j} + \eta_{C,j} \\
&= \lfloor q/2 \rceil C(\mu_1, \ldots, \mu_k) + e - \sum_{\alpha \in I_C} c_\alpha \tilde{r}_{C,\alpha} + \xi_{C,j} + \eta_{C,j} \\
&= \tilde{r}_{C,j} + e + \xi_{C,j} + \eta_{C,j} \\
&\approx_s \tilde{r}_{C,j} + \xi_{C,j} + \eta_{C,j} = y_{C,j}^{\mathsf{Hybrid}_3}
\end{aligned}$$

Here, the fourth equality follows from the correctness of FHE, which gives $\mathsf{FHE.decode}_0(\mathsf{fsk}, \mathsf{ct}_C) = \lfloor q/2 \rceil C(\mu_1, \ldots, \mu_k) + e$, where e is the FHE error and is bounded by B. Finally, the statistical indistinguishability in the last step follows from the smudging lemma 1, since $\xi_{C,j} \leftarrow [-B_{sm}, B_{sm}]$ and $B/B_{sm} \leq \mathsf{neg}(\lambda)$.

Claim 9 *Assume that* bSS *is a secure* $\{0,1\}$-LSSS. *Then* Hybrid_3 *and* Hybrid_4 *are statistically indistinguishable.*

Proof. The two hybrids differ only in the way $\{\mathsf{fhesk}\}_{j \in [\ell]}$ are generated. In Hybrid_3, they are generated as secret shares of fsk, while in Hybrid_4, they are generated as secret shares of $0^{|\mathsf{fsk}|}$. However, we observe that in Hybrid_3 and Hybrid_4, the key shares of fsk corresponding to only invalid party shares are used - in particular, only $\{\mathsf{fhesk}_j\}_{j \in T_{S^*}}$ are used. Hence, from the security of bSS, the two hybrids are statistically indistinguishable.

Claim 10 *Assume that* FHE *is a secure fully homomorphic encryption scheme. Then* Hybrid_4 *and* Hybrid_5 *are computationally indistinguishable.*

Proof. We observe that fsk is no longer used in Hybrid_4. Hence, the indistinguishability between Hybrid_4 and Hybrid_5 follows from FHE security.

Remark 2. In [ASY22], Agrawal *et. al* use Rényi divergence [BLL+15] based analysis to reduce the size of noise flooding in $\mathsf{BGGJKRS}$ threshold signature from exponential to polynomial, effectively reducing the size of modulus q to $\mathsf{poly}(\lambda)$. Rényi divergence is more suitable for search based primitives, like signatures. We remark that using similar analysis as in [ASY22], size of B_{sm} and E_{sm} can also be reduced to polynomial in case of threshold signatures. However, this does not apply for threshold FHE and UT which are indistinguishability based primitives.

6 Universal Thresholdizer

We first recall the definition of universal thresholdizer from [BGG+18]. Then we define our stronger security notion for universal thresholdizer needed to prove stronger notion of security for the primitives thresholdized using it - for example, threshold signatures and threshold CCA-PKE. We refer to [BGG+18] for the definitions of compactness, correctness, (weaker) security and robustness.

Definition 26 (Universal Thresholdizer). *Let* $P = \{P_1, \ldots, P_n\}$ *be a set of parties. A universal thresholdizer scheme for threshold access structure is a tuple of* PPT *algorithms* $\mathsf{UT} = (\mathsf{UT.Setup}, \mathsf{UT.Eval}, \mathsf{UT.Verify}, \mathsf{UT.Combine})$ *with the following properties:*

$\mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}_{t,n}, x) \to (\mathsf{pp}, s_1, \ldots, s_n)$*: On input the security parameter* λ*, a depth bound* d*, an access structure.* $\mathbb{A}_{t,n}$*, and a message* $x \in \{0,1\}^k$*, the setup algorithm outputs the public parameters* pp*, and a set of secret shares* $s_1, \ldots, s_n$*.*

$\mathsf{UT.Eval}(\mathsf{pp}, s_i, C) \to y_i$*: On input the public parameters* pp*, a share* s_i*, and a circuit* $C : \{0,1\}^k \to \{0,1\}$ *of depth at most* d*, the evaluation algorithm outputs a partial evaluation* y_i*.*

$\mathsf{UT.Verify}(\mathsf{pp}, y_i, C) \to \{0,1\}$*: On input the public parameters* pp*, a partial evaluation* y_i*, and a circuit* $C : \{0,1\}^k \to \{0,1\}$*, the verification algorithm accepts or rejects.*

$\mathsf{UT.Combine}(\mathsf{pp}, B) \to y$*: On input the public parameters* pp*, a set of partial evaluations* $B = \{y_i\}_{i \in S}$*, the combining algorithm outputs the final evaluation* y*.*

Definition 27 (UT Stronger Security). *We say that a* UT *scheme satisfies (stronger) security if for all* λ*, and depth bound* d*, the following holds. There exists a stateful* PPT *algorithm* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_I, \mathcal{S}_V)$ *such that for all* PPT *adversary* $\mathcal{A}$*, we have that the following experiments* $\mathsf{Expt}_{\mathcal{A},\mathsf{UT},\mathsf{Real}}(1^\lambda, 1^d)$ *and* $\mathsf{Expt}_{\mathcal{A},\mathsf{UT},\mathsf{Ideal}}(1^\lambda, 1^d)$ *are computationally indistinguishable:*

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

1. *On input the security parameter* 1^λ*, and circuit depth* 1^d*, the adversary* $\mathcal{A}$ *outputs an access structure* $\mathbb{A}_{t,n}$*, and a message* $x \in \{0,1\}^k$*.*
2. *The challenger runs* $(\mathsf{pp}, s_1, \ldots, s_n) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}_{t,n}, x)$ *and provides* pp *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *outputs an invalid (not necessarily maximal) party set* $S^* \subset \{P_1, \ldots, P_n\}$ *for* $\mathbb{A}_{t,n}$*.*
4. *The challenger provides the shares* $\{s_i\}_{i \in S^*}$ *to* $\mathcal{A}$*.*
5. $\mathcal{A}$ *issues a polynomial number of adaptive queries of the form* $(S \subseteq \{P_1, \ldots, P_n\}, C)$ *for circuits* $C : \{0,1\}^k \to \{0,1\}$ *of depth at most* d*. For each query, the challenger provides* $\{y_i \leftarrow \mathsf{UT.Eval}(\mathsf{pp}, s_i, C)\}_{i \in S}$ *to* $\mathcal{A}$*.*
6. *At the end of the experiment,* $\mathcal{A}$ *outputs a distinguishing bit* b*.*

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

1. *On input the security parameter* 1^λ*, and circuit depth* 1^d*, the adversary* $\mathcal{A}$ *outputs an access structure* $\mathbb{A}_{t,n}$*, and a message* $x \in \{0,1\}^k$*.*
2. *The challenger runs* $(\mathsf{pp}, s_1, \ldots, s_n, \mathsf{st}) \leftarrow \mathcal{S}_1(1^\lambda, 1^d, \mathbb{A}_{t,n})$ *and provides* pp *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *outputs an invalid (not necessarily maximal) party set* $S^* \subset \{P_1, \ldots, P_n\}$ *for* $\mathbb{A}_{t,n}$*.*
4. *The challenger provides the shares* $\{s_i\}_{i \in S^*}$ *to* $\mathcal{A}$*.*
5. $\mathcal{A}$ *issues polynomial number of adaptive queries of the form* $(S \subseteq \{P_1, \ldots, P_n\}, C)$ *for circuits* $C : \{0,1\}^k \to \{0,1\}$ *of depth at most* d*. For each query, the challenger provides* $\{y_i\}_{i \in S}$ *computed as follows:*
   - *If* C *is queried for the first time, then initialize* $S_C = S$*, else* $S_C = S_C \cup S$*.*
   - *If* $S_C \cup S^*$ *is an invalid party set, then* $(\{y_i\}_{i \in S}, \mathsf{st}') \leftarrow \mathcal{S}_I(\mathsf{pp}, C, S, \mathsf{st})$*.*
     Else, if $S_C \cup S^*$ *is a valid party set, then* $(\{y_i\}_{i \in S}, \mathsf{st}') \leftarrow \mathcal{S}_V(\mathsf{pp}, C, C(x), S, \mathsf{st})$*.*
   - *Update* $\mathsf{st} = \mathsf{st}'$*.*
6. *At the end of the experiment,* $\mathcal{A}$ *outputs a distinguishing bit* b*.*

Remark 3. The above definition differs from the security definition in [BGG+18] in the ideal experiment. In the weaker notion of [BGG+18], in response to any query (S, C), the UT simulator takes $C(x)$ as input, irrespective of whether or not the set S forms a valid party set along with S^* and sets corresponding to the previous queries for the circuit C. In our definition, we make this distinction - we define two simulators $\mathcal{S}_I$ and $\mathcal{S}_V$. $\mathcal{S}_I$ is used in case of invalid set and does not take $C(x)$ as input. $\mathcal{S}_V$ is used when S forms a valid set (along with S^* and sets in previous queries for C) and takes $C(x)$ as input.

6.1 Construction

The construction for universal thresholdizer with stronger security is the same as that in [BGG+18]. We show that the same construction satisfies our stronger notion of security if the underlying TFHE scheme satisfies strong simulation security. We recall the construction from [BGG+18] and prove its (stronger) security under the assumption that the underlying TFHE satisfies our stronger simulation security. We remark that the proof is very similar to the proof of [BGG+18, Theorem 7.11], except mainly in Hybrid_3, where we use TFHE simulators from our construction. However, we present the entire proof here for completeness.

[BGG+18] Construction of Universal Thresholdizer The construction relies on the following primitives:

- A threshold fully homomorphic encryption scheme $\mathsf{TFHE} = (\mathsf{TFHE.Setup}, \mathsf{TFHE.Encrypt}, \mathsf{TFHE.Eval}, \mathsf{TFHE.PartDec}, \mathsf{TFHE.FinDec})$
- A NIZK with pre-processing scheme $\mathsf{PZK} = (\mathsf{PZK.Pre}, \mathsf{PZK.Prove}, \mathsf{PZK.Verify})$
- A non-interactive commitment scheme $\mathsf{Com} = \mathsf{Com.commit}$.

A universal thresholdizer scheme $\mathsf{UT} = (\mathsf{UT.Setup}, \mathsf{UT.Eval}, \mathsf{UT.Verify}, \mathsf{UT.Combine})$ is constructed as follows:

$\mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}_{t,n}, x) \to (\mathsf{utpp}, \{\mathsf{uts}_i\}_{i\in[n]})$: On input the security parameter λ, depth bound d, access structure $\mathbb{A}_{t,n}$, and message $x = (x_1, \ldots, x_k) \in \{0,1\}^k$, the setup algorithm first generates the TFHE keys $(\mathsf{tfhepk}, \mathsf{tfhesk}_1, \ldots, \mathsf{tfhesk}_n) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}_{t,n})$ and ciphertexts $\mathsf{ct}_i \leftarrow \mathsf{TFHE.Encrypt}(\mathsf{tfhepk}, x_i)$ for $i = 1, \ldots, k$. Then, for all $i \in [n]$, it generates reference strings $(\sigma_{V,i}, \sigma_{P,i}) \leftarrow \mathsf{PZK.Pre}(1^\lambda)$, commitment randomness $r_i \leftarrow \{0,1\}^\lambda$, and commitments $com_i \leftarrow \mathsf{Com.commit}(\mathsf{tfhesk}_i; r_i)$.
Outputs $\mathsf{utpp} = (\mathsf{tfhepk}, \{\mathsf{ct}_i\}_{i\in[k]}, \{\sigma_{V,i}\}_{i\in[n]}, \{com_i\}_{i\in[n]})$, $\mathsf{uts}_i = (\mathsf{tfhesk}_i, \sigma_{P,i}, r_i)$.

$\mathsf{UT.Eval}(\mathsf{utpp}, \mathsf{uts}_i, C) \to y_i = (p_i, \pi_i)$: On input the public parameters utpp, a key share uts_i, and a circuit C, the evaluation algorithm does the following:

- Parses utpp as $(\mathsf{tfhepk}, \{\mathsf{ct}_i\}_{i\in[k]}, \{\sigma_{V,i}\}_{i\in[n]}, \{com_i\}_{i\in[n]})$ and uts_i as $(\mathsf{tfhesk}_i, \sigma_{P,i}, r_i)$.
- Computes the evaluated ciphertext $\mathsf{ct} \leftarrow \mathsf{TFHE.Eval}(\mathsf{tfhepk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ and partial decryption $p_i \leftarrow \mathsf{TFHE.PartDec}(\mathsf{tfhepk}, \mathsf{tfhesk}_i, \mathsf{ct})$.
- Then, it constructs the statement $\psi_i = \psi_i(com_i, \mathsf{ct}, p_i)$ asserting that the value p_i is consistent with the committed secret key tfhesk_i as

$$\exists(\mathsf{tfhesk}_i, r_i) : com_i = \mathsf{Com.commit}(\mathsf{tfhesk}_i; r_i) \wedge p_i = \mathsf{TFHE.PartDec}(\mathsf{tfhepk}, \mathsf{tfhesk}_i, \mathsf{ct}).$$

- It generates a NIZK proof $\pi_i \leftarrow \mathsf{PZK.Prove}(\sigma_{P,i}, \psi_i, (\mathsf{tfhesk}_i, r_i))$ and returns $y_i = (p_i, \pi_i)$.

$\mathsf{UT.Verify}(\mathsf{utpp}, y_i, C) \to \mathsf{accept/reject}$: On input the public parameters utpp, a partial evalaution y_i, and a circuit C, the verification algorithm does the following:

- Parses utpp as $(\mathsf{tfhepk}, \{\mathsf{ct}_i\}_{i\in[k]}, \{\sigma_{V,i}\}_{i\in[n]}, \{com_i\}_{i\in[n]})$.
- Computes the evaluated ciphertext $\mathsf{ct} \leftarrow \mathsf{TFHE.Eval}(\mathsf{tfhepk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$.
- Then, it constructs the statement $\psi_i = \psi_i(com_i, \mathsf{ct}, p_i)$ as described in the Eval algorithm.
- It then parses $y_i = (p_i, \pi_i)$ and returns the result of $\mathsf{PZK.Verify}(\sigma_{V,i}, \psi_i, \pi_i)$.

$\mathsf{UT.Combine}(\mathsf{utpp}, B) \to y$: On input the public parameters $\mathsf{utpp} = (\mathsf{tfhepk}, \{\mathsf{ct}_i\}_{i\in[k]}, \{\sigma_{V,i}\}_{i\in[n]}, \{com_i\}_{i\in[n]})$, and a set of partial evaluations $B = \{y_i\}_{i\in S}$ for some $S \subseteq [n]$, the combining algorithm first parses $y_i = (p_i, \pi_i)$ for $i \in S$ and outputs $y = \mathsf{TFHE.FinDec}(\mathsf{tfhepk}, \{p_i\}_{i\in S})$.

6.2 Security

Theorem 11. *Suppose* TFHE *satisfies semantic security (Definition 23) and stronger simulation security (Definition 25) with simulators (*$\mathsf{TFHE.}\mathcal{S}_1, \mathsf{TFHE.}\mathcal{S}_I, \mathsf{TFHE.}\mathcal{S}_V$*), PZK is a zero knowledge proof system with pre-processing that satisfies zero-knwoeldge (Definition 6), and C is a non-interactive commitment scheme that satisfies computational hiding. Then, the universal thresholdizer scheme by Boneh et. al [BGG+18] satisfies the stronger security (Definition 27).*

Proof. The proof uses the following hybrids. These hybrids are the same as those in the proof of [BGG+18, Theorem 7.11], except Hybrid_3.

Hybrid_0**:** This is the UT real security experiment from Definition 27. We define this hybrid in detail, to set up the notataions to be used in the following hybrids.
On input a threshold access structure $\mathbb{A}_{t,n}$, and a message $x \in \{0,1\}^k$ from $\mathcal{A}$, the challenger

- Runs $\mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}_{t,n}, x)$ by computing the TFHE keys $(\mathsf{tfhepk}, \mathsf{tfhesk}_1, \ldots, \mathsf{tfhesk}_n) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}_{t,n})$, ciphertexts $\{\mathsf{ct}_i \leftarrow \mathsf{TFHE.Encrypt}(\mathsf{tfhepk}, x_i)\}_{i\in[k]}$, $(\sigma_{V,i}, \sigma_{P,i}) \leftarrow \mathsf{PZK.Pre}(1^\lambda)$.

- It then samples $r_i \leftarrow \{0,1\}^\lambda$ and computes $com_i \leftarrow \mathsf{Com.commit}(\mathsf{tfhesk}_i; r_i)$ for $i = 1, \ldots, n$. Sets $\mathsf{utpp} = (\mathsf{tfhepk}, \{ct_i\}_{i\in[k]}, \{\sigma_{V,i}, \sigma_{P,i}\}_{i\in[n]}, \{com_i\}_{i\in[n]})$ and $\{\mathsf{uts}_i = (\mathsf{tfhesk}_i, r_i)\}_{i\in[n]}$.
- Sends utpp to the adversary.

When $\mathcal{A}$ outputs an invalid party set $S^* \subseteq [n]$, the challenger provides $\{\mathsf{uts}_i\}_{i\in S^*}$ to $\mathcal{A}$.

For each evaluation query (S, C) from $\mathcal{A}$, the challenger

- Computes $\mathsf{ct}_C \leftarrow \mathsf{TFHE.Eval}(\mathsf{tfhepk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$,
- Computes $\{p_i = \mathsf{TFHE.PartDec}(\mathsf{tfhesk}_i, \mathsf{ct}_C)\}_{i\in S}$ and $\{\pi_i\}_{i\in S}$ for statements $\{\psi_i\}_{i\in S}$ as defined in the construction using $\mathsf{PZK.Prove}$ algorithm. Sends $y_i = (p_i, \pi_i)$ to $\mathcal{A}$ for $i \in S$.

Hybrid_1**:** This is the same as the previous hybrid, except that for each query (S, C), for each $i \in S$, π_i is computed using the PZK simulator (i.e. without using the witness tfhesk_i).

Hybrid_0 and Hybrid_1 are computationally indistinguishable from the zero knowledge property of PZK.

Hybrid_2**:** This is the same as the previous hybrid except that the com_i in $\mathsf{UT.Setup}$ commits to zero string instead of tfhesk_i for all $i \in [n]$ as $com_i = \mathsf{Com.commit}(0^{|\mathsf{tfhesk}_i|}; r_i)$.

The computational indistinguishablity between Hybrid_1 and Hybrid_2 follows from the computational hiding property of Com.

Hybrid_3**:** In this hybrid the challenger uses TFHE simulators for generating the TFHE components in the $\mathsf{UT.Setup}$ and in computing partial evaluations as follows:

- In the $\mathsf{UT.Setup}$ algorithm, the challenger generates $(\mathsf{tfhepk}, \mathsf{tfhesk}_1, \ldots, \mathsf{tfhesk}_n, \mathsf{st}) \leftarrow \mathsf{TFHE}.\mathcal{S}_1(1^\lambda, 1^d, \mathbb{A}_{t,n})$ instead of running $\mathsf{TFHE.Setup}$. The rest of the $\mathsf{UT.Setup}$ is the same as in Hybrid_2.
- For each query (S, C), for each $i \in S$, p_i is computed using TFHE simulators $\mathsf{TFHE}.\mathcal{S}_I$ and $\mathsf{TFHE}.\mathcal{S}_V$ as follows:
  - If C is queried for the first time, initialize $S_C = \emptyset$, $\mathsf{st}_C = \emptyset$.
  - If $S \cup S_C \cup S^*$ is an invalid party set then the partial evaluations $\{p_i\}_{i\in S}$ are generated using $\mathcal{S}_I$ simulator for TFHE as $(\mathsf{st}'_C, \{p_i\}_{i\in S}) \leftarrow \mathsf{TFHE}.\mathcal{S}_I(C, \{\mathsf{ct}_\kappa\}_{\kappa\in[k]}, S, \mathsf{st}_C)$.
    Else, if $S \cup S_C \cup S^*$ is a valid party set then the partial evaluations $\{p_i^*\}_{i\in S}$ are generated using $\mathcal{S}_V$ simulator for TFHE as $(\mathsf{st}'_C, \{p_i\}_{i\in S}) \leftarrow \mathsf{TFHE}.\mathcal{S}_V(C, \{\mathsf{ct}_\kappa\}_{\kappa\in[k]}, C(x), S, \mathsf{st}_C)$.
  - Update st_C with st'_C, $S_C = S_C \cup S$.

The remaining components are computed as in the previous hybrid.

By the stronger simulation security of TFHE (Definition 25, Hybrid_2 and Hybrid_3 are statistically indistinguishable.

Hybrid_4**:** This is the same hybrid as the previous one, except that in the $\mathsf{UT.Setup}$, $\{\mathsf{ct}_i\}_{i\in[k]}\}$ encrypts zero string instead of x. That is, $\{\mathsf{ct}_i = \mathsf{TFHE.Encrypt}(\mathsf{tfhepk}, 0)\}_{i\in[k]}$.

The (computational) indistinguishability between Hybrid_3 and Hybrid_4 follows from the semantic security of TFHE. In particular, we observe that the $\mathsf{TFHE}.\mathcal{S}_1$ does not use tfhesk to generate the secret shares $\{\mathsf{tfhesk}_1, \ldots, \mathsf{tfhesk}_n\}$. Thus, tfhesk is not used in Hybrid_3 and hence, we can use semantic security of TFHE to argue indistinguishability.

Hybrid_4 is same as the ideal experiment, where the simulators $\mathsf{UT}.\mathcal{S}_1, \mathsf{UT}.\mathcal{S}_I, \mathsf{UT}.\mathcal{S}_V$ are defined as follows:

1. $\mathsf{UT}.\mathcal{S}_1(1^\lambda, 1^d, \mathbb{A}_{t,n})$:
   - Run $(\mathsf{tfhepk}, \{\mathsf{tfhesk}_i\}_{i\in[n]}, \mathsf{st}) \leftarrow \mathsf{TFHE}.\mathcal{S}_1(1^\lambda, 1^d, \mathbb{A}_{t,n})$.
   - Compute $\{\mathsf{ct}_i \leftarrow \mathsf{TFHE.Encrypt}(\mathsf{tfhepk}, 0)\}_{i\in[k]}$,
   - Generates $(\sigma_{V,i}, \sigma_{P,i}) \leftarrow \mathsf{PZK.Pre}(1^\lambda)$ for all $i \in [n]$.
   - It then samples $r_i \leftarrow \{0,1\}^\lambda$ and computes $com_i \leftarrow \mathsf{Com.commit}(0^{|\mathsf{tfhesk}_i|}; r_i)$ for $i = 1, \ldots, n$.
   - Sets $\mathsf{utpp} = (\mathsf{tfhepk}, \{ct_i\}_{i\in[k]}, \{\sigma_{V,i}, \sigma_{P,i}\}_{i\in[n]}, \{com_i\}_{i\in[n]})$ and $\{\mathsf{uts}_i = (\mathsf{tfhesk}_i, r_i)\}_{i\in[n]}$.
2. $\mathsf{UT}.\mathcal{S}_I(\mathsf{utpp}, C, S, \mathsf{st})$ Runs $\mathsf{TFHE}.\mathcal{S}_I(C, \{\mathsf{ct}_i\}_{i\in[k]}, S, \mathsf{st})$.
3. $\mathsf{UT}.\mathcal{S}_I(\mathsf{utpp}, C, C(x), S, \mathsf{st})$ Runs $\mathsf{TFHE}.\mathcal{S}_I(C, \{\mathsf{ct}_i\}_{i\in[k]}, C(x), S, \mathsf{st})$.

7 Applications

In this section we revisit the application of universal thresholdizer in thresholdizing different crypto primitives as considered in [BGG+18]. In particular we define and construct threshold signatures and threshold CCA PKE with stronger security properties. We note that the constructions for these primitives using universal thresholdizer is the same as in [BGG+18]. Our contribution lies in (defining and) proving stronger security for these primitives assuming that the underlying universal thresholdizer satisfies the stronger security as defined in section 6.

7.1 Threshold Signatures

We first recall the definition of threshold signatures and its desired properties. Then we describe selective unforgeability from [BGG+18] and its stronger notion from [BTZ22].

Definition 28 (Threshold Signatures). *Let $P = \{P_1, \ldots, P_n\}$ be a set of n parties. A threshold signature scheme for access structure $\mathbb{A}_{t,n}$ on P is a tuple of PPT algorithms denoted by* $\mathsf{TS} = (\mathsf{TS.KeyGen}, \mathsf{TS.PartSign}, \mathsf{TS.PartSignVerify}, \mathsf{TS.Combine}, \mathsf{TS.Verify})$ *defined as follows:*

- $\mathsf{TS.KeyGen}(1^\lambda, \mathbb{A}_{t,n}) \to (\mathsf{pp}, \mathsf{vk}, \{\mathsf{sk}_i\}_{i=1}^n)$*: On input the security parameter λ and an access structure $\mathbb{A}_{t,n}$, the* KeyGen *algorithm outputs public parameters* pp*, verification key* vk *and a set of key shares* $\{\mathsf{sk}_i\}_{i=1}^n$.
- $\mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m) \to \sigma_i$*: On input the public parameters* pp*, a partial signing key* sk_i *and a message $m \in \{0,1\}^*$ to be signed, the partial signing algorithm outputs a partial signature σ_i.*
- $\mathsf{TS.PartSignVerify}(\mathsf{pp}, m, \sigma_i) \to \mathsf{accept}/\mathsf{reject}$*: On input the public parameters* pp*, a message $m \in \{0,1\}^*$ and a partial signature σ_i, the partial signature verification algorithm outputs* accept *or* reject.
- $\mathsf{TS.Combine}(\mathsf{pp}, \{\sigma_i\}_{i\in S}) \to \sigma_m$*: On input the public parameters* pp *and the partial signatures $\{\sigma_i\}_{i\in S}$ for $S \in \mathbb{A}_{t,n}$, the combining algorithm outputs a full signature σ_m.*
- $\mathsf{TS.Verify}(\mathsf{vk}, m, \sigma_m) \to \mathsf{accept}/\mathsf{reject}$*: On input a verification key* vk*, a message m and a signature σ_m, the verification algorithm outputs accept or reject.*

A TS scheme should satisfy the following requirements.

Definition 29 (Compactness). *A* TS *scheme satisfies compactness if there exist polynomials* $\mathsf{poly}_1(\cdot), \mathsf{poly}_2(\cdot)$ *such that for all λ, for all $\mathbb{A}_{t,n}$ and $S \in \mathbb{A}_{t,n}$, the following holds. For* $(\mathsf{pp}, \mathsf{vk}, \{\mathsf{sk}_i\}_{i=1}^n) \leftarrow \mathsf{TS.KeyGen}(1^\lambda, \mathbb{A}_{t,n})$, $\sigma_i \leftarrow \mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m)$ *for $i \in S$, and* $\sigma \leftarrow \mathsf{TS.Combine}(\mathsf{pp}, \{\sigma_i\}_{i\in S})$*, we have that* $|\sigma| \le \mathsf{poly}_1(\lambda)$ *and* $|\mathsf{vk}| \le \mathsf{poly}_2(\lambda)$.

Definition 30 (Evaluation Correctness). *A signature scheme* TS *satisfies evaluation correctness if forall λ, for all $\mathbb{A}_{t,n}$ and $S \in \mathbb{A}_{t,n}$, the following holds. For* $(\mathsf{pp}, \mathsf{vk}, \{\mathsf{sk}_i\}_{i=1}^n) \leftarrow \mathsf{TS.KeyGen}(1^\lambda, \mathbb{A}_{t,n})$, $\sigma_i \leftarrow \mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m)$ *for $i \in [n]$ and* $\sigma_m \leftarrow \mathsf{TS.Combine}(\mathsf{pp}, \{\sigma_i\}_{i\in S})$*, we have:*

$$\Pr[\mathsf{TS.Verify}(\mathsf{vk}, m, \sigma_m) = \mathsf{accept}] \ge 1 - \mathit{neg}(\lambda).$$

Definition 31 (Partial Verification Correctness). *A signature scheme* TS *satisfies partial verification correctness if for all λ and and for all $\mathbb{A}_{t,n}$, the following holds. For* $(\mathsf{pp}, \mathsf{vk}, \{\mathsf{sk}_i\}_{i=1}^n) \leftarrow \mathsf{TS.KeyGen}(1^\lambda, \mathbb{A}_{t,n})$,

$$\Pr[\mathsf{TS.PartSignVerify}(\mathsf{pp}, m, \mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m)) = \mathsf{accept}] = 1 - \mathit{neg}(\lambda).$$

Definition 32 (Selective Unforgeability [BGG+18]). *A* TS *scheme is unforgeable if for all* PPT *adversary $\mathcal{A}$, the probability of winning in the following experiment,* $\mathsf{Expt}_{\mathcal{A},\mathsf{TS},uf}(1^\lambda)$ *is* $\mathit{neg}(\lambda)$.

1. *On input the security parameter λ and an access structure $\mathbb{A}_{t,n}$, the challenger runs the* $\mathsf{TS.KeyGen}(1^\lambda, \mathbb{A}_{t,n})$ *algorithm and generates public parameters* pp*, verification key* vk *and a set of n key shares* $\{\mathsf{sk}_i\}_{i=1}^n$*. It sends* pp *and* vk *to $\mathcal{A}$.*

2. *$\mathcal{A}$ then outputs a maximally invalid party set $S^* \subset [n]$, i.e. $|S^*| = t-1$, requesting key shares sk_i for $i \in S^*$.*
3. *Challenger provides the set of keys $\{\mathsf{sk}_i\}_{i\in S^*}$ to $\mathcal{A}$.*
4. *Adversary $\mathcal{A}$ issues polynomial number of adaptive queries of the form (m, i), where $i \in [n] \setminus S^*$, to get partial signature σ_i on m. For each query the challenger computes σ_i as $\mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m)$ and provides it to $\mathcal{A}$.*
5. *At the end of the experiment, $\mathcal{A}$ outputs a message-signature pair (m^*, σ^*). The adversary wins if the following conditions hold:*
   (a) m^* was never queried as a signing query.
   (b) $\mathsf{TS.Verify}(\mathsf{vk}, m^*, \sigma^*) = \mathsf{accept}$.

Definition 33 (Stronger Selective Unforgeability [BTZ22]). *In the stronger definition, the set S^* of corrupted parties is not necessarily* maximally *invalid, i.e. $|S^*| < t$. Another and the main difference is in the conditions under which the adversary wins as defined below:*

The adversary is allowed to issue partial signatures on the challenge message m^*, and wins if the following conditions hold:

1. *Let $S_{m^*} = \{i \ : \ (m^*, i)$ was queried as a signing query$\}$. Then $|S^* \cup S_{m^*}| < t$.*
2. *$\mathsf{TS.Verify}(\mathsf{vk}, m^*, \sigma^*) = \mathsf{accept}$.*

Similar to [BGG+18], we construct a threshold signature scheme from a universal thresholdizer and a signature scheme.

Construction 2 (Construction 8.16 in [BGG+18]) *The construction* $\mathsf{TS} = (\mathsf{TS.KeyGen}, \mathsf{TS.PartSign}, \mathsf{TS.PartSignVerify}, \mathsf{TS.Combine}, \mathsf{TS.Verify})$ *uses a signature scheme* $\text{SignScheme} = (\text{SGen}, \mathsf{Sign}, \mathsf{Verify})$ *and a universal thresholdizer* $\mathsf{UT} = (\mathsf{UT.Setup}, \mathsf{UT.Eval}, \mathsf{UT.Verify}, \mathsf{UT.Combine})$.

- $\mathsf{TS.KeyGen}(1^\lambda, \mathbb{A}_{t,n}) \to (\mathsf{pp}, \mathsf{vk}, \{\mathsf{sk}_i\}_{i=1}^n)$. *First it invokes* $\text{SGen}(1^\lambda)$ *to obtain a pair* $(\mathsf{pk}, \mathsf{sk})$ *and it sets* $\mathsf{vk} := \mathsf{pk}$. *Then it runs* $(\mathsf{upp}, \{\mathsf{usk}_i\}_{i=1}^n) \leftarrow \mathsf{UT.Setup}(1^\lambda, \mathbb{A}_{t,n}, \mathsf{sk})$. *Sets* $\mathsf{pp} = \mathsf{upp}$ *and* $\{\mathsf{sk}_i = \mathsf{usk}_i\}_{i=1}^n$.
- $\mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m) \to \sigma_i$. *On input the public parameters* pp*, a partial signing key* sk_i *and a message* m*, the partial signing algorithm outputs* $\mathsf{UT.Eval}(\mathsf{pp}, \mathsf{sk}_i, C_m)$ *where the circuit* C_m *is defined as*

$$C_m(\mathsf{sk}) := \mathsf{Sign}(\mathsf{sk}, m).$$

- $\mathsf{TS.PartSignVerify}(\mathsf{pp}, m, \sigma_i) \to \mathsf{accept}/\mathsf{reject}$. *On input the public parameters* pp*, message* m*, and a partial signature* σ_i*, the partial signature verification algorithm outputs* $\mathsf{UT.Verify}(\mathsf{pp}, \sigma_i, C_m)$.
- $\mathsf{TS.Combine}(\mathsf{pp}, \{\sigma_i\}_{i\in S}) \to \sigma_m$. *On input the public parameters* pp*, and a set of partial signatures* $\{\sigma_i\}_{i\in S}$*, the signature combining algorithm outputs* $\mathsf{UT.Combine}(\mathsf{pp}, \{\sigma_i\}_{i\in S})$.
- $\mathsf{TS.Verify}(\mathsf{vk}, m, \sigma_m) \to \mathsf{accept}/\mathsf{reject}$. *On input the signature verification key* $\mathsf{vk} = \mathsf{pk}$*, a message* m*, and a signature* σ*, the verification algorithm outputs* $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$.

Theorem 12. *If the universal thresholdizer* UT *satisfies the stronger security notion (Definition 27) and* SignScheme *is a signature scheme that satisfies unforgeability, then the construction above (Construction 2) satisfies the Stronger Selective Unforgeability (Definition 33).*

Proof. We prove the above theorem via the following hybrids. We start with Hybrid_0 which is the experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{TS},uf}(1^\lambda)$ for the construction 2.

Hybrid_1**.** Note that since UT is secure with respect to Definition 27, there exists a stateful PPT algorithm $\mathsf{UT}.\mathcal{S} = (\mathsf{UT}.\mathcal{S}_1, \mathsf{UT}.\mathcal{S}_I, \mathsf{UT}.\mathcal{S}_V)$ which can simulate answer to $\mathsf{UT.Eval}(\mathsf{pp}, \mathsf{sk}_i, \cdot)$ queries. We define Hybrid_1 that is similar to Hybrid_0 except that for the challenge queries of the form (m, i) made by $\mathcal{A}$, the challenger uses $\mathsf{UT}.\mathcal{S} = (\mathsf{UT}.\mathcal{S}_1, \mathsf{UT}.\mathcal{S}_I, \mathsf{UT}.\mathcal{S}_V)$ to generate partial signatures. It is straightforward to show these two hybrids are indistinguishable because UT is secure with respect to Definition 27.

Furthermore, we show that the advantage of $\mathcal{A}$ in Hybrid_1 is negligible. Let us assume that the advantage of $\mathcal{A}$ in Hybrid_1 is ϵ. We define a reduction adversary $\mathcal{B}$ to attack the unforgeability of the underlying signature scheme SignScheme = (SGen, Sign, Verify). Namely, the adversary $\mathcal{B}$ after receiving the public key pk from its challenger, sets $\mathsf{vk} = \mathsf{pk}$. It runs $(\mathsf{pp}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n, \mathsf{st}) \leftarrow \mathsf{UT}.\mathcal{S}_1(1^\lambda, 1^d, \mathbb{A}_{t,n})$ and provides pp to $\mathcal{A}$.

- When $\mathcal{A}$ outputs an invalid party set $S^* \subset [n]$, the adversary $\mathcal{B}$ provides the set of keys $\{\mathsf{sk}_i\}_{i \in S^*}$ to $\mathcal{A}$.
- Adversary $\mathcal{A}$ issues polynomial number of adaptive queries of the form (m, i), where $i \in [n] \setminus S^*$ to get partial signature σ_i for m. For each query the adversary $\mathcal{B}$ computes σ_i computed as follows:
  - If m is queried for the first time, then initialize $S_m = \{i\}$, else $S_m = S_m \cup \{i\}$.
  - If $S_m \cup S^*$ is an invalid party set, then $(\sigma_i, \mathsf{st}') \leftarrow \mathsf{UT}.\mathcal{S}_I(\mathsf{pp}, C_m, \{i\}, \mathsf{st})$
  - Else, if $S_m \cup S^*$ is a valid party set, then it queries m to its challenger to receive a signature σ_m on m, and returns $(\sigma_i, \mathsf{st}') \leftarrow \mathsf{UT}.\mathcal{S}_V(\mathsf{pp}, C_m, \sigma_m, \{i\}, \mathsf{st})$.
  - Update $\mathsf{st} = \mathsf{st}'$.
- When at the end of the experiment, $\mathcal{A}$ outputs a message-signature pair (m^*, σ^*), the adversary $\mathcal{B}$ returns (m^*, σ^*) as a forgery for SignScheme.

We observe that (m^*, σ^*) is a valid forgery by $\mathcal{B}$. This is because $|S^* \cup S_{m^*}| < t$ due to validity of $\mathcal{A}$'s forgery. Hence, $\mathcal{B}$ would never have asked a signature on m^* to its challenger to answer any partial signature query on m^* by $\mathcal{A}$. It is clear that if $\mathcal{A}$ wins with ϵ advantage, then the advantage of $\mathcal{B}$ in breaking the unforgeability of SignScheme is also ϵ. This finishes the proof.

7.2 Threshold CCA PKE

We first recall the definition of CCA threshold PKE, correctness and security from [BGG+18]. Then we define a stronger security notion, similar to stronger security of threshold signature.

Definition 34 (CCA Threshold PKE). *CCA threshold PKE is CCA secure PKE in which the decryption key is divided into key shares and distributed to multiple decryption servers. To decrypt a ciphertext, each decryption server creates its own decryption share. These shares can then be publicly combined to get the full decryption.*

Let $P = \{P_1, P_2, \ldots, P_n\}$ be a set of n participants. A CCA threshold PKE scheme for threshold access structure $\mathbb{A}_{t,n}$ on P and message space $\mathcal{M}$ is a tuple of PPT algorithms $\mathsf{TPKE} = (\mathsf{TPKE.KeyGen}, \mathsf{TPKE.Encrypt}, \mathsf{TPKE.PartDec}, \mathsf{TPKE.Combine})$ *defined as follows:*

- $\mathsf{TPKE.KeyGen}(1^\lambda, \mathbb{A}_{t,n}) \rightarrow (\mathsf{pp}, \mathsf{ek}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n)$: *On input the security parameter, λ and an access structure, $\mathbb{A}_{t,n}$, the keygen algorithm outputs the public parameters* pp *and the secret key shares* $\{\mathsf{sk}_i\}_{i=1}^n$
- $\mathsf{TPKE.Encrypt}(\mathsf{ek}, m) \rightarrow \mathsf{ct}$: *The encryption algorithm takes as input a message $m \in \mathcal{M}$ and the encryption key* ek *and outputs a ciphertext* ct.
- $\mathsf{TPKE.PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct}) \rightarrow m_i$:*The partial decryption algorithm takes as input the public parameters,* pp*, decryption key share* sk_i *and a ciphertext* ct *and outputs a partial message m_i*
- $\mathsf{TPKE.Combine}(\mathsf{pp}, \{m_i\}_{i \in S}) \rightarrow m'$:*The combining algorithm takes the public parameters,* pp *and set of partially decrypted messages $\{m_i\}_{i \in S}$ and outputs message m'*

Correctness: *A* TPKE *scheme is said to satisfy decryption correctness if for all λ, for all $\mathbb{A}_{t,n}$ and $S \in \mathbb{A}_{t,n}$, the following holds: for* $(\mathsf{pp}, \mathsf{ek}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{TPKE.KeyGen}(1^\lambda, \mathbb{A}_{t,n})$, $\mathsf{ct} \leftarrow \mathsf{TPKE.Encrypt}(\mathsf{ek}, m)$ *and* $m_i \leftarrow \mathsf{TPKE.PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct})$

$$\Pr[\mathsf{TPKE.Combine}(\mathsf{pp}, \{m_i\}_{i \in S})] \geq 1 - \mathit{neg}(\lambda).$$

Definition 35 (Security [BGG+18]). *A* TPKE *scheme for $\mathbb{A}_{t,n}$ is said to satisfy* CCA *security if for all λ, the following holds: for all* PPT *adversary $\mathcal{A}$, following experiments,* $\mathsf{Expt}^0_{\mathcal{A},\mathsf{TPKE}}(1^\lambda)$ *and* $\mathsf{Expt}^1_{\mathcal{A},\mathsf{TPKE}}(1^\lambda)$ *are computationally indistinguishable.*

$\mathsf{Expt}^b_{\mathcal{A},\mathsf{TPKE}}(1^\lambda)$

1. *On input a security parameter* λ*, and a threshold access structure* $\mathbb{A}_{t,n}$*, the challenger runs* $(\mathsf{pp}, \mathsf{ek}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{TPKE.KeyGen}(1^\lambda, \mathbb{A}_{t,n})$ *and provides* pp *and* ek *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *outputs a maximal invalid party set,* $S^* \subset [n]$*, that is,* $|S^*| = t - 1$*.*
3. *The challenger provides the set of secret keys,* $\{\mathsf{sk}_i\}_{i \in S^*}$ *to* $\mathcal{A}$*.*
4. $\mathcal{A}$ *issues a polynomial number of adaptive decryption queries of the form* (ct, i)*, where* $i \notin S^*$*.*
5. *The challenger computes* $s_i \leftarrow \mathsf{TFHE.PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct})$ *and sends* s_i *to* $\mathcal{A}$*.*
6. $\mathcal{A}$ *outputs a pair of challenge messages* (m_0^*, m_1^*)
7. *The challenger computes* $\mathsf{ct}^* \leftarrow \mathsf{TFHE.Encrypt}(\mathsf{ek}, m_b^*)$ *and sends* ct^* *to* $\mathcal{A}$*.*
8. $\mathcal{A}$ *continues issuing a polynomial number of adaptive decryption queries. However, the adversary is not allowed to issue a decryption query on the challenge ciphertext* ct^**.*
9. *At the end of the experiment,* $\mathcal{A}$ *outputs a guess bit* b'*.*

Similar to the stronger security of threshold signature, we define the stronger security for TPKE, where the adversary is allowed to issue partial decryption query on the challenge ciphertext as well, as long as it is for an invalid set of parties over all such queries.

Definition 36 (Stronger Security TPKE).
In the stronger definition, the set S^* *of corrupted parties is not necessarily* maximally *invalid, i.e.* $|S^*| < t$*. Another, and the main difference is in the admissibility condition for* $\mathcal{A}$ *as follows -* $\mathcal{A}$ *can issue queries of the form* (ct^*, i)*. Let* $S_{\mathsf{ct}^*} = \{i \ : \ \mathcal{A}$ *issued decryption query as* $(\mathsf{ct}^*, i)\}$*. Then,* $S^* \cup S_{\mathsf{ct}^*}$ *must be an invalid set, i.e.* $|S^* \cup S_{\mathsf{ct}^*}| < t$*.*

Similar to [BGG+18], we construct a threshold public-key encryption scheme from a universal thresholdizer and a public-key encryption.

Construction 3 (Construction 8.29 in [BGG+18]) *The construction* $\mathsf{TPKE} = (\mathsf{TPKE.KeyGen}, \mathsf{TPKE.Encrypt}, \mathsf{TPKE.PartDec}, \mathsf{TPKE.Combine})$ *uses a public key encryption scheme* $\mathsf{PKE} = (\mathsf{PKE.Gen}, \mathsf{Enc}, \mathsf{Dec})$ *and a universal thresholdizer* $\mathsf{UT} = (\mathsf{UT.Setup}, \mathsf{UT.Eval}, \mathsf{UT.Verify}, \mathsf{UT.Combine})$*.*

- $\mathsf{TPKE.KeyGen}(1^\lambda, \mathbb{A}_{t,n}) \to (\mathsf{pp}, \mathsf{ek}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n)$*. First it invokes* $\mathsf{PKE.Gen}(1^\lambda)$ *to obtain a pair* $(\mathsf{pk}, \mathsf{sk})$ *and it sets* $\mathsf{ek} := \mathsf{pk}$*. Then it runs* $(\mathsf{upp}, \{\mathsf{usk}_i\}_{i \in [n]}) \leftarrow \mathsf{UT.Setup}(1^\lambda, \mathbb{A}_{t,n}, \mathsf{sk})$*, and sets* $\mathsf{pp} = \mathsf{upp}$ *and* $\{\mathsf{sk}_i = \mathsf{usk}_i\}_{i=1}^n$*.*
- $\mathsf{TPKE.Encrypt}(\mathsf{ek}, m) \to \mathsf{ct}$*. The encryption algorithm takes as input a message* $m \in \mathcal{M}$ *and the encryption key* ek *and outputs a ciphertext* $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$*.*
- $\mathsf{TPKE.PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct}) \to m_i$*. The partial decryption algorithm takes as input the public parameters,* pp*, decryption key share* sk_i *and a ciphertext* ct *and outputs* $m_i \leftarrow \mathsf{UT.Eval}(\mathsf{pp}, \mathsf{sk}_i, C_{\mathsf{ct}})$ *where the circuit* C_{ct} *is defined as*
$$C_{\mathsf{ct}}(\mathsf{sk}) := \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}).$$
- $\mathsf{TPKE.Combine}(\mathsf{pp}, \{m_i\}_{i \in S}) \to m'$*. The combining algorithm takes the public parameters,* pp *and set of partially decrypted messages* $\{m_i\}_{i \in S}$ *and outputs* $m' \leftarrow \mathsf{UT.Combine}(\mathsf{pp}, \{m_i\}_{i \in S})$*.*

Theorem 13. *If the universal thresholdizer* UT *satisfies the stronger security definition (Definition 27) and* PKE *is CCA secure, then the construction above (Construction 3) satisfies the stronger security Definition 36.*

Proof. We prove the above theorem via the following hybrids. For simplicity, we consider a modified but equivalent version of $\mathsf{Expt}^b_{\mathcal{A},\mathsf{TPKE}}(1^\lambda)$ in which the advantage of the adversary is the probability that $b' = b$ when b is chosen uniformly random by the challenger. We start with Hybrid_0 which is the (modified) experiment $\mathsf{Expt}^b_{\mathcal{A},\mathsf{TPKE}}(1^\lambda)$ for the construction 2.

Hybrid_1. Note that since UT is secure with respect to Definition 27, there exists a stateful PPT algorithm $\mathsf{UT}.\mathcal{S} = (\mathsf{UT}.\mathcal{S}_1, \mathsf{UT}.\mathcal{S}_I, \mathsf{UT}.\mathcal{S}_V)$ which can simulate answer to $\mathsf{UT.Eval}(\mathsf{pp}, \mathsf{sk}_i, \cdot)$ queries. We define Hybrid_1 that is similar to Hybrid_0 except for the challenge decryption queries of the form (ct, i) made by $\mathcal{A}$. For

the challenge queries, the challenger uses $\mathsf{UT}.\mathcal{S} = (\mathsf{UT}.\mathcal{S}_1, \mathsf{UT}.\mathcal{S}_I, \mathsf{UT}.\mathcal{S}_V)$ to answer. It is straightforward to show these two hybrids are indistinguishable because UT is secure with respect to Definition 27.

Furthermore, we show that the advantage of $\mathcal{A}$ in Hybrid_1 is at most $1/2 +$ negligible. Let us assume that the advantage of $\mathcal{A}$ in Hybrid_1 is ϵ. We define a reduction adversary $\mathcal{B}$ to attack the CCA security of the underlying public-key encryption scheme $\mathsf{PKE} = (\mathsf{PKE}.\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. Namely, the adversary $\mathcal{B}$ after receiving the public key pk from its challenger, it sets $\mathsf{ek} = \mathsf{pk}$, it runs $(\mathsf{pp}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n, \mathsf{st}) \leftarrow \mathsf{UT}.\mathcal{S}_1(1^\lambda, 1^d, \mathbb{A}_{t,n})$ and provides pp, ek to $\mathcal{A}$.

- When $\mathcal{A}$ outputs an invalid party set $S^* \subseteq [n]$, the adversary $\mathcal{B}$ provides the set of keys $\{\mathsf{sk}_i\}_{i \in S^*}$ to $\mathcal{A}$.
- Adversary $\mathcal{A}$ issues a polynomial number of adaptive decryption queries of the form (ct, i), where $i \notin S^*$. For each query the adversary $\mathcal{B}$ returns m_i computed as follows:
  - If ct is queried for the first time, then initialize $S_{\mathsf{ct}} = \{i\}$, else $S_{\mathsf{ct}} = S_{\mathsf{ct}} \cup \{i\}$.
  - If $S_{\mathsf{ct}} \cup S^*$ is an invalid party set, then $(m_i, \mathsf{st}') \leftarrow \mathsf{UT}.\mathcal{S}_I(\mathsf{pp}, C_{\mathsf{ct}}, \{i\}, \mathsf{st})$.
  - Else, if $S_{\mathsf{ct}} \cup S^*$ is a valid party set, then it queries ct to its challenger to receive a decryption m_{ct}, and $(m_i, \mathsf{st}') \leftarrow \mathsf{UT}.\mathcal{S}_V(\mathsf{pp}, C_{\mathsf{ct}}, m_{\mathsf{ct}}, \{i\}, \mathsf{st})$.
  - Update $\mathsf{st} = \mathsf{st}'$.
- When $\mathcal{A}$ outputs a pair of challenge messages (m_0^*, m_1^*), the adversary $\mathcal{B}$ forwards it to its challenger and receives $\mathsf{ct}^* \leftarrow \mathsf{TFHE}.\mathsf{Encrypt}(\mathsf{ek}, m_b^*)$, where b is the challenge bit of PKE challenger. $\mathcal{B}$ sends ct^* to $\mathcal{A}$.
- $\mathcal{A}$ continues issuing a polynomial number of adaptive decryption queries (ct, i). The adversary $\mathcal{A}$ is allowed to issue a decryption query on the challenge ciphertext ct^* up to the gap between the threshold value and the number of corrupted parties. The adversary $\mathcal{B}$ answers similar to pre-challenge queries above, unless, when $\mathcal{A}$ queries ct^*. For ct^*, the adversary $\mathcal{B}$ stops and returns $\perp$ whenever $S_{\mathsf{ct}^*} \cup S^*$ is a valid party set. Note that as long as $S_{\mathsf{ct}^*} \cup S^*$ is an invalid party set, $\mathcal{B}$ uses $\mathsf{UT}.\mathcal{S}_I$, which does not need the decryption of ct^*.
- When at the end of the experiment, $\mathcal{A}$ outputs a bit b', the adversary $\mathcal{B}$ returns b' as its output.

It is clear that the advantage of $\mathcal{B}$ in breaking the CCA security of PKE is ϵ. This finishes the proof since since by the the CCA security of PKE, $\epsilon \leq 1/2 +$ negligible.

Acknowledgments. Ehsan Ebrahimi is supported by the Luxembourg National Research Fund under the Junior CORE project QSP (C22/IS/17272217/QSP/Ebrahimi).

References

ABV+12. Shweta Agrawal, Xavier Boyen, Vinod Vaikuntanathan, Panagiotis Voulgaris, and Hoeteck Wee. Functional encryption for threshold functions (or fuzzy ibe) from lattices. In *PKC*, volume 7293 of *LNCS*, pages 280–297. Springer,Berlin,Heidelberg, 2012. https://doi.org/10.1007/978-3-642-30057-8_17.

AMMR18. Shashank Agrawal, Payman Mohassel, Pratyay Mukherjee, and Peter Rindal. Dise: Distributed symmetric-key encryption. In *ACM-CCS*, pages 1993–2010. ACM, 2018. https://doi.org/10.1145/3243734.3243774.

ASY22. Shweta Agrawal, Damien Stehlé, and Anshu Yadav. Round-optimal lattice-based threshold signatures, revisited. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:20, 2022. https://doi.org/10.4230/LIPIcs.ICALP.2022.8.

BCK+22. Mihir Bellare, Elizabeth C. Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. Better than advertised security for non-interactive threshold signatures. In *CRYPTO*, volume 13510 of *LNCS*, pages 517–550. Springer, 2022. https://doi.org/10.1007/978-3-031-15985-5_18.

BGG+18. Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter MR Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In *CRYPTO*, volume 10991 of *LNCS*, pages 565–596. Springer, Cham, 2018. https://doi.org/10.1007/978-3-319-96884-1_19.

BLL+15. Shi Bai, Adeline Langlois, Tancrède Lepoint, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: Using the rényi divergence rather than the statistical distance. In *ASIACRYPT*, pages 3–24. Springer Berlin Heidelberg, 2015. https://doi.org/10.1007/978-3-662-48797-6_1.

BLMR13. Dan Boneh, Kevin Lewi, Hart Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In *CRYPTO*, volume 8042 of *LNCS*, pages 410–428. Springer, Berlin, Heidelberg, 2013. https://doi.org/10.1007/978-3-642-40041-4_23.

BLS04. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *J. Cryptol.*, 17(4):297–319, 2004. https://doi.org/10.1007/s00145-004-0314-9.

Bol02. Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *PKC*, pages 31–46. Springer, 2002. https://doi.org/10.1007/3-540-36288-6_3.

Bol03. Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *PKC*, pages 31–46. Springer, 2003. https://doi.org/10.1007/3-540-36288-6_3.

BP14. Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In *CRYPTO*, volume 8616 of *LNCS*, pages 353–370. Springer,Berlin,Heidelberg, 2014. https://doi.org/10.1007/978-3-662-44371-2_20.

BTZ22. Mihir Bellare, Stefano Tessaro, and Chenzhi Zhu. Stronger security for non-interactive threshold signatures: Bls and frost. *Cryptology ePrint Archive*, 2022.

BV11. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106, 2011. https://doi.org/10.1137/120868669.

CCK23. Jung Hee Cheon, Wonhee Cho, and Jiseung Kim. Improved universal thresholdizer from iterative shamir secret sharing. Cryptology ePrint Archive, Paper 2023/545, 2023.

CG99. Ran Canetti and Shafi Goldwasser. An Efficient *Threshold* Public Key Cryptosystem Secure Against Adaptive Chosen Ciphertext Attack. In *EUROCRYPT*, volume 1592 of *LNCS*, pages 90–106. Springer, 1999. https://doi.org/10.1007/3-540-48910-X_7.

DF89. Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In *CRYPTO*, volume 435 of *LNCS*, pages 307–315. Springer, 1989. https://doi.org/10.1007/0-387-34805-0_28.

DOTT21. Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. In *PKC*, pages 99–130, Cham, 2021. Springer International Publishing. https://doi.org/10.1007/978-3-030-75245-3_5.

dPKM+24. Rafael del Pino, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, and Markku-Juhani Saarinen. Threshold raccoon: Practical threshold signatures from standard lattice assumptions. In *EUROCRYPT*, pages 219–248. Springer, 2024. https://doi.org/10.1007/978-3-031-58723-8_8.

DSMP88. Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. *Non-interactive zero-knowledge with preprocessing.* Springer-Verlag, 1988. https://dl.acm.org/doi/10.5555/88314.88963.

GKS24. Kamil Doruk Gur, Jonathan Katz, and Tjerand Silde. Two-round threshold lattice-based signatures from threshold homomorphic encryption. In Markku-Juhani Saarinen and Daniel Smith-Tone, editors, *Post-Quantum Cryptography*, pages 266–300. Springer, 2024. https://doi.org/10.1007/978-3-031-62746-0_12.

GSW13. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, volume 8042 of *LNCS*, pages 75–92. Springer Berlin Heidelberg, 2013. https://doi.org10.1007/978-3-642-40041-4_5.

GWW+13. Yuanju Gan, Lihua Wang, Licheng Wang, Ping Pan, and Yixian Yang. Efficient construction of cca-secure threshold pke based on hashed diffie–hellman assumption. *The Computer Journal*, 56, 2013.

KG20. Chelsea Komlo and Ian Goldberg. FROST: flexible round-optimized schnorr threshold signatures. In *SAC*, LNCS, pages 34–65. Springer, 2020. https://doi.org/10.1007/978-3-030-81652-0_2.

Kim20. Sam Kim. Key-homomorphic pseudorandom functions from lwe with small modulus. In *EUROCRYPT*, volume 12106 of *LNCS*, pages 576–607. Springer,Cham, 2020. https://doi.org/10.1007/978-3-030-45724-2_20.

LS91. Dror Lapidot and Adi Shamir. Publicly verifiable non-interactive zero-knowledge proofs. In *CRYPTO*, pages 353–365. Springer, 1991. https://doi.org/10.1007/3-540-38424-3_26.

MPS+02. Keith M. Martin, Josef Pieprzyk, Reihaneh Safavi-Naini, Huaxiong Wang, and Peter R. Wild. Threshold MACs. In *ICISC*, volume 2587 of *LNCS*, pages 237–252. Springer, 2002. https://doi.org/10.1007/3-540-36552-4_17.

MS23. Daniele Micciancio and Adam Suhl. Simulation-secure threshold PKE from LWE with polynomial modulus. Cryptology ePrint Archive, Paper 2023/1728, 2023.

NPR99. Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and kdcs. In *EUROCRYPT*, volume 1592 of *LNCS*, pages 327–346. Springer,Berlin,Heidelberg, 1999. https://doi.org/10.1007/3-540-48910-X_23.

Sho00. Victor Shoup. Practical threshold signatures. In *EUROCRYPT*, volume 1807 of *LNCS*, pages 207–220. Springer, 2000. https://doi.org/10.1007/3-540-45539-6_15.

A Aditional Lemmas Related to KHPRF

A.1 Scurity lemma for perfect KHPRF

Lemma 5. *Let* F *be any secure* KHPRF *and* SS *is a secure linear secret sharing scheme (Definition 14). Then for all* PPT *adversary* $\mathcal{A}$, $\Pr[\mathcal{A}\ wins] \leq 1/2 + \mathsf{neg}(\lambda)$ *in the following experiment.*

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

1. *The adversary* $\mathcal{A}$ *outputs the threshold access structure* $\mathbb{A}_{t,n}$, *and a* KHPRF *key* K.
2. *The challenger* $\mathcal{C}$ *picks a share matrix* $\mathbf{M}$ *of dimensions* $\ell \times N$ *for* $\mathbb{A}_{t,n}$ *access structure and sends* $\mathsf{SS.pp} = (\mathbf{M}, \{T_i\}_{i\in[n]})$ *to* $\mathcal{A}$.
3. $\mathcal{A}$ *outputs an invalid share set* $T^* \subseteq [\ell]$.
4. $\mathcal{C}$ *runs* $\{k_1, \ldots, k_\ell\} \leftarrow \mathsf{SS.Share}(K, \mathbb{A}_{t,n})$ *and returns* $\{k_\alpha\}_{\alpha\in T^*}$ *to* $\mathcal{A}$. *It also samples a bit* $b \leftarrow \{0,1\}$.
5. $\mathcal{A}$ *issues polynomial number of evaluation queries of the form* (j, x) *adaptively, where* $j \in [\ell]$ *and* x *is an input to PRF* F.
   - *If* $b = 0$, $\mathcal{C}$ *returns* $y_{x,j} = F(k_j, x)$.
   - *Else (i.e.* $b = 1$*),*
     - *if* $j \in T^*$, *return* $y_{x,j} = F(k_j, x)$.
     - *else,* $\mathcal{C}$ *does the following:*
       * *if* x *is queried for the first time (for any party share* $j \in [\ell]$*), then it first computes* $y_{x,\alpha} = F(k_\alpha, x)$ *for all* $\alpha \in T^*$ *and runs* $\{y_{x,\alpha}\}_{\alpha\in[\ell]\setminus S} \leftarrow \mathsf{SS.ShareInt}(\mathsf{SS.pp}, T^*, \{y_{x,\alpha}\}_{\alpha\in T^*}, F(K, x), \mathbb{A}_{t,n})$. *Returns* $y_{x,j}$ *and saves* $\{y_{x,\alpha}\}_{\alpha\in[\ell]\setminus T^*}$ *for future queries.*
       * *else* $\mathcal{C}$ *returns the saved* $y_{x,j}$.
6. $\mathcal{A}$ *outputs* b' *and wins if* $b' = b$.

Note 2.

– Here, we are using a slight abuse of notation - we work directly at the level of party shares in $[\ell]$ without identifying the parties to which they belong. For example, we write $(k_1, \ldots, k_\ell) \leftarrow \mathsf{SS.Share}(K, \mathbb{A}_{t,n})$ instead of $\{\{k_j\}_{j\in T_i}\}_{i\in[n]} \leftarrow \mathsf{SS.Share}(K, \mathbb{A}_{t,n})$.
– In the above game, the adversary can issue evaluation queries for multiple party shares together as (T, x), where $T \subseteq [\ell]$.

A.2 Generalization of claim 2.

Claim 14 *For all adversary $\mathcal{A}$, $\mathit{Expt}^0_{\mathcal{A},\mathit{SSIntSim}}$ and $\mathit{Expt}^1_{\mathcal{A},\mathit{SSIntSim}}$, as described below are identical in $\mathcal{A}$'s view.*
$\underline{\mathit{Expt}^b_{\mathcal{A},\mathit{SSIntSim}}(1^\lambda)}$

1. *Upon input a threshold access structure $\mathbb{A}_{t,n}$, the challenger runs $(\mathbf{M}, \{T_i\}_{i\in[n]}) \leftarrow \mathsf{SS.Setup}(1^\lambda, \mathbb{A}_{t,n})$ and sends $\mathsf{pp} = (\mathbf{M}, \{T_i\}_{i\in[n]})$ to $\mathcal{A}$.*
2. *$\mathcal{A}$ outputs a secret s, an invalid party-share set $T \subseteq [\ell]$ along with the secret shares $\{w_j\}_{j\in S}$, already fixed by $\mathcal{A}$ for party-shares in T.*
3. *The challenger does the following:*
   – *If $b = 0$, generates $\{w_j\}_{j\in[\ell]} \leftarrow \mathsf{SS.ShareInt}(\mathsf{pp}, T, \{w_j\}_{j\in T}, s)$*
   – *Else if $b = 1$, initializes $W = T$ and $\mathsf{st} = \emptyset$.*
4. *For $\kappa = 1$ to $\ell - |T|$,*
   (a) *$\mathcal{A}$ outputs $j_\kappa \in [\ell]$ (wlog we assume that j_κ was not queried before and is not in T. Otherwise, the challenger returns the previously set value).*
   (b) *The challenger replies as following:*
      – *If $b = 0$, the challenger returns w_{j_κ} (generated in Step 3)*
      – *If $b = 1$ and $W \cup \{j_\kappa\}$ is an invalid share set, generate $(w_{j_\kappa}, \mathsf{st}') \leftarrow \mathsf{SSSimI}(\mathsf{pp}, W, \{w_\alpha\}_{\alpha\in W}, \{j_\kappa\}, \mathsf{st})$; updates $W = W \cup \{j_\kappa\}$, $\mathsf{st} = \mathsf{st}'$ and returns w_{j_κ}*
      – *If $b = 1$ and $W \cup \{j_\kappa\}$ is a valid share set, generate $(w_{j_\kappa}, \mathsf{st}') \leftarrow \mathsf{SSSimV}(\mathsf{pp}, W, \{w_\alpha\}_{\alpha\in W}, \{j_\kappa\}, s, \mathsf{st})$; updates $W = W \cup \{j_\kappa\}$, $\mathsf{st} = \mathsf{st}'$ and returns w_{j_κ}*
5. *$\mathcal{A}$ outputs its guess bit b' and wins if $b' = b$.*

Proof. The proof is same as that for claims 2 and 3 and hence omitted.

B TFHE for Stronger Simulation Security with Partially Adaptive Key Queries

In this section we combine our technique of using PRF for stronger simulation security with the technique in [ASY22] to achieve partially adaptive key queries in the random oracle model.

B.1 Definition

We first define the security definition in this setting.

Definition 37 (Partially Adaptive Stronger Simulation Security). *A TFHE scheme satisfies partially adaptive stronger simulation security if for all λ, depth bound d, and access structure $\mathbb{A}$, the following holds. There exists a stateful PPT algorithm $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_I, \mathcal{S}_V)$ such that for any PPT adversary $\mathcal{A}$, the following experiments $\mathsf{Expt}^{\mathsf{Real}}_{\mathcal{A},\mathsf{TFHE\text{-}partAdapt\text{-}strSim}}(1^\lambda, 1^d)$ and $\mathsf{Expt}^{\mathsf{Ideal}}_{\mathcal{A},\mathsf{TFHE\text{-}partAdapt\text{-}strSim}}(1^\lambda, 1^d)d)$ are indistinguishable:*
$\mathsf{Expt}^{\mathsf{Real}}_{\mathcal{A},\mathsf{TFHE\text{-}partAdapt\text{-}strSim}}(1^\lambda, 1^d)$

1. *On input the security parameter 1^λ and a circuit depth 1^d, the adversary $\mathcal{A}$ outputs $\mathbb{A} \in \mathbb{S}$.*
2. *The challenger runs $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A})$ and provides pk to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs messages $\mu_1, \ldots, \mu_k \in \{0, 1\}$.*
4. *The challenger provides $\{p_i \leftarrow \mathsf{TFHE.Encrypt}(\mathsf{pk}, \mu_i)\}_{i\in[k]}$ to $\mathcal{A}$.*

5. $\mathcal{A}$ *issues a polynomial number of adaptive* PartDec *queries of the form* $(S \subseteq \{P_1, \ldots, P_n\}, C)$ *for circuits* $C : \{0,1\}^k \to \{0,1\}$ *of depth at most* d. *For each query, the challenger computes* $\mathsf{ct}_C \leftarrow \mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ *and provides* $\mathsf{ct}_i \leftarrow \{\mathsf{TFHE.PartDec}(\mathsf{pk}, \mathsf{sk}_i, \mathsf{ct}_C)\}_{i \in S}$ *to* $\mathcal{A}$.
6. $\mathcal{A}$ *outputs an invalid (not necessarily maximal) party set* $S^* \subseteq \{P_1, \ldots, P_n\}$.
7. *The challenger provides the keys* $\{\mathsf{sk}_i\}_{i \in S^*}$.
8. $\mathcal{A}$ *continues to issue more* PartDec *queries, to which the challenger responds as before.*
9. *At the end of the experiment,* $\mathcal{A}$ *outputs a distinguishing bit* b.

$\mathsf{Expt}^{\mathsf{Ideal}}_{\mathcal{A},\mathsf{TFHE\text{-}partAdapt\text{-}strSim}}(1^\lambda, 1^d)$

1. *On input the security parameter* 1^λ *and a circuit depth* 1^d, *the adversary* $\mathcal{A}$ *outputs* $\mathbb{A} \in \mathbb{S}$.
2. *The challenger runs* $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathcal{S}_1(1^\lambda, 1^d, \mathbb{A})$ *and provides* pk *to* $\mathcal{A}$.
3. $\mathcal{A}$ *outputs messages* $\mu_1, \ldots, \mu_k \in \{0,1\}$.
4. *The challenger provides* $\{\mathsf{ct}_i \leftarrow \mathsf{TFHE.Encrypt}(\mathsf{pk}, \mu_i)\}_{i \in [k]}$ *to* $\mathcal{A}$.
5. $\mathcal{A}$ *issues a polynomial number of adaptive* PartDec *queries of the form* $(S \subseteq \{P_1, \ldots, P_n\}, C)$ *for circuits* $C : \{0,1\}^k \to \{0,1\}$ *of depth at most* d. *For each query, the challenger provides* $\{p_i\}_{i \in S}$ *computed as follows:*
   - *If* C *is queried for the first time, then initialize* $S_C = \emptyset$.
   - *If* $S \cup S_C$ *is an invalid party set, then* $(\mathsf{st}', \{p_i\}_{i \in S}) \leftarrow \mathcal{S}_I(C, \{\mathsf{ct}_1, \ldots, \mathsf{ct}_k\}, S, \mathsf{st})$
     Else, if $S \cup S_C$ *is a valid party set, then*
     $(\mathsf{st}', \{p_i\}_{i \in S}) \leftarrow \mathcal{S}_V(C, \{\mathsf{ct}_1, \ldots, \mathsf{ct}_k\}, C(\mu_1, \ldots, \mu_k), S, \mathsf{st})$.
   - *Update* $S_C = S_C \cup S$ *and* $\mathsf{st} = \mathsf{st}'$.
6. $\mathcal{A}$ *outputs an invalid (not necessarily maximal) party set* $S^* \subseteq \{P_1, \ldots, P_n\}$.
7. *The challenger provides the keys* $\{\mathsf{sk}_i\}_{i \in S^*}$.
8. $\mathcal{A}$ *continues to issue more* PartDec *queries. The challenger computes the response using the simulators as before - i.e., use* $\mathcal{S}_I$ *if* $S_C \cup S \cup S^*$ *is invalid, else use* $\mathcal{S}_V$.
9. *At the end of the experiment,* $\mathcal{A}$ *outputs a distinguishing bit* b.

We now construct a TFHE with above security.

B.2 Construction

Construction 4 (Partially Adaptive Stronger TFHE) *Our construction for* t*-out-of-*n *threshold access structure uses the same building blocks as in Construction 1. In addition, it also uses a keyed hash function* $H_1 : \{0,1\}^\lambda \times \{0,1\}^* \to \mathbb{Z}_q^\ell$, *where* ℓ *is the number of rows in the share matrix* $\mathbf{M}$.

$\mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}_{t,n})$**:** *On input the security parameter* λ, *depth bound* d, *and threshold access structure* $\mathbb{A}_{t,n}$, *the setup algorithm does the following:*
1. *Samples a random key for* H_1 *as* $R \leftarrow \{0,1\}^\lambda$; *secret shares* $\mathbf{0} \in \mathbb{Z}_q^\ell$ *as* $(\mathsf{rk}_1, \ldots \mathsf{rk}_n) \leftarrow \mathsf{bSS.Share}(0^\ell, \mathbb{A}_{t,n})$. *Here, each* $\mathsf{rk}_i = \{\mathbf{v}_j\}_{j \in T_i}$.
2. *Sample* $(\mathsf{fpk}, \mathsf{fsk}) \leftarrow \mathsf{FHE.Setup}(1^\lambda, 1^d)$.
3. *Secret share* $0 \in \mathcal{K}$ *as* $(K_1, \ldots, K_n) \leftarrow \mathsf{bSS.Share}(0, \mathbb{A}_{t,n})$ *and* fsk *as* $(\mathsf{fsk}_1, \ldots, \mathsf{fsk}_n) \leftarrow \mathsf{bSS.Share}(\mathsf{fsk}, \mathbb{A}_{t,n})$. *We let* $\mathsf{fsk}_i = \{\mathsf{fhesk}_j\}_{j \in T_i}$ *and* $K_i = \{k_j\}_{j \in T_i}$.
4. *Outputs* $\mathsf{tfpk} = \mathsf{fpk}$ *and* $\mathsf{tfsk}_i = (\mathsf{fsk}_i, K_i, \mathsf{rk}_i, R)$ *for all* $i \in [n]$.

$\mathsf{TFHE.Encrypt}(\mathsf{tfpk}, \mu)$**:** *On input the public key* tfpk *and input* $\mu \in \{0,1\}$, *the encryption algorithm computes and returns* $\mathsf{ct} = \mathsf{FHE.Encrypt}(\mathsf{tfpk}, \mu)$.

$\mathsf{TFHE.Eval}(\mathsf{tfpk}, C, \{\mathsf{ct}_1, \ldots, \mathsf{ct}_k\})$**:** *On input the public key* tfpk, *a circuit* $C : \{0,1\}^k \to \{0,1\}$ *of depth at most* d *and ciphertexts* $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$, *the evaluation algorithm computes and returns* $\mathsf{ct}_C = \mathsf{FHE.Eval}(\mathsf{tfpk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$.

$\mathsf{TFHE.PartDec}(\mathsf{tfpk}, \mathsf{tfsk}_i, \mathsf{ct})$**:** *On input the public key* tfpk, *a ciphertext* ct *and partial decryption key* tfsk_i, *the partial decryption algorithm does the following.*
1. *Parse* $\mathsf{tfsk}_i = (\{\mathsf{fhesk}_j\}_{j \in T_i}, \{k_j\}_{j \in T_i}$ *and sample* $\{\xi_j\}_{j \in T_i} \leftarrow [-B_{sm}, B_{sm}]$ *and* $\{\eta_j\}_{j \in T_i} \leftarrow [-E_{sm}, E_{sm}]$.

2. Compute and return $p_i = \{y_j = \mathsf{FHE.decode}_0(\mathsf{fhesk}_j, \mathsf{ct}) + \xi_j + H_1(R, \mathsf{ct})^\mathsf{T}\mathbf{v}_j + F(k_j, H(\mathsf{ct})) + \eta_j\}_{j\in T_i}$[18]

$\mathsf{TFHE.FinDec}(\mathsf{tfpk}, S, \{p_i\}_{i\in S})$**:** *On input a public key* tfpk*, a set* $S \subseteq [n]$*, and a set of partial decryption shares* $\{p_i\}_{i\in S}$*, it first checks if* $S \in \mathbb{A}_{t,n}$*. If no, then it outputs* $\perp$*. Else, it computes and returns*

$$\mu = \mathsf{FHE.decode}_1(\mathsf{bSS.Combine}(\{p_i\}_{i\in S}))$$

which involves following steps: parse $p_i = \{y_j\}_{j\in T_i}$ *for each* $i \in S$ *and compute a minimal valid shares set* $T \subseteq \bigcup_{i\in S} T_i$*. Then compute* $\mathsf{FHE.decode}_1(\sum_{j\in T} y_j)$*.*

The correctness is same as the correctness of Construction 1, due to the observation that for any valid set S, $\mathsf{bSS.Combine}(\{H_1(R, \mathsf{ct})\mathsf{rk}_i\}_{i\in S} = H_1(R, \mathsf{ct})\mathsf{bSS.Combine}(\{\mathsf{rk}_i\}_{i\in S} = H_1(R, \mathsf{ct}) \cdot 0^\ell = 0$. The second last equality follows from the fact that $\{\mathsf{rk}_i\}_{i\in[n]}$ are secret shares of 0 vector.

Security We prove the security via the following theorem.

Theorem 15. *Assume that* FHE *is secure (Definition 4),* F *is a secure* KHPRF *and* bSS *is a secure* $\{0,1\}$*-*LSSS*,* H *is collision resistant and* H_1 *is modeled as random oracle. Then the above construction of* TFHE *satisfies stronger simulation security with partially adaptive key queries(Definition 37).*

Proof. The theorem can be proved by combining the steps from proof of [ASY22, Theorem 5.1] and Theeorem 7. We provide an overview of the proof here. The proof is given via the following hybrids.

Hybrid_0**:** This is the real experiment.

- On input the access structure, $\mathbb{A}_{t,n}$, the challenger runs $(\mathsf{tfpk}, \mathsf{tfsk}_1, \ldots, \mathsf{tfsk}_n) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}_{t,n})$ and sends tfpk to $\mathcal{A}$.
- $\mathcal{A}$ outputs a set of messages $\mu_1, \ldots, \mu_k \in \{0,1\}$. The challenger then computes $\mathsf{ct}_i = \mathsf{TFHE.Encrypt}(\mathsf{tfpk}, \mu_i)$ for $i \in [k]$ and returns $\{\mathsf{ct}_i\}_{i\in[k]}$ to $\mathcal{A}$.
- $\mathcal{A}$ issues polynomially many PartDec queries of the form (S, C).
  For each query (S, C), the challenger computes $\mathsf{ct}_C = \mathsf{TFHE.Eval}(\mathsf{tfpk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ and returns $\{p_{C,i} = \mathsf{TFHE.PartDec}(\mathsf{tfpk}, \mathsf{tfsk}_i, \mathsf{ct}_C)\}_{i\in S}$ to $\mathcal{A}$.
  Each $p_{C,i} = \{y_{C,j}\}_{j\in T_i}$, where $y_{C,j} = \mathsf{FHE.decode}_0(\mathsf{fhesk}_j, \mathsf{ct}_C) + H_1(R, \mathsf{ct}_C)^\mathsf{T}\mathbf{v}_j + F(k_j, H(\mathsf{ct}_C)) + \xi_{C,j} + \eta_{C,j}$.
- $\mathcal{A}$ outputs a set $S^* \subseteq [n]$ such that $|S^*| < t$. The challenger returns $\{\mathsf{tfsk}_i\}_{i\in S^*}$.
- $\mathcal{A}$ continues to query more partial decryptions, to which the challenger replies honestly as before.
- Whenever $\mathcal{A}$ issues a query (x, y) to the oracle for H_1, the challenger samples a random value r from $\mathbb{Z}_q^\ell$, sets and returns $H_1(x, y) = r$. If a query is repeated, the challenger returns the already set value.

Hybrid_1**:** Let us call the partial decryption queries before the key queries as pre-queries and those after as post queries. Then, Hybrid_1 differes from Hybrid_0 in the following:

- for each pre query (S, C), the challenger computes $p_{C,i} = \{y_{C,j}\}_{j\in T_i}$ differently as $y_{C,j} = \tilde{r}_{C,j} + \xi_{C,j} + \eta_{C,j}$, where $\tilde{r}_{C,j}$ are generated as follows:
  - If C is queried for the first time then initialize $S_C = \emptyset$, $\mathsf{st}_C = \emptyset$, $\mathcal{L}_C = \emptyset$.
  - If $S_C \cup S$ is an invalid party set then generate

    $$(\{\tilde{r}_{C,j}\}_{j\in T_S}, \mathsf{st}'_C) \leftarrow \mathsf{bSS.SSSimI}(T_{S_C}, \{\tilde{r}_{C,j}\}_{j\in T_{S_C}}, T_S, \mathsf{st}_C).$$

    Else, if $S_C \cup S$ is a valid party set then generate

    $$(\{\tilde{r}_{C,j}\}_{j\in T_S}, \mathsf{st}'_C) \leftarrow \mathsf{bSS.SSSimV}(T_{S_C}, \{\tilde{r}_{C,j}\}_{j\in T_{S_C}}, T_S, \lfloor q/2 \rceil C(\mu_1, \ldots, C_k), \mathsf{st}_C).$$

    Here, for $j \in T_{S_C}$, $\tilde{r}_{C,j}$ is computed during previous queries for C and is saved in $\mathcal{L}_C$.
  - Update $S_C = S_C \cup S$ and $\mathsf{st}_C = \mathsf{st}'_C$ and save $\{\tilde{r}_{C,j}\}_{j\in T_S}$ in $\mathcal{L}_C$.
- When $\mathcal{A}$ issues key queries for a set S^*, the challenger does the following.

[18] The two smudging noises, ξ_j and η_j can in fact be merged together by appropriately setting the parameters.

- For each C for which a partial decryption has been returned, it does the following:
  - Implicitly assumes a $\mathsf{PartDec}$ query of the form $(S^* \setminus S_C, C)$ and generates $\{\tilde{r}_{C,j}\}_{j \in T_{S^* \setminus S_C}}$.
  - For each $j \in T_{S^*}$, computes $\hat{r}_{C,j} = \tilde{r}_{C,j} - \mathsf{FHE.decode}_0(\mathsf{fhesk}_j, \mathsf{ct}_C) - F(k_j, H(\mathsf{ct}_C))$. It then finds a $\mathbf{h}_C$ such that $\mathbf{h}_C^{\mathsf{T}} \mathbf{v}_j = \hat{r}_{C,j}$ for all $j \in T_{S^*}$. Observe that $\mathbf{h}_C$ is of length ℓ, we have more unknowns than the number of equations, and hence such a $\mathbf{h}_C$ can be found. Moreover since S^* is an invalid set, $\{\hat{r}_{C,j}\}_{j \in T_{S^*}}$ are random[19] Thus $\mathbf{h}_C$ is also random. The challenger then programs $H_1(R, \mathsf{ct}_C) = \mathbf{h}_C$ and returns $\{\mathsf{fsk}_j\}_{j \in T_{S^*}}$ to $\mathcal{A}$. We note that since R is a secret with high entropy, the probability that the adversary would have queried the random oracle H_1 on input $(R, \cdot)$ before getting the keys is negligible.
  - For each $j \in T_{S_C \setminus S^*}$, $\tilde{r}_{C,j}$ is viewed as $\mathsf{FHE.decode}_0(\mathsf{fhesk}_j, \mathsf{ct}_C) + H_1(R, \mathsf{ct}_C) + r'_{C,j}$. Since, $\tilde{r}_{C,j}$ is random, $r'_{C,j}$ is also random. Thus, effectively, the PRF component is replaced with a random value.

– For each post query (S, C), if $S = S^*$, the challenger returns an honestly computed value. For $i \in S \setminus S^*$, $p_{C,i} = \{y_{C,j}\}_{j \in T_i}$ is computed as $y_{C,j} = \tilde{r}_{C,j} + \xi_{C,j} + \eta_{C,j}$, where $\tilde{r}_{C,j}$ are again generated using the simulators as follows:
  - If C is queried for the first time then initialize $S_C = \emptyset$, $\mathsf{st}_C = \emptyset$, $\mathcal{L}_C = \emptyset$.
  - If $S^* \cup S_C \cup S$ is an invalid party set then generate $(\{\tilde{r}_{C,j}\}_{j \in T_{S \setminus S^*}}, \mathsf{st}'_C) \leftarrow \mathsf{bSS.SSSimI}(T_{S^* \cup S_C}, \{\tilde{r}_{C,j}\}_{j \in T_{S^* \cup S_C}}, T_{S \setminus S^*}, \mathsf{st}_C)$.
    Else, if $S^* \cup S_C \cup S$ is a valid party set then generate

$$(\{\tilde{r}_{C,j}\}_{j \in T_{S \setminus S^*}}, \mathsf{st}'_C) \leftarrow \mathsf{bSS.SSSimV}(\lfloor q/2 \rceil C(\mu_1, \ldots, C_k), T_{S^* \cup S_C}, \{\tilde{r}_{C,j}\}_{j \in T_{S^* \cup S_C}}, T_{S^* \setminus S}, \mathsf{st}_C)$$

    Again, for each $j \in T_{S \setminus S^*}$, $\tilde{r}_{C,j}$ can be viewed as $\mathsf{FHE.decode}_0(\mathsf{fhesk}_j, \mathsf{ct}_C) + H_1(R, \mathsf{ct}_C) + r'_{C,j}$. Since, $\tilde{r}_{C,j}$ is random, $\hat{r}_{C,j}$ is also random.

  We observe that this hybrid has the same distribution as Hybrid_3 in the proof of theorem 7. And hence, by similar arguments in theorem 5, Hybrid_0 and Hybrid_1 are indistinguishable.

Hybrid_2**:** In this hybrid, the challenger generates secret shares of $0^{|\mathsf{fsk}|}$ in place of fsk. Hybrid_2 is indistinguishable from Hybrid_1 from the security of bSS.

Hybrid_3**:** In this hybrid, ct_i encrypts 0 instead of μ_i. The indistinguishability from the previous hybrid follows from FHE security, since fsk is no longer used.

Finally, the proof completes by observing that since the challenger does not use fsk or $\{\mu_i\}_{i \in [k]}$ in Hybrid_3, it represents the simulator in the ideal experiment of Definition 37.

[19] except for the dependencies due to the share matrix $\mathbf{M}$, which are there in $\{\mathbf{v}_j\}_{j \in T_{S^*}}$ as well.