Threshold Cryptosystems From Threshold Fully Homomorphic Encryption[*]

Dan Boneh[†] Rosario Gennaro[‡] Steven Goldfeder[§] Aayush Jain[¶] Sam Kim[‖]
Peter M. R. Rasmussen[**] Amit Sahai[††]

Abstract

We develop a general approach to adding a threshold functionality to a large class of (non-threshold) cryptographic schemes. A threshold functionality enables a secret key to be split into a number of shares, so that only a threshold of parties can use the key, without reconstructing the key. We begin by constructing a *threshold* fully-homomorphic encryption scheme (TFHE) from the learning with errors (LWE) problem. We next introduce a new concept, called a *universal thresholdizer*, from which many threshold systems are possible. We show how to construct a universal thresholdizer from our TFHE. A universal thresholdizer can be used to add threshold functionality to many systems, such as CCA-secure public-key encryption (PKE), signature schemes, pseudorandom functions, and others primitives. In particular, by applying this paradigm to a (non-threshold) lattice signature system, we obtain the first single-round threshold signature scheme from LWE.

1 Introduction

Threshold cryptography [DF89, Fra89, DDFY94] is a general technique used to protect a cryptographic secret by splitting it into N shares and storing each share on a different server. Any subset of t servers can use the secret without re-constructing it. However, an adversary that compromises $t-1$ servers should not be able to recover or use the secret. Two examples of threshold tasks are:

- **Threshold signatures**: distribute the signing key of a signature system among N servers, so that any t servers can generate a signature. The scheme must provide *anonymity* and *succinctness*. Anonymity means that the same signature is produced, no matter which subset of t servers is used. Succinctness means that the signature size can depend on the security parameter, but must be independent of N and t.
- **Threshold decryption**: distribute the decryption key of a CCA-secure public-key encryption scheme among N servers, so that any t servers can decrypt. The scheme must be succinct, meaning that ciphertext size must be independent of N and t.

[*]This work is a merge of two closely related papers [BGGK17, JRS17].
[†]Stanford University. Email: `dabo@cs.stanford.edu`.
[‡]City College of New York. Email: `rosario@cs.ccny.cuny.edu`.
[§]Princeton University. Email: `stevenag@cs.princeton.edu`.
[¶]UCLA and Center for Encrypted Functionalities. Email `aayushjain@cs.ucla.edu`
[‖]Stanford University. Email: `skim13@cs.stanford.edu`.
[**]UCLA and Center for Encrypted Functionalities. Email `rasmussen@cs.ucla.edu`
[††]UCLA and Center for Encrypted Functionalities. Email `sahai@cs.ucla.edu`

Moreover, the time to verify signatures or encrypt messages should be independent of N and t. Other threshold tasks include threshold (H)IBE key generation, threshold ABE key generation, threshold pseudorandom functions, and many others (see Section 1.2). All have similar anonymity, succinctness, and efficiency requirements.

A common goal for threshold systems is to minimize the amount of interaction in the system, and in particular, construct *one-round* schemes. For example, in the case of signatures, an entity called a *combiner* wishes to sign a message m. The combiner sends m to all N servers, and some t of them reply. The combiner combines the t replies, and obtains the signature. No other interaction is allowed. In particular, the servers may not communicate with one another, or interact further with the combiner. Similarly, for threshold decryption, the combiner sends the ciphertext to all N servers, some t servers reply, and the combiner combines the replies to obtain the plaintext. No other interaction is allowed. We will often refer to the servers as *partial signers* or *partial decryptors*.

Many signature and encryption schemes have been thresholdized. For example, RSA signatures and encryption [Fra89, DDFY94, GRJK07, Sho00], Schnorr signatures [SS01], (EC)DSA signatures [GJKR01, GGN16], BLS signatures [BLS04, Bol03], Cramer-Shoup encryption [CG99], Regev encryption [BD10], and many more [SG02, DK05, BBH06]. Despite this great success, thresholdizing many basic lattice-based cryptographic primitives has been challenging. For example, it is still an open problem to construct lattice-based *one-round* threshold signatures or CCA-secure threshold PKE satisfying strong succinctness properties, as discussed in related work (Section 1.2). Thresholdizing more advanced lattice-based primitives, such as fully homomorphic encryption or functional encryption, has been largely unexplored.

1.1 Our Contributions

Our main contributions are twofold. First, we define the notion of *threshold fully homomorphic encryption* (TFHE) and construct it from the learning with errors assumption (LWE). As in a threshold PKE, a threshold FHE scheme allows the decryption key to be split into shares such that any t-out-of-N partial decryptions can be combined into a complete decryption of a given ciphertext in a single round. Furthermore, an evaluated ciphertext should be compact meaning that its size is independent of the original message and the number of decryptors N (Section 5.1). Second, we present general framework for universally thresholdizing many (non-threshold) cryptographic schemes using threshold-FHE. This framework lets us resolve the long-standing problem of one-round threshold signatures from lattices.

Threshold FHE. A general obstacle to constructing succinct threshold cryptosystems from lattice assumptions is the noise blow up that results from a multiplication by a large Lagrange coefficient. We handle this difficulty in two different ways. Our first method relies on linear secret sharing schemes where the reconstruction coefficients are always binary. We show that this class of secret sharing schemes is compatible with the decryption operation of a fully homomorphic encryption scheme and is also expressive enough to contain threshold access structures, along with other more general structures. In our second method, we focus on using the standard t-out-of-N Shamir secret sharing scheme, but we modify the noise distribution such that a multiplication by a Lagrange coefficient does not blow up the noise too much. By combining our methods with a suitable FHE scheme, we obtain a secure TFHE (Section 5.2 and 5.3) with strong compactness properties.

A universal thresholdizer. Our second contribution is a general framework for universally thresholdizing many (non-threshold) cryptographic schemes using a TFHE. For this, we define a new primitive called a *universal thresholdizer*. We show how to construct a universal thresholdizer with strong compactness properties from our TFHE scheme (Section 7). A universal thresholdizer takes in a cryptographic key and produces a number of key shares that can be used to individually evaluate a cryptographic function. Each of these individual evaluation shares can then be combined to result in the final evaluation of the function. We require that the scheme guarantees *privacy* meaning that no $t-1$ key shares or their evaluation shares reveal any information about the original key. Furthermore, we require that the scheme satisfies *robustness*, meaning that a maliciously generated evaluation share can always be detected.

With these guarantees, a universal thresholdizer scheme can be used to thresholdize many different types of systems. For example, we can take *any* (non-threshold) signature scheme as a black box and construct from it a one-round threshold signature scheme (Section 8.2). Since a universal thresholdizer can be proven secure based on LWE, and because there are known (non-threshold) signature schemes based on LWE [GPV08, Boy10, Lyu12], we obtain the first one-round threshold signature scheme based on LWE that is both succinct and anonymous. This resolves a long-standing open problem in lattice-based cryptography.

Beyond signatures, a universal thresholdizer can be composed with an existing CCA-secure PKE scheme [PW11, GPV08, Pei09, ABB10, MP12] to obtain the first lattice-based (one-round) threshold CCA-secure PKE where the public key size and encryption time are independent of the number of servers (Section 8.3). Similarly, composing universal thresholdizer with a functional encryption scheme gives functional encryption with threshold key generation. A universal thresholdizer, on its own, gives a function secret sharing scheme [BGI15, BGI16] that can support threshold access structures (Section 8.1).

Decentralized threshold FHE. Our basic TFHE scheme requires a trusted setup procedure to split the secret FHE key into shares. In Section 6, we define a *decentralized threshold fully homomorphic encryption* (dTFHE). In a dTFHE scheme, each decryption server generates its own public/secret key pair. At encryption time, the encryptor specifies a set of public keys and a threshold t to produce a ciphertext that can only be decrypted by combining t partial decryptions corresponding to the specific public keys. We construct a dTFHE in Section 6.2. However, while it achieves the added flexibility of decentralized key generation, our dTFHE scheme does not satisfy as strong compactness properties as our TFHE and universal thresholdizer. We leave improving the compactness of our dTFHE as an important open problem.

1.2 Related Work on Threshold Lattice Systems

Before describing our results, we first survey the existing work on threshold lattice cryptosystems.

Non-compact systems. We begin with threshold systems that have public key and ciphertext/signature sizes that are linear in N. Bendlin and Damgård [BD10] gave a threshold version of Regev's CPA-secure encryption scheme [Reg09], and Myers et al. [MSS11] applied the technique to fully homomorphic encryption. Xie et al. [XXZ11] gave a threshold CCA secure PKE scheme from lossy trapdoor functions, which can be instantiated from LWE [PW11]. In all these schemes, both the size of the public key and the ciphertext scales at least linearly in the number of decryptors. For signatures, Cayrel et al. [CLRS10] gave a lattice-based threshold ring signature scheme in which at least t signers are needed to create an anonymous signature. In this system, each signer has its own public key, and the verification time of a signature grows linearly with the number of signers.

Online/offline systems. The threshold Gaussian sampling protocol of Bendlin et al. [BKP13] as well as the commitment and zero-knowledge protocols of Baum et al. [BDOP16] provide compact threshold cryptosystems. However, the limitation of these systems is that the servers can only perform an *a priori* bounded number of *online* non-interactive decryption/signing operations before they must perform an *offline* interactive step.

N**-out-of-**N **systems.** Recent advances in low-round MPC from LWE [AJLA$^+$12, GLS15, MW16, BP16, PS16] give threshold cryptosystems for N-out-of-N thresholds. In these constructions, each party that is involved in the protocol encrypts an input to a joint function using an FHE scheme and broadcasts the ciphertexts to other parties. Then, each party homomorphically evaluates on the ciphertexts that it receives and participates in a single round distributed decryption protocol. If a trusted authority thresholdize an FHE key at setup and distribute the keys to each servers, then the single round distributed decryption protocol gives a N-out-of-N threshold FHE with trusted setup. The goal of this work is to construct cryptosystems that support *arbitrary* thresholds.

Fully-thresholdized systems. Very few existing lattice cryptosystems overcome all the limitations described above. One exception is threshold distributed PRFs [BLMR13] built from key-homomorphic PRFs [BLMR13, BP14, BV15].

2 Overview of the Main Construction

In this section, we provide an overview of the main threshold fully homomorphic encryption (TFHE) construction and its applications to thresholdizing cryptographic systems through a universal thresholdizer. We provide the full TFHE construction in Section 5.2 and 5.3. We define and construct a universal thresholdizer scheme in Section 7 and discuss its applications in greater depth in Section 8.

2.1 Distributing FHE Decryption

Our starting point is a standard LWE based fully homomorphic encryption schemes such as GSW [GSW13]. Recall that a ciphertext ct is a matrix in $\mathbb{Z}_q^{n\times m}$ and a secret key sk is a vector in $\mathbb{Z}_q^n$ for appropriately chosen LWE parameters n, m, q. To decrypt a ciphertext ct, the decryptor takes a specific column ct_m of the ciphertext matrix and computes its inner product with the secret key sk. That is, the decryptor computes $\langle \mathsf{ct}_m, \mathsf{sk}\rangle \in \mathbb{Z}_q$. If the resulting value is small, the underlying plaintext is interpreted as 0; otherwise, it is interpreted as 1 (Definition 3.9).

Since inner product is linear, one might try to thresholdize FHE decryption by applying Shamir t-out-of-N secret sharing to sk. This will produce N keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$, one for each user. Then to decrypt a ciphertext ct, each user can compute the inner product $\langle \mathsf{ct}_m, \mathsf{sk}_i\rangle$ as its partial decryption. The combiner can then compute the Lagrange coefficients $\lambda_i^{(S)}$ for some subset $S \subseteq \{1, \ldots, N\}$ of size t and recombine the shares as

$$\sum_{i\in S} \lambda_i^{(S)} \cdot \Big\langle \mathsf{ct}_m, \mathsf{sk}_i \Big\rangle = \Big\langle \mathsf{ct}_m, \sum_{i\in S} \lambda_i^{(S)} \cdot \mathsf{sk}_i \Big\rangle = \Big\langle \mathsf{ct}_m, \mathsf{sk} \Big\rangle.$$

Unfortunately, this construction is insecure. For $i \in \{1, \ldots, N\}$, every time decryptor i computes a partial decryption, it leaks information about its secret share sk_i by publishing the inner product of sk_i with a public vector ct_m.

One way to resolve this issue is for decryptor i to add small additive noise to the inner product

$$\mathsf{p}_i = \langle \mathsf{ct}_m, \mathsf{sk}_i \rangle + \mathsf{noise}.$$

However, for a t-out-of-N threshold scheme, this additive error prevents correct reconstruction of the key. The Lagrange coefficients, when interpreted as elements in $\mathbb{Z}_q$, are large and therefore, blow up the noise when multiplied to the partial decryptions.

A standard solution to this problem is called bit decomposition. For example, every decryptor can compute the inner product and scale it by powers of two as:

$$\mathsf{p}_i^{(j)} = 2^j \cdot \langle \mathsf{ct}_m, \mathsf{sk}_i \rangle + \mathsf{noise}^{(j)}, \qquad \text{for } j = 1, \ldots, \log_2 q.$$

It sends $\{\mathsf{p}_i^{(j)}\}_j$ to the combiner. However, such bit decomposition not only makes simulation of partial decryptions difficult in the security proof, but also leads to direct attacks due to the malleability of the partial decryptions.

In this work, we show how to handle the noise blow up in two different ways. We provide the high level overview of the two techniques below.

Using $\{0,1\}$-LSSS. In our first method, instead of coping with the Lagrange coefficients directly in the construction, we abstract it out by using a different secret sharing scheme. In particular, we first define a class of access structures, denoted $\{0,1\}$-LSSS, that consists of the set of access structures that can be supported by a linear secret sharing scheme where the reconstruction coefficients are always binary (Definition 4.13). More precisely, such secret sharing scheme divides a secret sk into a set of shares $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$ according to an access structure such that each share consists of a set of field elements in $\mathbb{Z}_q$, $\mathsf{sk}_i = \{\mathsf{s}_{i,j}\}_{j\in[\ell]}$ for a fixed bound ℓ. For any set $S \subseteq [N]$ that satisfies the access structure, there exists a subset $S' \subseteq \bigcup_{i\in S'} \mathsf{sk}_i$ such that $\sum_{S'} \mathsf{s}_{i,j} = \mathsf{sk}$. It is not difficult to see that with this property, we can construct a correct TFHE scheme for any access structures in $\{0,1\}$-LSSS. With careful analysis (Section 5.2), we can also show that using such special linear secret sharing scheme, all the partial decryptions are simulatable.

The remaining question is how expressive is the class $\{0,1\}$-LSSS? Two obvious access structures which are contained in $\{0,1\}$-LSSS are undirected s-t-connectivity and N-out-of-N, but other than these, it is not clear what other useful access structures are contained in $\{0,1\}$-LSSS. However, we show that, in fact, the class is fairly large and contains the set of access structures defined by *monotone Boolean formulas* [LW11]. A classic result of Valiant [Val84, Gol14] shows that every threshold function can be expressed as a polynomial size monotone formula. Therefore, a $\{0,1\}$-LSSS contains the set of threshold access structures that we need.

We start with the observation that the set of access structures defined by monotone Boolean formulas with input fan-out 1 (*special* monotone Boolean formula) belongs to $\{0,1\}$-LSSS through a folklore algorithm [Bei96, GPSW06] Let $C : \{0,1\}^N \to \{0,1\}$ be a special monotone Boolean formula with an associated tree T whose internal nodes are assigned either AND or OR, and the N leaf nodes are INPUT gates that are assigned x_i. Then, we can define a linear secret sharing scheme for $\mathsf{s} \in \mathbb{Z}_q$ described as follows.

1. Assign the root r of T with the secret to be shared s.

2. If r is an INPUT gate, then simply return. Otherwise:

    - If r is an AND gate, then additively secret share k by sampling $\alpha \xleftarrow{R} \mathbb{Z}_q$ and define two shares $\mathsf{s}_\ell = \alpha$ and $\mathsf{s}_r = \mathsf{k} - \alpha$.

- If r is an OR gate, then duplicate k into shares by setting $\mathsf{s}_\ell = \mathsf{k}$ and $\mathsf{s}_r = \mathsf{k}$.

3. For each child node v_ℓ and v_r, let T_ℓ and T_r be the sub-trees having v_ℓ and v_r as roots respectively. Then, recurse on the sub-trees T_ℓ and T_r with secrets s_ℓ and s_r respectively.

At the end of the recursive process, each leaf node that is assigned x_i is assigned with a secret share s_i. It is not difficult to see that for $x \in \{0,1\}^N$, the secret s can be reconstructed from the set of shares $\{\mathsf{s}_i\}_{x_i=1}$ if and only if $C(x) = 1$. Furthermore, given $\{\mathsf{s}_i\}_{x_i=1}$ for $C(x) = 1$, the reconstruction procedure consists of simply identifying a subset of the shares $S \subseteq \{\mathsf{s}_i\}_{x_i=1}$ (for the OR gates), and summing up the shares $\mathsf{s} = \sum_{i\in S} \mathsf{s}_i$ (for the AND gates). In Section 4.2, we prove that this construction indeed yields a correct and secure secret sharing scheme for special monotone Boolean formulas.

Our next observation is that the secret sharing mechanism above can also be used for *regular* monotone Boolean formulas, which have multiple input fan-out. Consider a monotone Boolean formula $C : \{0,1\}^N \to \{0,1\}$ with multiple input fan-out that is bounded by ℓ. Then, we can derive a new special monotone Boolean formula $\tilde{C} : \{0,1\}^{\ell N} \to \{0,1\}$ by letting every fan-out of an input gate of C to be a separate input. Now, applying the secret sharing mechanism above to $\tilde{C}$ yields a set of shares $\{\mathsf{s}_i\}_{i\in[\ell N]}$ shares. Partitioning this set into the corresponding input x_i in C, we get a set of N shares $\{\mathsf{s}_{i,j}\}$ that still abides to the syntax of a linear secret sharing scheme required for $\{0,1\}$-LSSS. Furthermore, since the circuit C can still be evaluated from $\tilde{C}$, the secret s can be reconstructed from the union of the set of shares $\bigcup_{i\in S}\{\mathsf{s}_{i,j}\}$ for any satisfying set S.

It remains to prove that this secret sharing scheme is secure under collusion. If so, then we obtain a *secure* TFHE scheme. We indeed show that this is the case.

Clearing out denominators. Although the use of $\{0,1\}$-LSSS to achieve threshold decryption results in a clean construction that does not require any significant modification to the existing fully homomorphic construction, the use of monotone Boolean formulas to express threshold access structure introduces significant overhead to the resulting TFHE construction. In particular, the size of the key shares sk_i for $i = 1, \dots, N$ is at least $\Omega(N^4)$, introducing significant space overhead. In Section 5.3, we introduce another approach where the share sizes are quasilinear $\tilde{O}(N)$. We describe the precise trade-offs between the two methods in Section 5.4.

The high level idea of our second method is to use the technique of "clearing out the denominators" [Sho00, ABV$^+$12]. The observation is that since the Lagrange coefficients are rational numbers, we can scale them to be integers. In particular, for a t-out-of-N secret sharing, for any set S of size t and $i \in S$, the term $(N!)^2 \cdot \lambda_i^{(S)}$ is an integer. By modifying the construction so that every signer first scales the noise that it adds by $(N!)^2$, and sufficiently increasing the modulus of the scheme to support its additional noise growth, we can preserve correct reconstruction. In fact, with careful analysis (Section 5.3), such a TFHE construction can be made secure.

An evident limitation of the method above is the increase in the modulus and hence, an increase in the size of the ciphertext. Since the size of elements in $\mathbb{Z}_q$ increases by $\log N! = O(N \log N)$ bits, the size of the ciphertext depends linearly on the number of servers N, violating our compactness requirement (Definition 5.2). However, we show that any non-compact TFHE can be boosted to a compact one by combining it with any compact (non-threshold) FHE. The idea is to first construct the notion of universal thresholdizer (Definition 7.1) via a (non-compact) TFHE scheme and then use the thresholdizer to thresholdize a compact FHE. We provide a high level description of the universal thresholdizer in Section 2.2 and provide the formal details of this boosting step in Section 8.4.

2.2 Universal Thresholdizer: A General Tool

We next put our new TFHE to use. We define a new primitive called a *universal thresholdizer* (UT) that can be used to thresholdize many existing systems including signatures (Section 8.2). The resulting systems are secure one-round threshold systems that also provide robustness guarantees against malicious key share holders. Our universal thresholdizer abstraction provides a modular design for threshold systems and also simplifies the proof of security.

A UT scheme consists of a setup algorithm, an evaluation algorithm, and a combining algorithm. The setup of a UT scheme takes in a secret message x and divides it into a set of shares $\mathsf{s}_1, \ldots, \mathsf{s}_N$, which are distributed to N users. On input a circuit C, each user can independently compute an evaluation share y_i of $C(x)$ using their shares s_i. For a set $S = \{\mathsf{y}_i\}$ for which $|S| \geq t$, the evaluation shares can be combined to produce $\mathsf{y} = C(x)$. For robustness, we define an extra verification algorithm that given C and y_i, checks whether y_i was computed correctly.

The privacy guarantee of a UT scheme states that the shares $\mathsf{s}_1, \ldots, \mathsf{s}_N$ as well as the evaluation shares y_i can be simulated only given access to the circuit C and $C(x)$. The robustness guarantee of a UT scheme simply states that it is hard for an adversary to produce an improperly computed evaluation share y_i for a circuit C such that the verification algorithm accepts.

With these security guarantees, it is easy thresholdize existing cryptographic functions. To demonstrate the idea, consider the case of distributed PRF where a key k can be divided into a number of key shares such that independent PRF evaluations using these key shares can be combined into a final PRF evaluation. To construct a distributed PRF $\tilde{F}$ from a regular PRF $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$, we sample a key $\mathsf{k} \overset{\text{R}}{\leftarrow} \mathcal{K}$ and invoke UT setup with k to generate the key shares $\mathsf{s}_1, \ldots, \mathsf{s}_N$. Then, to evaluate $\tilde{F}$ on an input $x \in \mathcal{X}$, each party generates the evaluation share y_i for the circuit $C_x(\mathsf{k}) = F(\mathsf{k}, x)$. The evaluation share can then be combined in a threshold manner to produce the final PRF evaluation $\mathsf{y} = F(\mathsf{k}, x)$.

The robustness of $\tilde{F}$ follows from the robustness condition of UT straightforwardly. To prove pseudorandomness of $\tilde{F}$, we simply erase the original PRF key k from the security experiment by invoking the privacy simulator of UT. This allows us to reduce pseudorandomness directly to the underlying PRF security game of F.

In Section 7.1, we construct a robust universal thresholdizer using non-interactive zero knowledge proofs (NIZK). We note that constructing NIZK from lattices is still an open problem. However, our setting allows the use of NIZK with preprocessing [DMP88, LS90], which can be constructed from one-way functions and therefore, can be instantiated from lattices. A better way to ensure robustness is using *homomorphic signatures* (Section 7.2). Because homomorphic signatures [BF11, GVW15] give more compact proofs than NIZKs, we can get partial evaluations y_i whose size is independent of the original secret message x and the size of circuit C that is used for the evaluation. Unlike NIZK, homomorphic signatures can be constructed from the SIS problem [GVW15].

3 Preliminaries

Basic Notations. For an integer n, we write $[n]$ to denote the set $\{1, \ldots, n\}$. We use bold lowercase letters (*e.g.*$\mathbf{v}, \mathbf{w}$) to denote vectors and bold uppercase letters (*e.g.*$\mathbf{A}, \mathbf{B}$) to denote matrices. Throughout this work, we will always use infinity norm for vectors. This means that for a vector $\mathbf{x}$, the norm $\|\mathbf{x}\|$ is the maximal absolute value of an element in $\mathbf{x}$. For any set X, we denote by $\mathcal{P}(X)$ as the power set of X. For any $Y, Z \in \{0,1\}^n$, we say that $Y \subseteq Z$ if for each index $i \in [n]$

such that $Y_i = 1$, we have $Z_i = 1$.

We write λ for the security parameter. We say that a function $\epsilon(\lambda)$ is negligible in λ if $\epsilon(\lambda) = o(1/\lambda^c)$ for every $c \in \mathbb{N}$, and we write $\mathsf{negl}(\lambda)$ to denote a negligible function in λ. We say that an event occurs with *negligible probability* if the probability of the event is $\mathsf{negl}(\lambda)$, and an event occurs with *overwhelming probability* if its complement occurs with negligible probability. For a distribution X over a finite domain Ω, we write $\omega \leftarrow X$ to denote that ω is sampled at random according to distribution X. For a uniform distribution, we simply write $\omega \overset{\mathrm{R}}{\leftarrow} \Omega$. For a distribution ensemble $\chi = \chi(\lambda)$ over the integers, and an integer bound $B = B(\lambda)$, we say that χ is B-bounded if $\Pr_{x \leftarrow \chi(\lambda)}[|x| \leq B(\lambda)] = 1$.

3.1 Statistical Distance

Definition 3.1 (Statistical Distance)**.** Let E be a finite set, Ω a probability space, and $X, Y : \Omega \to E$ random variables. We define the *statistical distance* between X and Y to be the function Δ defined by

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

We next recall a technique of "smuding out" or hiding the presense of noise by overwhelming it with a much larger noise.

Lemma 3.2 (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 $\Delta(E_1, E_2) < B_1/B_2$.*

3.2 Learning with Errors

Definition 3.3 (LWE)**.** Let n, m, q be positive integers and χ be some noise distribution over $\mathbb{Z}_q$. In the $\mathsf{LWE}(n, m, q, \chi)$ problem, the adversary's goal is to distinguish between the two distributions

$$(\mathbf{A}, \mathbf{A}^T\mathbf{s} + \mathbf{e}) \quad \text{and} \quad (\mathbf{A}, \mathbf{u})$$

where $\mathbf{A} \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \chi^m$, and $\mathbf{u} \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_q^m$.

For certain B_{lwe}-bounded error distributions χ, the $\mathsf{LWE}(n, m, q, \chi)$ problem is as hard as approximating certain worst-case lattice problems such as GapSVP and SIVP on n-dimensional lattices to within $\tilde{O}(n \cdot q/B_{\mathsf{lwe}})$ factor [Reg09, Pei09, ACPS09, MM11, MP12, BLP+13].

3.3 Cryptographic Primitives

In this section, we review *non-interactive zero knowledge* proofs and *fully homomorphic encryption*. Additional definitions of basic cryptographic primitives are provided in Appendix A.

3.3.1 Zero Knowledge with Pre-Processing

In this work, we consider a 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, which we call *zero knowledge proof system with pre-processing* (PZK), can be constructed from a much weaker assumption than a standard NIZK proof system and is sufficient for this work. In particular, [DMP88, LS90] construct PZK from one-way functions.

Definition 3.4. 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 $(\sigma_V, \sigma_P) \leftarrow \mathsf{PZK.Pre}(1^\lambda)$:

1. **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.

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

3. **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:
$$\{\sigma_V, \mathsf{PZK.Prove}(\sigma_P, x, w)\} \approx_c \{\mathcal{S}(x)\}$$

3.3.2 Homomorphic Encryption

In this section, we review the notion of fully homomorphic encryption.

Definition 3.5 (FHE)**.** A fully homomorphic encryption FHE scheme is a tuple of PPT algorithms $\mathsf{FHE} = (\mathsf{FHE.Setup}, \mathsf{FHE.Encrypt}, \mathsf{FHE.Eval}, \mathsf{FHE.Decrypt})$ with the following properties:

- $\mathsf{FHE.Setup}(1^\lambda, 1^d) \rightarrow (\mathsf{pk}, \mathsf{sk})$: On input the security parameter λ, and a depth bound d, the setup algorithm outputs a key pair $(\mathsf{pk}, \mathsf{sk})$.

- $\mathsf{FHE.Encrypt}(\mathsf{pk}, \mu) \rightarrow \mathsf{ct}$: On input a public key pk, and a message $\mu \in \{0, 1\}$, the encryption algorithm outputs a ciphertext ct.

- $\mathsf{FHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k) \rightarrow \hat{\mathsf{ct}}$: On input a public key pk, a circuit $C : \{0, 1\}^k \rightarrow \{0, 1\}$ of depth at most d, and a tuple of ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$, the evaluation algorithm outputs an evaluated ciphertext $\hat{\mathsf{ct}}$.

- $\mathsf{FHE.Decrypt}(\mathsf{pk}, \mathsf{sk}, \hat{\mathsf{ct}}) \rightarrow \hat{\mu}$: On input a public key pk, secret key sk, and a ciphertext $\hat{\mathsf{ct}}$, the decryption algorithm outputs a message $\hat{\mu} \in \{0, 1, \perp\}$.

We require an FHE scheme to satisfy compactness, correctness, and security as follows.

Definition 3.6 (Compactness)**.** We say that a FHE scheme is compact if there exists a polynomial $\mathsf{poly}(\cdot)$ such that for all λ, depth bound d, circuit $C : \{0,1\}^k \to \{0,1\}$ of depth at most d, and $\mu_i \in \{0,1\}$ for $i \in [k]$, the following holds. For $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.Setup}(1^\lambda, 1^d)$, $\mathsf{ct}_i \leftarrow \mathsf{FHE.Encrypt}(\mathsf{pk}, \mu_i)$ for $i \in [k]$, $\hat{\mathsf{ct}} \leftarrow \mathsf{FHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$, we have that $|\hat{\mathsf{ct}}| \leq \mathsf{poly}(\lambda, d)$.

Definition 3.7 (Correctness)**.** We say that a FHE scheme is correct if for all λ, depth bound d, circuit $C : \{0,1\}^k \to \{0,1\}$ of depth at most d, and $\mu_i \in \{0,1\}$ for $i \in [k]$, the following holds. For $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.Setup}(1^\lambda, 1^d)$, $\mathsf{ct}_i \leftarrow \mathsf{FHE.Encrypt}(\mathsf{pk}, \mu_i)$ for $i \in [k]$, and $\hat{\mathsf{ct}} \leftarrow \mathsf{FHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$,

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

Definition 3.8 (Security)**.** We say that a FHE scheme satisfies security if for all λ, and depth bound d, the following holds. For any PPT adversary $\mathcal{A}$, the following experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{FHE}}(1^\lambda, 1^d)$ outputs 1 with negligible probability:

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

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

Additional properties. In this work, we base our constructions on an FHE scheme satisfying a few additional properties. We refer to this type of FHE scheme as a *special* FHE scheme.

Definition 3.9 (Special FHE)**.** Let FHE be a fully homomorphic encryption scheme (Definition 3.5). We call FHE a *special* FHE scheme if it satisfies the following properties:

1. On input 1^λ and 1^d, the setup algorithm $\mathsf{FHE.Setup}$ outputs $(\mathsf{pk}, \mathsf{sk})$ where the public key contains a prime q, and the secret key sk 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:
   - $\mathsf{FHE.Decode}_0(\mathsf{sk}, \mathsf{ct}) \to p$: On input an encryption of a message $\mu \in \{0,1\}$, and a secret key vector sk, outputs $p \in \mathbb{Z}_q$ where $p = \mu \left\lfloor \frac{q}{2} \right\rceil + e$ for $e \in [-cB, cB]$, $B = B(\lambda, d, q)$, and e is an integer multiple of c.
   - $\mathsf{FHE.Decode}_1(p) \to \mu$: On input $p \in \mathbb{Z}_q$, it outputs $\mu = 1$ if $p \in [-\left\lfloor \frac{q}{4} \right\rceil, \left\lfloor \frac{q}{4} \right\rceil]$, and $\mu = 1$ otherwise.
3. Decode_0 is a linear operation over $\mathbb{Z}_q$ in the secret key sk.

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

When context is clear, we will often drop the rounding notation $\lfloor \cdot \rceil$ and implicitly assume it for simplicity.

As observed in [MW16], these properties are satisfied by simple modification on the constructions that are based on [GSW13]. We provide a high level description of the modification in Appendix B for reference. Constructions that are based on [BV14, BGV12] can also be adapted to satisfy these properties.

Theorem 3.10. *Fix the security parameter λ and depth bound d. Then, there exists a special* FHE *scheme satisfying Definition 3.9 with an associated noise bound $B = 2^{\tilde{O}(d)}$ assuming the hardness of* $\mathsf{LWE}(n, m, q, \chi)$ *where $n(\lambda), m(\lambda) = \mathsf{poly}(\lambda)$, χ is a B_{lwe}-bounded distribution for $B_{\mathsf{lwe}} = \mathsf{poly}(\lambda)$ and $q = 2^{\tilde{O}(d)}$.*

In particular, the FHE scheme can be based on approximating worst-case lattice problems on n-dimensional lattices to within $2^{\tilde{O}(d)}$ factor.

4 Secret Sharing for Threshold Access Structures

In this section, we provide general results on secret sharing that we use throughout this work. We start by establishing basic notations and terms in secret sharing in Section 4.1 and define threshold access structures in Section 4.2. Then, we define a special class of access structures that we call $\{0, 1\}$-LSSS and show that it contains the class of threshold access structures in Section 4.3.

4.1 Secret Sharing Preliminaries

In this section, we provide an introduction to the basic notations and terms in secret sharing. Some of our definitions are adapted from [LW11].

4.1.1 Access Structures

Definition 4.1 (Monotone Access Structure)**.** Let $P = \{P_1, \ldots, P_N\}$ be a set of parties. A collection $\mathbb{A} \subseteq \mathcal{P}(P)$ is monotone if for any sets B, C satisfying $B \in \mathbb{A}$ and $B \subseteq C \subseteq P$, we have $C \in \mathbb{A}$. A monotone access structure on P is a monotone collection $\mathbb{A} \subseteq \mathcal{P}(P) \backslash \emptyset$. The sets in $\mathbb{A}$ are called the valid sets and the sets in $\mathcal{P}(P) \backslash \mathbb{A}$ are called the invalid sets.

Since we only consider monotone access structures in this work, we use the terms monotone access structure and just access structure interchangeably.

For simplicity of notation, we generally identify a party with its index. Additionally, for a set of parties $P = \{P_1, \ldots, P_N\}$, and $S \subseteq P$, we denote by $\mathbf{x}_S$ the vector $\mathbf{x}_S = (x_1, \ldots, x_N)$ where $x_i = 1$ if $P_i \in S$ and $x_i = 0$ if $P_i \notin S$.

Definition 4.2 (Efficient Access Structure)**.** Let $P = \{P_1, \ldots, P_N\}$ be a set of parties and $\mathbb{A} \subseteq \mathcal{P}(P)$ a monotone access structure on P. We say that $\mathbb{A}$ is efficient if there exists a polynomial size circuit $f_{\mathbb{A}} : \{0, 1\}^N \to \{0, 1\}$ such that for all $S \subseteq P$, $f_{\mathbb{A}}(\mathbf{x}_S) = 1$ if and only if $S \in \mathbb{A}$.

In this work, we will only consider efficient access structures.

Definition 4.3 (Class of Monotone Access Structures)**.** Let $P = \{P_1, \ldots, P_N\}$ be a set of parties. A class of monotone access structures is a collection $\mathbb{S} = \{\mathbb{A}_1, \ldots, \mathbb{A}_t\} \subseteq \mathcal{P}(\mathcal{P}(P))$ of monotone access structures on P.

4.1.2 Secret Sharing

We now define secret sharing for a class of access structures [CK93, BC94, BR07].

Definition 4.4. Let $P = \{P_1, \ldots, P_N\}$ be a set of parties and $\mathbb{S}$ be a class of efficient access structures. 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:

- $\mathsf{SS.Share}(\mathsf{k}, \mathbb{A}) \to (\mathsf{s}_1, \ldots, \mathsf{s}_N)$: On input a secret $\mathsf{k} \in \mathcal{K}$ and an access structure $\mathbb{A}$, the sharing algorithm returns a set of shares $\mathsf{s}_1, \ldots, \mathsf{s}_N$ for each parties.
- $\mathsf{SS.Combine}(B) \to \mathsf{k}$: On input a set of shares $B = \{\mathsf{s}_i\}_{i \in S}$, the combining algorithm outputs a secret $\mathsf{k} \in \mathcal{K}$.

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

Definition 4.5 (Correctness)**.** For all $S \in \mathbb{A}$, and $\mathsf{k} \in \mathcal{K}$, $(\mathsf{s}_1, \ldots, \mathsf{s}_N) \leftarrow \mathsf{SS.Share}(\mathsf{k}, \mathbb{A})$, we have

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

Definition 4.6 (Privacy)**.** For all $S \notin \mathbb{A}$, and $\mathsf{k}_0, \mathsf{k}_1 \in \mathcal{K}$, $(\mathsf{s}_{b,1}, \ldots, \mathsf{s}_{b,N}) \leftarrow \mathsf{SS.Share}(\mathsf{k}_b, \mathbb{A})$ for $b \in \{0, 1\}$, the following distributions are identical

$$\{\mathsf{s}_{0,i}\}_{i \in S} \approx \{\mathsf{s}_{1,i}\}_{i \in S}.$$

A useful definition for secret sharing schemes is the notion of *maximal invalid party set* and *minimal valid party set.*

Definition 4.7. Let $P = \{P_1, \ldots, P_N\}$ be a set of parties and $\mathbb{A}$ be a monotone access structure on P. Then, we define the following:

- A set of parties $S \subseteq P$ is a *maximal invalid party set* if $S \notin \mathbb{A}$ but for every $P_i \in P \backslash S$, we have $S \cup \{P_i\} \in \mathbb{A}$.
- A set of parties $S \subseteq P$ is a *minimal valid party set* if $S \in \mathbb{A}$ and for every $S' \subsetneq S$, we have $S' \notin \mathbb{A}$.

Linear Secret Sharing. In this work, we base our constructions on a special class of secret sharing scheme called linear secret sharing scheme where the combining algorithm $\mathsf{SS.Combine}$ consists of linear operations.

Definition 4.8 (Linear Secret Sharing Scheme)**.** Let $P = \{P_1, \ldots, P_N\}$ be a set of parties and $\mathbb{S}$ a class of efficient access structures on P. 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 the following properties are satisfied:

- $\mathsf{SS.Share}(\mathsf{k}, \mathbb{A})$: 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 $T_i \subseteq [\ell]$. To create the shares on a secret k, the sharing algorithm first samples random values $r_2, \ldots, r_n \xleftarrow{\text{R}} \mathbb{Z}_p$, and define a vector $\mathbf{w} = \mathbf{M} \cdot (\mathsf{k}, r_2, \ldots, r_n)^T$. The share for P_i consists of the entries $\{w_j\}_{j \in T_i}$.
- $\mathsf{SS.Combine}(B)$: 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 jth row of $\mathbf{M}$. Note that 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

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

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

In a linear secret sharing scheme, each party P_i holds a set of scalars in $\mathbb{Z}_p$ as shares. For our definitions and proofs, it is convenient to define an analogous definition to Definition 4.7 for these set of scalar elements.

Definition 4.9. Let $P = \{P_1, \dots, P_N\}$ be a set of parties, $\mathbb{S}$ a class of efficient access structures on P, and SS a linear secret sharing scheme for $\mathcal{S}$ with share matrix $\mathbf{M} \in \mathbb{Z}_q^{\ell \times N}$. For a set of indices $T \subseteq [\ell]$, we say that T is a *valid share set* if $(1, 0, \dots, 0) \in \mathsf{span}(\{\mathbf{M}[j]\}_{j \in T})$, and an *invalid share set* otherwise. We also define the following:

- A set of indices $T \subseteq [\ell]$ is a *maximal invalid share set* if T is an invalid share set, but for any $i \in [\ell] \backslash T$, 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.

For simplicity, we slightly abuse notation and refer to LSSS_N as the *class of access structure* that *can be supported* by a linear secret sharing scheme on N parties. When the context is clear, we simply refer to LSSS_N as just LSSS.

Secret Sharing Vectors. In our application of linear secret sharing, we will always be sharing a vector $\mathbf{s} \in \mathbb{Z}_q^n$ instead of a single scalar in $\mathbb{Z}_p$. Simply sharing each entry of the vector $\mathbf{s}$ using fresh randomness for each entry yields shares $\mathbf{s}_1, \dots, \mathbf{s}_\ell \in \mathbb{Z}_p^n$. It is easy to see that the secret $\mathbf{s} \in \mathbb{Z}_q^n$ can be reconstructed as a linear combination of the shares $\mathbf{s}_i$ using the same coefficients as for a single field element. It is also easy to see that the privacy property of the scheme is also maintained.

4.2 Threshold Access Structures

In this section, we define the class of threshold access structures TAS and describe Shamir secret sharing [Sha79].

Definition 4.10 (TAS)**.** Let $P = \{P_1, \dots, P_N\}$ be a set of parties. An access structure $\mathbb{A}_t$ is called a threshold access structure if for every set of parties $S \subseteq P$, we have $S \in \mathbb{A}_t$ if and only if $|S| \geq t$. We define TAS to be the class of all access structures $\mathbb{A}_t$ for all $t \in \mathbb{N}$.

Instead of defining the algorithms of Shamir secret sharing formally, we just describe the properties of the scheme that we need.

Theorem 4.11 (Shamir Secret Sharing)**.** *Let $P = \{P_1, \dots, P_N\}$ 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 4.8) SS with secret space $\mathcal{K} = \mathbb{Z}_p$ for some prime p satisfying the following properties:*

- *For any secret* $\mathsf{k} \in \mathbb{Z}_p$ *and* $\mathbb{A}_t \in \mathsf{TAS}$*, each share for party* P_i *consists of a single element* $w_i \in \mathbb{Z}_p$*. For convenience of notation, we denote* $w_0 = \mathsf{k}$*.*
- *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 $\lambda_{i,j}^S \in \mathbb{Z}_q$ *such that*

$$w_j = \sum_{i \in S} \lambda_{i,j}^S \cdot w_i.$$

For our purposes, we want the Lagrange coefficients to be "low-norm" values. However, a regular Lagrange coefficient have no bound on its norm. Therefore, for our construction, we take advantage of the fact that the Lagrange coefficients can be defined to be rational numbers and therefore, we can "clear out their denominators" [Sho00, ABV+12].

Lemma 4.12 ([ABV+12])**.** *Let* $P = \{P_1, \ldots, P_N\}$ *be a set of parties,* TAS *the class of threshold access structures on* P*, and* SS *a Shamir secret sharing scheme with secret space* $\mathbb{Z}_p$ *for some prime* p *with* $(N!)^3 \leq p$*. Then, for any set* $S \subset [N] \cup \{0\}$ *of size* t*, and for any* $i, j \in [N]$*, the product* $(N!)^2 \cdot \lambda_{i,j}^S$ *is bound*

$$\left|(N!)^2 \cdot \lambda_{i,j}^S\right| \leq (N!)^3$$

interpretted as an integer.

4.3 Access Structures $\{0,1\}$-LSSS

In this section, we define a special class of access structures that we denote by $\{0,1\}$-LSSS that is contained in LSSS. This is the class of access structures that can be supported by a linear secret sharing scheme where the recovery coefficients are always binary. In Section 5.2, we construct a threshold fully homomorphic encryption scheme for these classes of access structures. We show in Section 4.2 that the class $\{0,1\}$-LSSS contains the class of threshold access structures.

Definition 4.13 ($\{0,1\}$-LSSS)**.** Let $P = \{P_1, \ldots, P_N\}$ be a set of parties. The class of access structure $\{0,1\}$-LSSS_N is the collection of access structures $\mathbb{A} \in \mathsf{LSSS}_N$ (Definition 4.8) for which there exists an efficient linear secret sharing scheme $\mathsf{SS} = (\mathsf{SS.Share}, \mathsf{SS.Combine})$ over the secret space $\mathcal{K} = \mathbb{Z}_p$ satsifying the following property:

- Let k 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 $S \in \mathbb{A}$, there exists a subset $T \subseteq \bigcup_{i \in S} T_i$ such that $\mathsf{k} = \sum_{j \in T} w_j$.

We call a linear secret sharing scheme that satisfy the properties above as a *special linear secret sharing scheme.*

We note that for any special linear secret sharing scheme SS, and for any minimal valid share set $T \subseteq [\ell]$, we have that $\sum_{j \in T} w_j = \mathsf{k}$.

Now, the fact that every access structure $\mathbb{A} \in \{0,1\}$-LSSS is efficient follows directly from the efficiency of the LSSS class. However, it is less clear that the set T of the definition above can be computed efficiently given any $S \subseteq \mathbb{A}$. We show that this is indeed the case in the following lemma. We provide the proof in Appendix C.1

Lemma 4.14. *Let $P = \{P_1, \ldots, P_N\}$ be a set of parties, and* SS *a special linear secret sharing scheme for* $\{0,1\}$-LSSS*. Then, for any access structure* $\mathbb{A} \in \{0,1\}$-LSSS*, and* $S \in \mathbb{A}$*, the set* $T \subseteq S$ *as specified in Definition 4.13 can be computed efficiently.*

We now state the main theorem of this section.

Theorem 4.15. $\mathsf{TAS} \subseteq \{0,1\}$-$\mathsf{LSSS}$.

To prove the lemma, we first define the class of access structures induced by monotone Boolean formulas.

Definition 4.16 (Monotone Boolean Formula)**.** A *monotone Boolean formula* $C : \{0,1\}^N \to \{0,1\}$ is a Boolean circuit with the following properties:

- There is a single output gate.
- Every gates is one of AND or OR gate with fan-in 2 and fan-out 1.
- The input wires can have multiple fan-out.

Definition 4.17 (MBF)**.** Let $P = \{P_1, \ldots, P_N\}$ be a set of parties and $C : \{0,1\}^N \to \{0,1\}$ a monotone Boolean formula. An access structure $\mathbb{A}_C$ is called a monotone boolean formula access structure if for every set of parties $S \subseteq P$, we have $S \in \mathbb{A}$ if and only if $C(\mathbf{x}) = 1$. We define MBF to be the class of all access structures $\mathbb{A}_C$ for all monotone Boolean formula C.

Now, Theorem 4.15 is implied by the following.

Theorem 4.18 ([Val84, Gol14])**.** $\mathsf{TAS} \subseteq \mathsf{MBF}$.

Theorem 4.19 ([LW11])**.** $\mathsf{MBF} \subseteq \{0,1\}$-$\mathsf{LSSS}$.

Although Theorem 4.19 is folklore, we provide the formal proof in Appendix C.2 for completeness.

5 Threshold Fully Homomorphic Encryption

In this section, we present the definition of threshold fully homomorphic encryption (TFHE) for any class of access structures. Then, in Sections 5.2 and 5.3, we construct TFHE for the class of threshold access structure TAS.

5.1 Definitions

Definition 5.1 (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 single bit plaintext $\mu \in \{0,1\}$, the encryption algorithm outputs a ciphertext ct.

- $\mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots \mathsf{ct}_k) \to \hat{\mathsf{ct}}$: 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 $\hat{\mathsf{ct}}$.

- $\mathsf{TFHE.PartDec}(\mathsf{pk}, \mathsf{ct}, \mathsf{sk}_i) \to \mathsf{p}_i$: On input a public key pk, a ciphertext ct, and a secret key share sk_i, the partial decryption algorithm outputs a partial decryption p_i related to the party P_i.

- $\mathsf{TFHE.FinDec}(\mathsf{pk}, B) \to \hat{\mu}$: On input a public key pk, and a set $B = \{\mathsf{p}_i\}_{i \in S}$ for some $S \subseteq \{P_1, \ldots, P_N\}$, the final decryption algorithm outputs a plaintext $\hat{\mu} \in \{0, 1, \perp\}$.

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

Definition 5.2 (Compactness)**.** We say that 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 $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A})$, $\mathsf{ct}_i \leftarrow \mathsf{TFHE.Encrypt}(\mathsf{pk}, \mu_i)$ for $i \in [k]$, $\hat{\mathsf{ct}} \leftarrow \mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$, $\mathsf{p}_j \leftarrow \mathsf{TFHE.PartDec}(\mathsf{pk}, \mathsf{ct}, \mathsf{sk}_j)$ for any $j \in [N]$, $|\hat{\mathsf{ct}}| \leq \mathsf{poly}(\lambda, d)$ and $|\mathsf{p}_j| \leq \mathsf{poly}(\lambda, d, N)$.

Definition 5.3 (Evaluation Correctness)**.** We say that a TFHE scheme satisfies evaluation correctness if for all λ, depth bound d, access structure $\mathbb{A}$, 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 $i \in [k]$, $\hat{\mathsf{ct}} \leftarrow \mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$,

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

Definition 5.4 (Semantic Security)**.** We say that 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 $b \xleftarrow{\mathrm{R}} \{0,1\}$ 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 5.5 (Simulation Security)**.** We say that 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 $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}(1^\lambda, 1^d)$ and $\mathsf{Expt}_{\mathcal{A},\mathsf{Ideal}}(1^\lambda, 1^d)$ are indistinguishable:

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

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 $\{\mathsf{TFHE.Encrypt}(\mathsf{pk}, \mu_i)\}_{i \in [k]}$ 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 $C : \{0,1\}^k \rightarrow \{0,1\}$ of depth at most d. For each query, the challenger computes $\hat{\mathsf{ct}} \leftarrow \mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ and provides $\{\mathsf{TFHE.PartDec}(\mathsf{pk}, \hat{\mathsf{ct}}, \mathsf{sk}_i)\}_{i \in S}$ to $\mathcal{A}$.
6. At the end of the experiment, $\mathcal{A}$ outputs a distinguishing bit b.

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

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 $\{\mathsf{TFHE.Encrypt}(\mathsf{pk}, \mu_i)\}_{i \in [k]}$ 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 $C : \{0,1\}^k \rightarrow \{0,1\}$ of depth at most d. For each query, the challenger runs the simulator $\{\mathsf{p}_i\}_{i \in S} \leftarrow \mathcal{S}_2(C, \{\mathsf{ct}_1, \ldots, \mathsf{ct}_k\}, C(\mu_1, \ldots, \mu_k), S, \mathsf{st})$ and sends $\{\mathsf{p}_i\}_{i \in S}$ to $\mathcal{A}$.
6. At the end of the experiment, $\mathcal{A}$ outputs a distinguishing bit b.

5.2 TFHE using $\{0,1\}$-LSSS

In this section, we present our construction of TFHE for the class of access structures $\{0,1\}$-LSSS. We note that by Theorem 4.15, this gives a TFHE scheme for the class of threshold access structures TAS.

Construction 5.6. Let $P = \{P_1, \ldots, P_N\}$ be a set of parties. Our TFHE construction relies on the following primitives:

- Let $\mathsf{FHE} = (\mathsf{FHE.Setup}, \mathsf{FHE.Encrypt}, \mathsf{FHE.Eval}, \mathsf{FHE.Decrypt})$ be a special fully homomorphic encryption scheme with noise bound $B = B(\lambda, d, q)$ and multiplicative constant 1 (Definition 3.9).

- Let $\mathsf{SS} = (\mathsf{SS.Share}, \mathsf{SS.Combine})$ be a special linear secret sharing scheme (Definition 4.13). We use T_i to denote a partition of the share matrix and use $\{\mathbf{s}_j\}_{j \in T_i}$ to denote a share associated with P_i consisting of elements in $\mathbb{Z}_q$. We also use $\ell = \ell(\lambda, N)$ to denote a fixed polynomial bound on the size of the share: $|T_i| \leq \ell$ for all $i \in [N]$.

We also fix a parameter B_{sm} that specifies the bound on the smuding noise (see Section 5.2.1). We construct $\mathsf{TFHE} = (\mathsf{TFHE.Setup}, \mathsf{TFHE.Encrypt}, \mathsf{TFHE.Eval}, \mathsf{TFHE.PartDec}, \mathsf{TFHE.FinDec})$ as follows:

- $\mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A})$: On input the security parameter λ, depth bound d, and an access structure $\mathbb{A}$, the setup algorithm generates the FHE keys $(\mathsf{fhepk}, \mathsf{fhesk}) \leftarrow \mathsf{FHE.Setup}(1^\lambda, 1^d)$.

Then, it divides the key fhesk into shares $(\mathsf{fhesk}_1, \dots, \mathsf{fhesk}_N) \leftarrow \mathsf{SS.Share}(\mathsf{fhesk}, \mathbb{A})$. It sets $\mathsf{pk} = \mathsf{fhepk}$ and $\mathsf{sk}_i = \mathsf{fhesk}_i$ for $i = 1, \dots, N$.

- $\mathsf{TFHE.Encrypt}(\mathsf{pk}, \mu)$: On input the public key pk, and a message $\mu \in \{0, 1\}$, the encryption algorithm computes $\mathsf{ct} \leftarrow \mathsf{FHE.Encrypt}(\mathsf{pk}, \mu)$ and outputs ct.
- $\mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \dots, \mathsf{ct}_k)$: On input a public key pk, a circuit C, and a set of ciphertexts $\mathsf{ct}_1, \dots, \mathsf{ct}_k$ the evaluation algorithm computes $\hat{\mathsf{ct}} \leftarrow \mathsf{FHE.Eval}(C, \mathsf{ct}_1, \dots, \mathsf{ct}_k)$ and outputs $\hat{\mathsf{ct}}$.
- $\mathsf{TFHE.PartDec}(\mathsf{pk}, \mathsf{ct}, \mathsf{sk}_i)$: On input a public key pk, a ciphertext ct, and a decryption key share $\mathsf{sk}_i = \{\mathbf{s}_j\}_{j \in T_i}$ for each $\mathbf{s}_j \in \mathbb{Z}_q^n$, the partial decryption algorithm samples a smudging error $e_j \overset{\text{R}}{\leftarrow} [-B_{\mathsf{sm}}, B_{\mathsf{sm}}]$ and computes $\tilde{\mathbf{p}}_j = \mathsf{FHE.Decode}_0(\mathbf{s}_j, \mathsf{ct}) + e_j \in \mathbb{Z}_q$ for $j \in T_i$. It outputs the set $\mathsf{p}_i = \{\tilde{\mathbf{p}}_j\}_{j \in T_i}$ as its partial decryption.
- $\mathsf{TFHE.FinDec}(\mathsf{pk}, B)$: On input a public key pk and a set of partial decryption shares $\{\mathsf{p}_i\}_{i \in S}$, it first checks if $S \in \mathbb{A}$. If this is not the case, then it outputs $\perp$. Otherwise, it computes a minimal valid share set (Definition 4.9) $T \subseteq \bigcup_{i \in S} T_i$ and computes $\mu \leftarrow \mathsf{FHE.Decode}_1\big(\sum_{j \in T} \tilde{\mathbf{p}}_j\big)$. It outputs μ.

We now state the compactness, correctness, and security theorems for Construction 5.6.

Theorem 5.7. *Suppose* FHE *is a compact fully homomorphic encryption scheme (Definition 3.6). Then, the* TFHE *scheme from Construction 5.6 satisfies compactness (Definition 5.2).*

Theorem 5.8. *Suppose* FHE *is a special fully homomorphic encryption scheme that satisfies correctness (Definition 3.7) with noise bound B and* SS *is a secret sharing scheme that satisfies correctness (Definition 4.5). Then, the* TFHE *scheme from Construction 5.6 with parameter B_{sm} such that $B + \ell \cdot B_{\mathsf{sm}} \leq \lfloor \frac{q}{4} \rceil$ satisfies evaluation correctness (Definition 5.3).*

Theorem 5.9. *Suppose* FHE *is a fully homomorphic encryption scheme that satisfies security (Definition 3.8). Then, the* TFHE *scheme from Construction 5.6 satisfies semantic security (Definition 5.4).*

Theorem 5.10. *Suppose* FHE *is a fully homomorphic encryption scheme that satisfies security (Definition 3.8) and* SS *is a secret sharing scheme that satisfies security (Definition 4.6). Then, the* TFHE *scheme from Construction 5.6 with parameter B_{sm} such that $B / B_{\mathsf{sm}} = \mathsf{negl}(\lambda)$ satisfies simulation security (Definition 5.5).*

The compactness and semantic security of Construction 5.6 (Theorems 5.7 and 5.9) follow from the compactness and security of the underlying FHE and SS schemes in a straightforward way. We provide the formal proofs of evaluation correctness and simulation security (Theorems 5.8 and 5.10) in Section D.

5.2.1 Parameter Instantiation

For correctness and security, we require the parameters to satisfy:

- $B + \ell \cdot B_{\mathsf{sm}} \leq \frac{q}{4}$ (Theorem 5.8).
- $B / B_{\mathsf{sm}} = \mathsf{negl}(\lambda)$ (Theorem 5.10).

By Theorem 3.10, for a depth bound d, there exists a special FHE scheme with an associated noise bound $B = 2^{\tilde{O}(d)}$ assuming the hardness of $\mathsf{LWE}(n, m, q, \chi)$ for $B_{\mathsf{lwe}} = \mathsf{poly}(\lambda)$ and $q = 2^{\tilde{O}(d)}$. Then, if we set $B_{\mathsf{sm}} = 2^{\tilde{O}(d)-\omega(1)}$, the two conditions above are satisfied. In particular, this translates to approximating worst-case lattice problems with sub-exponential approximation factors.

5.3 TFHE from Shamir secret sharing

In this section, we present our construction of TFHE using a standard Shamir secret sharing scheme. This construction does not satisfy our notion of compactness 5.2. However, in Section 8.4, we show how to transform a non-compact TFHE scheme to a compact one generically using UT.

Construction 5.11. Let $P = \{P_1, \ldots, P_N\}$ be a set of parties. Our TFHE construction relies on the following primitives:

- Let $\mathsf{FHE} = (\mathsf{FHE.Setup}, \mathsf{FHE.Encrypt}, \mathsf{FHE.Eval}, \mathsf{FHE.Decrypt})$ be a special fully homomorphic encryption scheme with noise bound $B = B(\lambda, d, q)$ and multiplicative constant $(N!)^2$ (Definition 3.9).
- Let $\mathsf{SS} = (\mathsf{SS.Share}, \mathsf{SS.Combine})$ be a Shamir secret sharing scheme (Theorem 4.11).

We also fix a parameter B_{sm} that specifies the bound on the smudging noise (see Section 5.3.1). We construct $\mathsf{TFHE} = (\mathsf{TFHE.Setup}, \mathsf{TFHE.Encrypt}, \mathsf{TFHE.Eval}, \mathsf{TFHE.PartDec}, \mathsf{TFHE.FinDec})$ as follows:

- $\mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}_t)$: On input the security parameter λ, depth bound d, and an access structure $\mathbb{A}_t \in \mathsf{TAS}$, the setup algorithm generates the FHE keys $(\mathsf{fhepk}, \mathsf{fhesk}) \leftarrow \mathsf{FHE.Setup}(1^\lambda, 1^d)$. Then, it divides the key fhesk into shares using Shamir secret sharing $(\mathsf{fhesk}_1, \ldots, \mathsf{fhesk}_N) \leftarrow \mathsf{SS.Share}(\mathsf{fhesk}, \mathbb{A}_t)$. It sets $\mathsf{pk} = \mathsf{fhepk}$ and $\mathsf{sk}_i = \mathsf{fhesk}_i \in \mathbb{Z}_q^n$ for $i = 1, \ldots, N$.
- $\mathsf{TFHE.Encrypt}(\mathsf{pk}, \mu)$: On input the public key pk, and a message $\mu \in \{0, 1\}$, the encryption algorithm computes $\mathsf{ct} \leftarrow \mathsf{FHE.Encrypt}(\mathsf{pk}, \mu)$ and outputs ct.
- $\mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$: On input a public key pk, a circuit C, and a set of ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$ the evaluation algorithm computes $\hat{\mathsf{ct}} \leftarrow \mathsf{FHE.Eval}(C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ and outputs $\hat{\mathsf{ct}}$.
- $\mathsf{TFHE.PartDec}(\mathsf{pk}, \mathsf{ct}, \mathsf{sk}_i)$: On input a public key pk, a ciphertext ct, and a decryption key share $\mathsf{sk}_i \in \mathbb{Z}_q^n$, the partial decryption algorithm samples a smudging error $e \overset{\mathrm{R}}{\leftarrow} [-B_{\mathsf{sm}}, B_{\mathsf{sm}}]$ and computes $\mathsf{p}_i = \mathsf{FHE.Decode}_0(\mathsf{sk}_i, \mathsf{ct}) + (N!)^2 \cdot e \in \mathbb{Z}_q$. It outputs p_i.
- $\mathsf{TFHE.FinDec}(\mathsf{pk}, B)$: On input a public key pk and a set of partial decryption shares $\{\mathsf{p}_i\}_{i \in S}$, it first checks if $S \in \mathbb{A}$. If this is not the case, then it output $\perp$. Otherwise, it arbitrary chooses a satisfying set $S' \subseteq S$ of size t and computes the Lagrange coefficients $\lambda_{i,0}^{S'}$ for all $i \in S'$. Then, it computes $\mu \leftarrow \mathsf{FHE.Decode}_1\big(\sum_{i \in S'} \lambda_{i,0}^{S'} \cdot \mathsf{p}_i\big)$, and outputs μ.

We now state the correctness and security theorems for Construction 5.11.

Theorem 5.12. *Suppose* FHE *is a compact fully homomorphic encryption scheme (Definition 3.7) with noise bound B and* SS *is a Shamir secret sharing scheme that satisfies correctness (Theorem 4.11). Then, the* TFHE *scheme from Construction 5.11 with parameter $B + (N!)^3 \cdot N \cdot B_{\mathsf{sm}} \leq \frac{q}{4}$ satisfies evaluation correctness (Definition 5.3).*

Theorem 5.13. *Suppose* FHE *is a fully homomorphic encryption scheme that satisfies security (Definition 3.8). Then, the* TFHE *scheme from Construction 5.6 satisfies semantic security (Definition 5.4).*

Theorem 5.14. *Suppose* FHE *is a fully homomorphic encryption scheme that satisfies security (Definition 3.8) and* SS *is a secret sharing scheme that satisfies security (Definition 4.6). Then, the* TFHE *scheme from Construction 5.11 with parameter B_{sm} such that $B/B_{\mathsf{sm}} = \mathsf{negl}(\lambda)$ satisfies simulation security (Definition 5.5).*

The semantic security of Construction 5.11 (Theorem 5.13) follows from the semantic security of the underlying FHE in a straightforward way. We provide the formal proofs of evaluation correctness and simulation security (Theorems 5.8 and 5.10) in Section D.

5.3.1 Parameter Instantiation

For correctness and security, we require the parameters to satisfy:

- $B + (N!)^3 \cdot N \cdot B_{\mathsf{sm}} \leq \frac{q}{4}$ (Theorem 5.12).
- $B/B_{\mathsf{sm}} = \mathsf{negl}(\lambda)$ (Theorem 5.14).

By Theorem 3.10, for a depth bound d, there exists a special FHE scheme with an associated noise bound $B = 2^{\tilde{O}(d)}$ assuming the hardness of $\mathsf{LWE}(n, m, q, \chi)$ for $B_{\mathsf{lwe}} = \mathsf{poly}(\lambda)$ and $q = 2^{\tilde{O}(d)}$. Then, if we set $B_{\mathsf{sm}} = 2^{\tilde{O}(d)-\omega(1)}/(N!)^3$, the two conditions above are satisfied. In particular, this translates to approximating worst-case lattice problems with sub-exponential approximation factors.

5.4 Performance Comparisons

We briefly describe the performance tradeoffs between our two constructions. The main advantage of Construction 5.6 is the compact size of the ciphertext, which has no additional overhead compared to any (non-threshold) FHE ciphertext. The main drawback of using $\{0,1\}$-LSSS, however, is the blow up of the key share sizes. To represent a threshold access structure in terms of monotone Boolean formula, we require a formula of size $O(N^{5.2})$ [Val84, Gol14]. This means that when translated to $\{0,1\}$-LSSS, each decryption key share consists of $O(N^{4.2})$ equivalent of a regular FHE keys on average, which is a significant space overhead.

Construction 5.11, compared to a regular (non-threshold) FHE scheme, does not introduce any additional overhead to the size of the ciphertext and the decryption key shares with respect to the *number of components*. However, the main overhead is in the increase of the modulus, which increases by $O(N!^3)$ multiplicative factor. This means that the representation of the ciphertext and the key shares increase additively by $O(N \log N)$. In terms of the size of the key shares, this is a significantly more compact than the share sizes in Construction 5.6. However, the size of the ciphertext now depends on N, thereby violating our compactness requirement (Definition 5.2).

In Section 8.4, we show how to convert a *non-compact* TFHE scheme to a *compact* TFHE scheme. The idea is to construct a *non-compact* universal thresholdizer (Definition 7.1) from a *non-compact* TFHE, and then use the universal thresholdizer to thresholdize a *compact* (non-threshold) FHE scheme. Applying this transformation to Construction 5.11 gives a construction where there is no additional overhead in the ciphertext size and only $O(N \log N)$ overhead to the key share sizes.

6 Decentralized TFHE

In Section 5, we defined the notion of a threshold fully homomorphic encryption scheme to have a central setup. Namely, the setup algorithm takes in an access structure $\mathbb{A}$ for a fixed set of parties as input and produces a set of decryption key shares $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$ for the servers. In practice, the set of parties that participate in the decryption protocol can always change and the access structure updated. When using a standard TFHE scheme in this dynamic setting, a trusted setup algorithm must be run each time a new decryption server enters or leaves a protocol.

In this section, we define and construct an extension to the notion of TFHE that we name *decentralized threshold fully homomorphic encryption* (dTFHE). In a dTFHE scheme, there is no setup algorithm. Rather, each party can generate its own $(\mathsf{pk}_i, \mathsf{sk}_i)$ key pair from a public key encryption scheme of its choice. The encryption algorithm then takes in a set of public keys $\{\mathsf{pk}_i\}_{i \in [N]}$ and an access structure $\mathbb{A}$ to encrypt to a message x. A ciphertext that is generated in this way can only be decrypted with a set of keys $\{\mathsf{sk}_i\}_{i \in S}$ for a satisfying set $S \in \mathbb{A}$.

6.1 Definition

In this subsection, we define our notion of decentralized fully homomorphic encryption. To capture the fact that a party can use any general public key encryption scheme, we allow the dTFHE encryption algorithm to take in the actual PKE encryption algorithms of party P_i denoted Enc_i. We assume that Enc_i consists of the description of the PKE encryption algorithm as well as a hardcoded public key pk_i. We denote a decryption algorithm by Dec_i similarly.

Definition 6.1. A decentralized threshold fully homomorphic encryption scheme for $\mathbb{S}$ is a tuple of PPT algorithms $\mathsf{dTFHE} = (\mathsf{dTFHE.Encrypt}, \mathsf{dTFHE.Eval}, \mathsf{dTFHE.PartDec}, \mathsf{dTFHE.FinDec})$ with the following properties:

- $\mathsf{dTFHE.Encrypt}(1^\lambda, 1^d, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_N, \mathbb{A}, x) \to \mathsf{ct}$: On input the security parameter λ, a depth bound d, a set of encryption algorithms $\mathsf{Enc}_1, \ldots, \mathsf{Enc}_N$, an access structure $\mathbb{A}$ on $\{P_1, \ldots, P_N\}$, and a message $x \in \{0,1\}^k$, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{dTFHE.Eval}(C, \mathsf{ct}) \to \hat{\mathsf{ct}}$: On input a circuit $C : \{0,1\}^k \to \{0,1\}$, and a ciphertext ct, the evaluation algorithm outputs an evaluated ciphertext $\hat{\mathsf{ct}}$.
- $\mathsf{dTFHE.PartDec}(\hat{\mathsf{ct}}, \mathsf{Dec}_i)$: On input a ciphertext $\hat{\mathsf{ct}}$, and a secret key sk_i, the partial decryption algorithm outputs a partial decryption p_i associated with party P_i.
- $\mathsf{dTFHE.FinDec}(B)$: On input a set of partial decryptions $\{\mathsf{p}_i\}_{i \in S}$, the final decryption algorithm outputs a message x'.

We require a dTFHE scheme to satisfy the following compactness, correctness, and security properties. We note that our compactness notion for dTFHE is weaker than Definition 5.2 as we allow the size of an evaluated ciphertext to depend on N.

Definition 6.2 (Weak Compactness)**.** We say that a dTFHE scheme for $\mathbb{S}$ is compact if there exists a polynomial $\mathsf{poly}(\cdot)$ such that for all λ, depth bound d, circuit $C : \{0,1\}^k \to \{0,1\}$ of depth at most d, encryption algorithms Enc_i for $i \in [N]$, access structure $\mathbb{A} \in \mathbb{S}$, and $x \in \{0,1\}^k$, the following holds. For $\mathsf{ct} \leftarrow \mathsf{dTFHE.Encrypt}(1^\lambda, 1^d, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_N, \mathbb{A}, x)$, $\hat{\mathsf{ct}} \leftarrow \mathsf{dTFHE.Eval}(C, \mathsf{ct})$, and $\mathsf{p}_i \leftarrow \mathsf{dTFHE.PartDec}(\hat{\mathsf{ct}}, \mathsf{Dec}_i)$ for $i \in [N]$, we have $|\hat{\mathsf{ct}}|, |\mathsf{p}_i| \leq \mathsf{poly}(\lambda, d, N)$.

Definition 6.3 (Evaluation Correctness)**.** We say that a dTFHE scheme for $\mathbb{S}$ satisfies evaluation correctness if for all λ, depth bound d, circuit $C : \{0,1\}^k \to \{0,1\}$ of depth at most d, correct encryption and decryption algorithms $(\mathsf{Enc}_i, \mathsf{Dec}_i)$ for $i \in [N]$, access structure $\mathbb{A} \in \mathbb{S}$, and $x \in \{0,1\}^k$, the following holds. For $\mathsf{ct} \leftarrow \mathsf{dTFHE.Encrypt}(1^\lambda, 1^d, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_N, \mathbb{A}, x)$, and $\hat{\mathsf{ct}}, \leftarrow \mathsf{dTFHE.Eval}(C, \mathsf{ct})$, we have

$$\Pr[\mathsf{dTFHE.FinDec}(\{\mathsf{dTFHE.PartDec}(\hat{\mathsf{ct}}, \mathsf{Dec}_i)\}_{i \in S}) = C(x)] = 1 - \mathsf{negl}(\lambda).$$

Definition 6.4 (Semantic Security)**.** We say that a dTFHE scheme for $\mathbb{S}$ satisfies semantic security if for all λ, depth bound d, and secure encryption algorithms Enc_i for $i \in [N]$, the following holds. For any PPT adversary $\mathcal{A}$, the following experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{dTFHE},\mathsf{sem}}(1^\lambda, 1^d, \{\mathsf{Enc}_i\}_{i \in [N]})$ outputs 1 with negligible probability:

$\mathsf{Expt}_{\mathcal{A},\mathsf{dTFHE},\mathsf{sem}}(1^\lambda, 1^d, \{\mathsf{Enc}_i\}_{i \in [N]})$:

1. On input the security parameter 1^λ, depth bound 1^d, and encryption algorithms $\{\mathsf{Enc}_i\}_{i \in [N]}$, the challenger provides $\mathsf{Enc}_1, \ldots, \mathsf{Enc}_N$ to $\mathcal{A}$.
2. $\mathcal{A}$ outputs an access structure $\mathbb{A} \in \mathbb{S}$, a pair of messages $x_0, x_1 \in \{0,1\}^k$, and an unsatisfying set $S \subseteq \{P_1, \ldots, P_N\}$.
3. The challenger encrypts $\mathsf{ct}_b \leftarrow \mathsf{dTFHE.Encrypt}(1^\lambda, 1^d, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_N, \mathbb{A}, x_b)$ for $b \xleftarrow{\text{R}} \{0,1\}$ and sends it to $\mathcal{A}$ along with $\{\mathsf{Dec}_i\}_{i \in S}$.
4. $\mathcal{A}$ outputs its guess b'. The experiment outputs 1 if $b' = b$.

Definition 6.5 (Simulation Security)**.** We say that a dTFHE scheme for $\mathbb{S}$ satisfies simulation security if for all λ, depth bound d, and secure encryption and decryption algorithms $(\mathsf{Enc}_i, \mathsf{Dec}_i)$ for $i \in [N]$, the following holds. There exists a stateful simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for any PPT adversary $\mathcal{A}$, the following two experiments $\mathsf{Expt}_{\mathcal{A},\mathsf{dTFHE},\mathsf{Real}}(1^\lambda, 1^d, \{(\mathsf{Enc}_i, \mathsf{Dec}_i)\}_{i \in [N]})$ and $\mathsf{Expt}_{\mathcal{A},\mathsf{dTFHE},\mathsf{Ideal}}(1^\lambda, 1^d, \{(\mathsf{Enc}_i, \mathsf{Dec}_i)\}_{i \in [N]})$ are computationally indistinguishable:

$\mathsf{Expt}_{\mathcal{A},\mathsf{dTFHE},\mathsf{Real}}(1^\lambda, 1^d, \{(\mathsf{Enc}_i, \mathsf{Dec}_i)\}_{i \in [N]})$:

1. On input the security parameter 1^λ, depth bound 1^d, and a set of algorithms $\{(\mathsf{Enc}_i, \mathsf{Dec}_i)\}_{i \in [N]}$, the challenger provides $\mathsf{Enc}_1, \ldots, \mathsf{Enc}_N$ to $\mathcal{A}$.
2. $\mathcal{A}$ outputs an access structure $\mathbb{A}$, a message $x \in \{0,1\}^k$, and a maximal invalid party set $S^* \subseteq \{P_1, \ldots, P_N\}$.
3. The challenger encrypts $\mathsf{ct} \leftarrow \mathsf{dTFHE.Encrypt}(1^\lambda, 1^d, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_N, \mathbb{A}, x)$ and provides $(\mathsf{ct}, \{\mathsf{Dec}_i\}_{i \in S^*})$ to $\mathcal{A}$.
4. $\mathcal{A}$ 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 $\hat{\mathsf{ct}} \leftarrow \mathsf{dTFHE.Eval}(C, \mathsf{ct})$ and provides $\{\mathsf{dTFHE.PartDec}(\hat{\mathsf{ct}}, \mathsf{Dec}_i)\}_{i \in S}$ to $\mathcal{A}$.
5. At the end of the experiment, $\mathcal{A}$ outputs a distinguishing bit b.

$\mathsf{Expt}_{\mathcal{A},\mathsf{dTFHE},\mathsf{Ideal}}(1^\lambda, 1^d, \{(\mathsf{Enc}_i, \mathsf{Dec}_i)\}_{i \in [N]})$:

1. On input the security parameter 1^λ, depth bound 1^d, and a set of algorithms $\{(\mathsf{Enc}_i, \mathsf{Dec}_i)\}_{i \in [N]}$, the challenger provides $\mathsf{Enc}_1, \ldots, \mathsf{Enc}_N$ to $\mathcal{A}$.
2. $\mathcal{A}$ outputs an access structure $\mathbb{A}$, a message $X \in \{0,1\}^k$, and a maximal invalid party set $S^* \subseteq \{P_1, \ldots, P_N\}$.
3. The challenger computes $(\mathsf{ct}, \mathsf{st}) \leftarrow \mathcal{S}_1(1^\lambda, 1^d, \{\mathsf{Enc}_i\}_{i \in [N]}, \{\mathsf{Dec}_i\}_{i \in S^*}, \mathbb{A})$ and provides $(\mathsf{ct}, \{\mathsf{Dec}_i\}_{i \in S^*})$ to $\mathcal{A}$.

4. $\mathcal{A}$ 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 $\{\mathsf{p}_i\}_{i \in S} \leftarrow \mathcal{S}_2(C, C(x), \mathsf{st})$ and sends $\{\mathsf{p}_i\}_{i \in S}$ to $\mathcal{A}$.
5. At the end of the experiment, $\mathcal{A}$ outputs a distinguishing bit b.

6.2 Construction

We construct a decentralized threshold fully homomorphic encryption scheme from a standard TFHE (Section 5.1).

Construction 6.6. Our decentralized fully homomorphic encryption scheme relies on the following primitives:

- Let $\mathsf{TFHE} = (\mathsf{TFHE.Setup}, \mathsf{TFHE.Encrypt}, \mathsf{TFHE.Eval}, \mathsf{TFHE.PartDec}, \mathsf{TFHE.FinDec})$ be a threshold fully homomorphic encryption scheme for the class of access structures $\mathbb{S}$.

We construct a dynamic threshold fully homomorphic encryption scheme $\mathsf{dTFHE} = (\mathsf{dTFHE.Encrypt}, \mathsf{dTFHE.Eval}, \mathsf{dTFHE.PartDec}, \mathsf{dTFHE.FinDec})$ for $\mathbb{S}$ as follows:

- $\mathsf{dTFHE.Encrypt}(1^\lambda, 1^d, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_N, \mathbb{A}, x)$: On input the security parameter λ, depth bound d, a set of encryption algorithms $\mathsf{Enc}_1, \ldots, \mathsf{Enc}_N$, an access structure $\mathbb{A}$ and a message $x \in \{0,1\}^k$, the encryption algorithm first generates the keys $(\mathsf{tfhepk}, \mathsf{tfhesk}_1, \ldots, \mathsf{tfhesk}_N) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A})$. Then, it encrypts the message using TFHE as $\tilde{\mathsf{ct}}_i \leftarrow \mathsf{TFHE.Encrypt}(\mathsf{pk}, x_i)$ and encrypts each key share $\mathsf{ct}_i \leftarrow \mathsf{Enc}_i(\mathsf{tfhesk}_i)$. It sets

$$\mathsf{ct} = (\mathsf{tfhepk}, \{\mathsf{ct}'_i\}_{i \in [k]}, \{\mathsf{ct}_i\}_{i \in [N]}).$$

- $\mathsf{dTFHE.Eval}(C, \mathsf{ct})$: On input a circuit $C : \{0,1\}^k \to \{0,1\}$, and a ciphertext $\mathsf{ct} = (\mathsf{tfhepk}, \{\mathsf{ct}'_i\}_{i \in [k]}, \{\mathsf{ct}_i\}_{i \in [N]})$, the evaluation algorithm computes $\hat{\mathsf{ct}}' \leftarrow \mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$, and sets $\hat{\mathsf{ct}} = (\mathsf{tfhepk}, \hat{\mathsf{ct}}', \{\mathsf{ct}_i\}_{i \in [N]})$.

- $\mathsf{dTFHE.PartDec}(\hat{\mathsf{ct}}, \mathsf{Dec}_i)$: On input a ciphertext $\hat{\mathsf{ct}} = (\mathsf{pk}, \hat{\mathsf{ct}}', \{\mathsf{ct}_i\}_{i \in [N]})$, and a secret key sk_i, the partial decryption algorithm decrypts $\mathsf{tfhesk}_i \leftarrow \mathsf{Dec}_i(\mathsf{ct}_i)$, computes $\mathsf{p}_i \leftarrow \mathsf{TFHE.PartDec}($ $\mathsf{tfhepk}, \mathsf{tfhesk}_i, \hat{\mathsf{ct}})$, and returns p_i.

- $\mathsf{dTFHE.FinDec}(B)$: On input a set of partial decryptions $\{\mathsf{p}_i\}_{i \in S}$, the final decryption algorithm computes $m \leftarrow \mathsf{TFHE.FinDec}(\mathsf{tfhepk}, \{\mathsf{p}_i\}_{i \in S})$.

We now state the compactness, correctness, and security theorems for Construction 6.6.

Theorem 6.7. *Suppose* TFHE *is a compact threshold fully homomorphic encryption scheme (Definition 5.2). Then, the* dTFHE *scheme from Construction 6.6 satisfies weak compactness (Definition 6.2).*

Theorem 6.8. *Suppose* TFHE *is a threshold fully homomorphic encryption scheme that satisfies evaluation correctness (Definition 5.3). Then, the* dTFHE *scheme from Construction 6.6 satisfies evaluation correctness (Definition 6.3).*

Theorem 6.9. *Suppose* TFHE *is a threshold fully homomorphic encryption scheme that satisfies semantic security (Definition 5.4). Then, the* dTFHE *scheme from Construction 6.6 satisfies semantic security (Definition 6.4).*

Theorem 6.10. *Suppose* TFHE *is a threshold fully homomorphic encryption scheme that satisfies simulation security (Definition 5.4). Then, the* dTFHE *scheme from Construction 6.6 satisfies simulation security (Definition 6.5).*

The proofs of the theorems above follow immediately from the properties of the underlying TFHE scheme.

6.3 Multi-Key TFHE

We note that a threshold variant of a multi-key fully homomorphic encryption can be defined as a natural extension of a regular TFHE (Definition 5.1). In fact, our decentralized threshold fully homomorphic encryption scheme can be seen as a special case of a multi-key TFHE. In a multi-key TFHE, one can homomorphically evaluate on ciphertexts that are encrypted under the same access pattern $\mathbb{A}$, but with different keys. Combining the existing multi-key fully homomorphic encryption schemes [CM15, MW16] with the dTFHE scheme above, a multi-key TFHE can be constructed in a straightforward way.

7 Universal Thresholdizer

The notion of TFHE is a natural generalization of a standard fully homomorphic encryption scheme that has numerous applications in threshold cryptography. Specifically, it can be used to *generically* construct a *thresholdized* variant of any basic cryptographic function. For these type of applications, it is natural to view the notion of TFHE as a *thresholdizer* mechanism. In these settings, we do not require the full generality of the TFHE syntax. Furthermore, for TFHE to be useful as a thresholdizer tool, we require it to be robust, meaning that there exists an efficient public mechanism to verify whether a partial decryption was done correctly. Therefore, we define a natural notion of *universal thresholdizer* (UT) that captures these properties. We use universal thresholdizer for our applications in Section 8.

7.0.1 Definition

Informally, the setup and the encryption algorithms for TFHE is merged into a single UT setup algorithm, and the evaluation and partial decryption algorithms for TFHE is merged into a single UT evaluation algorithm. Furthermore, semantic security (Definition 5.4) and simulation security (Definition 5.5) is merged into a single definition for simplicity. Finally, there is an additional verification algorithm that checks whether an evaluation was done correctly.

Definition 7.1 (Universal Thresholdizer)**.** 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 universal thresholdizer scheme for $\mathbb{S}$ and $\mathcal{M}$ is a tuple of PPT algorithms $\mathsf{UT} = (\mathsf{UT.Setup}, \mathsf{UT.Eval}, \mathsf{UT.Verify}, \mathsf{UT.Combine})$ with the following properties:

- $\mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}, x) \to (\mathsf{pp}, \mathsf{s}_1, \ldots, \mathsf{s}_N)$: On input the security parameter λ, a depth bound d, an access structure $\mathbb{A}$, and a message $x \in \{0,1\}^k$, the setup algorithm outputs the public parameters pp, and a set of shares $\mathsf{s}_1, \ldots, \mathsf{s}_N$.
- $\mathsf{UT.Eval}(\mathsf{pp}, \mathsf{s}_i, C) \to \mathsf{y}_i$: 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.

- $\mathsf{UT.Verify}(\mathsf{pp}, \mathsf{y}_i, C) \to \{0,1\}$: 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.
- $\mathsf{UT.Combine}(\mathsf{pp}, B) \to \mathsf{y}$: On input the public parameters pp, a set of partial evaluations $B = \{\mathsf{y}_i\}_{i \in S}$, the combining algorithm outputs the final evaluation y.

We require a UT scheme satisfy the following compactness, correctness, and security properties. The compactness and evaluation correctness definitions are natural analogues of the TFHE definitions. The security requirement of a TFHE scheme combines the semantic and simulation security definitions of TFHE. Verification correctness and robustness are additions to the definition to capture verifiable evalaution.

Definition 7.2 (Compactness)**.** We say that a UT scheme is compact if there exists a polynomial $\mathsf{poly}(\cdot)$ 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 $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}, x)$, $\mathsf{y}_i \leftarrow \mathsf{UT.Eval}(\mathsf{pp}, \mathsf{s}_i, C)$ for any $i \in [N]$, we have $|\mathsf{y}_i| \leq \mathsf{poly}(\lambda, d, N)$.

Definition 7.3 (Evaluation Correctness)**.** We say that a UT scheme satisfies evaluation correctness if for all λ, depth bound d, access structure $\mathbb{A}$, message $x \in \{0,1\}^k$, circuit $C : \{0,1\}^k \to \{0,1\}$ of depth at most d, and $S \in \mathbb{A}$, the following condition holds. For $(\mathsf{pp}, \mathsf{s}_1, \ldots, \mathsf{s}_N) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}, x)$,

$$\Pr[\mathsf{UT.Combine}(\mathsf{pp}, \{\mathsf{UT.Eval}(\mathsf{pp}, \mathsf{s}_i, C)\}_{i \in S}) = C(x)] = 1 - \mathsf{negl}(\lambda).$$

Definition 7.4 (Verification Correctness)**.** We say that a UT scheme satisfies verification correctness if for all λ, depth bound d, access structure $\mathbb{A}$, message $x \in \{0,1\}^k$, and circuit $C : \{0,1\}^k \to \{0,1\}$ of depth at most d, the following holds. For $(\mathsf{pp}, \mathsf{s}_1, \ldots, \mathsf{s}_N) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}, x)$, $\mathsf{y}_i \leftarrow \mathsf{UT.Eval}(\mathsf{pp}, \mathsf{s}_i, C)$ for any $i \in [N]$, we have that

$$\Pr[\mathsf{UT.Verify}(\mathsf{pp}, \mathsf{y}_i, C) = 1] = 1.$$

Definition 7.5 (Security)**.** We say that a UT scheme satisfies security if for all λ, and depth bound d, the following holds. There exists a stateful PPT algorithm $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for any 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 $\mathbb{A} \in \mathbb{S}$, and a message $x \in \{0,1\}^k$.
2. The challenger runs $(\mathsf{pp}, \mathsf{s}_1, \ldots, \mathsf{s}_N) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}, \underline{x})$ and provides pp to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a maximal invalid party set $S^* \subseteq \{P_1, \ldots, P_N\}$ for $\mathbb{A}$.
4. The challenger provides the shares $\{\mathsf{s}_i\}_{i \in S^*}$ 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 $\{\mathsf{y}_i \leftarrow \mathsf{UT.Eval}(\mathsf{pp}, \mathsf{s}_i, C)\}_{i \in S}$ 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 a circuit depth 1^d, the adversary $\mathcal{A}$ outputs an access structure $\mathbb{A} \in \mathbb{S}$, and a message $x \in \{0,1\}^k$.

2. The challenger runs $(\mathsf{pp}, \mathsf{s}_1, \dots, \mathsf{s}_N, \mathsf{st}) \leftarrow \mathcal{S}_1(1^\lambda, 1^d, \mathbb{A})$ and provides pp to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a maximal invalid party set $S^* \subseteq \{P_1, \dots, P_N\}$ for $\mathbb{A}$.
4. The challenger provides the shares $\{\mathsf{s}_i\}_{i \in S^*}$ to $\mathcal{A}$.
5. $\mathcal{A}$ issues a polynomial number of adaptive queries of the form $(S \subseteq \{P_1, \dots, P_N\}, C)$ for circuits $C : \{0,1\}^k \to \{0,1\}$ of depth at most d. For each query, the challenger runs the simulator $\{\mathsf{y}_i\}_{i \in S} \leftarrow \mathcal{S}_2(\mathsf{pp}, C, C(x), S, \mathsf{st})$ and sends $\{\mathsf{y}_i\}_{i \in S}$ to $\mathcal{A}$.
6. At the end of the experiment, $\mathcal{A}$ outputs a distinguishing bit b.

Definition 7.6 (Robustness)**.** We say that a UT scheme satisfies robustness if for all λ, and depth bound d, the following holds. For any PPT adversary $\mathcal{A}$, the following experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{Robust}}(1^\lambda, 1^d)$ outputs 1 with negligible probability:

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

1. On input the security parameter 1^λ and circuit depth 1^d, the adversary $\mathcal{A}$ outputs a message $x \in \{0,1\}^k$ and $\mathbb{A} \in \mathbb{S}$.
2. The challenger runs $(\mathsf{pp}, \mathsf{s}_1, \dots, \mathsf{s}_N) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}, x)$ and provides $(\mathsf{pp}, \mathsf{s}_1, \dots, \mathsf{s}_N)$ to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a fake partial evaluation y_i^*.
4. The challenger returns 1 if $\mathsf{y}_i^* \neq \mathsf{UT.Eval}(\mathsf{pp}, \mathsf{s}_i, C)$ and $\mathsf{UT.Verify}(\mathsf{pp}, \mathsf{y}_i^*, C) = 1$.

7.1 Universal Thresholdizer from TFHE and PZK

In this section, we construct universal thresholdizer generically from threshold fully homomorphic encryption (Section 5) and NIZK with pre-processing (Section 3.3.1).

Construction 7.7. Our universal thresholdizer construction relies on the following primitives:

- Let $\mathsf{TFHE} = (\mathsf{TFHE.Setup}, \mathsf{TFHE.Encrypt}, \mathsf{TFHE.Eval}, \mathsf{TFHE.PartDec}, \mathsf{TFHE.FinDec})$ be a threshold fully homomorphic encryption scheme.

- Let $\mathsf{PZK} = (\mathsf{PZK.Pre}, \mathsf{PZK.Prove}, \mathsf{PZK.Verify})$ be a NIZK with pre-processing scheme.

- Let $\mathsf{C} = (\mathsf{C.Com})$ be a non-interactive commitment scheme.

We construct a universal thresholdizer scheme $\mathsf{UT} = (\mathsf{UT.Setup}, \mathsf{UT.Eval}, \mathsf{UT.Verify}, \mathsf{UT.Combine})$ as follows:

- $\mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}, x)$: On input the security parameter λ, depth bound d, access structure $\mathbb{A}$, and message $x \in \{0,1\}^k$, the setup algorithm first generates the TFHE keys $(\mathsf{tfhepk}, \mathsf{tfhesk}_1, \dots, \mathsf{tfhesk}_N) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A})$ and ciphertexts $\mathsf{ct}_i \leftarrow \mathsf{TFHE.Encrypt}(\mathsf{tfhepk}, x_i)$ for $i = 1, \dots k$. Then, it generates reference strings $(\sigma_{V,i}, \sigma_{P,i}) \leftarrow \mathsf{PZK.Pre}(1^\lambda)$, commitment randomness $r_i \xleftarrow{\text{R}} \{0,1\}^\lambda$, and commitments $\mathsf{com}_i \leftarrow \mathsf{C.Com}(\mathsf{tfhesk}_i; r_i)$ for $i = 1, \dots N$. It sets
$$\mathsf{pp} = \big(\mathsf{tfhepk}, \{\mathsf{ct}_i\}_{i \in [k]}, \{\sigma_{V,i}\}_{i \in [N]}, \{\mathsf{com}_i\}_{i \in [N]}\big) \qquad \mathsf{s}_i = \big(\mathsf{tfhesk}_i, \sigma_{P,i}, r_i\big).$$

- $\mathsf{UT.Eval}(\mathsf{pp}, \mathsf{s}_i, C)$: On input the public parameters pp, a share s_i, and a circuit C, the evaluation algorithm first computes the evaluated ciphertext $\hat{\mathsf{ct}} \leftarrow \mathsf{TFHE.Eval}(\mathsf{tfhepk}, C, \mathsf{ct}_1, \dots, \mathsf{ct}_k)$ and partial decryption $\mathsf{p}_i \leftarrow \mathsf{TFHE.PartDec}(\mathsf{tfhepk}, \hat{\mathsf{ct}}, \mathsf{tfhesk}_i)$. Then, it constructs the statement

$\Psi_i = \Psi_i(\mathsf{com}_i, \hat{\mathsf{ct}}, \mathsf{p}_i)$ asserting that the value p_i is consistent with the committed secret key tfhesk_i:

$$\exists \ (\mathsf{tfhesk}_i, r_i) : \mathsf{com}_i = \mathsf{C.Com}(\mathsf{tfhesk}_i; r_i) \wedge \mathsf{p}_i = \mathsf{TFHE.PartDec}(\mathsf{pp}, \hat{\mathsf{ct}}, \mathsf{tfhesk}_i).$$

It generates a NIZK proof $\pi_i \leftarrow \mathsf{PZK.Prove}(\sigma_{P,i}, \Psi_i, (\mathsf{tfhesk}_i, r_i))$ and returns $\mathsf{y}_i = (\mathsf{p}_i, \pi_i)$.

- $\mathsf{UT.Verify}(\mathsf{pp}, \mathsf{y}_i, C)$: On input the public parameters pp, a partial evalaution y_i, and a circuit C, the verification algorithm first computes the evaluated ciphertext $\hat{\mathsf{ct}} \leftarrow \mathsf{TFHE.Eval}(\mathsf{pp}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ and constructs the statement $\Psi_i = \Psi_i(\mathsf{com}_i, \hat{\mathsf{ct}}, \mathsf{p}_i)$. It then parses $\mathsf{y}_i = (\mathsf{p}_i, \pi_i)$ and returns the result of $\mathsf{PZK.Verify}(\sigma_{V,i}, \Psi_i, \pi_i)$.

- $\mathsf{UT.Combine}(\mathsf{pp}, B)$: On input the public parameters pp, and a set of partial evaluations $B = \{\mathsf{y}_i\}_{i \in S}$ for some $S \subseteq \{P_1, \ldots, P_N\}$, the combining algorithm first parses $\mathsf{y}_i = (\mathsf{p}_i, \pi_i)$ for $i \in S$ and outputs $\mathsf{TFHE.FinDec}(\mathsf{tfhepk}, \{\mathsf{p}_i\}_{i \in S})$.

We now state the compactness, correctness, and security theorems for Construction 7.7.

Theorem 7.8. *Suppose* TFHE *is a compact threshold fully homomorphic encryption scheme (Definition 5.2). Then, the universal thresholdizer scheme from Construction 7.7 satisfies compactness (Definition 7.2).*

Theorem 7.9. *Suppose* TFHE *is a threshold fully homomorphic encryption scheme that satisfies evaluation correctness (Definition 5.3). Then, the universal thresholdizer scheme from Construction 7.7 satisfies evaluation correctness (Definition 5.3).*

Theorem 7.10. *Suppose* PZK *is a complete zero knowledge proof system with pre-processing (Definition 3.4). Then, the universal thresholdizer scheme from Construction 7.7 satisfies verification correctness (Definition 7.4).*

Theorem 7.11. *Suppose* TFHE *satisfies semantic security (Definition 5.4) and simulation security (Definition 5.5),* PZK *is a zero knowledge proof system with pre-processing that satisfies zero-knwoeldge (Definition 3.4), and* C *is a non-interactive commitment scheme that satisfies computational hiding (Definition A.1). Then, the universal thresholdizer scheme from Construction 7.7 satisfies security (Definition 7.5).*

Theorem 7.12. *Suppose* PZK *is a zero knowledge proof system with pre-processing that satisfies soundness (Definition 3.4) and* C *is a non-interactive commitment scheme that satisfies perfect binding (Definition A.1). Then, the universal thresholdizer scheme from Construction 7.7 satisfies robustness 7.6.*

We provide the formal proofs of the theorems above in Appendix E.

7.2 Robustness from Homomorphic Signatures

In Section 7.1, we used NIZK with pre-processing (Section 3.3.1) to enforce robustness. Another way to enforce robustness is to use homomorphic signatures [BF11, GVW15]. A homomorphic signature is like a regular signature scheme, but it additionally allows a signature σ_x of a message x to be homomorphically evaluated with a circuit C. The resulting signature $\sigma_{C(x)}$ is compact in that its size depends only on the depth of C and $|C(x)|$; and it certifies that a value $y = C(x)$ is indeed the output of C evaluated on the original message x. Furthermore, the signature $\sigma_{C(x)}$ itself does not leak any information about the original message x other than what can be inferred from C and $C(x)$.

To enforce robustness for the construction in Section 7.1, the setup algorithm can simply use a homomorphic signature to sign each decryption key share of a TFHE scheme and include it as part of each party's share. Then, to evaluate on the shares, each user can homomorhpically compute on the TFHE ciphertexts and compute the partial decryption as before, but at the same time homomorphically evaluate on the signatures to derive a new signature that certifies correct partial decryption. The unforgeability property of the homomorphic signature scheme guarantees that no cheating adversary can generate a false signature on a value $y \neq C(x)$.

The benefit of using a homomorphic signature is that the proof size depends only on the depth of the circuit C to be computed and the evaluation share y. Using NIZK's, on the other hand, the proof size grows in the secret size $|x|$ and size of the circuit $|C|$. For applications that require long secret x, homomorphic signatures can give significant savings in the size of the evaluation shares. Since homomorphic signatures for circuits can be constructed from LWE [GVW15], its use does not introduce any new assumption to our construction.

8 Applications

In this section, we describe our applications of a universal thresholdizer scheme. We start by showing that a universal thresholdizer scheme for a class of access structures immediately gives rise to a function secret sharing scheme for the same class of access structure (Section 8.1). Next, we show that a universal thresholdizer scheme for the class of threshold access structures can be combined with existing cryptographic primitives to produce their thresholdized variants. As discussed in Section 1.1, these give rise to *threshold signatures*, *CCA threshold PKE*, *distributed PRFs*, and even *functional encryption* with thresholdized key generation. In this work, we provide just two of these applications: threshold signatures (Section 8.2) and CCA threshold PKE (Section 8.3). These two notions demonstrate how to use a universal thresholdizer as a general tool. The method that we develop in this section can be applied to a wide range of other applications in a straightforward way. Finally, in Section 8.4, we show that a non-compact universal thresholdizer scheme can be used to thresholdize a compact fully homomorphic encryption scheme to construct a compact TFHE scheme. This can be viewed as a way to boost a non-compact TFHE scheme to a compact one.

For full generality, we define the notions of functional secret sharing, threshold signatures, and CCA threshold PKE with respect to general access structures. By Theorem 4.15, all applications in this section can be instantiated for the class of threshold access structure TAS (Definition 4.10).

8.1 Function Secret Sharing

In this section, we construct a function secret sharing scheme from universal thresholdizers. The study of functional secret sharing was first initiated in [BGI15, BGI16]. In their definition, reconstruction is simply an addition of partial evaluations in a fixed group. We study a relaxed variant where the complexity of the reconstruction function is allowed to depend on the depth of the function (but is independent of the size of the circuit).

8.1.1 Definitions

The following definition is a natural generalization of the definition in [BGI15] to arbitrary access structures.

Definition 8.1. Let $P = \{P_1, \dots, P_N\}$ be a set of parties, let $\mathbb{S}$ be a class of efficient access structure on P, and let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of circuit classes. A function secret sharing scheme for $\mathbb{S}$ and $\mathcal{C}$ is a tuple of PPT algorithms $\mathsf{FSS} = (\mathsf{FSS.Gen}, \mathsf{FSS.Eval}, \mathsf{FSS.Decode})$ defined as follows:

- $\mathsf{FSS.Gen}(1^\lambda, f, \mathbb{A}) \to (\mathsf{s}_1, \dots, \mathsf{s}_N)$: On input the security parameter λ, a circuit $f : \{0,1\}^n \to \{0,1\} \in \mathcal{C}_\lambda$, and an access structure $\mathbb{A} \in \mathbb{S}$, the share generation algorithm outputs function shares $\mathsf{s}_1, \dots, \mathsf{s}_N$.
- $\mathsf{FSS.Eval}(\mathsf{s}_i, x) \to \mathsf{y}_i$: On input a function share s_i, and an input $x \in \{0,1\}^n$, the evaluation algorithm (deterministically) outputs a partial evaluation y_i.
- $\mathsf{FSS.Decode}(B) \to \mathsf{y}$: On input a set $B = \{\mathsf{y}_i\}_{i \in S}$ for some $S \subseteq \{P_1, \dots, P_N\}$, the decoding algorithm outputs an evaluation $\mathsf{y} \in \{0, 1, \perp\}$.

We require an FSS scheme to satisfy correctness, compactness, and security properties defined as follows.

Definition 8.2 (Compactness)**.** We say that a FSS scheme satisfies compactness if for all λ, circuit $f : \{0,1\}^n \to \{0,1\} \in \mathcal{C}_\lambda$, $S \subseteq \{P_1, \dots, P_N\}$, and $x \in \{0,1\}^n$, the following condition holds. There exists a polynomial poly such that for $(\mathsf{s}_1, \dots, \mathsf{s}_N) \leftarrow \mathsf{FSS.Gen}(1^\lambda, f, \mathbb{A})$, $\mathsf{y}_i \leftarrow \mathsf{FSS.Eval}(\mathsf{s}_i, x)$ for $i \in [N]$, we have $|\mathsf{y}_i| \leq \mathsf{poly}(\lambda, d)$.[1]

Definition 8.3 (Evaluation Correctness)**.** We say that a FSS scheme satisfies evaluation correctness if for all λ, circuit $f : \{0,1\}^n \to \{0,1\} \in \mathcal{C}_\lambda$, $S \subseteq \{P_1, \dots, P_N\}$, and $x \in \{0,1\}^n$, the following condition holds. For $(\mathsf{s}_1, \dots, \mathsf{s}_N) \leftarrow \mathsf{FSS.Gen}(1^\lambda, f, \mathbb{A})$, if $S \in \mathbb{A}$,

$$\Pr[\mathsf{FSS.Decode}(\{\mathsf{FSS.Eval}(\mathsf{s}_i, x)\}_{i \in S}) = f(x)] = 1 - \mathsf{negl}(\lambda)$$

and if $S \notin \mathbb{A}$,

$$\Pr[\mathsf{FSS.Decode}(\{\mathsf{FSS.Eval}(\mathsf{s}_i, x)\}_{i \in S}) = \perp] = 1 - \mathsf{negl}(\lambda).$$

Definition 8.4 (Security)**.** We say that a FSS scheme is secure if for all λ, the following holds. For any PPT adversary $\mathcal{A}$, the following experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{FSS}}(1^\lambda)$ outputs 1 with negligible probability:

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

1. On input the security parameter 1^λ, the adversary $\mathcal{A}$ outputs an access structure $\mathbb{A} \in \mathbb{S}$, a pair of functions (f_0, f_1) of same depth d and size s, and a set $S \subseteq P$ such that $S \notin \mathbb{A}$ and S is a maximal invalid party set.
2. The challenger samples a random bit $b \in \{0,1\}$, computes $(\mathsf{s}_1, \dots, \mathsf{s}_N) \leftarrow \mathsf{FSS.Gen}(1^\lambda, f_b, \mathbb{A})$, and provides $\{\mathsf{s}_i\}_{i \in S}$ to $\mathcal{A}$.
3. $\mathcal{A}$ issues a polynomial number of adaptive evaluation queries x_j satisfying $f_0(x_j) = f_1(x_j)$. For each query, the challenger computes $\mathsf{y}_i \leftarrow \mathsf{FSS.Eval}(\mathsf{s}_i, x)$ for $i \in [N]$ and provides $\{\mathsf{y}_i\}_{i \in [N]}$ to $\mathcal{A}$.
4. At the end of the experiment, $\mathcal{A}$ outputs a guess b'. The experiment outputs 1 if $b = b'$.

[1]Note that in function secret sharing proposed in [BGI15], the reconstruction function computes an addition of the partial evaluations, which are of a fixed size. Here we allow the size of the shares and the running time of the reconstruction to grow with the depth, but not the size, of the evaluated circuit.

8.1.2 Construction

We construct function secret sharing scheme for any access structures $\mathbb{S}$ from a universal thresholdizer (Section 7) and a pseudorandom function (Section A.2).

Construction 8.5. Our function secret sharing construction relies on the following primitives:

- Let $\mathsf{UT} = (\mathsf{UT.Setup}, \mathsf{UT.Eval}, \mathsf{UT.Verify}, \mathsf{UT.Combine})$ be a universal thresholdizer scheme where $\mathsf{len}(\lambda)$ is the bit-length of the randomness used by the $\mathsf{UT.Eval}$ algorithm.
- Let $F : \mathcal{K} \times \{0,1\}^* \to \{0,1\}^{\mathsf{len}(\lambda)}$ be a PRF.

Additionally, in our construction description, we use $\mathcal{U}$ to denote the universal circuit that takes in a function description f and an input x, and outputs $f(x)$. We denote $\mathcal{U}_x$ to denote the universal circuit with a hardcoded input x that takes in a function f and outputs $f(x)$. We construct a function secret sharing scheme $\mathsf{FSS} = (\mathsf{FSS.Gen}, \mathsf{FSS.Eval}, \mathsf{FSS.Decode})$ as follows:

- $\mathsf{FSS.Gen}(1^\lambda, f, \mathbb{A})$: On input the security parameter λ, a circuit $f : \{0,1\}^n \to \{0,1\}$, and an access structure $\mathbb{A} \in \mathbb{S}$, the share generation algorithm computes $(\mathsf{utpp}, \mathsf{uts}_1, \ldots, \mathsf{uts}_N) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}, f)$ where d is the depth of the universal circuit $\mathcal{U}$ accepting a circuit of size $|f|$ and input of size n. Then, for each $i \in [N]$, it samples a PRF key $k_i \overset{\text{R}}{\leftarrow} \mathcal{K}$ and sets $\mathsf{s}_i = (\mathsf{utpp}, \mathsf{uts}_i, k_i)$.
- $\mathsf{FSS.Eval}(\mathsf{s}_i, x)$: On input a function share s_i, and input $x \in \{0,1\}^n$, the evaluation algorithm parses $\mathsf{s}_i = (\mathsf{utpp}, \mathsf{uts}_i, k_i)$, generates evaluation randomness $r_i \leftarrow F(k_i, x)$, and outputs $\mathsf{y}_i \leftarrow \mathsf{UT.Eval}(\mathsf{utpp}, \mathsf{uts}_i, \mathcal{U}_x; F(k_i, x))$.
- $\mathsf{FSS.Decode}(B)$: On input a set of partial evaluations $B = \{\mathsf{s}_i\}_{i \in S}$, the decoding algorithm outputs $\mathsf{y} \leftarrow \mathsf{UT.Combine}(\mathsf{utpp}, B)$.

We now state the correctness and security theorems for Construction 8.5.

Theorem 8.6. *Suppose* UT *is a universal thresholdizer scheme that satisfies compactness (Definition 7.2). Then, the function secret sharing scheme from Construction 8.5 satisfies compactness (Definition 8.2).*

Theorem 8.7. *Suppose* UT *is a universal thresholdizer scheme that satisfies evaluation correctness (Definition) 7.3). Then, the function secret sharing scheme from Construction 8.5 satisfies evaluation correctness (Definition 8.3).*

Theorem 8.8. *Suppose* UT *is a universal thresholdizer scheme that satisfies security (Definition 7.5), and F is a PRF (Definition A.2). Thenn, the function secret sharing scheme from Construction 8.5 satisfies security (Definition 8.4).*

The proofs of the theorems above follow immediately from the properties of the underlying UT scheme.

8.2 Threshold Signatures

In this section, we construct a threshold signatures scheme from universal thresholdizers. In a threshold signature scheme, the signing key of a signer is divided into a number of key shares and are distributed to multiple signers. When sigining a message, each of the signers creates a partial signature with its own share of the signing key. Then, a combining algorithm combines the partial signatures into a full signature. For generality, we present the definition of threshold signatures with respect to a general class of access structures.

8.2.1 Definitions

Definition 8.9 (Threshold Signatures)**.** Let $P = \{P_1, \dots, P_N\}$ be a set of parties and let $\mathbb{S}$ be a class of efficient access structure on P. A threshold signature scheme for $\mathbb{S}$ is a tuple of PPT algorithms $\mathsf{TS} = (\mathsf{TS.Setup}, \mathsf{TS.PartSign}, \mathsf{TS.PartSignVerify}, \mathsf{TS.Combine}, \mathsf{TS.Verify})$ defined as follows:

- $\mathsf{TS.Setup}(1^\lambda, \mathbb{A}) \to (\mathsf{pp}, \mathsf{vk}, \mathsf{sk}_1, \dots, \mathsf{sk}_N)$: On input the security parameter λ, and an access structure $\mathbb{A}$, the setup algorithm outputs the public parameters pp, a signature verification key vk, and a set of key shares $\mathsf{sk}_1, \dots, \mathsf{sk}_N$.
- $\mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m) \to \sigma_i$: On input the public parameters pp, a partial signing key sk_i, and a message $m \in \{0,1\}^*$, the partial signing algorithm outputs a partial signature σ_i.
- $\mathsf{TS.PartSignVerify}(\mathsf{pp}, m, \sigma_i) \to \{0,1\}$: On input the public parameters pp, a message $m \in \{0,1\}^*$, and a partial signature σ_i, the partial signature verification algorithm accepts or rejects.
- $\mathsf{TS.Combine}(\mathsf{pp}, B) \to \sigma$: On input the public parameters pp, and a set of partial signatures $B = \{\sigma_i\}_{i \in S}$, the signature combining algorithm outputs a full signature σ.
- $\mathsf{TS.Verify}(\mathsf{vk}, m, \sigma)$: On input a signature verification key vk, a message $m \in \{0,1\}^*$, and a signature σ, the verification algorithm accepts or rejects.

We require a TS scheme to satisfy the following compactness, correctness, and security requirements.

Definition 8.10 (Compactness)**.** We say that a TS scheme for $\mathbb{S}$ satisfies compactness if there exist polynomials $\mathsf{poly}_1(\cdot), \mathsf{poly}_2(\cdot)$ such that for all λ and $\mathbb{A} \in \mathbb{S}$, the following holds. For $(\mathsf{pp}, \mathsf{vk}, \mathsf{sk}_1, \dots, \mathsf{sk}_N) \leftarrow \mathsf{TS.Setup}(1^\lambda, \mathbb{A})$, $\sigma_i \leftarrow \mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m)$ for $i \in S$, $\sigma \leftarrow \mathsf{TS.Combine}(\mathsf{pp}, \{\sigma_i\}_{i \in S})$, we have that $|\sigma| \leq \mathsf{poly}_1(\lambda)$ and $|\mathsf{vk}| \leq \mathsf{poly}_2(\lambda)$.

Definition 8.11 (Evaluation Correctness)**.** We say that a TS scheme for $\mathbb{S}$ satisfies evaluation correctness if for all λ, $\mathbb{A} \in \mathbb{S}$, and $S \in \mathbb{A}$, the following holds. For $(\mathsf{pp}, \mathsf{vk}, \mathsf{sk}_1, \dots, \mathsf{sk}_N) \leftarrow \mathsf{TS.Setup}(1^\lambda, \mathbb{A})$, $\sigma_i \leftarrow \mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m)$ for $i \in S$,

$$\Pr[\mathsf{TS.Verify}(\mathsf{vk}, m, \mathsf{TS.Combine}(\mathsf{pp}, \{\sigma_i\}_{i \in S})) = 1] = 1 - \mathsf{negl}(\lambda).$$

Definition 8.12 (Partial Verification Correctness)**.** We say that a TS scheme for $\mathbb{S}$ satisfies partial verification correctness if for all λ and $\mathbb{A} \in \mathbb{S}$, the following holds. For $(\mathsf{pp}, \mathsf{vk}, \mathsf{sk}_1, \dots, \mathsf{sk}_N) \leftarrow \mathsf{TS.Setup}(1^\lambda, \mathbb{A})$,

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

Definition 8.13 (Unforgeability)**.** We say that a TS scheme for $\mathbb{S}$ satisfies unforgeability if for all λ, the following holds. For any PPT adversary $\mathcal{A}$, the following experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{TS},\mathsf{uf}}(1^\lambda)$ outputs 1 with negligible probability:

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

1. On input the security parameter 1^λ, the adversary $\mathcal{A}$ outputs an access structure $\mathbb{A} \in \mathbb{S}$.
2. The challenger samples $(\mathsf{pp}, \mathsf{vk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{TS.Setup}(1^\lambda, \mathbb{A})$ and provides pp and vk to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a set $S \subseteq P$ such that $S \notin \mathbb{A}$ and S is a maximal invalid party set.
4. The challenger provides the set of keys $\{\mathsf{sk}_i\}_{i \in S}$ to $\mathcal{A}$.
5. $\mathcal{A}$ issues a polynomial number of adaptive signing queries of the form (m, i) where $i \in [N] \backslash S$. For each query, the challenger computes $\sigma_i \leftarrow \mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m)$ and provides σ_i to $\mathcal{A}$.
6. At the end of the experiment, $\mathcal{A}$ outputs a forgery (m^*, σ^*). The experiment outputs 1 if $\mathsf{TS.Verify}(\mathsf{vk}, m^*, \sigma^*) = 1$ and m was not previously queried as a signing query.

Definition 8.14 (Robustness)**.** We say that a TS scheme for $\mathbb{S}$ satisfies robustness if for all λ, the following holds. For any PPT adversary $\mathcal{A}$, the following experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{TS},\mathsf{rb}}(1^\lambda)$ outputs 1 with negligible probability:

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

1. On input the security parameter 1^λ, the adversary $\mathcal{A}$ outputs an access structure $\mathbb{A} \in \mathbb{S}$.
2. The challenger samples $(\mathsf{pp}, \mathsf{vk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{TS.Setup}(1^\lambda, \mathbb{A})$ and provides $(\mathsf{pp}, \mathsf{vk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N)$ to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a partial signature forgery (m^*, σ_i^*, i).
4. The experiment outputs 1 if $\mathsf{TS.PartSignVerify}(\mathsf{pp}, m^*, \sigma_i^*) = 1$ and $\sigma_i \neq \mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m^*)$.

Definition 8.15 (Anonymity)**.** We say that a TS scheme for $\mathbb{S}$ satisfies anonymity if for λ, $\mathbb{A} \in \mathbb{S}$, and $S \in \mathbb{A}$, the following holds. For $(\mathsf{pp}, \mathsf{vk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{TS.Setup}(1^\lambda, \mathbb{A})$, $B_b = \{\mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m)\}_{i \in S_b}$ for $b \in \{0, 1\}$, we have

$$\Pr[\mathsf{TS.Combine}(\mathsf{pp}, B_0) \neq \mathsf{TS.Combine}(\mathsf{pp}, B_1)] \leq \mathsf{negl}(\lambda).$$

8.2.2 Construction

We construct threshold signature scheme from a universal thresholdizer (Section 7) and a signature scheme (Section A.3).

Construction 8.16. Our threshold signature construction relies on the following primitives:

- Let $\mathsf{UT} = (\mathsf{UT.Setup}, \mathsf{UT.Eval}, \mathsf{UT.Verify}, \mathsf{UT.Combine})$ be a universal thresholdizer scheme for the class of access structures $\mathbb{S}$.
- Let $\mathsf{S} = (\mathsf{S.KeyGen}, \mathsf{S.Sign}, \mathsf{S.Verify})$ be a signature scheme. For our construction, we assume that the signing algorithm $\mathsf{S.Sign}$ is a deterministic algorithm. This is without loss of generality since any randomized signature scheme can be derandomized (i.e. using PRFs).

Now, we construct a threshold signature scheme $\mathsf{TS} = (\mathsf{TS.Setup}, \mathsf{TS.PartSign}, \mathsf{TS.PartSignVerify}, \mathsf{TS.Combine}, \mathsf{TS.Verify})$ for $\mathbb{S}$ as follows:

- $\mathsf{TS.Setup}(1^\lambda, \mathbb{A})$: On input the security parameter λ, and an access structure $\mathbb{A}$, the setup algorithm first generates the keys for the signature scheme $(\mathsf{ssk}, \mathsf{svk}) \leftarrow \mathsf{S.KeyGen}(1^\lambda)$. Then it instantiates the universal thresholdizer scheme $(\mathsf{utpp}, \mathsf{uts}_1, \ldots, \mathsf{uts}_N) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}, \mathsf{ssk})$ where d is the depth of the signing algorithm $\mathsf{S.Sign}$. Then, it sets
$$\mathsf{pp} = \mathsf{utpp}, \quad \mathsf{vk} = \mathsf{svk}, \quad \mathsf{sk}_i = \mathsf{uts}_i \quad \forall i \in [N].$$
- $\mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m)$: On input the public parameters $\mathsf{pp} = \mathsf{utpp}$, a partial signing key $\mathsf{sk}_i = \mathsf{uts}_i$, and a message $m \in \{0,1\}^*$, the partial signing algorithm outputs $\sigma_i \leftarrow \mathsf{UT.Eval}(\mathsf{utpp}, \mathsf{uts}_i, C_m)$ where the circuit C_m is defined as
$$C_m(\mathsf{ssk}) = \mathsf{S.Sign}(\mathsf{ssk}, m).$$
- $\mathsf{TS.PartSignVerify}(\mathsf{pp}, m, \sigma_i)$: On input the public parameters pp, message $m \in \{0,1\}^*$, and a partial signature σ_i, the partial signature verification algorithm outputs $\mathsf{UT.Verify}(\mathsf{utpp}, \sigma_i, C_m)$.
- $\mathsf{TS.Combine}(\mathsf{pp}, B)$: On input the public parameters pp, and a set of partial signatures $B = \{\sigma_i\}_{i \in S}$, the signature combining algorithm outputs $\mathsf{UT.Combine}(\mathsf{utpp}, B)$.
- $\mathsf{TS.Verify}(\mathsf{vk}, m, \sigma)$: On input the signature verification key $\mathsf{vk} = \mathsf{svk}$, a message $m \in \{0,1\}^*$, and a signature σ, the verification algorithm outputs $\mathsf{S.Verify}(\mathsf{vk}, m, \sigma)$.

We now state the compactness, correctness, and security theorems for Construction 8.16.

Theorem 8.17. *Suppose* UT *is a universal thresholdizer scheme that satisfies evaluation correctness (Definition 7.3). Then, the threshold signature scheme from Construction 8.16 satisfies compactness (Definition 8.10).*

Theorem 8.18. *Suppose* UT *is a universal thresholdizer scheme that satisfies evaluation correctness (Definition 7.3) and* S *is a signature scheme that satisfies correctness (Definition A.4). Then, the threshold signature scheme from Construction 8.16 satisfies evaluation correctness (Definition 8.11).*

Theorem 8.19. *Suppose* UT *is a universal thresholdizer scheme that satisfies evaluation verification correctness (Definition 7.4). Then, the threshold signature scheme from Construction 8.16 satisfies partial verification correctness (Definition 8.12).*

Theorem 8.20. *Suppose* UT *is a universal thresholdizer scheme that satisfies security (Definition 7.5) and* S *is a signature scheme that satisfies unforgeability (Definition A.5). Then, the threshold signature scheme from Construction 8.16 satisfies unforgeability (Definition 8.13).*

Theorem 8.21. *Suppose* UT *is a universal thresholdizer scheme that satisfiesd robustness (Definition 7.6). Then, the threshold signature scheme from Construction 8.16 satisfies robustness (Definition 8.14).*

Theorem 8.22. *Suppose* UT *is a universal thresholdizer scheme that satisfies evaluation correctness (Definition 7.3). Then, the threshold signature scheme from Construction 8.16 satisfies anonymity (Definition 8.15).*

We provide formal proofs of the theorems above in Appendix F.

8.3 CCA Threshold PKE

In this section, we construct a CCA secure threshold PKE scheme from universal thresholdizers. In a threshold PKE scheme, the decryption key is divided into a number of key shares and are distributed to multiple decryption servers. When decrypting a message, each of the decryption servers creates its own decryption share, and the decryption shares can be publicly combined to result in a full decryption. As was done for threshold signatures, we present the definition of thresholdPKE with respect to a general class of access structures.

8.3.1 Definitions

Definition 8.23 (Threshold PKE)**.** Let $P = \{P_1, \dots, P_N\}$ be a set of parties and let $\mathbb{S}$ be a class of efficient access structures on P. A threshold PKE scheme for a message space $\mathcal{M}$ is a tuple of PPT algorithms $\mathsf{TPKE} = (\mathsf{TPKE.Setup}, \mathsf{TPKE.Enc}, \mathsf{TPKE.PartDec}, \mathsf{TPKE.PartVerify}, \mathsf{TPKE.Combine})$ defined as follows:

- $\mathsf{TPKE.Setup}(1^\lambda, \mathbb{A}) \to (\mathsf{pp}, \mathsf{ek}, \mathsf{sk}_1, \dots, \mathsf{sk}_N)$: On input the security parameter λ, and an access structure $\mathbb{A}$, the setup algorithm outputs the public parameters pp, an encryption key ek, and a set of key shares $\mathsf{sk}_1, \dots, \mathsf{sk}_N$.

- $\mathsf{TPKE.Enc}(\mathsf{ek}, m) \to \mathsf{ct}$: On input the encryption key ek, and a message $m \in \mathcal{M}$, the encryption algorithm outputs a ciphertext ct.

- $\mathsf{TPKE.PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct}) \to \mathsf{s}_i$: On input the public parameters pp, a key share sk_i, and a ciphertext ct, the partial decryption algorithm outputs a partial decryption share m_i.

- $\mathsf{TPKE.PartVerify}(\mathsf{pp}, \mathsf{ct}, \mathsf{m}_i) \to \{0, 1\}$: On input the public parameters pp, a ciphertext ct, and a partial decryption share m_i, the part verification algorithm accepts or rejects.

- $\mathsf{TPKE.Combine}(\mathsf{pp}, B) \to m'$: On input the public parameters pp, and a set of decryption shares $B = \{\mathsf{m}_i\}_{i \in S}$, the combining algorithm outputs a message m'.

We require a TPKE scheme to satisfy the following compactness, correctness, and security requirements.

Definition 8.24 (Compactness)**.** We say that a TPKE scheme for $\mathbb{S}$ satisfies compactness if there exist polynomials $\mathsf{poly}_1(\cdot)$, $\mathsf{poly}_2(\cdot)$ such that for all λ and $\mathbb{A} \in \mathbb{S}$, the following holds. For $(\mathsf{pp}, \mathsf{ek}, \mathsf{sk}_1, \dots, \mathsf{sk}_N) \leftarrow \mathsf{TPKE.Setup}(1^\lambda, \mathbb{A})$, $\mathsf{ct} \leftarrow \mathsf{TPKE.Enc}(\mathsf{ek}, m)$, we have that $|ct| \leq \mathsf{poly}_1(\lambda)$ and $|\mathsf{ek}| \leq \mathsf{poly}_2(\lambda)$.

Definition 8.25 (Decryption Correctness)**.** We say that a TPKE scheme for $\mathbb{S}$ satisfies decryption correctness if for all λ, $\mathbb{A} \in \mathbb{S}$, and $S \in \mathbb{A}$, the following holds. For $(\mathsf{pp}, \mathsf{ek}, \mathsf{sk}_1, \dots, \mathsf{sk}_N) \leftarrow \mathsf{TPKE.Setup}(1^\lambda, \mathbb{A})$, $\mathsf{ct} \leftarrow \mathsf{TPKE.Enc}(\mathsf{ek}, m)$, $\mathsf{m}_i \leftarrow \mathsf{TPKE.PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct})$ for $i \in S$,

$$\Pr[\mathsf{TPKE.Combine}(\mathsf{pp}, \{\mathsf{m}_i\}_{i \in S}) = m] = 1 - \mathsf{negl}(\lambda).$$

Definition 8.26 (Partial Verification Correctness)**.** We say that a TPKE scheme for $\mathbb{S}$ satisfies partial verification correctness if for all λ, $\mathbb{A} \in \mathbb{S}$, and $S \in \mathbb{A}$, the following holds. For $(\mathsf{pp}, \mathsf{ek}, \mathsf{sk}_1, \dots, \mathsf{sk}_N) \leftarrow \mathsf{TPKE.Setup}(1^\lambda, \mathbb{A})$, $\mathsf{ct} \leftarrow \mathsf{TPKE.Enc}(\mathsf{ek}, m)$,

$$\Pr[\mathsf{TPKE.PartVerify}(\mathsf{pp}, \mathsf{TPKE.PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct})) = 1] = 1 - \mathsf{negl}(\lambda).$$

Definition 8.27 (Security)**.** We say that a TPKE scheme for $\mathbb{S}$ satisfies security if for all λ, the following holds. For any PPT adversary $\mathcal{A}$ the following experiments $\mathsf{Expt}_{\mathcal{A},\mathsf{TPKE},0}(1^\lambda)$ and $\mathsf{Expt}_{\mathcal{A},\mathsf{TPKE},1}(1^\lambda)$ are computationally indistinguishable:

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

1. On input the security parameter 1^λ, the adversary $\mathcal{A}$ outputs an access structure $\mathbb{A} \in \mathbb{S}$.
2. The challenger samples $(\mathsf{pp}, \mathsf{ek}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{TS.Setup}(1^\lambda, \mathbb{A})$ and provides pp and ek to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a set $S \subseteq P$ such that $S \notin \mathbb{A}$ and S is a maximal invalid set.
4. The challenger provides the set of keys $\{\mathsf{sk}_i\}_{i \in S}$ to $\mathcal{A}$.
5. $\mathcal{A}$ issues a polynomial number of adaptive decryption queries of the form (ct, i) where $i \in [N] \backslash S$. For each query, the challenger computes $\mathsf{s}_i \leftarrow \mathsf{TPKE.PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct})$ and provides s_i to $\mathcal{A}$.
6. $\mathcal{A}$ outputs a pair of challenge messages (m_0^*, m_1^*).
7. The challenger computes $\mathsf{ct}_b^* \leftarrow \mathsf{TPKE.Enc}(\mathsf{ek}, m)$ and provides ct_b^* to $\mathcal{A}$.
8. $\mathcal{A}$ continues issuing a polynomial number of adaptive decryption queries.
9. At the end of the experiment, $\mathcal{A}$ outputs a guess b', which is the output of the experiment.

Definition 8.28 (Robustness)**.** We say that a TPKE scheme for $\mathbb{S}$ satisfies robustness if for all λ, the following holds. For any PPT adversary $\mathcal{A}$, the following experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{TPKE},\mathsf{rb}}(1^\lambda)$ outputs 1 with negligible probability:

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

1. On input the security parameter 1^λ, the adversary $\mathcal{A}$ outputs an access structure $\mathbb{A} \in \mathbb{S}$.
2. The challenger samples $(\mathsf{pp}, \mathsf{ek}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{TPKE.Setup}(1^\lambda, \mathbb{A})$ and provides $(\mathsf{pp}, \mathsf{ek}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N)$ to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a partial decryption forgery $(\mathsf{ct}^*, \mathsf{m}_i^*, i)$.
4. The experiment outputs 1 if $\mathsf{TPKE.PartVerify}(\mathsf{pp}, \mathsf{ct}^*, \mathsf{m}_i^*) = 1$ and $\mathsf{m}_i^* \neq \mathsf{TPKE.PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct}^*)$.

8.3.2 Construction

We construct a threshold PKE from a universal thresholdizer (Section 7) and a CCA secure PKE scheme (Section A.4).

Construction 8.29. Our threshold PKE construction relies on the following primitives:

- Let $\mathsf{UT} = (\mathsf{UT.Setup}, \mathsf{UT.Eval}, \mathsf{UT.Verify}, \mathsf{UT.Combine})$ be a universal thresholdizer scheme for the class of access structures $\mathbb{S}$.
- Let $\mathsf{PKE} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Encrypt}, \mathsf{PKE.Decrypt})$ be a public key encryption scheme.

Now, we construct a threshold PKE scheme $\mathsf{TPKE} = (\mathsf{TPKE.Setup}, \mathsf{TPKE.Enc}, \mathsf{TPKE.PartDec}, \mathsf{TPKE.PartVerify}, \mathsf{TPKE.Combine})$ for $\mathbb{S}$ as follows:

- $\mathsf{TPKE.Setup}(1^\lambda, \mathbb{A})$: On input the security parameter λ, and an access structure $\mathbb{A}$, the setup algorithm first generates the keys for the PKE scheme $(\mathsf{pkesk}, \mathsf{pkepk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$. Then it instantiates the universal thresholdizer scheme $(\mathsf{utpp}, \mathsf{uts}_1, \ldots, \mathsf{uts}_N) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}, \mathsf{pkesk})$ where d is the depth of the decryption algorithm $\mathsf{PKE.Decrypt}$. Then, it sets

$$\mathsf{pp} = \mathsf{utpp}, \quad \mathsf{ek} = \mathsf{pkepk}, \quad \mathsf{sk}_i = \mathsf{uts}_i \quad \forall i \in [N].$$

- $\mathsf{TPKE.Enc}(\mathsf{ek}, m)$: On input the encryption key ek, and a message $m \in \mathcal{M}$, the encryption algorithm computes $\mathsf{ct} \leftarrow \mathsf{PKE.Encrypt}(\mathsf{pk}, m)$ and outputs ct.
- $\mathsf{TPKE.PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct})$: On input the public parameters pp, key share $\mathsf{sk}_i = \mathsf{uts}_i$, and a ciphertext ct, the partial decryption algorithm computes $\mathsf{m}_i \leftarrow \mathsf{UT.Eval}(\mathsf{utpp}, \mathsf{uts}_i, C_{\mathsf{ct}})$ where the circuit C_m is defined as

$$C_{\mathsf{ct}}(\mathsf{pkesk}) = \mathsf{PKE.Decrypt}(\mathsf{pkesk}, \mathsf{ct}).$$

  It then outputs m_i.
- $\mathsf{TPKE.PartVerify}(\mathsf{pp}, \mathsf{ct}, \mathsf{m}_i)$: On input the public parameters pp, a ciphertext ct, and a decryption share m_i, the part verification algorithm outputs $\mathsf{UT.Verify}(\mathsf{utpp}, \mathsf{m}_i, C_{\mathsf{ct}})$.
- $\mathsf{TPKE.Combine}(\mathsf{pp}, B)$: On input the public parameters pp, and a set of partial decryption shares $B = \{\mathsf{m}_i\}_{i \in S}$, the combining algorithm computes $m' \leftarrow \mathsf{UT.Combine}(\mathsf{utpp}, B)$ and outputs a message m'.

We now state the compactness, correctness, and security theorems for Construction 8.29.

Theorem 8.30. *Suppose* UT *is a universal thresholdizer scheme that satisfies evaluation correctness (Definition 7.3). Then, the* TPKE *scheme from Construction 8.29 satisfies compactness (Definition 8.24).*

Theorem 8.31. *Suppose* UT *is a universal thresholdizer scheme that satisfies evaluation correctness (Definition 7.3) and* PKE *is a public key encryption scheme that satisfies correctness (Definition A.7). Then, the* TPKE *scheme from Construction 8.29 satisfies decryption correctness (Definition 8.25).*

Theorem 8.32. *Suppose* UT *is a universal thresholdizer scheme that satisfies verification correctness (Definition 7.4). Then, the* TPKE *scheme from Construction 8.29 satisfies decryption correctness (Definition 8.26).*

Theorem 8.33. *Suppose* UT *is a universal thresholdizer scheme that satisfies security (Definition 7.5) and* PKE *is a public key encryption scheme that satisfies security (Definition A.8). Then, the* TPKE *scheme from Construction 8.29 satisfies security (Definition 8.26).*

Theorem 8.34. *Suppose* UT *is a universal thresholdizer scheme that satisfies robustness (Definition 7.6). Then, the* TPKE *scheme from Construction 8.29 satisfies robustness (Definition 8.28).*

We provide formal proofs of the theorems above in Appendix G.

8.4 TFHE from UT

In this section, we show how to use a *non-compact* universal thresholdizer to construct a *compact* TFHE scheme by thresholdizing a compact (non-threshold) FHE scheme. Since we can construct a non-compact universal thresholdizer from a non-compact TFHE scheme, the construction can be viewed as a *boosting* step that takes a non-compact TFHE scheme and converting it into a compact one.

Construction 8.35. Our TFHE construction relies on the following primitives:

- Let $\mathsf{UT} = (\mathsf{UT.Setup}, \mathsf{UT.Eval}, \mathsf{UT.Verify}, \mathsf{UT.Combine})$ be a universal thresholdizer scheme for the class of access structure $\mathbb{S}$ (Definition 7.1)
- Let $\mathsf{FHE} = (\mathsf{FHE.Setup}, \mathsf{FHE.Encrypt}, \mathsf{FHE.Eval}, \mathsf{FHE.Decrypt})$ be a fully homomorphic encryption scheme (Definition 3.5).

We construct $\mathsf{TFHE} = (\mathsf{TFHE.Setup}, \mathsf{TFHE.Encrypt}, \mathsf{TFHE.Eval}, \mathsf{TFHE.PartDec}, \mathsf{TFHE.FinDec})$ as follows:

- $\mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A})$: On input the security parameter λ, depth bound d, and an access structure $\mathbb{A}$, the setup algorithm first generates the keys for the FHE scheme $(\mathsf{fhepk}, \mathsf{fhesk}) \leftarrow \mathsf{FHE.Setup}(1^\lambda, 1^d)$. Then, it instantiates the universal thresholdizer scheme $(\mathsf{utpp}, \mathsf{uts}_1, \ldots, \mathsf{uts}_N) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^{d'}, \mathbb{A}, \mathsf{fhesk})$ where d' is the depth of the decryption algorithm $\mathsf{FHE.Decrypt}$. Then, it sets
$$\mathsf{pk} = (\mathsf{utpp}, \mathsf{fhepk}), \qquad \mathsf{sk}_i = \mathsf{uts}_i \quad \forall i \in [N].$$
- $\mathsf{TFHE.Encrypt}(\mathsf{pk}, \mu)$: On input a public key pk, and a single bit plaintext $\mu \in \{0, 1\}$, the encryption algorithm computes $\mathsf{ct} \leftarrow \mathsf{FHE.Encrypt}(\mathsf{fhepk}, m)$ and outputs ct.
- $\mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$: On input a public key $\mathsf{pk} = (\mathsf{utpp}, \mathsf{fhepk})$, circuit C, and a set of ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$, the evaluation algorithm computes $\hat{\mathsf{ct}} \leftarrow \mathsf{FHE.Eval}(\mathsf{fhepk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ and outputs $\hat{\mathsf{ct}}$.
- $\mathsf{TFHE.PartDec}(\mathsf{pk}, \mathsf{ct}, \mathsf{sk}_i)$: On input a public key $\mathsf{pk} = (\mathsf{utpp}, \mathsf{fhepk})$, a ciphertext ct, and a secret key share $\mathsf{sk}_i = \mathsf{uts}_i$, the partial decryption algorithm computes $\mathsf{p}_i \leftarrow \mathsf{UT.Eval}(\mathsf{utpp}, \mathsf{uts}_i, C_{\mathsf{ct}})$ where the circuit C_{ct} is defined as
$$C_{\mathsf{ct}}(\mathsf{fhesk}) = \mathsf{FHE.Decrypt}(\mathsf{fhesk}, \mathsf{ct}).$$
It then outputs p_i.
- $\mathsf{TFHE.FinDec}(\mathsf{pk}, B)$: On input a public key pk, and a set $B = \{\mathsf{p}_i\}_{i \in S}$, the final decryption algorithm computes $\hat{\mu} \leftarrow \mathsf{UT.Combine}(\mathsf{utpp}, B)$ and outputs $\hat{\mu}$.

We now state the compactness, correctness, and security theorems for Construction 8.35.

Theorem 8.36. *Suppose* UT *is a universal thresholdizer scheme that satisfies evaluation correctness (Definition 7.3). Then, the* TFHE *scheme from Construction 8.35 satisfies compactness (Definition 5.2).*

Theorem 8.37. *Suppose* UT *is a universal thresholdizer scheme that satisfies evaluation correctness (Definition 7.3) and* FHE *is a fully homomorphic encryption scheme that satisfies correctness (Definition 3.7). Then, the* TFHE *scheme from Construction 8.35 satisfies evaluation correctness (Definition 5.3).*

Theorem 8.38. *Suppose* UT *is a universal thresholdizer scheme that satisfies security (Definition 7.5) and* FHE *is a fully homomorphic encryption scheme that satisfies semantic security (Definition 3.8). Then, the* TFHE *scheme from Construction 8.35 satisfies semantic security (Definition 5.4).*

Theorem 8.39. *Suppose* UT *is a universal thresholdizer scheme that satisfies security (Definition 7.5). Then, the* TFHE *scheme from Construction 8.35 satisfies simulation security (Definition 5.5).*

The proofs of the theorems above follow immediately from the properties of the underlying UT and FHE schemes.

Acknowledgements

D. Boneh and S. Kim are supported by NSF, DARPA, the Simons foundation, and a grant from ONR. R. Gennaro is supported by NSF grant 1545759. S. Goldfeder is supported by NSF Graduate Research Fellowship under grant number DGE 1148900. A. Jain, P. Rasmussen, and A. Sahai are supported by a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1619348, 1228984, 1136174, and 1065276, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the ARL under Contract W911NF-15-C-0205. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, 2010.

[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*, 2012.

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*. 2009.

[AJLA$^+$12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold fhe. In *EUROCRYPT*, 2012.

[BBH06] Dan Boneh, Xavier Boyen, and Shai Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. In *CT-RSA*, 2006.

[BC94] Amos Beimel and Benny Chor. Universally ideal secret-sharing schemes. *IEEE Transactions on Information Theory*, 40(3):786–794, 1994.

[BD10] Rikke Bendlin and Ivan Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In *TCC*, 2010.

[BDOP16] Carsten Baum, Ivan Damgård, Sabine Oechsner, and Chris Peikert. Efficient commitments and zero-knowledge protocols from ring-sis with applications to lattice-based threshold cryptosystems. *IACR Cryptology ePrint Archive*, 2016:997, 2016.

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

[BF11] Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In *EUROCRYPT*, 2011.

[BGGK17] Dan Boneh, Rosario Gennaro, Steven Goldfeder, and Sam Kim. A lattice-based universal thresholdizer for cryptographic systems. Cryptology ePrint Archive, Report 2017/251, 2017. http://eprint.iacr.org/2017/251.

[BGI15] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In *EUROCRYPT*, 2015.

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under ddh. In *CRYPTO*, 2016.

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

[Bit17] Nir Bitansky. Verifiable random functions from non-interactive witness-indistinguishable proofs. In *TCC*, 2017.

[BKP13] Rikke Bendlin, Sara Krehbiel, and Chris Peikert. How to share a lattice trapdoor: threshold protocols for signatures and (h) ibe. In *ACNS*, 2013.

[BLMR13] Dan Boneh, Kevin Lewi, Hart Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In *CRYPTO*. 2013.

[BLP$^+$13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, 2013.

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *J. Cryptology*, 17(4):297–319, September 2004.

[Blu83] Manuel Blum. Coin flipping by telephone a protocol for solving impossible problems. *ACM SIGACT News*, 15(1):23–27, 1983.

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *PKC*, 2003.

[BOV03] Boaz Barak, Shien Jin Ong, and Salil P Vadhan. Derandomization in cryptography. In *CRYPTO*, 2003.

[Boy10] Xavier Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In *PKC*, 2010.

[BP14] Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In *CRYPTO*, 2014.

[BP16] Zvika Brakerski and Renen Perlman. Lattice-based fully dynamic multi-key fhe with short ciphertexts. CRYPTO, 2016.

[BR07] Mihir Bellare and Phillip Rogaway. Robust computational secret sharing and a unified account of classical secret-sharing goals. In *CCS*. ACM, 2007.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *SIAM Journal on Computing*, 43(2):831–871, 2014.

[BV15] Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic prfs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In *TCC*, 2015.

[CG99] Ran Canetti and Shafi Goldwasser. An efficient *Threshold* public key cryptosystem secure against adaptive chosen ciphertext attack. In *EUROCRYPT*, 1999.

[CK93] Benny Chor and Eyal Kushilevitz. Secret sharing over infinite domains. *Journal of Cryptology*, 6(2):87–95, 1993.

[CLRS10] Pierre-Louis Cayrel, Richard Lindner, Markus Rückert, and Rosemberg Silva. A lattice-based threshold ring signature scheme. In *LATINCRYPT*, 2010.

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

[DDFY94] Alfredo DeSantis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to share a function securely. In *STOC*, 1994.

[DF89] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In *CRYPTO*, 1989.

[DK05] Yevgeniy Dodis and Jonathan Katz. Chosen-ciphertext security of multiple encryption. In *TCC*, 2005.

[DMP88] Alfredo De-Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge with preprocessing. In *CRYPTO*, 1988.

[Fra89] Yair Frankel. A practical protocol for large group oriented networks. In *EUROCRYPT*, 1989.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM (JACM)*, 33(4):792–807, 1986.

[GGN16] Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In *ACNS*, 2016.

[GHKW17] Rishab Goyal, Susan Hohenberger, Venkata Koppula, and Brent Waters. A generic approach to constructing and proving verifiable random functions. In *TCC*, 2017.

[GJKR01] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Robust threshold DSS signatures. *Inf. Comput.*, 164(1):54–84, 2001.

[GLS15] S. Dov Gordon, Feng-Hao Liu, and Elaine Shi. Constant-round MPC with fairness and guarantee of output delivery. In *CRYPTO*, pages 63–82, 2015.

[Gol14] Oded Goldreich. Valiant's polynomial-size monotone formula for majority. 2014.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, 2006.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, 2008.

[GRJK07] Rosario Gennaro, Tal Rabin, Stanislaw Jarecki, and Hugo Krawczyk. Robust and efficient sharing of RSA functions. *J. Cryptology*, 20(3):393, 2007.

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

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In *STOC*, 2015.

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

[LS90] Dror Lapidot and Adi Shamir. Publicly verifiable non-interactive zero-knowledge proofs. In *CRYPTO*, 1990.

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

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, 2012.

[MM11] Daniele Micciancio and Petros Mol. Pseudorandom knapsacks and the sample complexity of lwe search-to-decision reductions. In *CRYPTO*. 2011.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*. 2012.

[MSS11] Steven Myers, Mona Sergi, and Abhi Shelat. Threshold fully homomorphic encryption and secure computation. *IACR Cryptology ePrint Archive*, 2011:454, 2011.

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

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, 2009.

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

[PW11] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. *SIAM Journal on Computing*, 40(6):1803–1844, 2011.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009.

[SG02] Victor Shoup and Rosario Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *J. Cryptology*, 15(2):75–96, 2002.

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

[Sho00] Victor Shoup. Practical threshold signatures. In *EUROCRYPT*, 2000.

[SS01] Douglas R. Stinson and Reto Strobl. Provably secure distributed schnorr signatures and a (t, n) threshold scheme for implicit certificates. In *ACISP*, 2001.

[Val84] Leslie G. Valiant. Short monotone formulae for the majority function. *J. Algorithms*, 1984.

[XXZ11] Xiang Xie, Rui Xue, and Rui Zhang. Efficient threshold encryption from lossy trapdoor functions. In *PQCrypto*, 2011.

A Other Basic Cryptographic Primitives

In this section, we recall the definitions of additional cryptographic primitives.

A.1 Non-Interactive Commitments

We recall the definition of (perfectly binding) non-interactive commitment schemes. The presentation is based on [Bit17].

Definition A.1 (Non-Interactive Commitments [Blu83])**.** An algorithm $\mathsf{C} = (\mathsf{C.Com})$ is a non-interactive commitment scheme if the following conditions are true. For a message $x \in \{0,1\}^*$ and randomness $r \in \{0,1\}^\lambda$, let $\mathsf{com} \leftarrow \mathsf{C.Com}(x; r)$. Then:

1. **Perfect Binding**: For every security parameter $\lambda \in \mathbb{N}$, and string $\mathsf{com} \in \{0,1\}^*$, there exists at most a single $x \in \{0,1\}^*$ such that com is a commitment to x:
$$\forall r_0, r_1 \in \{0,1\}^\lambda \quad \text{if} \quad \mathsf{C.Com}(x_0; r_0) = \mathsf{C.Com}(x_1; r_1) \quad \text{then} \quad x_0 = x_1.$$
2. **Computational Hiding**: For every $\lambda \in \mathbb{N}$, $x_0, x_1 \in \{0,1\}^{\mathsf{poly}(\lambda)}$, the following distributions are computationally indistinguishable:
$$\left\{ \mathsf{com}_0 : \begin{array}{c} r \xleftarrow{\text{R}} \{0,1\}^\lambda \\ \mathsf{com}_0 \leftarrow \mathsf{C.Com}(x_0; r) \end{array} \right\} \approx_c \left\{ \mathsf{com}_1 : \begin{array}{c} r \xleftarrow{\text{R}} \{0,1\}^\lambda \\ \mathsf{com}_1 \leftarrow \mathsf{C.Com}(x_0; r) \end{array} \right\}.$$

The work of [Blu83] constructs such non-interactive commitments from injective one-way functions and [BOV03] constructs them from regular one-way functions and the worst-case assumption that there exists a problem solvable in deterministic time $2^{O(n)}$ with non-deterministic circuit complexity $2^{\Omega(n)}$. Recently, [GHKW17] constructed such non-interactive commitment schemes from the LWE.

A.2 Pseudorandom Functions

Definition A.2 (PRF [GGM86])**.** Fix the security parameter λ. A PRF $F : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^m$ with key space $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$ is secure if for all efficient algorithms $\mathcal{A}$,
$$\left| \Pr\left[k \leftarrow \mathcal{K} : \mathcal{A}^{F(k,\cdot)}(1^\lambda) = 1 \right] - \Pr\left[f \xleftarrow{\$} \mathsf{Funcs}(\mathcal{X}, \mathcal{Y}) : \mathcal{A}^{f(\cdot)}(1^\lambda) = 1 \right] \right| = \mathsf{negl}(\lambda)$$
where $\mathsf{Funcs}(\mathcal{X}, \mathcal{Y})$ denotes the set of all functions with domain $\mathcal{X}$ and range $\mathcal{Y}$.

A.3 Signature Scheme

Definition A.3 (Signature Scheme)**.** A signature scheme S is a tuple of algorithms $\mathsf{S} = (\mathsf{S.KeyGen}, \mathsf{S.Sign}, \mathsf{S.Verify})$ defined as follows:

- $\mathsf{S.KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{vk})$: On input the security parameter λ, the key generation algorithm outputs a signing key sk and a verification key vk.
- $\mathsf{S.Sign}(\mathsf{sk}, m) \to \sigma$: On input a signing key sk, and a message $m \in \{0,1\}^*$, the signing algorithm outputs a signature σ.

- $\mathsf{S.Verify}(\mathsf{vk}, m, \sigma) \to \{0,1\}$: On input a verification key vk, a message m and, a signature σ, the verification algorithm accepts or rejects.

We require a signature scheme S to satisfy the following correctness and security properties.

Definition A.4 (Correctness)**.** We say that a signature scheme S is correct if for all $\lambda \in \mathbb{N}$, $\mu \in \mathcal{M}$, $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{S.KeyGen}(1^\lambda)$, we have that

$$\Pr[\mathsf{S.Verify}(\mathsf{vk}, \mathsf{S.Sign}(\mathsf{sk}, m)) = 1] = 1.$$

Definition A.5 (Unforgeability)**.** We say that a signature scheme satisfies unforgeability if for any PPT adversary $\mathcal{A}$, the following experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{S},\mathsf{uf}}(1^\lambda)$ outputs 1 with negligible probability:

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

1. The challenger runs $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{S.KeyGen}(1^\lambda)$, and provides vk to $\mathcal{A}$.
2. $\mathcal{A}$ issues a polynomial number of adaptive queries m. For each query, the challenger computes $\sigma \leftarrow \mathsf{S.Sign}(\mathsf{sk}, m)$ and provides σ to $\mathcal{A}$.
3. At the end of the experiment, $\mathcal{A}$ outputs a forgery (m^*, σ^*). The experiment outputs 1 if $\mathsf{S.Verify}(\mathsf{vk}, m^*, \sigma^*)$ accepts and m^* was not previously queried by $\mathcal{A}$.

A.4 Public Key Encryptions

Definition A.6 (PKE)**.** A public key encryption (PKE) scheme PKE is a tuple of algorithms $\mathsf{PKE} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Encrypt}, \mathsf{PKE.Decrypt})$ defined as follows:

- $\mathsf{PKE.KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$: On input the security parameter λ, the key generation algorithm outputs a secret key sk and a public key pk.
- $\mathsf{PKE.Encrypt}(\mathsf{pk}, m) \to \mathsf{ct}$: On input a public key pk, and a message $m \in \mathcal{M}$, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{PKE.Decrypt}(\mathsf{sk}, \mathsf{ct}) \to m'$: On input a secret key sk, and a ciphertext ct, the decryption algorithm outputs a message m'.

We require a public key encryption scheme PKE to satisfy the following correctness and security properties.

Definition A.7 (Correctness)**.** We say that a PKE scheme PKE is correct if for all $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$, $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, we have that

$$\Pr[\mathsf{PKE.Decrypt}(\mathsf{sk}, \mathsf{PKE.Encrypt}(\mathsf{pk}, m)) = m] = 1.$$

For this work, we use the standard notion of chosen ciphertext attack (CCA2) security for public key encryption.

Definition A.8 (Security)**.** We say that a PKE scheme satisfies CCA security if for any PPT adversary $\mathcal{A}$, the following experiments $\mathsf{Expt}_{\mathcal{A},\mathsf{PKE},0}(1^\lambda)$ and $\mathsf{Expt}_{\mathcal{A},\mathsf{PKE},1}(1^\lambda)$ are computationally indistinguishable:

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

1. The challenger runs $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, and provides pk to $\mathcal{A}$.
2. $\mathcal{A}$ makes a polynomial number of adaptive decryption queries ct. For each query ct, the challenger provides $\mathsf{PKE.Decrypt}(\mathsf{sk}, \mathsf{ct})$ to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a pair of messages (m_0^*, m_1^*).
4. The challenger computes $\mathsf{ct}_b^* \leftarrow \mathsf{PKE.Encrypt}(\mathsf{pk}, m_b^*)$, and provides ct_b^* to $\mathcal{A}$.
5. $\mathcal{A}$ continues making a polynomial number of adaptive decryption queries ct for which $\mathsf{ct} \neq \mathsf{ct}_b^*$.
6. $\mathcal{A}$ outputs a guess b', which is the output of the experiment.

B GSW Modification

We briefly recall the GSW construction. We describe the construction ignoring the precise parameters, and the specifics of the "gadget matrix." A more formal description can be found in [GSW13].

Fix the security parameter λ, and let n, m, q, and χ be an appropriately chosen LWE parameters where q is a prime. Also, let the matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ be the standard *gadget matrix* and let $\mathbf{w} \in \mathbb{Z}_q^m$ be a vector with entry 1 in the mth component and 0 elsewhere. We have the property that $\mathbf{G} \cdot \mathbf{w} = q/2$.

We define the GSW encryption scheme as follows:

- $\mathsf{FHE.Setup}(1^\lambda, 1^d)$: The key generation algorithm samples a uniformly random matrix $\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n \times m}$. It also samples a uniformly random vector $\mathbf{s} \xleftarrow{R} \mathbb{Z}_q^n$ and an error vector $\mathbf{e} \leftarrow \chi$ and defines $\tilde{\mathbf{A}} = \begin{pmatrix} \mathbf{A} \\ \mathbf{s}^T\mathbf{A} + \mathbf{e}^T \end{pmatrix}$. Then, it sets
$$\mathsf{pp} = \tilde{\mathbf{A}} \qquad \mathsf{sk} = \begin{pmatrix} -s & 1 \end{pmatrix}.$$
- $\mathsf{FHE.Encrypt}(\mathsf{pk}, \mu)$: The encryption algorithm generates a uniform matrix $\mathbf{R} \xleftarrow{R} \{0,1\}^{m \times m}$ and outputs
$$\mathsf{ct} = \tilde{\mathbf{A}} \cdot \mathbf{R} + \mu \cdot \mathbf{G}.$$
- $\mathsf{FHE.Decrypt}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct})$: The decryption algorithm computes $y = \langle \mathsf{sk}, \mathsf{ct} \cdot \mathbf{w}_m \rangle$ and outputs 0 if $y \in [-q/4, q/4]$ and outputs 1 otherwise.

Given messages $\mu_1, \mu_2 \in \mathbb{Z}_q$, we can homomorphically add and multiply on the ciphertexts as follows:

$$\begin{aligned} \mathsf{ct}_1 \;\; \mathsf{ADD} \;\; \mathsf{ct}_2 &= \mathsf{ct}_1 + \mathsf{ct}_2 \\ &= \left(\tilde{\mathbf{A}} \cdot \mathbf{R}_1 + \mu_1 \cdot \mathbf{G}\right) + \left(\tilde{\mathbf{A}} \cdot \mathbf{R}_2 + \mu_2 \cdot \mathbf{G}\right) \\ &= \tilde{\mathbf{A}} \cdot (\mathbf{R}_1 + \mathbf{R}_2) + (\mu_1 + \mu_2) \cdot \mathbf{G} \end{aligned}$$

$$\begin{aligned} \mathsf{ct}_1 \;\; \mathsf{MULT} \;\; \mathsf{ct}_2 &= \mathsf{ct}_1 \cdot \mathbf{G}^{-1}(\mathsf{ct}_2) \\ &= \left(\tilde{\mathbf{A}}\mathbf{R}_1 + \mu_1 \cdot \mathbf{G}\right) \cdot \mathbf{G}^{-1}(\mathsf{ct}_2) \\ &= \tilde{\mathbf{A}}\mathbf{R}_1 \cdot \mathbf{G}^{-1}(\mathsf{ct}_2) + \mu_1 \cdot \mathsf{ct}_2 \\ &= \tilde{\mathbf{A}}(\mathbf{R}_1 \cdot \mathbf{G}^{-1}(\mathsf{ct}_2) + \mu_1(\tilde{\mathbf{A}}\mathbf{R}_2 + \mu_2 \cdot \mathbf{G}) \\ &= \tilde{\mathbf{A}}\left(\mathbf{R}_1 \cdot \mathbf{G}^{-1}(\mathsf{ct}_2) + \mu_1 \cdot \mathbf{R}_2\right) + \mu_1\mu_2 \cdot \mathbf{G} \end{aligned}$$

Note that for a ciphertext $\hat{\mathsf{ct}} = \tilde{\mathbf{A}}\hat{\mathbf{R}} + \hat{\mu} \cdot \mathbf{G}$, the decryption procedure can be described as computing the inner product

$$\langle \mathsf{sk}, \hat{\mathsf{ct}} \cdot \mathbf{w}_m \rangle = (q/2) \cdot \hat{\mu} + \mathbf{e}^T \cdot \hat{\mathbf{R}} \cdot \mathbf{w}_m$$

and rounding the decryption noise $\mathbf{e}^T \cdot \mathbf{r}^*$ for some low-norm integer vector $\hat{\mathbf{r}} = \hat{\mathbf{R}} \cdot \mathbf{w}_m$.

Tweak: For the decryption noise to always be a integer multiple of a multiplicative constant c, we can modify the construction by simply modifying the error vector $\mathbf{e}$ by c at setup. Namely, the algorithm $\mathsf{FHE.Setup}$ defines the public key

$$\tilde{\mathbf{A}} = \begin{pmatrix} \mathbf{A} \\ \mathbf{s}^T\mathbf{A} + c \cdot \mathbf{e}^T \end{pmatrix}.$$

Now, for any homomorphic computation, the decryption noise is the inner product $c \cdot \mathbf{e}^T \cdot \hat{\mathbf{r}}$, which is an integer multiple of c.

The security of the GSW scheme relies on the public parameter matrix $\tilde{\mathbf{A}}$ being computationally indistinguishable from a uniformly random matrix in $\mathbb{Z}_q^{(n+1)\times m}$ by LWE. It is easy to see that multiplying the error vector $\mathbf{e}$ by c does not effect the reduction. In particular, given an LWE sample $(\mathbf{A}, \mathbf{u})$, the challenger can define the public matrix

$$\tilde{\mathbf{A}} = \begin{pmatrix} \mathbf{A} \\ c \cdot \mathbf{u}^T \end{pmatrix}.$$

If $\mathbf{u} = \mathbf{s}^T\mathbf{A} + \mathbf{e}^T$, then this is the correct distribution as in the real scheme as a scalar multiplication by a nonzero integer over a prime modulus is bijective. If $\mathbf{u}$ is a uniformly random vector in $\mathbb{Z}_q^m$, then again, since we are working over a prime modulus, the matrix $\tilde{\mathbf{A}}$ is a uniformly random matrix in $\mathbb{Z}_q^{(n+1)\times m}$. Now, applying the leftover hash lemma, the ciphertext is statistically uniform and therefore, the message is hidden.

C Proofs in Section 4

C.1 Proof of Lemma 4.14

Fix an access structure $\mathbb{A} \in \{0,1\}$-LSSS, and let $\mathbf{M}$ be a share matrix for $\mathbb{A}$ with partitions $T_1, \ldots, T_N \subseteq [\ell]$ as specified in Definition 4.8. To prove the lemma, it is sufficient to show that for any $S \in \mathbb{A}$, we can efficiently find a minimal valid share set $T \subseteq \bigcup_{i \in S} T_i$. This can be computed recursively by setting $T = \bigcup_{i \in S} T_i$ and defining

$$T = \begin{cases} T & \text{if } (1,0,\ldots,0) \notin \mathsf{span}(\{\mathbf{M}[j]\}_{j \in T \setminus \{i\}}) \\ T \setminus \{i\} & \text{if } (1,0,\ldots,0) \in \mathsf{span}(\{\mathbf{M}[j]\}_{j \in T \setminus \{i\}}) \end{cases}$$

for all $i \in T$. The iteration terminates if no more parties can be removed. At each recursive step, an element $P_i \in S'$ is removed or the iteration finishes. Therefore, it is easy to see that the procedure terminates in polynomial time. It is also easy to see that the resulting set S' is a minimal valid share set.

C.2 Proof of Theorem 4.19

Given a monotone Boolean formula $C : \{0,1\}^N \to \{0,1\}$, we construct a share matrix for the linear secret sharing scheme that satisfies Definition 4.8 and 4.13.

We assume that all the input wires of C have fan-out 1.[2] This is without loss of generality since given a monotone Boolean formula $C : \{0,1\}^N \to \{0,1\}$, we can always construct a new formula $C' : \{0,1\}^{N\ell} \to \{0,1\}$ with input fan-out 1 by replicating each input wire as a separate input to C (here, ℓ is bounded by $|C| = \mathsf{poly}(N)$). This provides a mapping $\phi : [N] \to \mathcal{P}([N])$ of each input wire x_i of C to a partition of input wires $\{x_{i,1}, \ldots, x_{i,\ell}\}$ of C'. Now, let $(\mathsf{s}'_{1,1}, \ldots, \mathsf{s}'_{N,\ell}) \leftarrow \mathsf{SS}'.\mathsf{Share}(\mathsf{k}, \mathbb{A}_{C'})$ be the shares of C' via a linear secret sharing scheme SS'. Then, it is easy to see that defining a secret sharing scheme SS that produces shares by running $(\mathsf{s}'_{1,1}, \ldots, \mathsf{s}'_{N,\ell}) \leftarrow \mathsf{SS}'.\mathsf{Share}(\mathsf{k}, \mathbb{A}_{C'})$ and setting $\mathsf{s}_i = \{\mathsf{s}'_{i,1}, \ldots, \mathsf{s}'_{i,\ell}\}$ results in a special linear secret sharing scheme that satisfies correctness and security.

Now, given a special monotone Boolean formula $C : \{0,1\}^N \to \{0,1\}$, we describe how to achieve a special linear secret sharing scheme using a "folklore algorithm." To describe the algorithm formally, we interpret C as a tree with a unique root node r being the single output gate and the leaf nodes being the input variables to C. We let $\mathcal{V}$ be the set of nodes of the tree, $\mathcal{V}_{\mathrm{OR}} \subset \mathcal{V}$ the set of all nodes that correspond to OR gates and $\mathcal{V}_{\mathrm{AND}} \subset \mathcal{V}$ the set of all nodes that correspond to AND gates.

We let $n = |\mathcal{V}_{\mathrm{OR}} \bigcup \mathcal{V}_{\mathrm{AND}}|$ be the number of all nodes in the tree excluding the input nodes. Then, we fix a canonical indexing v that identifies each node $v \in \mathcal{V}_{\mathrm{OR}} \bigcup \mathcal{V}_{\mathrm{AND}}$ with an index $i \in [n]$. This assigns the root node $r = \mathsf{v}(1)$ and for each pair of nodes $\mathsf{v}(i), \mathsf{v}(j)$ for $i < j$, the depth of $\mathsf{v}(i)$ is less than the depth of $\mathsf{v}(j)$.

The share algorithm $\mathsf{SS}.\mathsf{Share}(\mathsf{k}, \mathbb{A}_C)$ generates the sharing matrix using the folklore algorithm as follows:

"Folklore" Algorithm

Input: A special monotone Boolean formula $C : \{0,1\}^N \to \{0,1\}$.
Output: An LSSS share matrix $\mathbf{M}$ for the access structure induced by C.

1. Label r with a length one vector $\mathbf{m}_r = (1)$.
2. Initialize a counter $\mathsf{count} = 1$.
3. For $v = \mathsf{v}(i)$ for $i = 1, \ldots, n$:
   (a) If $v \in \mathcal{V}_{\mathrm{OR}}$, label its children with the same vector as m.
   (b) If $v \in \mathcal{V}_{\mathrm{AND}}$ with vector $\mathbf{m}_v$, then pad $\mathbf{m}_v$ with 0's at the end (if necessary) to make it of length count. Denote the new vector by $\mathbf{m}'$. Then, label one of its children with the vector $(\mathbf{m}', 1)$ and the other with the vector $(0, \ldots, 0, -1)$ of length $\mathsf{count} + 1$. Then increase count by 1.
4. Once the entire tree is labeled, the vectors associated with the leaf nodes form the rows of the sharing matrix $\mathbf{M}$. If these vectors have different lengths, the shorter vectors are padded with 0's.

[2] In particular, C is a *special monotone Boolean formula.*

At the end of the algorithm, the matrix $\mathbf{M}$ consists of N rows and n' columns where n' is the total number of AND gates in C.

Let $P = \{P_1, \ldots, P_N\}$ be a set of parties. Then, it is easy to see that the algorithm above outputs a linear secret sharing share matrix $\mathbf{M}$ with each party P_i assigned a row of $\mathbf{M}$. We now show that the share matrix $\mathbf{M}$ satisfies the properties of a share matrix required by a special linear secret sharing scheme (Definition 4.13).

We note that since each share of P consists of a single element in $\mathbb{Z}_p$, it is sufficient to show that for any secret k shared into $\mathsf{s}_1, \ldots, \mathsf{s}_N$, and any $S \in \mathbb{A}$, there exists a set $S' \subseteq S$ such that $S' \in \mathbb{A}$ and $\mathsf{k} = \sum_{i \in S'} \mathsf{s}_i$. Let $v_i \in \mathcal{V}$ be an input node identified with party P_i and $\mathbf{m}_{v_i}$ its associated vector in $\mathbf{M}$. Then, we show the following claim, which proves that $\mathbf{M}$ satisfies the requirements of a special linear secret sharing scheme.

Claim C.1. *For a minimal valid party set $S \subseteq P$, we have $\sum_{i \in S} \mathbf{m}_{v_i} = \mathbf{m}_r$.*

Proof. We prove the claim by induction on the height of T.

- The claim certainly holds when T has height 1 or 2, which is the case where C simply consists of a single input gate or two input. and a gate.
- Now, consider the case when the height of T is strictly greater than 2. Let r be the root of T and let s and t be the children of r. Then, denote by C_s and C_t the circuits induced by the subtree of T with root s and t, respectively and let $\mathbb{A}_s$ and $\mathbb{A}_t$ be the access structures induced by C_s and C_t. Assume without loss of generality that the leafs of C_s and C_t correspond to parties $P_s = \{P_1, \ldots, P_k\}$ and $P_t = \{P_{k+1}, \ldots, P_N\}$. We must consider two cases.
  - If r is an AND gate, then for any minimal valid share set $S \subseteq P$, we must have that $S \cap P_s$ and $S \cap P_t$ are minimal valid share sets for $\mathbb{A}_s$ and $\mathbb{A}_t$ respectively, simply by the minimality of S. Therefore, by construction, we have $\mathbf{m}_r = \mathbf{m}_s + \mathbf{m}_t$ and it follows that
  $$\sum_{i \in S} \mathbf{m}_{v_i} = \sum_{i \in S \cap P_s} \mathbf{m}_{v_i} + \sum_{i \in S \cap P_t} \mathbf{m}_{v_i} = \mathbf{m}_r$$
  by induction.
  - If r is an OR gate, then for any minimal valid share set $S \subseteq P$, we must again either have $S \cap P_s$ is a minimal valid share set for $\mathbb{A}_s$ and $S \cap P_t = \emptyset$ or vice versa. Since $\mathbf{m}_r = \mathbf{m}_s = \mathbf{m}_t$ by construction, it follows that $\sum_{i \in S} \mathbf{m}_{v_i} = v_r$ by induction.

This concludes the proof of the claim. □

This concludes the proof of the theorem.

D Proofs and Discussions in Section 5

D.1 Proof of Theorem 5.8

Fix the security parameter 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 message $\mu_i \in \{0,1\}$ for $i \in [k]$. Let $(\mathsf{pp}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow$

$\mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A})$, $\mathsf{ct}_i \leftarrow \mathsf{TFHE.Encrypt}(\mathsf{pk}, \mu_i)$ for $i \in [k]$, and $\hat{\mathsf{ct}} \leftarrow \mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$. We must show that

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

By definition, $\mathsf{pk} = \mathsf{fhepk}$ and $\mathsf{sk}_i = \mathsf{fhesk}_i$ where $(\mathsf{fhepk}, \mathsf{fhesk}) \leftarrow \mathsf{FHE.Setup}(1^\lambda, 1^d)$ and $(\mathsf{fhesk}_1, \ldots, \mathsf{fhesk}_N) \leftarrow \mathsf{SS.Share}(\mathsf{fhesk}, \mathbb{A})$. The TFHE encryption algorithm $\mathsf{TFHE.Encrypt}(\mathsf{pk}, \mu_i)$ computes the ciphertexts $\mathsf{ct}_i \leftarrow \mathsf{FHE.Encrypt}(\mathsf{fhepk}, \mu_i)$, and $\mathsf{TFHE.Eval}(C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ computes $\hat{\mathsf{ct}} \leftarrow \mathsf{FHE.Eval}(C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$. Finally, given $\hat{\mathsf{ct}}$ and $\mathsf{sk}_i = \{\mathbf{s}_j\}_{j \in T_i}$, the partial decryption algorithm returns $\tilde{\mathbf{p}}_j \leftarrow \mathsf{FHE.Decode}_0(\mathbf{s}_j, \hat{\mathsf{ct}}) + e_j$ for $j \in T_i$ where $e_j \overset{\text{R}}{\leftarrow} [-B_{\mathsf{sm}}, B_{\mathsf{sm}}]$, and $\mathsf{TFHE.FinDec}(\{\mathsf{p}_i\}_{i \in S})$ computes a minimal valid share set $T \subseteq \bigcup_{i \in S} T_i$ for $\mathbb{A}$ and returns $\mathsf{FHE.Decode}_1\big(\sum_{j \in T} \tilde{\mathbf{p}}_j\big)$.

Now, by linearity of $\mathsf{FHE.Decode}_0$, we have

$$\begin{aligned} \sum_{j \in T} \tilde{\mathbf{p}}_j &= \sum_{j \in T} \mathsf{FHE.Decode}_0(\mathbf{s}_j, \hat{\mathsf{ct}}) + e_j \\ &= \mathsf{FHE.Decode}_0\left(\sum_{j \in T} \mathbf{s}_j, \hat{\mathsf{ct}}\right) + \underbrace{\sum_{j \in T} e_j}_{e_T} \end{aligned}$$

where $|e_T| \le \ell \cdot B_{\mathsf{sm}}$. By the property of minimal valid share set T (Definition 4.9) and the correctness of SS (Definition 4.5), $\mathsf{fhesk} = \sum_{i \in T} \mathsf{fhesk}_i$. Therefore,

$$\begin{aligned} \sum_{i \in T} \mathsf{p}_i &= \mathsf{FHE.Decode}_0(\mathsf{fhesk}, \hat{\mathsf{ct}}) + e_T \\ &= \mu \left\lfloor \frac{q}{2} \right\rceil + e + e_T \end{aligned}$$

by the correctness of $\mathsf{FHE.Decode}_0$ (Definition 3.9). Then, by correctness of $\mathsf{FHE.Decode}_1$ (Definition 3.9), as long as $|e + e_T| = B + \ell \cdot B_{\mathsf{sm}} \le \left\lfloor \frac{q}{4} \right\rceil$, we have

$$\Pr\left[\mathsf{FHE.Decode}_1\left(\sum_{i \in T} \mathsf{p}_i\right) = C(\mu_1, \ldots, \mu_k)\right] = 1 - \mathsf{negl}(\lambda).$$

The theorem follows.

D.2 Proof of Theorem 5.10

Our proof proceeds via a sequence of hybrid experiments between an adversary $\mathcal{A}$ and a challenger.

- H_0: This is the TFHE real security experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}(1^\lambda, 1^d)$ from Definition 5.5. On input an access structure $\mathbb{A} \in \mathbb{S}$ from $\mathcal{A}$, the challenger runs the setup $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A})$ and provides pk to $\mathcal{A}$. It then receives a maximal invalid party set $S^* \subseteq \{P_1, \ldots, P_N\}$ and a set of messages $\mu_1, \ldots, \mu_k \in \{0, 1\}$. The challenger then provides the keys $\{\mathsf{sk}_i\}_{i \in S^*}$ and the ciphertexts $\{\mathsf{TFHE.Encrypt}(\mathsf{pk}, \mu_i)\}_{i \in [k]}$ to $\mathcal{A}$. For each query (S, C) that $\mathcal{A}$ makes, the challenger computes $\hat{\mathsf{ct}} \leftarrow \mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ and provides $\{\mathsf{TFHE.PartDec}(\mathsf{pk}, \hat{\mathsf{ct}}, \mathsf{sk}_i)\}_{i \in S}$ to $\mathcal{A}$.

- H_1: Same as H_0, except that the challenger simulates the partial decryptions for each of $\mathcal{A}$'s queries. Specifically, when $\mathcal{A}$ provides the challenge maximal invalid party set $S^* \subseteq \{P_1, \ldots, P_N\}$, the challenger commits to a *maximal invalid share set* T^* containing $\bigcup_{i \in S^*} T_i$ (Definition 4.9). Then, for each query (S, C) that $\mathcal{A}$ makes, the challenger computes $\hat{\mathsf{ct}} \leftarrow \mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ as before, but now it derives each of $\mathsf{TFHE.PartDec}(\mathsf{pk}, \hat{\mathsf{ct}}, \mathsf{sk}_i)$ for $i \in S^*$ only from the set of elements $\{\mathbf{s}_j\}_{j \in T^*}$ and $C(x)$ as follows.

  Recall that the partial decryptions $\mathsf{TFHE.PartDec}(\mathsf{pk}, \hat{\mathsf{ct}}, \mathsf{sk}_i)$ consists of a set of $\mathbb{Z}_q$ elements $\mathsf{p}_i = \{\tilde{\mathbf{p}}_j\}_{j \in T_i}$. Then, for each $j \in T_i$, the challenger computes $\tilde{\mathbf{p}}_j$:

  - If $j \in T_i \cap T^*$, then it sets $\tilde{\mathbf{p}}_j = \mathsf{FHE.Decode}_0(\mathbf{s}_j, \hat{\mathsf{ct}}) + e_j$ for $e_j \xleftarrow{\text{R}} [-B_{\mathsf{sm}}, B_{\mathsf{sm}}]$ as in the real scheme.
  - If $j \notin T_i \cap T^*$, then the challenger first computes a minimal valid share set $T \subseteq T^* \cup \{j\}$. Such set T exists since T^* is a maximal invalid share set and therefore, $T^* \cup \{j\}$ is a valid share set. The challenger then sets

    $$\tilde{\mathbf{p}}_j = \frac{q}{2} \cdot C(x) - \sum_{j' \in T \setminus \{j\}} \mathsf{FHE.Decode}_0(\mathbf{s}_{j'}, \hat{\mathsf{ct}}) + \tilde{e}$$

    for $\tilde{e} \xleftarrow{\text{R}} [-B_{\mathsf{sm}}, B_{\mathsf{sm}}]$.

  The challenger then provides $\mathsf{p}_i = \{\tilde{\mathbf{p}}_j\}_{j \in T_i}$ for $i \in S$ to $\mathcal{A}$.

  In Lemma D.1 below, we show that the hybrid experiments H_0 and H_1 are statistically indistinguishable.

- H_2: Same as H_1, except that the challenger now samples each key share sk_i uniformly at random. Specifically, on input an access structure $\mathbb{A} \in \mathbb{S}$ from $\mathcal{A}$, the challenger runs $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A})$ by generating the FHE keys $(\mathsf{fhepk}, \mathsf{fhesk}) \leftarrow \mathsf{FHE.Setup}(1^\lambda, 1^d)$, but now, instead of secret sharing fhesk, it secret shares the zero strings $\mathsf{fhesk}_1, \ldots, \mathsf{fhesk}_N \leftarrow \mathsf{SS.Share}(\mathbf{0}, \mathbb{A})$. The rest of the experiment remains unchanged.

  In Lemma D.2, below, we show that the hybrid experiments H_1 and H_2 are perfectly indistinguishable. Note that in this experiment, the challenger generates the keys $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N)$ and answers $\mathcal{A}$'s queries without using sk or $x_1, \ldots, x_k$. Therefore, the challenger in this experiment corresponds to the simulator in $\mathsf{Expt}_{\mathcal{A},\mathsf{Ideal}}(1^\lambda, 1^d)$.

We now argue that the consecutive hybrid experiments above are indistinguishble. For an adversary $\mathcal{A}$, we write $\mathsf{H}_i(\mathcal{A})$ to denote the output of H_i.

Lemma D.1. *If $B/B_{\mathsf{sm}} = \mathsf{negl}(\lambda)$, then for all adversaries $\mathcal{A}$, $|\Pr[\mathsf{H}_0(\mathcal{A}) = 1] - \Pr[\mathsf{H}_1(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.*

Proof. The only difference $\mathcal{A}$'s views in H_0 and H_1 is the way the challenger computes $\mathsf{p}_i \leftarrow \mathsf{TFHE.PartDec}(\mathsf{pk}, \hat{\mathsf{ct}}, \mathsf{sk}_i)$ for each $i \in S$ on $\mathcal{A}$'s adaptive queries (S, C). In H_0, the challenger generates p_i by computing $\tilde{\mathbf{p}}_j = \mathsf{FHE.Decode}_0(\mathbf{s}_j, \hat{\mathsf{ct}}) + e_j$ for $j \in T_i$, $e_j \leftarrow [-B_{\mathsf{sm}}, B_{\mathsf{sm}}]$ and setting $\mathsf{p}_i = \{\tilde{\mathbf{p}}_j\}_{j \in T_i}$. In H_1, the challenger first commits to a maximal invalid share set T^* at setup. Then, it computes $\tilde{\mathbf{p}}_j$ for $j \in T_i \cap T^*$ as in the real scheme, but deviates from the real scheme for $j \notin T_i \cap T^*$. Therefore, we restrict our attention to the case of $j \notin T_i \cap T^*$.

In H_1, the challenger first computes a minimal valid share set $T \subseteq T^* \cup \{j\}$ and sets

$$\tilde{\mathbf{p}}_j = \frac{q}{2} \cdot C(x) - \sum_{j' \in T \setminus \{j\}} \mathsf{FHE.Decode}_0(\mathbf{s}_{j'}, \hat{\mathsf{ct}}) + \tilde{e}$$

for $\tilde{e} \xleftarrow{\mathrm{R}} [-B_{\mathsf{sm}}, B_{\mathsf{sm}}]$. We first note that by the correctness of FHE (Definition 3.9), we have $q/2 \cdot C(x) = \mathsf{FHE.Decode}_0(\mathsf{sk}, \hat{\mathsf{ct}}) + e$ for some noise $e \in [-B, B]$. Therefore, we can rewrite the relation

$$\begin{aligned}\tilde{\mathbf{p}}_j &= \frac{q}{2} \cdot C(x) - \sum_{j' \in T \setminus \{j\}} \mathsf{FHE.Decode}(\mathbf{s}_{j'}, \hat{\mathsf{ct}}) + \tilde{e} \\ &= \mathsf{FHE.Decode}_0(\mathsf{sk}, \hat{\mathsf{ct}}) + e - \sum_{j' \in T \setminus \{j\}} \mathsf{FHE.Decode}(\mathbf{s}_{j'}, \hat{\mathsf{ct}}) + \tilde{e} \\ &= \mathsf{FHE.Decode}\left(\mathsf{sk} - \sum_{j' \in T \setminus \{j\}} \mathbf{s}_j, \hat{\mathsf{ct}}\right) + e + \tilde{e}\end{aligned}$$

Now, by the property of a minimal valid share set $T \subseteq T^* \cup \{j\}$, we have $\sum_{j' \in T} \mathbf{s}_j = \mathsf{sk}$, and therefore, as long as $j \in T$, we have $\mathsf{sk} - \sum_{j' \in T \setminus \{j\}} \mathbf{s}_{j'} = \mathbf{s}_j$. This is indeed the case since T^* is a maximal invalid share set.

We now have the relation $\tilde{\mathbf{p}}_j = \mathsf{FHE.Decode}_0(\mathbf{s}_j, \hat{\mathsf{ct}}) + e + \tilde{e}$, which is identically distributed as in H_0 except for the additive noise term e. By assumption, $B/B_{\mathsf{sm}} = \mathsf{negl}(\lambda)$ and the lemma follows from the smudging lemma (Lemma 3.2). $\square$

Lemma D.2. *If* SS *is a secret sharing scheme satisfying privacy (Definition 4.6), then for all adversaries* $\mathcal{A}$, $|\Pr[\mathsf{H}_1(\mathcal{A}) = 1] - \Pr[\mathsf{H}_2(\mathcal{A}) = 1]| = 0$.

Proof. The lemma follows from the privacy property of the secret sharing scheme SS in a straightforward way. The only difference between H_1 and H_2 is in the way the key shares $\mathsf{fhesk}_1, \ldots, \mathsf{fhesk}_N$ are generated. In H_1, the challenger sets $\mathsf{fhesk}_1, \ldots, \mathsf{fhesk}_N \leftarrow \mathsf{SS.Share}(\mathsf{fhesk}, \mathbb{A})$, while in H_2, the challenger sets $\mathsf{fhesk}_1, \ldots, \mathsf{fhesk}_N \leftarrow \mathsf{SS.Share}(\mathsf{fhesk}, \mathbb{A})$. Then, by the privacy property of SS (Definition 4.6), as long as $\mathcal{A}$ is given access to $\{\mathsf{fhesk}_i\}_{i \in S^*}$ for an invalid set S^*, the two distributions are identical. This is indeed the case by the specification of the security game and the lemma follows. $\square$

Combining Lemmas D.1, and D.2, the theorem follows.

D.3 Proof of Theorem 5.12

Fix the security parameter λ, depth bound d, access structure $\mathbb{A}_t \in \mathsf{TAS}$, a circuit $C : \{0,1\}^k \to \{0,1\}$ of depth at most d, $S \in \mathbb{A}$, and message $\mu_i \in \{0,1\}$ for $i \in [k]$. Let $(\mathsf{pp}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A})$, $\mathsf{ct}_i \leftarrow \mathsf{TFHE.Encrypt}(\mathsf{pk}, \mu_i)$ for $i \in [k]$, and $\hat{\mathsf{ct}} \leftarrow \mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$. We must show that

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

By definition, $\mathsf{pk} = \mathsf{fhepk}$, and $\mathsf{sk}_i = \mathsf{fhesk}_i$ where $(\mathsf{fhepk}, \mathsf{fhesk}) \leftarrow \mathsf{FHE.Setup}(1^\lambda, 1^d)$ and $(\mathsf{fhesk}_1, \ldots, \mathsf{fhesk}_N) \leftarrow \mathsf{SS.Share}(\mathsf{fhesk}, \mathbb{A}_t)$. The TFHE encryption algorithm $\mathsf{TFHE.Encrypt}(\mathsf{pk}, \mu_i)$ computes the ciphertexts $\mathsf{ct}_i \leftarrow \mathsf{FHE.Encrypt}(\mathsf{fhepk}, \mu_i)$, and $\mathsf{TFHE.Eval}(C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ computes $\hat{\mathsf{ct}} \leftarrow \mathsf{FHE.Eval}(C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$. Finally, given $\hat{\mathsf{ct}}$ and $\mathsf{sk}_i \in \mathbb{Z}_q$, the partial decryption algorithm computes $\mathsf{p}_i \leftarrow \mathsf{FHE.Decode}_0(\mathsf{sk}_i, \hat{\mathsf{ct}}) + (N!)^2 \cdot e_i$ for $e_i \xleftarrow{\text{R}} [-B_{\mathsf{sm}}, B_{\mathsf{sm}}]$, and $\mathsf{TFHE.FinDec}(\{\mathsf{p}_i\}_{i \in S})$ chooses a subset $S' \subseteq S$ of size t and returns $\mathsf{FHE.Decode}_1(\sum_{i \in S'} \lambda_{i,0}^{S'} \cdot \mathsf{p}_i)$.

Now, by linearity of $\mathsf{FHE.Decode}_0$, we have

$$\begin{aligned}\sum_{i \in S'} \lambda_{i,0}^{S'} \cdot \mathsf{p}_i &= \sum_{i \in S'} \lambda_{i,0}^{S'} \cdot (\mathsf{FHE.Decode}_0(\mathsf{sk}_i, \hat{\mathsf{ct}}) + (N!) e_i) \\ &= \mathsf{FHE.Decode}_0\left(\sum_{i \in S'} \lambda_{i,0}^{S'} \cdot \mathsf{sk}_i, \hat{\mathsf{ct}}\right) + \underbrace{\sum_{i \in S'} \lambda_{i,0}^{S'} \cdot (N!)^2 \cdot e_i}_{e_{S'}} .\end{aligned}$$

By Lemma 4.12, we have $|e_{S'}| \leq (N!)^3 \cdot N \cdot B_{\mathsf{sm}}$. By correctness of SS (Theorem 4.11), $\mathsf{fhesk} = \sum_{i \in S'} \lambda_{i,0}^{S'} \cdot \mathsf{sk}_i$. Therefore,

$$\begin{aligned}\sum_{i \in S'} \lambda_{i,0}^{S'} \cdot \mathsf{p}_i &= \mathsf{FHE.Decode}_0(\mathsf{fhesk}, \hat{\mathsf{ct}}) + e_{S'} \\ &= \mu \left\lfloor \frac{q}{2} \right\rceil + e + e_{S'}\end{aligned}$$

by the correctness of $\mathsf{FHE.Decode}_0$ (Definition 3.9). Then, by correctness of $\mathsf{FHE.Decode}_1$ (Definition 3.9), as long as $|e + e_T| = B + (N!)^3 \cdot N \cdot B_{\mathsf{sm}} \leq \lfloor \frac{q}{4} \rceil$, we have

$$\Pr\left[\mathsf{FHE.Decode}_1\left(\sum_{i \in S'} \lambda_{i,0}^{S'} \cdot \mathsf{p}_i\right) = C(\mu_1, \ldots, \mu_k)\right] = 1 - \mathsf{negl}(\lambda).$$

The theorem follows.

D.4 Proof of Theorem 5.14

Our proof proceeds via a sequence of hybrid experiments between an adversary $\mathcal{A}$ and a challenger.

- H_0: This is the TFHE real security experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}(1^\lambda, 1^d)$ from Definition 5.5. On input an access structure $\mathbb{A}_t \in \mathsf{TAS}$ from $\mathcal{A}$, the challenger runs the setup $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}_t)$ and provides pk to $\mathcal{A}$. It then receives a party set $S^* \subseteq \{P_1, \ldots, P_N\}$ of size $t-1$ and a set of messages $\mu_1, \ldots, \mu_k \in \{0,1\}$. The challenger then provides the keys $\{\mathsf{sk}_i\}_{i \in S^*}$ and the ciphertexts $\{\mathsf{TFHE.Encrypt}(\mathsf{pk}, \mu_i)\}_{i \in [k]}$ to $\mathcal{A}$. For each query (S, C) that $\mathcal{A}$ makes, the challenger computes $\hat{\mathsf{ct}} \leftarrow \mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ and provides $\{\mathsf{TFHE.PartDec}(\mathsf{pk}, \hat{\mathsf{ct}}, \mathsf{sk}_i)\}_{i \in S}$ to $\mathcal{A}$.

- H_1: Same as H_0, except that the challenger simulates the partial decryptions for each of $\mathcal{A}$'s queries. Specifically, for each query (S, C) that $\mathcal{A}$ makes, the challenger computes $\hat{\mathsf{ct}} \leftarrow \mathsf{TFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ as before, but now it derives each of $\mathsf{TFHE.PartDec}(\mathsf{pk}, \hat{\mathsf{ct}}, \mathsf{sk}_i)$ for $i \in S$ only from the set of elements $\{\mathsf{s}_j\}_{j \in S^*}$ and $C(x)$ as follows.

  - If $i \in S^*$, then the challenger simply outputs $\mathsf{p}_i = \mathsf{FHE.Decode}_0(\mathsf{sk}_i, \hat{\mathsf{ct}}) + (N!)^2 \cdot e$ for $e \xleftarrow{\text{R}} [-B_{\mathsf{sm}}, B_{\mathsf{sm}}]$ as in the real scheme.

– If $i \notin S^*$, then the challenger first computes the Lagrange coefficients $\lambda_{j,i}^{S^*}$ for $j \in S^*$. Then it outputs

$$\mathsf{p}_i = \lambda_{0,i}^{S^*} \cdot \frac{q}{2} \cdot C(x) + \sum_{j \in S^*} \lambda_{j,i}^{S^*} \cdot \mathsf{FHE.Decode}_0(\mathsf{sk}_j, \hat{\mathsf{ct}}) + (N!)^2 \cdot e$$

for $e \xleftarrow{\text{R}} [-B_{\mathsf{sm}}, B_{\mathsf{sm}}]$.

In Lemma D.3 below, we show that the hybrid experiments H_0 and H_1 are statistically indistinguishable.

- H_2: Same as H_1, except that the challenger now samples each key share sk_i uniformly at random. Specifically, on input an access structure $\mathbb{A}_t \in \mathsf{TAS}$ from $\mathcal{A}$, the challenger runs $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}_t)$ by generating the FHE keys $(\mathsf{fhepk}, \mathsf{fhesk}) \leftarrow \mathsf{FHE.Setup}(1^\lambda, 1^d)$, but now, instead of secret sharing fhesk, it secret shares the zero strings $\mathsf{fhesk}_1, \ldots, \mathsf{fhesk}_N \leftarrow \mathsf{SS.Share}(\mathbf{0}, \mathbb{A}_t)$. The rest of the experiment remains unchanged.

  In Lemma D.4, below, we show that the hybrid experiments H_1 and H_2 are perfectly indistinguishable. Note that in this experiment, the challenger generates the keys $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N)$ and answers $\mathcal{A}$'s queries without using sk or $x_1, \ldots, x_k$. Therefore, the challenger in this experiment corresponds to the simulator in $\mathsf{Expt}_{\mathcal{A},\mathsf{Ideal}}(1^\lambda, 1^d)$.

We now argue that the consecutive hybrid experiments above are indistinguishble. For an adversary $\mathcal{A}$, we write $\mathsf{H}_i(\mathcal{A})$ to denote the output of H_i.

Lemma D.3. *If $B/B_{\mathsf{sm}} = \mathsf{negl}(\lambda)$, then for all adversaries $\mathcal{A}$,* $|\Pr[\mathsf{H}_0(\mathcal{A}) = 1] - \Pr[\mathsf{H}_1(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.

Proof. The only difference in $\mathcal{A}$'s views in H_0 and H_1 is the way the challenger computes $\mathsf{p}_i \leftarrow \mathsf{TFHE.PartDec}(\mathsf{pk}, \hat{\mathsf{ct}}, \mathsf{sk}_i)$ for each $i \in S$ on $\mathcal{A}$'s adaptive queries (S, C). In H_0, the challenger generates p_i by computing $\mathsf{p}_i = \mathsf{FHE.Decode}_0(\mathsf{sk}_i, \hat{\mathsf{ct}}) + e$ for $e \xleftarrow{\text{R}} [-B_{\mathsf{sm}}, B_{\mathsf{sm}}]$. In H_1, the challenger computes p_i for $i \in S^*$ as in the real scheme, but deviates from the real scheme for $i \notin S^*$. Therefore, we restrict our attention to the case of $i \notin S^*$.

In H_1, for $i \notin S^*$, the challenger first computes the Lagrange coefficients $\lambda_{j,i}^{S^*}$ for $j \in S^*$. Then it outputs

$$\mathsf{p}_i = \lambda_{0,i}^{S^*} \cdot \frac{q}{2} \cdot C(x) + \sum_{j \in S^*} \lambda_{j,i}^{S^*} \cdot \mathsf{FHE.Decode}_0(\mathsf{sk}_j, \hat{\mathsf{ct}}) + (N!)^2 \cdot e$$

for $e \xleftarrow{\text{R}} [-B_{\mathsf{sm}}, B_{\mathsf{sm}}]$. We first note that by the correctness of FHE (Definition 3.9), we have $q/2 \cdot C(x) = \mathsf{FHE.Decode}_0(\mathsf{sk}, \hat{\mathsf{ct}}) + \tilde{e}$ for some noise $\tilde{e} \in [-B, B]$. Therefore, we can rewrite the relation

$$\begin{aligned}
\mathsf{p}_i &= \lambda_{0,i}^{S^*} \cdot \frac{q}{2} \cdot C(x) + \sum_{j \in S^*} \lambda_{j,i}^{S^*} \cdot \mathsf{FHE.Decode}(\mathsf{sk}_j, \hat{\mathsf{ct}}) + (N!)^2 \cdot e \\
&= \lambda_{0,i}^{S^*} \cdot \mathsf{FHE.Decode}_0(\mathsf{sk}, \hat{\mathsf{ct}}) + \tilde{e} + \sum_{j \in S^*} \lambda_{j,i}^{S^*} \cdot \mathsf{FHE.Decode}(\mathsf{sk}_j, \hat{\mathsf{ct}}) + (N!)^2 \cdot e \\
&= \mathsf{FHE.Decode}\left(\lambda_{0,i}^{S^*} \cdot \mathsf{sk} + \sum_{j \in S^*} \lambda_{j,i}^{S^*} \cdot \mathsf{sk}_j, \hat{\mathsf{ct}} \right) + \tilde{e} + (N!)^2 \cdot e
\end{aligned}$$

Now, by the correctness property of secret sharing, we have $\lambda_{0,i}^{S^*} \cdot \mathsf{sk} + \sum_{j \in S^*} \lambda_{j,i}^{S^*} \cdot \mathsf{sk}_j = \mathsf{sk}_i$. Therefore, we have the relation $\mathsf{p}_i = \mathsf{FHE.Decode}_0(\mathsf{sk}_i, \hat{\mathsf{ct}}) + \tilde{e} + (N!)^2 \cdot e$, which is identically distributed as in H_0 except for the additive noise term $\tilde{e}$. By assumption, $\tilde{e}$ is a multiple of $(N!)^2$, which is the multiplicative constant of FHE. The lemma now follows from the fact $B/B_{\mathsf{sm}} = \mathsf{negl}(\lambda)$ and the smudging lemma (Lemma 3.2). □

Lemma D.4. *If* SS *is a secret sharing scheme satisfying privacy (Definition 4.6), then for all adversaries* $\mathcal{A}$*,* $|\Pr[\mathsf{H}_1(\mathcal{A}) = 1] - \Pr[\mathsf{H}_2(\mathcal{A}) = 1]| = 0$.

Proof. The lemma follows from the privacy property of the secret sharing scheme SS in a straightforward way. The only difference between H_1 and H_2 is in the way the key shares $\mathsf{fhesk}_1, \dots, \mathsf{fhesk}_N$ are generated. In H_1, the challenger sets $\mathsf{fhesk}_1, \dots, \mathsf{fhesk}_N \leftarrow \mathsf{SS.Share}(\mathsf{fhesk}, \mathbb{A})$, while in H_2, the challenger sets $\mathsf{fhesk}_1, \dots, \mathsf{fhesk}_N \leftarrow \mathsf{SS.Share}(\mathsf{fhesk}, \mathbb{A})$. Then, by the privacy property of SS (Definition 4.6), as long as $\mathcal{A}$ is given access to $\{\mathsf{fhesk}_i\}_{i \in S^*}$ for an invalid set S^*, the two distributions are identical. This is indeed the case by the specification of the security game and the lemma follows. □

Combining Lemmas D.1, and D.2, the theorem follows.

E Proofs in Section 7

E.1 Proof of Theorem 7.8

The compactness of the universal thresholdizer scheme from Construction 7.7 follows from the compactness of the underlying TFHE scheme in a straightforward way. Fix the security parameter λ, depth bound d, access structure $\mathbb{A}$, message $x \in \{0,1\}^k$ and circuit $C : \{0,1\}^k \to \{0,1\}$ of depth at most d. Let $(\mathsf{pp}, \mathsf{s}_1, \dots, \mathsf{s}_N) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}, x)$, and $\mathsf{y}_i \leftarrow \mathsf{UT.Eval}(\mathsf{pp}, \mathsf{s}_i, C)$ for $i \in [N]$.

By construction, $\mathsf{y}_i = (\mathsf{p}_i, \pi_i)$ where p_i is the output of $\mathsf{TFHE.Eval}$, and π_i is a PZK proof of the statement $\Pi_i(\mathsf{com}_i, \hat{\mathsf{ct}}, \mathsf{p}_i)$. By the compactness property TFHE, there exists a polynomial $\mathsf{poly}_1(\cdot)$ such that $|\hat{\mathsf{ct}}|, |\mathsf{p}_i| \le \mathsf{poly}_1(\lambda, d, N)$. Also, the statement $\Pi_i(\mathsf{com}_i, \hat{\mathsf{ct}}, \mathsf{p}_i)$ is parameterized by $\mathsf{com}_i, \hat{\mathsf{ct}}, \mathsf{p}_i$. We have $\mathsf{com}_i = \mathsf{C.Com}(\mathsf{tfhesk}_i; r_i)$ and since $|\mathsf{tfhesk}_i|, |r_i| \le \mathsf{poly}(\lambda, d, N)$, there exists a polynomial $\mathsf{poly}_2(\cdot)$ such that $|\pi_i| \le \mathsf{poly}(|\mathsf{com}_i| + |\hat{\mathsf{ct}}| + |\mathsf{p}_i|) \le \mathsf{poly}_2(\lambda, d, N)$. Finally, defining $\mathsf{poly} = \mathsf{poly}_1 + \mathsf{poly}_2$, we have

$$|\mathsf{y}_i| = |\mathsf{p}_i| + |\pi_i| \le \mathsf{poly}_1(\lambda, d) + \mathsf{poly}_2(\lambda, d, N) \le \mathsf{poly}(\lambda, d, N).$$

This concludes the proof of the theorem.

E.2 Proof of Theorem 7.9

Fix the security parameter λ, depth bound d, access structure $\mathbb{A}$, message $x \in \{0,1\}^k$ and circuit $C : \{0,1\}^k \to \{0,1\}$ of depth at most d. Let $(\mathsf{pp}, \mathsf{s}_1, \dots, \mathsf{s}_N) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}, x)$, and $\mathsf{y}_i \leftarrow \mathsf{UT.Eval}(\mathsf{pp}, \mathsf{s}_i, C)$ for $i \in [N]$.

By construction, $\mathsf{pp} = \big(\mathsf{tfhepk}, \{\mathsf{ct}_i\}_{i \in [k]}, \{\sigma_{V,i}\}_{i \in [N]}, \{\mathsf{com}_i\}_{i \in [N]}\big)$ and $\mathsf{s}_i = \big(\mathsf{tfhesk}_i, \sigma_{P,i}, r_i\big)$ where $(\mathsf{tfhepk}, \mathsf{tfhesk}_1, \dots, \mathsf{tfhesk}_N) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A})$ and $\mathsf{ct}_i \leftarrow \mathsf{TFHE.Encrypt}(\mathsf{tfhepk}, x_i)$ for $i \in [k]$. The evaluation algorithm $\mathsf{UT.Eval}$ computes $\mathsf{y}_i = (\mathsf{p}_i, \pi_i)$ where $\mathsf{p}_i \leftarrow \mathsf{TFHE.PartDec}(\mathsf{tfhepk},$

$\mathsf{TFHE.Eval}(\mathsf{tfhepk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k))$ and the combining algorithm $\mathsf{UT.Combine}$ parses $\mathsf{y}_i = (\mathsf{p}_i, \pi_i)$ and outputs $\mathsf{TFHE.FinDec}(\mathsf{tfhepk}, \{\mathsf{p}_i\}_{i\in S})$. Therefore, by evaluation correctness of TFHE, we have

$$\Pr[\mathsf{UT.Combine}(\mathsf{pp}, \{\mathsf{y}_i\}_{i\in S}) = C(x)] = \Pr[\mathsf{TFHE.FinDec}(\mathsf{tfhepk}, \{\mathsf{p}_i\}_{i\in S}) = C(x)] = 1 - \mathsf{negl}(\lambda).$$

This completes the proof of the theorem.

E.3 Proof of Theorem 7.10

Fix the security parameter λ, depth bound d, an access structure $\mathbb{A}$, and $x \in \{0,1\}^k$. Let $(\mathsf{pp}, \mathsf{s}_1, \ldots, \mathsf{s}_N) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}, x)$, and $\mathsf{y}_i \leftarrow \mathsf{UT.Eval}(\mathsf{pp}, \mathsf{s}_i, C)$ for $i \in [N]$.

By construction, $\mathsf{pp} = \big(\mathsf{tfhepk}, \{\mathsf{ct}_i\}_{i\in[k]}, \{\sigma_{V,i}\}_{\in[N]}, \{\mathsf{com}_i\}_{i\in[N]}\big)$ and $\mathsf{s}_i = \big(\mathsf{tfhesk}_i, \sigma_{P,i}, r_i\big)$ where $(\sigma_{V,i}, \sigma_{P,i}) \leftarrow \mathsf{PZK.Pre}(1^\lambda)$, $\mathsf{com}_i \leftarrow \mathsf{C.Com}(\mathsf{tfhesk}_i; r_i)$. The evaluation algorithm $\mathsf{UT.Eval}$ computes $\hat{\mathsf{ct}} \leftarrow \mathsf{TFHE.Eval}(\mathsf{tfhepk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$, constructs the statement $\Psi_i(\mathsf{com}_i, \hat{\mathsf{ct}}, \mathsf{p}_i)$, and computes $\mathsf{y}_i = (\mathsf{p}_i, \pi_i)$ where $\pi_i \leftarrow \mathsf{PZK.Prove}(\sigma_{P,i}, \Psi_i, (\mathsf{tfhesk}_i, r_i))$. By definition of the statement $\Psi_i(\mathsf{tfhesk}_i, r_i)$, the pair (tfhesk_i, r_i) is a valid witness and since $\mathsf{UT.Eval}$ simply returns the output of $\mathsf{PZK.Verify}(\sigma_{P,i}, \Psi_i, \pi_i)$, we have

$$\Pr[\mathsf{UT.Verify}(\mathsf{pp}, \mathsf{y}_i, C) = 1] = \Pr[\mathsf{PZK.Verify}(\sigma_{P,i}, \Psi_i, \pi_i) = 1] = 1$$

by completeness of PZK. This completes the proof of the theorem.

E.4 Proof of Theorem 7.11

Our proof proceeds via a sequence of hybrid experiments between an adversary $\mathcal{A}$ and a challenger.

- H_0: This is the UT real security experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{UT},\mathsf{Real}}(1^\lambda, 1^d)$ from Definition 7.5. On input an access structure $\mathbb{A} \in \mathbb{S}$, and a message $x \in \{0,1\}^k$ from $\mathcal{A}$, the challenger runs $(\mathsf{pp}, \mathsf{s}_1, \ldots, \mathsf{s}_N) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}, x)$ and provides pp to $\mathcal{A}$. It receives a maximal invalid party set $S \subseteq \{P_1, \ldots, P_N\}$ and provides $\{\mathsf{s}_i\}_{i\in S^*}$ to $\mathcal{A}$. For each evaluation query (S, C) that $\mathcal{A}$ makes, the challenger provides $\{\mathsf{y}_i \leftarrow \mathsf{UT.Eval}(\mathsf{pp}, \mathsf{s}_i, C)\}_{i\in S}$ to $\mathcal{A}$.

- H_1: Same as H_0, except that the challenger simulates the PZK proof in $\mathsf{UT.Eval}$. Specifically, on each evaluation query (S, C) that $\mathcal{A}$ makes, the challenger computes $\hat{\mathsf{ct}} \leftarrow \mathsf{TFHE.Eval}(\mathsf{tfhepk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ and partial decryption $\mathsf{p}_i \leftarrow \mathsf{TFHE.PartDec}(\mathsf{tfhepk}, \hat{\mathsf{ct}}, \mathsf{tfhesk}_i)$ for $i \in S$ as before, but it now runs the PZK simulator $\pi_i^* \leftarrow \mathsf{PZK}.\mathcal{S}(\Psi_i(\mathsf{com}_i, \hat{\mathsf{ct}}, \mathsf{p}_i))$ and sets $\mathsf{y}_i = (\mathsf{p}_i, \pi_i^*)$ for $i \in S$.

  By the zero knowledge property of PZK, the hybrid experiments H_0 and H_1 are computationally indistinguishable.

- H_2: Same as H_1, except that the challenger generates pp with commitments to the zero string. Specifically, on input an access structure $\mathbb{A} \in \mathbb{S}$, and a message $x \in \{0,1\}^k$ from $\mathcal{A}$, the challenger runs setup $\mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}, x)$ by computing the TFHE keys $(\mathsf{tfhepk}, \mathsf{tfhesk}_1, \ldots, \mathsf{tfhesk}_N) \leftarrow \mathsf{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A})$ ciphertexts $\mathsf{ct}_i \leftarrow \mathsf{TFHE.Encrypt}(\mathsf{tfhepk}, \mathsf{tfhepk}, x_i)$ for $i \in [k]$, $(\sigma_{V,i}, \sigma_{P,i}) \leftarrow \mathsf{PZK.Pre}(1^\lambda)$ and $r_i \leftarrow \{0,1\}^\lambda$ for $i = 1, \ldots, N$ as before. But now, it defines $\mathsf{com}_i^* \leftarrow \mathsf{C.Com}(0^{|\mathsf{tfhesk}_i|}; r_i)$ for $i = 1, \ldots N$ and sets $\mathsf{pp} = \big(\mathsf{tfhepk}, \{\mathsf{ct}_i\}_{i\in[k]}, {\sigma_{V,i}}_{i\in[N]}, \{\mathsf{com}_i^*\}_{i\in[N]}\big)$.

  By the computational hiding property of C, the hybrid experiments H_1 and H_2 are computationally indistinguishable.

- H_3: Same as H_2, except that the challenger simulates the TFHE setup and partial decryptions. Specifically, on input an access structure $\mathbb{A} \in \mathbb{S}$, and a message $x \in \{0,1\}^k$ from $\mathcal{A}$, the challenger runs setup $\mathsf{UT.Setup}(1^\lambda 1^d, \mathbb{A}, x)$ by first invoking the simulator $(\mathsf{tfhepk}, \mathsf{tfhesk}'_1, \dots, \mathsf{tfhesk}'_N, \mathsf{st}) \leftarrow \mathsf{TFHE}.\mathcal{S}_1(1^\lambda, 1^d, \mathbb{A})$ instead of running $\mathsf{TFHE.Setup}$. Then, it carries out the rest of $\mathsf{UT.Setup}$ and the experiment as before in H_2. Then, on each evaluation query (S, C) that $\mathcal{A}$ makes, instead of computing $\hat{\mathsf{ct}} \leftarrow \mathsf{TFHE.Eval}(\mathsf{tfhepk}, C, \mathsf{ct}_1, \dots, \mathsf{ct}_k)$ and partial decryptions $\mathsf{p}_i \leftarrow \mathsf{TFHE.PartDec}(\mathsf{tfhepk}, \hat{\mathsf{ct}}, \mathsf{tfhesk}_i)$ for $i \in S$, the challenger computes $\{\mathsf{p}_i^*\}_{i \in S} \leftarrow \mathsf{TFHE}.\mathcal{S}_2(C, \{\mathsf{ct}_1, \dots, \mathsf{ct}_k\}, C(\mu_1, \dots, \mu_k), S, \mathsf{st})$. It simulates PZK proofs as before.

  By the simulation security of TFHE, the hybrid experiments H_2 and H_3 are statistically indistinguishable.

- H_4: Same as H_3, except that the challenger encrypts the zero string $\mathbf{0}$ instead of x during setup. Specifically, on input an access structure $\mathbb{A} \in \mathbb{S}$, and a message $x \in \{0,1\}^k$ from $\mathcal{A}$, the challenger ignores x and runs setup as in H_3 but with the zero string $\mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}, \mathbf{0})$. The challenger carries out the rest of the experiments as before.

  By semantic security of TFHE, the hybrid experiments H_3 and H_4 are computationally indistinguishable. Also, note that in H_4, the challenger simulates the setup algorithm $\mathsf{UT.Setup}$ as well as the partial evaluation algorithm $\mathsf{UT.Eval}$ without requiring access to the secret x. Therefore, H_4 corresponds to the UT ideal security experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{UT},\mathsf{Ideal}}(1^\lambda, 1^d)$.

Combining the indistinguishability of the consecutive hybrid experiments, we conclude that the UT scheme in Construction 7.7 satisfies security as in Definition 7.5.

E.5 Proof of Theorem 7.12

In the robustness security experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{Robust}}(1^\lambda, 1^d)$, the adversary $\mathcal{A}$ first outputs a message $x \in \{0,1\}^k$ and $\mathbb{A} \in \mathbb{S}$. The challenger then computes $(\mathsf{pp}, \mathsf{s}_1, \dots, \mathsf{s}_N) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}, x)$ and provides $(\mathsf{pp}, \mathsf{s}_1, \dots, \mathsf{s}_N)$. Now, the experiment outputs 1 only if $\mathcal{A}$ outputs a fake partial evaluation y_i^* such that the following conditions hold:

- $\mathsf{y}_i^* \neq \mathsf{UT.Eval}(\mathsf{pp}, \mathsf{s}_i, C)$.
- $\mathsf{UT.Verify}(\mathsf{pp}, \mathsf{y}_i^*, C) = 1$.

Recall that the verification algorithm $\mathsf{UT.Verify}$ on input (y_i, C) first evaluates the ciphertext $\hat{\mathsf{ct}} \leftarrow \mathsf{TFHE.Eval}(\mathsf{pp}, C, \mathsf{ct}_1, \dots, \mathsf{ct}_k)$, and constructs the statement $\Psi_i(\mathsf{com}_i, \hat{\mathsf{ct}}, \mathsf{p}_i)$:

$$\exists\ (\mathsf{tfhesk}_i, r_i) : \mathsf{com}_i = \mathsf{C.Com}(\mathsf{tfhesk}_i; r_i) \wedge \mathsf{p}_i = \mathsf{TFHE.PartDec}(\mathsf{pp}, \hat{\mathsf{ct}}, \mathsf{tfhesk}_i).$$

Then, it parses $\mathsf{y}_i = (\mathsf{p}_i, \pi_i)$ and returns the result of $\mathsf{PZK.Verify}\big(\sigma_{V,i}, \Psi_i(\mathsf{com}_i, \hat{\mathsf{ct}}, \mathsf{p}_i), \pi_i\big)$. Therefore, since $\mathsf{UT.Verify}(\mathsf{y}_i^*, C) = 1$, we have that $\mathsf{PZK.Verify}\big(\sigma_{V,i}, \Psi_i(\mathsf{com}_i, \hat{\mathsf{ct}}, \mathsf{p}_i^*), \pi_i^*\big) = 1$ for $\mathsf{y}_i^* = (\mathsf{p}_i^*, \pi_i^*)$. This means that $\mathsf{y}_i^* = (\mathsf{p}_i^*, \pi_i^*)$ satisfies one of the following:

- **Case 1**: The statement $\Psi_i(\mathsf{com}_i, \hat{\mathsf{ct}}, \mathsf{p}_i^*)$ is a true statement (has a valid witness).
- **Case 2**: The statement $\Psi_i(\mathsf{com}_i, \hat{\mathsf{ct}}, \mathsf{p}_i^*)$ is a false statement, but $\mathsf{PZK.Verify}(\sigma_{V,i}, \Psi_i(\mathsf{com}_i, \hat{\mathsf{ct}}, \mathsf{p}_i^*), \pi^*) = 1$.

It suffices to show that the probability of each of these cases occurring is negligible.

First assume that the statement $\Psi_i(\mathsf{com}_i, \hat{\mathsf{ct}}, \mathsf{p}_i^*)$ is a true statement. By the assumption that $\mathsf{y}_i^* \neq \mathsf{UT.Eval}(\mathsf{pp}, \mathsf{s}_i, C)$, we have $\mathsf{p}_i^* \neq \mathsf{TFHE.PartDec}(\mathsf{pp}, \hat{\mathsf{ct}}, \mathsf{tfhesk}_i)$. $\Psi_i(\mathsf{com}_i, \hat{\mathsf{ct}}, \mathsf{p}_i^*)$ is a true statement only if there exists some randomness r_i^* such that $\mathsf{com}_i = \mathsf{C.Com}(\mathsf{tfhesk}_i^*, r_i^*)$. However, this is a contradiction since the commitment scheme C is perfectly binding.

Now, assume that $\Psi_i(\mathsf{com}_i, \hat{\mathsf{ct}}, \mathsf{p}_i^*)$ is a false statement, but $\mathsf{PZK.Verify}(\sigma_{V,i}, \Psi_i(\mathsf{com}_i, \hat{\mathsf{ct}}, \mathsf{p}_i^*), \pi^*) = 1$. However, since the proof system PZK is a sound, the probability that there exists a proof π_i^* such that $\mathsf{PZK.Verify}(\sigma_{V,i}, \Psi_i(\mathsf{com}_i, \hat{\mathsf{ct}}, \mathsf{p}_i^*), \pi^*) = 1$ is negligible. We conclude that the UT scheme in Construction 7.7 satisfies robustness as in Definition 7.6.

F Proofs in Section 8.2.2

The proofs of Theorems 8.17, 8.18, 8.19, and 8.22 follow from the correctness properties of the underlying universal thresholdizer UT. We provide the formal proofs of the security properties of Construction 8.16.

F.1 Proof of Theorem 8.20

Our proof proceeds via a sequence of hybrid experiments between an adversary $\mathcal{A}$ and a challenger.

- H_0: This is the real TS unforgeability experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{TS},\mathsf{uf}(1^\lambda)}$ from Definition 8.13. On input an access structure $\mathbb{A} \in \mathbb{S}$ from $\mathcal{A}$, the challenger runs $(\mathsf{pp}, \mathsf{vk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{TS.Setup}(1^\lambda, \mathbb{A})$ and provides $(\mathsf{pp}, \mathsf{vk})$ to $\mathcal{A}$. It then receives a set $S \subseteq \{P_1, \ldots, P_N\}$ such that S is a maximal invalid set and provides $\{\mathsf{sk}_i\}_{i \in S}$ to $\mathcal{A}$. For each signing query (m, i) that $\mathcal{A}$ makes, the challenger computes $\sigma_i \leftarrow \mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m)$ and provides σ_i to $\mathcal{A}$. At the end of the experiment, the adversary outputs a forgery (m^*, σ^*) and wins if $\mathsf{TS.Verify}(\mathsf{vk}, m^*, \sigma^*) = 1$ and m^* was not previously queried as a signing query.

- H_1: This experiment is the same as H_0, except that now, the challenger invokes the UT simulator $\mathsf{UT}.\mathcal{S}$ from Definition 7.5 for setup and partial evaluations. Specifically, to run setup $\mathsf{TFHE.Setup}(1^\lambda, acc)$, the challenger sets $\mathsf{pp} = \mathsf{utpp}$ and $\mathsf{sk}_i = \mathsf{uts}_i$ where $(\mathsf{utpp}, \mathsf{uts}_1, \ldots, \mathsf{uts}_N, \mathsf{st}) \leftarrow \mathsf{UT}.\mathcal{S}_1(1^\lambda, 1^d, \mathbb{A})$. Then, it generates the keys $(\mathsf{ssk}, \mathsf{svk}) \leftarrow \mathsf{S.KeyGen}(1^\lambda)$, sets $\mathsf{vk} = \mathsf{svk}$ and keeps ssk. Then, for each signing query (m, i) that $\mathcal{A}$ makes, the challenger computes $\sigma \leftarrow \mathsf{S.Sign}(\mathsf{ssk}, m)$ and invokes the simulator $\sigma_i \leftarrow \mathsf{UT}.\mathcal{S}_2(\mathsf{utpp}, C_m, \sigma, S, \cup\{P_i\}, \mathsf{st})$.[3] It provides σ_i to $\mathcal{A}$. The rest of the experiment remains the same.

We now argue that the hybrid experiments $\mathsf{H}_0, \mathsf{H}_1$ are computationally indistinguishable. It is easy to see that in both hybrid experiments above, the output of the experiment is efficiently computable by the challenger. Therefore, the following two lemmas are sufficient to prove the theorem.

Below, for an adversary $\mathcal{A}$, we write $\mathsf{H}_i(\mathcal{A})$ to denote the output of H_i.

Lemma F.1. *If* UT *is a secure universal thresholdizer, then for all efficient adversaries* $\mathcal{A}$, $|\Pr[\mathsf{H}_0(\mathcal{A}) = 1] - \Pr[\mathsf{H}_1(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.

Proof. The only difference between the experiments H_0 and H_1 is the way the challenger runs UT setup and the way it answers the signing queries (m, i). In H_0, the challenger sets $\mathsf{pp} = \mathsf{utpp}$ and $\mathsf{sk}_i =$

[3]Recall that the circuit C_m is defined as the signing circuit $C_m(\mathsf{ssk}) = \mathsf{S.Sign}(\mathsf{ssk}, m)$ (See Construction 8.16).

uts_i where $(\mathsf{utpp}, \mathsf{uts}_1, \ldots, \mathsf{uts}_N) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}, x)$. It computes $\sigma_i \leftarrow \mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m)$ by invoking $\sigma_i \leftarrow \mathsf{UT.Eval}(\mathsf{uts}_i, C_m)$. Therefore, the view of $\mathcal{A}$ in H_0 is precisely the view in the experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{UT},\mathsf{Real}}(1^\lambda, 1^d)$ where the message $m = \mathsf{ssk}$. In H_1, the challenger sets $\mathsf{pp} = \mathsf{utpp}$ and $\mathsf{sk}_i = \mathsf{uts}_i$ where $(\mathsf{utpp}, \mathsf{uts}_1, \ldots, \mathsf{uts}_N, \mathsf{st}) \leftarrow \mathsf{UT}.\mathcal{S}_1(1^\lambda, 1^d, \mathbb{A})$. For the signing queries (m, i), it invokes the simulator $\mathsf{UT}.\mathcal{S}_2(\mathsf{utpp}, C_m, \sigma, S \cup \{P_i\}, \mathsf{st})$ where $\sigma \leftarrow \mathsf{S.Sign}(\mathsf{ssk}, m)$. This is precisely the view in the experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{UT},\mathsf{Ideal}}(1^\lambda, 1^d)$ where the message $m = \mathsf{ssk}$. Therefore, we have

$$\left|\Pr[\mathsf{H}_0(\mathcal{A}) = 1] - \Pr[\mathsf{H}_1(\mathcal{A}) = 1]\right| = \left|\Pr[\mathsf{Expt}_{\mathcal{A},\mathsf{UT},\mathsf{Real}}(1^\lambda, 1^d) = 1] - \Pr[\mathsf{Expt}_{\mathcal{A},\mathsf{UT},\mathsf{Ideal}}(1^\lambda, 1^d) = 1]\right|$$

which is negligible. □

Lemma F.2. *If* S *is an unforgeable signture scheme, then for all efficient adversaries* $\mathcal{A}$*,* $\Pr[\mathsf{H}_1(\mathcal{A}) = 1] = \mathsf{negl}(\lambda)$*.*

Proof. Let $\mathcal{A}$ be any adversary in H_1. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that wins the unforgeability experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{S},\mathsf{uf}}(1^\lambda)$ (Definition A.5). Algorithm $\mathcal{B}$ works as follows:

1. At the beginning of the game, $\mathcal{B}$ receives svk from the unforgeability challenger. It instantiates UT setup $(\mathsf{utpp}, \mathsf{uts}_1, \ldots, \mathsf{uts}_N, \mathsf{st}) \leftarrow \mathsf{UT}.\mathcal{S}_1(1^\lambda, 1^d, \mathbb{A})$. It provides $\mathsf{pp} = \mathsf{utpp}$ and $\mathsf{vk} = \mathsf{svk}$ to $\mathcal{A}$. Note that by the definition of the unforgeability challenger, the view of $\mathcal{A}$ until this point is exactly the view in H_1.

2. When $\mathcal{A}$ outputs a maximal invalid share set S, $\mathcal{B}$ provides the UT shares $\{\mathsf{uts}_i\}_{i \in S}$. Again, by definition, this is a perfect simulation of H_1.

3. For each query (m, i) that $\mathcal{A}$ makes, $\mathcal{B}$ submits m as its own signing query to the unforgeability challenger and receives $\sigma = \mathsf{S.Sign}(\mathsf{ssk}, m)$. It then simulates partial signatures by running UT simulator $\sigma_i \leftarrow \mathsf{UT}.\mathcal{S}_2(\mathsf{utpp}, C_m, \sigma, S \cup \{P_i\}, \mathsf{st})$. By definition, we have $C_m(\mathsf{ssk}) = \mathsf{S.Sign}(\mathsf{ssk}, m)$. Therefore, each partial signatures σ_i is simulated exactly as in H_1.

4. At the end of the experiment, $\mathcal{A}$ outputs a forgery (m^*, σ^*). Recall that (m^*, σ^*) is a valid forgery if

   - $\mathsf{TS.Verify}(\mathsf{vk}, m^*, \sigma^*) = 1$.
   - m^* was not previously queried as a signing query.

   By definition, $\mathsf{TS.Verify}(\mathsf{vk}, m^*, \sigma^*) = \mathsf{S.Verify}(\mathsf{vk}, m^*, \sigma^*)$. Furthermore, $\mathcal{B}$ invokes the signing query to the unforgeability challenger only when $\mathcal{A}$ makes its partial signing queries. Therefore, the message, signature pair (m^*, σ^*) is a valid forgery for S. The algorithm $\mathcal{B}$ outputs (m^*, σ^*) as its valid forgery.

Finally, by the correctness of $\mathcal{B}$, we have

$$\Pr[\mathsf{H}_1(\mathcal{A}) = 1] = \Pr[\mathsf{Expt}_{\mathcal{A},\mathsf{S},\mathsf{uf}}(1^\lambda) = 1]$$

which is negligible. This concludes the proof of the lemma. □

This concludes the proof of the theorem.

F.2 Proof of Theorem 8.21

In the robustness security experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{TPKE},\mathsf{rb}}(1^\lambda)$, $\mathcal{A}$ first outputs an access structure $\mathbb{A} \in \mathbb{S}$. The challenger then computes $(\mathsf{pp}, \mathsf{vk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{TS.Setup}(1^\lambda, \mathbb{A})$ and provides all keys $(\mathsf{pp}, \mathsf{vk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N)$ to $\mathcal{A}$. Now, the experiment returns 1 only if $\mathcal{A}$ outputs a partial signature forgery (m^*, σ_i^*, i) such that the following conditions are satisfied:

- $\mathsf{TS.PartSignVerify}(\mathsf{pp}, m^*, \sigma_i^*) = 1$.
- $\sigma_i \neq \mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m^*)$.

Recall that the partial signing algorithm, on input $(\mathsf{pp}, \mathsf{sk}_i = \mathsf{uts}_i, m)$ outputs $\sigma_i = \mathsf{UT.Eval}(\mathsf{utpp}, \mathsf{uts}_i, C_m)$. Therefore, if (m^*, σ_i^*) is a valid forgery, then $\sigma_i^* \neq \mathsf{UT.Eval}(\mathsf{utpp}, \mathsf{uts}_i, C_{m^*})$. However, by definition, the partial signature verification algorithm $\mathsf{TS.PartSignVerify}$ outputs 1 if the UT verification algorithm accepts $\mathsf{UT.Verify}(\mathsf{utpp}, \sigma_i^*, C_{m^*})$. Therefore, if $\mathsf{TS.PartSignVerify}(\mathsf{pp}, m^*, \sigma_i^*) = 1$, then (m^*, σ_i^*, i) is also a valid forgery for UT. This shows that

$$\Pr[\mathsf{Expt}_{\mathcal{A},\mathsf{TS},\mathsf{rb}}(1^\lambda) = 1] = \Pr[\mathsf{Expt}_{\mathcal{A},\mathsf{UT},\mathsf{rb}}(1^\lambda, 1^d) = 1]$$

which is negligible. This concludes the proof of the theorem.

G Proofs in Section 8.3.2

The proofs of Theorems 8.30, 8.31, and 8.32 follow from the correctness properties of the underlying universal thresholdizer UT. For robustness, the proof Theorem 8.34 follows from the same argument used in the proof of Theorem 8.21. We provide the formal proof of Theorem 8.33.

G.1 Proof of Theorem 8.33

Our proof proceeds via a sequence of hybrid experiments between an adversary $\mathcal{A}$ and a challenger.

- H_0: This is the real TPKE CCA security experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{PKE},0}(1^\lambda)$ from Definition A.8. On input an access structure $\mathbb{A} \in \mathbb{S}$ from $\mathcal{A}$, the challenger runs $(\mathsf{pp}, \mathsf{ek}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N \leftarrow \mathsf{TPKE.Setup}(1^\lambda, \mathbb{A})$ and provides $(\mathsf{pp}, \mathsf{ek})$ to $\mathcal{A}$. It then receives a set $S \subseteq \{P_1, \ldots, P_N\}$ such that S is a maximal invalid set and provides $\{\mathsf{sk}_i\}_{i \in S}$ to $\mathcal{A}$. For each decryption queries (ct, i), the challenger computes $\mathsf{m}_i \leftarrow \mathsf{TPKE.PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct})$ and provides s_i to $\mathcal{A}$. In the process, it receives a pair of challenge messages (m_0^*, m_1^*) and provides $\mathsf{ct}_0^* \leftarrow \mathsf{TPKE.Enc}(\mathsf{ek}, m_0^*)$ to $\mathcal{A}$.
- H_1: This experiment is the same as H_0 except that now, the challenger invokes the UT simulator $\mathsf{UT}.\mathcal{S}$ from Definition 7.5 for setup and partial evaluations. Specifically, to run setup $\mathsf{TFHE.Setup}(1^\lambda, \mathbb{A})$, the challenger sets $\mathsf{pp} = \mathsf{utpp}$ and $\mathsf{sk}_i = \mathsf{uts}_i$ where $(\mathsf{utpp}, \mathsf{uts}_1, \ldots, \mathsf{uts}_N, \mathsf{st}) \leftarrow \mathsf{UT}.\mathcal{S}_1(1^\lambda, 1^d, \mathbb{A})$. Then, it generates $(\mathsf{pkesk}, \mathsf{pkepk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, sets $\mathsf{ek} = \mathsf{pkepk}$ and keeps pkesk. Then, for each decryption query (ct, i), the challenger computes $m \leftarrow \mathsf{PKE.Decrypt}(\mathsf{pkesk}, \mathsf{ct})$ and invokes the simulator $\mathsf{m}_i \leftarrow \mathsf{UT}.\mathcal{S}_2(\mathsf{utpp}, C_{\mathsf{ct}}, m, S \cup \{P_i\}, \mathsf{st})$.[4] It provides m_i to $\mathcal{A}$. The rest of the experiment remains the same.

[4]Recall that the circuit C_{ct} is defined as the decryption circuit $C_{\mathsf{ct}}(\mathsf{pkesk}) = \mathsf{PKE.Decrypt}(\mathsf{pkesk}, \mathsf{ct})$ (see Construction 8.29)

- H_2: This experiment is the same as H_1 except that now, the challenger generates the challenge ciphertext as the encryption of m_1^* instead of m_0^*. More precisely, when $\mathcal{A}$ submits a pair of challenge message (m_0^*, m_1^*), the challenger computes $\mathsf{ct}_1^* \leftarrow \mathsf{TPKE.Enc}(\mathsf{ek}, m_1^*)$ and provides ct_1^* to $\mathcal{A}$.

- H_3: This experiment is the same as H_2 except that now, the challenger runs UT setup as in the real scheme with respect to pkesk and answers the decryption queries exactly as in the real scheme using $\mathsf{TPKE.PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct})$. This corresponds to the real TPKE CCA security experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{PKE},1}(1^\lambda)$ from Definition A.8.

We now argue that the consecutive hybrid experiments above are indistinguishable. For an adversary $\mathcal{A}$, we write $\mathsf{H}_1(\mathcal{A})$ to denote the output of H_i.

Lemma G.1. *If* UT *is a secure universal thresholdizer, then for all efficient adversaries* $\mathcal{A}$, $\big|\Pr[\mathsf{H}_0(\mathcal{A}) = 1] - \Pr[\mathsf{H}_1(\mathcal{A}) = 1]\big| = \mathsf{negl}(\lambda)$.

Proof. The only difference between the experiments H_0 and H_1 is the way the challenger runs UT setup and the way it answers the decryption queries (ct, i). In H_0, the challenger sets $\mathsf{pp} = \mathsf{utpp}$ and $\mathsf{sk}_i = \mathsf{uts}_i$ where $(\mathsf{utpp}, \mathsf{uts}_1, \ldots, \mathsf{uts}_N) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}, \mathsf{pkesk})$. It computes $\mathsf{m}_i \leftarrow \mathsf{TPKE.PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct})$ by invoking $\mathsf{m}_i \leftarrow \mathsf{UT.Eval}(\mathsf{uts}_i, C_{\mathsf{ct}})$. Therefore, the view of $\mathcal{A}$ in H_0 is precisely the view in the experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{UT},\mathsf{Real}}(1^\lambda, 1^d)$ where the message $m = \mathsf{pkesk}$. In H_1, the challenger sets $\mathsf{pp} = \mathsf{utpp}$ and $\mathsf{sk}_i = \mathsf{uts}_i$ where $(\mathsf{utpp}, \mathsf{uts}_1, \ldots, \mathsf{uts}_N, \mathsf{st}) \leftarrow \mathsf{UT}.\mathcal{S}_1(1^\lambda, 1^d, \mathbb{A})$. For the decryption queries (ct, i), it invokes the simulator $\mathsf{UT}.\mathcal{S}_2(\mathsf{utpp}, C_{\mathsf{ct}}, m, S \cup \{P_i\}, \mathsf{st})$ for $m \leftarrow \mathsf{PKE.Decrypt}(\mathsf{pkesk}, \mathsf{ct})$. This is precisely the view in the experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{UT},\mathsf{Ideal}}(1^\lambda, 1^d)$ where the message $m = \mathsf{pkesk}$. Therefore, we have

$$\big|\Pr[\mathsf{H}_0(\mathcal{A}) = 1] - \Pr[\mathsf{H}_1(\mathcal{A}) = 1]\big| = \big|\Pr[\mathsf{Expt}_{\mathcal{A},\mathsf{UT},\mathsf{Real}}(1^\lambda, 1^d)] - \Pr[\mathsf{Expt}_{\mathcal{A},\mathsf{UT},\mathsf{Ideal}}(1^\lambda, 1^d) = 1]\big|$$

which is negligible. □

Lemma G.2. *If* PKE *is a secure public key encryption scheme, then for all efficient adversaries* $\mathcal{A}$, $\big|\Pr[\mathsf{H}_1(\mathcal{A}) = 1] - \Pr[\mathsf{H}_2(\mathcal{A}) = 1]\big| = \mathsf{negl}(\lambda)$.

Proof. Let $\mathcal{A}$ be any adversary that interacts with experiments H_1 and H_2. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that distinguishes experiments $\mathsf{Expt}_{\mathcal{A},\mathsf{PKE},0}(1^\lambda)$ and $\mathsf{Expt}_{\mathcal{A},\mathsf{PKE},1}(1^\lambda)$ (Definition A.8). Algorithm $\mathcal{B}$ works as follows:

1. At the beginning of the game, $\mathcal{B}$ receives pkepk from the PKE challenger. It instantiates UT setup $(\mathsf{utpp}, \mathsf{uts}_1, \ldots, \mathsf{uts}_N, \mathsf{st}) \leftarrow \mathsf{UT}.\mathcal{S}_1(1^\lambda, 1^d, \mathbb{A})$. It provides $\mathsf{pp} = \mathsf{utpp}$ and $\mathsf{ek} = \mathsf{pkepk}$ to $\mathcal{A}$. Note that by the definition of the PKE challenger, the view of $\mathcal{A}$ at this point is exactly the view in H_1 and H_2.

2. When $\mathcal{A}$ outputs a maximal invalid share set S, $\mathcal{B}$ provides the UT shares $\{\mathsf{uts}_i\}_{i \in S}$. Again, by definition, this is a perfect simulation of H_1 and H_2.

3. For each query (ct, i) that $\mathcal{A}$ makes, $\mathcal{B}$ submits ct as its own decryption query to the PKE challenger and receives the decryption $m = \mathsf{PKE.Decrypt}(\mathsf{pkesk}, \mathsf{ct})$. It then simulates the partial decryptions by running the UT simulator $\mathsf{m}_i \leftarrow \mathsf{UT}.\mathcal{S}_2(\mathsf{utpp}, C_{\mathsf{ct}}, m, S \cup \{P_i\}, \mathsf{st})$. By definition, we have $C_m(\mathsf{pkesk}) = \mathsf{PKE.Decrypt}(\mathsf{pkesk}, \mathsf{ct})$. Therefore, each partial decryptions m_i is simulated exactly as in H_1.

4. During the query phase, $\mathcal{A}$ outputs a pair of challenge messages (m_0^*, m_1^*). Then, $\mathcal{B}$ sends (m_0^*, m_1^*) as its own challenge pair to the PKE challenger and receives ct_b^*. It provides ct_b^* to $\mathcal{A}$.

5. At the end of the experiment, $\mathcal{A}$ outputs its guess b'. Now, if $\mathcal{B}$ is interacting in $\mathsf{Expt}_{\mathcal{A},\mathsf{PKE},0}(1^\lambda)$, then by definition of PKE challenger, we have $\mathsf{ct}_b^* = \mathsf{PKE.Encrypt}(\mathsf{pkepk}, m_0^*)$. Therefore, in this case, $\mathcal{B}$ provides a perfect simulation of H_1 to $\mathcal{A}$. Correspondingly, if $\mathcal{B}$ is interacting in $\mathsf{Expt}_{\mathcal{A},\mathsf{PKE},1}(1^\lambda)$, then $\mathsf{ct}_b^* = \mathsf{PKE.Encrypt}(\mathsf{pkepk}, m_1^*)$, and $\mathcal{B}$ provides a perfect simulation of H_2. Therefore, with the distinguishing advantage of $\mathcal{A}$, the algorithm $\mathcal{B}$ distinguishes $\mathsf{Expt}_{\mathcal{A},\mathsf{PKE},0}(1^\lambda)$ and $\mathsf{Expt}_{\mathcal{A},\mathsf{PKE},1}(1^\lambda)$.

   What remains to check is whether $\mathcal{B}$ is an admissible adversary. By definition, admissible adversary $\mathcal{A}$ does not submit ct^* as part of its partial decryption queries. Since $\mathcal{B}$ submits a ciphertext ct to its decryption oracle only when $\mathcal{A}$ submits its partial decryption query with ct, $\mathcal{B}$ is admissible. This concludes the proof of the lemma.

□

Lemma G.3. *It* UT *is a secure universal thresholdizer, then for all efficient adversaries* $\mathcal{A}$, $\big|\Pr[\mathsf{H}_0(\mathcal{A}) = 1] - \Pr[\mathsf{H}_1(\mathcal{A}) = 1]\big| = \mathsf{negl}(\lambda)$.

Proof. Follows from the proof of Lemma G.1. □

Combining Lemmas G.1, G.2, and G.3, the theorem follows.