Threshold Cryptosystems From Threshold Fully Homomorphic Encryption*

Dan Boneh[1], Rosario Gennaro[2], Steven Goldfeder[3], Aayush Jain[4], Sam Kim[1]
Peter M. R. Rasmussen[4], and Amit Sahai[4]

[1] Stanford University
[2] City College of New York
[3] Princeton University
[4] UCLA and Center for Encrypted Functionalities

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 (ThFHE) 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 ThFHE. 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 [26, 28, 25] 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.

*The full version of this paper is available at [12].

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 [28, 25, 31, 50], Schnorr signatures [52], (EC)DSA signatures [30, 29], BLS signatures [14, 9], Cramer-Shoup encryption [21], Regev encryption [7], and many more [51, 27, 10]. 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* (ThFHE) 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 a 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, which prevents correct reconstruction of the message by the combiner.[1] We handle this difficulty in two different

[1] In the weaker model where the set of t servers that will respond to the combiner is known ahead of time, there exist simple methods to preserve correctness. In this work,

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, to achieve better efficiency, 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 ThFHE (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 ThFHE. 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 ThFHE 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 8.1). Since a universal thresholdizer can be proven secure based on LWE, and because there are known (non-threshold) signature schemes based on LWE [32, 15, 41], 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 [47, 32, 45, 1, 42] 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 . 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 [16, 17] that can support threshold access structures . We provide the details of these constructions in [12].

Decentralized threshold FHE. Our basic ThFHE 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* (dThFHE). In a dThFHE scheme, each decryption server generates its own public/secret key pair. At

we work in the traditional model of threshold cryptography where the combiner *does not* know the set of t servers ahead of time.

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 dThFHE in Section 6. However, while it achieves the added flexibility of decentralized key generation, our dThFHE scheme does not satisfy as strong compactness properties as our ThFHE and universal thresholdizer. We leave improving the compactness of our dThFHE as an important open problem following from our work.

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 [7] gave a threshold version of Regev's CPA-secure encryption scheme [48], and Myers et al. [44] applied the technique to fully homomorphic encryption. Xie et al. [54] gave a threshold CCA secure PKE scheme from lossy trapdoor functions, which can be instantiated from LWE [47]. 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. [22] 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. [8] as well as the commitment and zero-knowledge protocols of Baum et al. [5] 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.

MPC systems. Recent advances in low-round MPC from LWE [3, 23, 36, 43, 19, 46] give t-out-of-N threshold cryptosystems for a restricted set of t. 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 thresholdizes an FHE key at setup and distributes the keys to each servers, then the single round distributed decryption protocol can be used to construct a threshold FHE scheme. A crucial limitation of these single-round distributed decryption protocols, however, is that in order for decryption to be successful, *all* parties must participate in the decryption protocol. Even if only a single party is corrupt or is absent from the protocol, the rest of the parties cannot recover the encrypted message, which results in only an N-out-of-N threshold FHE scheme.

We note that a t-out-of-N threshold FHE for small thresholds can be constructed from a N-out-of-N scheme generically. For each $\binom{N}{t}$ sets of size t, a trusted authority can thresholdize an FHE key using a t-out-of-t scheme and provide the corresponding key shares to each parties that are contained in the set. To decrypt a ciphertext, each party can provide the partial decryptions using each of the key shares it has, which a combiner can re-combine only when at least t users provide a correct partial decryption. We note, however, that such extension clearly does not work for larger values of t.

Fully-thresholdized systems. Very few existing lattice cryptosystems overcome all the limitations described above. One exception is threshold distributed PRFs [13] built from key-homomorphic PRFs [13, 4, 20].

Decentralized key generation. Generally, in threshold cryptosystems, the key shares for each parties must be generated either by a trusted authority or via a highly interactive multiparty computation. In this work, we study threshold cryptosystems with *non-interactive* decentralized key generation. In the setting of functional encryption, a non-interative decentralized key generation for threshold access structures was considered in [18].

2 Overview of the Main Construction

In this section, we provide an overview of the main threshold fully homomorphic encryption (ThFHE) construction and its applications to thresholdizing cryptographic systems through a universal thresholdizer. We provide the full ThFHE 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 [33]. 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.

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 $\mathsf{sk}_1, \dots, \mathsf{sk}_N$, 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 $S \subseteq \{1, \dots, N\}$ 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.

Problem with Bit Decomposition. A typical solution to the problem of noise blow-up 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. Using the binary representation of the Lagrange coefficients, the combiner can recombine the shares without blowing up the noise. However, such bit decomposition introduces a problem in the security proof. In order to prove that the partial decryption shares $\{\mathsf{p}_i^{(j)}\}_j$ do not leak information about the underlying key sk_i, it should be possible to statistically simulate them given only $t-1$ secret keys of the other parties and the plaintext m. Interpreting the $t-1$ partial decryptions of the other parties and the message m as t shares of decryptor i's partial decryption, we can simulate the inner product $\lambda_i^{(S)} \cdot \langle \mathsf{ct}_m, \mathsf{sk}_i \rangle + \mathsf{noise}$, but not the individually scaled inner products $\{\mathsf{p}_i^{(j)}\}_j$.

The difficulty with the simulation is, in fact, warranted as there are direct attacks that can recover information about the key share sk_i from the scaled partial decryptions. One such attack can be described as follows. Consider an adversary that has access to a number of partial decryptions produced by decryptor i on ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_n \in \mathbb{Z}_q^n$ where the ciphertexts form a linearly independent set as vectors over $\mathbb{Z}_q$. Then, since the matrix $\mathbf{C} = [\mathsf{ct}_1 | \ldots, | \mathsf{ct}_n] \in \mathbb{Z}_q^{n \times n}$ is full rank, there exists a vector $\mathbf{u} \in \mathbb{Z}_q^n$ such that $\mathbf{C} \cdot \mathbf{u} = \mathbf{e}_1$ for the elementary basis vector $\mathbf{e}_1 = (1, 0, \ldots, 0)$. To recover the first vector position of $\mathsf{sk}_i \in \mathbb{Z}_q^n$, the adversary can bit-decompose the vector $\mathbf{u}$ and linearly combine the noisy inner products

$$\begin{gathered} \{\langle 2^j \cdot \mathsf{ct}_1, \mathsf{sk}_i \rangle + \mathsf{noise}^{(j)}\}_{j \in \log q} \\ \{\langle 2^j \cdot \mathsf{ct}_2, \mathsf{sk}_i \rangle + \mathsf{noise}^{(j)}\}_{j \in \log q} \\ \vdots \\ \{\langle 2^j \cdot \mathsf{ct}_n, \mathsf{sk}_i \rangle + \mathsf{noise}^{(j)}\}_{j \in \log q} \end{gathered}$$

to form $\langle \mathbf{e}_1, \mathsf{sk}_i \rangle + \mathsf{noise}$. Since $\mathbf{u}$ is bit-decomposed, noise remains small and hence, the high-ordered bits of the first vector position of sk_i is leaked. The adversary can then repeat the attack (with different elementary basis vectors $\mathbf{e}_i$) to recover the high-ordered bits of the rest of the components of sk_i, which can completely compromise security.

As such attack demonstrates, managing the noise blow up to achieve both correct decryption and security is difficult. In this work, we show how to handle this 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 itself, we abstract it out by using a different secret sharing scheme. Specifically, 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 4.4). More precisely, for a fixed access structure, such secret sharing scheme divides a secret sk into a set of shares $\mathsf{sk}_1, \dots, \mathsf{sk}_N$ such that each sk_i itself consists of a set of individual shares in $\mathbb{Z}_q$, $\mathsf{sk}_i = \{\mathsf{s}_{i,j}\}_{j \in [\ell]}$ for some fixed bound ℓ. Then, we require that for any set $S \subseteq [N]$ that satisfies the access structure, there exists a subset of the individual shares $S' \subseteq \bigcup_{i \in S} \mathsf{sk}_i$ such that $\sum_{S'} \mathsf{s}_{i,j} = \mathsf{sk}$. With such requirement, it is straightforward to construct a correct threshold FHE scheme for $\{0,1\}$-LSSS following the approach outlined above. For simulatability, we use the fact that for a linear secret sharing scheme for $\{0,1\}$-LSSS, there exists a *maximal invalid share set* $S^* \subseteq \bigcup_{i \in [N]} \mathsf{sk}_i$ such that S^* itself does not reveal any information about the secret sk or any other individual shares $\mathsf{s}_{i,j} \in \bigcup_{i \in [N]} \mathsf{sk}_i$, but any set that strictly contains S^* completely determines the secret sk as well as all *individual* shares $\mathsf{s}_{i,j} \in \bigcup_{i \in [N]} \mathsf{sk}_i$. With careful analysis (Section 5.2), we show that this property allows the simulation of each partial decryptions.

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* [40]. A classic result of Valiant [53, 34] 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 [6, 37] 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 $\alpha \xleftarrow{\text{R}} \mathbb{Z}_q$ 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 4.1, 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* ThFHE 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 ThFHE construction. In particular, the size of the key shares sk_i for $i = 1, \ldots, N$ is at least $\Omega(N^4)$, introducing significant space overhead. In Section 5.3, we introduce another approach where the share sizes are quasilinear $\tilde{O}(N)$.[2]

The high level idea of our second method is to use the technique of "clearing out the denominators" [50, 2]. 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. This means that even when interpretted as an element in $\mathbb{Z}_q$, the term $(N!)^2 \cdot \lambda_i^{(S)}$ is bounded by a fixed positive integer. Hence, 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

[2]We describe the precise trade-offs between the two methods in [12].

growth, we can preserve correct reconstruction. In fact, with careful analysis (Section 5.3), such a ThFHE 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 ThFHE 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) ThFHE 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 the full version [12].

2.2 Universal Thresholdizer: A General Tool

We next put our new ThFHE to use. We define a new primitive called a *universal thresholdizer* (UT) that can be used to thresholdize many existing systems including signatures (Section 8.1). 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 7.2, we construct a robust universal thresholdizer using non-interactive zero knowledge proofs (NIZK). We note that constructing NIZKs from lattices is still an open problem. However, our setting allows the use of NIZK with preprocessing [24, 39], which can be constructed from lattices [38]. A better way to ensure robustness is using *homomorphic signatures* (Section 7.3). Because homomorphic signatures [11, 35] 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 [35].

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 $\mathcal{P}(X)$ 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. 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 $\omega \xleftarrow{\text{R}} \Omega$. 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 $\Pr_{x \leftarrow \chi(\lambda)}[|x| \leq B(\lambda)] = 1$. In the full version of this paper [12], we provide additional preliminaries in statistical distance, lattice cryptography, as well as definitions of basic cryptographic primitives.

4 Secret Sharing for Threshold Access Structures

In this section, we provide general results on secret sharing that we use throughout this work. We provide additional background on basic notations and terms that we use in the full version [12]. In Section 4.1, we define threshold access structures and recall Shamir secret sharing. In Section 4.2, 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.

4.1 Threshold Access Structures

In this section, we define the class of threshold access structures TAS and describe Shamir secret sharing [49].

Definition 4.1 (TAS). *Let $P = \{P_1, \ldots, P_N\}$ 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 4.2 (Shamir Secret Sharing). *Let $P = \{P_1, \ldots, P_N\}$ be a set of parties and let TAS be the class of threshold access structures on P. Then, there exists a linear secret sharing scheme 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" [50, 2].

Lemma 4.3 ([2]). *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 an integer and is bounded*

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

4.2 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. We show that the class $\{0,1\}$-LSSS contains the class of threshold access structures. In Section 5.2, we construct a threshold fully homomorphic encryption scheme for these classes of access structures.

Definition 4.4 **($\{0,1\}$-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 $\mathbb{A} \in \mathsf{LSSS}_N$ for which there exists an efficient linear secret sharing scheme $\mathsf{SS} = (\mathsf{SS.Share}, \mathsf{SS.Combine})$ over the secret space $\mathcal{K} = \mathbb{Z}_p$ satisfying 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 satisfies 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 the full version [12].

Lemma 4.5. *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 4.4 can be computed efficiently.*

We now state the main theorem of this section.

Theorem 4.6. $\mathsf{TAS} \subseteq \{0,1\}$-$\mathsf{LSSS}$.

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

Definition 4.7 (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 gate is one of AND or OR gate with fan-in 2 and fan-out 1.*
– *The input wires can have multiple fan-out.*

Definition 4.8 **(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 4.6 is implied by the following.

Theorem 4.9 ([53, 34]). $\mathsf{TAS} \subseteq \mathsf{MBF}$.

Theorem 4.10 ([40]). $\mathsf{MBF} \subseteq \{0,1\}$-$\mathsf{LSSS}$.

Although Theorem 4.10 is folklore, we provide the formal proof in the full version [12].

5 Threshold Fully Homomorphic Encryption

In this section, we present the definition of threshold fully homomorphic encryption (ThFHE) for any class of access structures. Then, in Sections 5.2 and 5.3, we construct ThFHE for the class of threshold access structure TAS. In the full version of this work [12], we provide the performance comparisons of the two constructions.

5.1 Definitions

Definition 5.1 (Threshold Fully Homomorphic Encryption (ThFHE)). *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* $\mathsf{ThFHE} = (\mathsf{ThFHE.Setup}, \mathsf{ThFHE.Encrypt}, \mathsf{ThFHE.Eval}, \mathsf{ThFHE.PartDec}, \mathsf{ThFHE.FinDec})$ *with the following properties:*

- $\mathsf{ThFHE.Setup}(1^\lambda, 1^d, \mathbb{A}) \to (\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N)$*: 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* $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$.
- $\mathsf{ThFHE.Encrypt}(\mathsf{pk}, \mu) \to \mathsf{ct}$*: On input a public key* pk*, and a single bit plaintext $\mu \in \{0,1\}$, the encryption algorithm outputs a ciphertext* ct.
- $\mathsf{ThFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots \mathsf{ct}_k) \to \hat{\mathsf{ct}}$*: 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 ciphertext* $\hat{\mathsf{ct}}$.
- $\mathsf{ThFHE.PartDec}(\mathsf{pk}, \mathsf{ct}, \mathsf{sk}_i) \to \mathsf{p}_i$*: 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.*
- $\mathsf{ThFHE.FinDec}(\mathsf{pk}, B) \to \hat{\mu}$*: 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 ThFHE scheme satisfies compactness, correctness, and security.

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

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

Definition 5.4 (Semantic Security). *We say that a* ThFHE *scheme satisfies semantic security if for all* λ*, and depth bound* d*, the following holds. For any PPT adversary* $\mathcal{A}$*, the following experiment* $\mathsf{Expt}_{\mathcal{A},\mathsf{ThFHE},\mathsf{sem}}(1^\lambda, 1^d)$ *outputs* 1 *with negligible probability:*

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

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* $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{ThFHE.Setup}(1^\lambda, 1^d, \mathbb{A})$ *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* $\mathsf{ThFHE.Encrypt}(\mathsf{pk}, b)$ *for* $b \xleftarrow{\text{R}} \{0, 1\}$ *to* $\mathcal{A}$*.*
5. $\mathcal{A}$ *outputs a guess* b'*. The experiment outputs* 1 *if* $b = b'$*.*

In addition to the standard semantic security notion for ThFHE, we require a ThFHE scheme to satisfy simulation security. Semantic security guarantees that the *ciphertexts* of a ThFHE scheme does not reveal any information to an adversary with an unqualified set of partial decryption keys. For most use cases for ThFHE, the adversary additionally gets access to valid *partial decryptions* of ciphertexts. Simulation security guarantees that the partial decryptions also do not leak information to the adversary.

Definition 5.5 (Simulation Security). *We say that a* ThFHE *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* $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{ThFHE.Setup}(1^\lambda, 1^d, \mathbb{A})$ *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* $\{\mathsf{ThFHE.Encrypt}(\mathsf{pk}, \mu_i)\}_{i \in [k]}$ *to* $\mathcal{A}$*.*
5. $\mathcal{A}$ *issues a polynomial number of adaptive queries of the form* $(S \subseteq \{P_1, \ldots, P_N\}, C)$ *for circuits* $C : \{0, 1\}^k \to \{0, 1\}$ *of depth at most* d*. For each query, the challenger computes* $\hat{\mathsf{ct}} \leftarrow \mathsf{ThFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ *and provides* $\{\mathsf{ThFHE.PartDec}(\mathsf{pk}, \hat{\mathsf{ct}}, \mathsf{sk}_i)\}_{i \in S}$ *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* $(\mathsf{pk}, \mathsf{sk}_1, \dots \mathsf{sk}_N, \mathsf{st}) \leftarrow \mathcal{S}_1(1^\lambda, 1^d, \mathbb{A})$ *and provides* pk *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *outputs a maximal invalid party set* $S^* \subseteq \{P_1, \dots, P_N\}$ *and messages* $\mu_1, \dots, \mu_k \in \{0,1\}$*.*
4. *The challenger provides the keys* $\{\mathsf{sk}_i\}_{i \in S^*}$ *and* $\{\mathsf{ThFHE.Encrypt}(\mathsf{pk}, \mu_i)\}_{i \in [k]}$ *to* $\mathcal{A}$*.*
5. $\mathcal{A}$ *issues a polynomial number of adaptive queries of the form* $(S \subseteq \{P_1, \dots, P_N\}, C)$ *for circuits* $C : \{0,1\}^k \to \{0,1\}$ *of depth at most* d*. For each query, the challenger runs the simulator* $\{\mathsf{p}_i\}_{i \in S} \leftarrow \mathcal{S}_2(C, \{\mathsf{ct}_1, \dots, \mathsf{ct}_k\}, C(\mu_1, \dots, \mu_k), S, \mathsf{st})$ *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*.*

We note that it is possible to unify the two security definitions, and we do so for our definition of universal thresholdizers (Section 7). However, for ThFHE, we present the two definitions separately for more intuition and modularity in the security proof.

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

In this section, we present our construction of ThFHE for the class of access structures $\{0,1\}$-LSSS. We note that by Theorem 4.6, this gives a ThFHE scheme for the class of threshold access structures TAS.

Construction 5.6 *Let* $P = \{P_1, \dots, P_N\}$ *be a set of parties. Our* ThFHE *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 [12, Def. 3.9].*
- *Let* $\mathsf{SS} = (\mathsf{SS.Share}, \mathsf{SS.Combine})$ *be a special linear secret sharing scheme (Definition 4.4). 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 smudging noise (see Section 5.2.1). We construct* $\mathsf{ThFHE} = (\mathsf{ThFHE.Setup}, \mathsf{ThFHE.Encrypt}, \mathsf{ThFHE.Eval}, \mathsf{ThFHE.PartDec}, \mathsf{ThFHE.FinDec})$ *as follows:*

- $\mathsf{ThFHE.Setup}(1^\lambda, 1^d, \mathbb{A})$*: 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* $i = 1, \dots, N$*.*
- $\mathsf{ThFHE.Encrypt}(\mathsf{pk}, \mu)$*: 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*.*

- $\mathsf{ThFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$*: 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* $\hat{\mathsf{ct}}$*.*
- $\mathsf{ThFHE.PartDec}(\mathsf{pk}, \mathsf{ct}, \mathsf{sk}_i)$*: 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* $e_j \xleftarrow{\mathrm{R}} [-B_{\mathsf{sm}}, B_{\mathsf{sm}}]$ *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.*
- $\mathsf{ThFHE.FinDec}(\mathsf{pk}, B)$*: 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* $T \subseteq \bigcup_{i \in S} T_i$ *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 [12, Def. 3.6]. Then, the* ThFHE *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 [12, Def. 3.7] with noise bound* B *and* SS *is a secret sharing scheme that satisfies correctness [12, Def. 4.5]. Then, the* ThFHE *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 [12, Def. 3.8]. Then, the* ThFHE *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 [12, Def 3.8] and* SS *is a secret sharing scheme that satisfies security [12, Def 4.6]. Then, the* ThFHE *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 the full version [12].

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).

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 $B = \mathsf{poly}(\lambda)$ and $q = 2^{\tilde{O}(d)+\omega(\log n)}$. Then, if we set $B_{\mathsf{sm}} = 2^{\tilde{O}(d)+\omega(\log n)}$, the two conditions above are satisfied. In particular, this translates to approximating worst-case lattice problems with sub-exponential approximation factors.

5.3 ThFHE from Shamir secret sharing

In this section, we present our construction of ThFHE using a standard Shamir secret sharing scheme. This construction does not satisfy our notion of compactness 5.2. However, in the full version [12], we show how to transform a non-compact ThFHE scheme to a compact one generically using UT.

Construction 5.11 *Let $P = \{P_1, \ldots, P_N\}$ be a set of parties. Our* ThFHE *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 $(N!)^2$ ([12, Def. 3.9]).*
- *Let* $\mathsf{SS} = (\mathsf{SS.Share}, \mathsf{SS.Combine})$ *be a Shamir secret sharing scheme (Theorem 4.2).*

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

- $\mathsf{ThFHE.Setup}(1^\lambda, 1^d, \mathbb{A}_t)$*: 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 $i = 1, \ldots, N$.*
- $\mathsf{ThFHE.Encrypt}(\mathsf{pk}, \mu)$*: 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*.*
- $\mathsf{ThFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$*: 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* $\hat{\mathsf{ct}}$*.*
- $\mathsf{ThFHE.PartDec}(\mathsf{pk}, \mathsf{ct}, \mathsf{sk}_i)$*: 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 $e \xleftarrow{R} [-B_{\mathsf{sm}}, B_{\mathsf{sm}}]$ 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*.*
- $\mathsf{ThFHE.FinDec}(\mathsf{pk}, B)$*: 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 ([12, Def. 3.7]) with noise bound* B *and* SS *is a Shamir secret sharing scheme that satisfies correctness (Theorem 4.2). Then, the* ThFHE *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 ([12, Def. 3.8]). Then, the* ThFHE *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 ([12, Def. 3.8]) and* SS *is a secret sharing scheme that satisfies security ([12, Def. 4.6]). Then, the* ThFHE *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 the full version [12].

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).

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 $B = \mathsf{poly}(\lambda)$ and $q = 2^{\tilde{O}(d)+\omega(\log n)}$. Then, if we set $B_{\mathsf{sm}} = 2^{\tilde{O}(d)+\omega(\log n)}/(N!)^3$, the two conditions above are satisfied. In particular, this translates to approximating worst-case lattice problems with sub-exponential approximation factors.

6 Decentralized ThFHE

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 ThFHE 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 ThFHE that we name *decentralized threshold fully homomorphic encryption* (dThFHE). In a dThFHE 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 $S \in \mathbb{A}$. Due to space limitations, we provide the formal definition of dThFHE in Section 6.1 and provide the construction in the full version [12].

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 dThFHE 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 a class of access structures* $\mathbb{S}$ *is a tuple of PPT algorithms* $\mathsf{dThFHE} = (\mathsf{dThFHE.Encrypt}, \mathsf{dThFHE.Eval}, \mathsf{dThFHE.PartDec}, \mathsf{dThFHE.FinDec})$ *with the following properties:*

- $\mathsf{dThFHE.Encrypt}(1^\lambda, 1^d, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_N, \mathbb{A}, x) \rightarrow \mathsf{ct}$*: 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*.*
- $\mathsf{dThFHE.Eval}(C, \mathsf{ct}) \rightarrow \hat{\mathsf{ct}}$*: On input a circuit* $C : \{0,1\}^k \rightarrow \{0,1\}$*, and a ciphertext* ct*, the evaluation algorithm outputs an evaluated ciphertext* $\hat{\mathsf{ct}}$*.*
- $\mathsf{dThFHE.PartDec}(\hat{\mathsf{ct}}, \mathsf{Dec}_i)$*: 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*.*
- $\mathsf{dThFHE.FinDec}(B)$*: On input a set of partial decryptions* $\{\mathsf{p}_i\}_{i\in S}$*, the final decryption algorithm outputs a message* x'*.*

We require a dThFHE scheme to satisfy the following compactness, correctness, and security properties. We note that our compactness notion for dThFHE 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* dThFHE *scheme for* $\mathbb{S}$ *is compact if there exists a polynomial* $\mathsf{poly}(\cdot)$ *such that for all* λ*, depth bound* d*, circuit* $C : \{0,1\}^k \rightarrow \{0,1\}$ *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* $\mathsf{ct} \leftarrow \mathsf{dThFHE.Encrypt}(1^\lambda, 1^d, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_N, \mathbb{A}, x)$*,* $\hat{\mathsf{ct}} \leftarrow \mathsf{dThFHE.Eval}(C, \mathsf{ct})$*, and* $\mathsf{p}_i \leftarrow \mathsf{dThFHE.PartDec}(\hat{\mathsf{ct}}, \mathsf{Dec}_i)$ *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* dThFHE *scheme for* $\mathbb{S}$ *satisfies evaluation correctness if for all* λ*, depth bound* d*, circuit* $C : \{0,1\}^k \rightarrow$

$\{0,1\}$ *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* $\mathsf{ct} \leftarrow \mathsf{dThFHE.Encrypt}(1^\lambda, 1^d, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_N, \mathbb{A}, x)$*, and* $\hat{\mathsf{ct}}, \leftarrow \mathsf{dThFHE.Eval}(C, \mathsf{ct})$*, we have*

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

Definition 6.4 (Semantic Security). *We say that a* dThFHE *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* $\mathsf{Expt}_{\mathcal{A},\mathsf{dThFHE,sem}}(1^\lambda, 1^d, \{\mathsf{Enc}_i\}_{i \in [N]})$ *outputs 1 with negligible probability:*

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

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* $\mathsf{ct}_b \leftarrow \mathsf{dThFHE.Encrypt}(1^\lambda, 1^d, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_N, \mathbb{A}, x_b)$ *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* dThFHE *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* $\mathsf{Expt}_{\mathcal{A},\mathsf{dThFHE,Real}}(1^\lambda, 1^d, \{(\mathsf{Enc}_i, \mathsf{Dec}_i)\}_{i \in [N]})$ *and* $\mathsf{Expt}_{\mathcal{A},\mathsf{dThFHE,Ideal}}(1^\lambda, 1^d, \{(\mathsf{Enc}_i, \mathsf{Dec}_i)\}_{i \in [N]})$ *are computationally indistinguishable:*

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

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* $\mathsf{ct} \leftarrow \mathsf{dThFHE.Encrypt}(1^\lambda, 1^d, \mathsf{Enc}_1, \ldots, \mathsf{Enc}_N, \mathbb{A}, x)$ *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* $\hat{\mathsf{ct}} \leftarrow \mathsf{dThFHE.Eval}(C, \mathsf{ct})$ *and provides* $\{\mathsf{dThFHE.PartDec}(\hat{\mathsf{ct}}, \mathsf{Dec}_i)\}_{i \in S}$ *to* $\mathcal{A}$*.*
5. *At the end of the experiment,* $\mathcal{A}$ *outputs a distinguishing bit* b*.*

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

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.*

7 Universal Thresholdizer

The notion of ThFHE 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 ThFHE as a *thresholdizer* mechanism. In these settings, we do not require the full generality of the ThFHE syntax. Furthermore, for ThFHE 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 thresholdizers for our applications in Section 8.

7.1 Definition

Informally, the setup and the encryption algorithms for ThFHE are merged into a single UT setup algorithm, and the evaluation and partial decryption algorithms for ThFHE 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 ThFHE definitions. The security requirement of a ThFHE scheme combines the semantic and simulation security definitions of ThFHE. Verification correctness and robustness are additions to the definition to capture verifiable evaluation.

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* $|\mathsf{y}_i| \le \mathsf{poly}(\lambda, d, N)$.

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 $(\mathsf{pp}, \mathsf{s}_1, \ldots, \mathsf{s}_N, \mathsf{st}) \leftarrow \mathcal{S}_1(1^\lambda, 1^d, \mathbb{A})$ and provides pp to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs a maximal invalid party set $S^* \subseteq \{P_1, \ldots, P_N\}$ 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 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 $(\mathsf{pp}, \mathsf{s}_1, \ldots, \mathsf{s}_N) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^d, \mathbb{A}, x)$ and provides $(\mathsf{pp}, \mathsf{s}_1, \ldots, \mathsf{s}_N)$ 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 $\mathsf{UT.Verify}(\mathsf{pp}, \mathsf{y}_i^*, C) = 1$.*

7.2 Universal Thresholdizer from ThFHE and PZK

In this section, we construct a universal thresholdizer generically from threshold fully homomorphic encryption (Section 5) and NIZK with pre-processing (see [12]).

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

- *Let* $\mathsf{ThFHE} = (\mathsf{ThFHE.Setup}, \mathsf{ThFHE.Encrypt}, \mathsf{ThFHE.Eval}, \mathsf{ThFHE.PartDec}, \mathsf{ThFHE.FinDec})$ *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* ThFHE *keys* $(\mathsf{tfhepk}, \mathsf{tfhesk}_1, \ldots, \mathsf{tfhesk}_N) \leftarrow \mathsf{ThFHE.Setup}(1^\lambda, 1^d, \mathbb{A})$ *and ciphertexts* $\mathsf{ct}_i \leftarrow \mathsf{ThFHE.Encrypt}(\mathsf{tfhepk}, x_i)$ *for* $i = 1, \ldots k$. *Then, it generates reference strings* $(\sigma_{V,i}, \sigma_{P,i}) \leftarrow \mathsf{PZK.Pre}(1^\lambda)$, *commitment randomness* $r_i \overset{\mathrm{R}}{\leftarrow} \{0,1\}^\lambda$, *and commitments* $\mathsf{com}_i \leftarrow \mathsf{C.Com}(\mathsf{tfhesk}_i; r_i)$ *for* $i = 1, \ldots N$. *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* $\hat{\mathsf{ct}} \leftarrow \mathsf{ThFHE.Eval}(\mathsf{tfhepk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ *and partial decryption* $\mathsf{p}_i \leftarrow \mathsf{ThFHE.PartDec}(\mathsf{tfhepk}, \hat{\mathsf{ct}}, \mathsf{tfhesk}_i)$. *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:

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

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 evaluation* y_i, *and a circuit* C, *the verification algorithm first computes the evaluated ciphertext* $\hat{\mathsf{ct}} \leftarrow \mathsf{ThFHE.Eval}(\mathsf{pp}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ *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* $\mathsf{ThFHE.FinDec}(\mathsf{tfhepk}, \{\mathsf{p}_i\}_{i \in S})$.

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

Theorem 7.8. *Suppose* ThFHE *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* ThFHE *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 ([12, Def. 3.4]). Then, the universal thresholdizer scheme from Construction 7.7 satisfies verification correctness (Definition 7.4).*

Theorem 7.11. *Suppose* ThFHE *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-knowledge ([12, Def. 3.4]), and* C *is a non-interactive commitment scheme that satisfies computational hiding ([12, Def. 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 ([12, Def. 3.4]) and* C *is a non-interactive commitment scheme that satisfies perfect binding ([12, Def. 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 the full version [12].

7.3 Robustness from Homomorphic Signatures

In Section 7.2, we used NIZK with pre-processing to enforce robustness. Another way to enforce robustness is to use homomorphic signatures [11, 35]. A homomorphic signature scheme 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 7.2, the setup algorithm can simply use a homomorphic signature to sign each decryption key share of a ThFHE scheme and include it as part of each party's share. Then, to evaluate on the shares, each user can homomorphically compute on the ThFHE 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 falsen 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 [35], its use does not introduce any new assumption to our construction. We provide the formal construction from homomorphic signatures in the full version [12].

8 Applications

In this section, we describe our applications of a universal thresholdizer scheme. Due to space constraints, we defer some of our applications in the full version [12]. In [12], we show that a universal thresholdizer scheme for a class of access structures immediately give rise to a function secret sharing scheme for the same class of access structures. We also 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 8.1) and CCA threshold PKE ([12]). These two notions demonstrate how to use a universal thresholdizer as a general tool. The methods that we develop in this section can be applied to a wide range of other applications in a straightforward way. In [12], we also show that a non-compact universal threholsidzer scheme can be used to thresholdize a compact fully homomorphic encryption scheme to construct a compact ThFHE 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 4.6, all applications in this section can be instantiated for the class of threshold access structure TAS (Definition 4.1).

8.1 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.

We provide the full definition of threshold signatures in the full version [12].

8.1.1 Construction

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

Construction 8.1 *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 8.1.

Theorem 8.2. *Suppose* UT *is a universal thresholdizer scheme that satisfies evaluation correctness (Definition 7.3). Then, the threshold signature scheme from Construction 8.1 satisfies compactness ([12, Def. 8.10]).*

Theorem 8.3. *Suppose* UT *is a universal thresholdizer scheme that satisfies evaluation correctness (Definition 7.3) and* S *is a signature scheme that satisfies correctness ([12, Def. A.4]). Then, the threshold signature scheme from Construction 8.1 satisfies evaluation correctness ([12, Def. 8.11]).*

Theorem 8.4. *Suppose* UT *is a universal thresholdizer scheme that satisfies evaluation verification correctness (Definition 7.4). Then, the threshold signature scheme from Construction 8.1 satisfies partial verification correctness ([12, Def. 8.12]).*

Theorem 8.5. *Suppose* UT *is a universal thresholdizer scheme that satisfies security (Definition 7.5) and* S *is a signature scheme that satisfies unforgeability ([12, Def. A.5]). Then, the threshold signature scheme from Construction 8.1 satisfies unforgeability ([12, Def. 8.13]).*

Theorem 8.6. *Suppose* UT *is a universal thresholdizer scheme that satisfiesd robustness (Definition 7.6). Then, the threshold signature scheme from Construction 8.1 satisfies robustness ([12, Def. 8.14]).*

Theorem 8.7. *Suppose* UT *is a universal thresholdizer scheme that satisfies evaluation correctness (Definition 7.3). Then, the threshold signature scheme from Construction 8.1 satisfies anonymity ([12, Def. 8.15]).*

We provide formal proofs of the theorems above in the full version [12].

9 Conclusion and Open Problems

In this work, we proposed a general framework for constructing various threshold cryptosystems from standard lattice assumptions. We first defined the notion of *threshold fully homomorphic encryption* (ThFHE) and constructed it from LWE. Then, we showed that ThFHE can be used to instantiate a new abstraction called *universal thresholdizers*, which can be combined with existing cryptographic primitives like digital signatures and CCA-secure PKEs to form new *threshold signatures* and *CCA-secure threshold PKEs* from LWE.

Our work gives rise to many new open problems in threshold cryptography. A universal thresholdizer can be used as a tool to construct a variety of different primitives in threshold cryptography. Can universal thresholdizers be realized from other standard assumptions such as DDH or assumptions on bilinear maps? Are there more efficient constructions of universal thresholdizers from LWE?

On the theoretical side, we show how to construct threshold signatures or threshold PKEs via a generic, but primitive dependent transformation using universal thresholdizers. Is it possible to formalize what it means to thresholdize *any* cryptographic function?

Acknowledgements

We thank the anonymous Crypto reviewers for their helpful comments. 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

1. S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, 2010.
2. S. Agrawal, X. Boyen, V. Vaikuntanathan, P. Voulgaris, and H. Wee. Functional encryption for threshold functions (or fuzzy ibe) from lattices. In *PKC*, 2012.
3. G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs. Multiparty computation with low communication, computation and interaction via threshold fhe. In *EUROCRYPT*, 2012.
4. A. Banerjee and C. Peikert. New and improved key-homomorphic pseudorandom functions. In *CRYPTO*, 2014.

5. C. Baum, I. Damgård, S. Oechsner, and C. Peikert. Efficient commitments and zero-knowledge protocols from ring-sis with applications to lattice-based threshold cryptosystems. *IACR Cryptology ePrint Archive*, 2016:997, 2016.
6. A. Beimel. Phd thesis. *Israel Institute of Technology, Technion, Haifa, Israel,*, 1996.
7. R. Bendlin and I. Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In *TCC*, 2010.
8. R. Bendlin, S. Krehbiel, and C. Peikert. How to share a lattice trapdoor: threshold protocols for signatures and (h) ibe. In *ACNS*, 2013.
9. A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *PKC*, 2003.
10. D. Boneh, X. Boyen, and S. Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. In *CT-RSA*, 2006.
11. D. Boneh and D. M. Freeman. Homomorphic signatures for polynomial functions. In *EUROCRYPT*, 2011.
12. D. Boneh, R. Gennaro, S. Goldfeder, A. Jane, S. Kim, P. M. R. Rasmussen, and A. Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. Cryptology ePrint Archive, Report 2017/956, 2017. `https://eprint.iacr.org/2017/956`.
13. D. Boneh, K. Lewi, H. Montgomery, and A. Raghunathan. Key homomorphic prfs and their applications. In *CRYPTO*. 2013.
14. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. *J. Cryptology*, 17(4):297–319, Sept. 2004.
15. X. Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In *PKC*, 2010.
16. E. Boyle, N. Gilboa, and Y. Ishai. Function secret sharing. In *EUROCRYPT*, 2015.
17. E. Boyle, N. Gilboa, and Y. Ishai. Breaking the circuit size barrier for secure computation under ddh. In *CRYPTO*, 2016.
18. Z. Brakerski, N. Chandran, V. Goyal, A. Jain, A. Sahai, and G. Segev. Hierarchical functional encryption. In *ITCS*, 2016.
19. Z. Brakerski and R. Perlman. Lattice-based fully dynamic multi-key fhe with short ciphertexts. CRYPTO, 2016.
20. Z. Brakerski and V. Vaikuntanathan. Constrained key-homomorphic prfs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In *TCC*, 2015.
21. R. Canetti and S. Goldwasser. An efficient *Threshold* public key cryptosystem secure against adaptive chosen ciphertext attack. In *EUROCRYPT*, 1999.
22. P.-L. Cayrel, R. Lindner, M. Rückert, and R. Silva. A lattice-based threshold ring signature scheme. In *LATINCRYPT*, 2010.
23. A. Choudhury, J. Loftus, E. Orsini, A. Patra, and N. P. Smart. Between a rock and a hard place: Interpolating between mpc and fhe. In *ASIACRYPT*, 2013.
24. A. De-Santis, S. Micali, and G. Persiano. Non-interactive zero-knowledge with preprocessing. In *CRYPTO*, 1988.
25. A. DeSantis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely. In *STOC*, 1994.
26. Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *CRYPTO*, 1989.
27. Y. Dodis and J. Katz. Chosen-ciphertext security of multiple encryption. In *TCC*, 2005.
28. Y. Frankel. A practical protocol for large group oriented networks. In *EUROCRYPT*, 1989.

29. R. Gennaro, S. Goldfeder, and A. Narayanan. Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In *ACNS*, 2016.
30. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. *Inf. Comput.*, 164(1):54–84, 2001.
31. R. Gennaro, T. Rabin, S. Jarecki, and H. Krawczyk. Robust and efficient sharing of RSA functions. *J. Cryptology*, 20(3):393, 2007.
32. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, 2008.
33. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*. 2013.
34. O. Goldreich. Valiant's polynomial-size monotone formula for majority. 2014.
35. S. Gorbunov, V. Vaikuntanathan, and D. Wichs. Leveled fully homomorphic signatures from standard lattices. In *STOC*, 2015.
36. S. D. Gordon, F. Liu, and E. Shi. Constant-round MPC with fairness and guarantee of output delivery. In *CRYPTO*, pages 63–82, 2015.
37. V. Goyal, O. Pandey, A. Sahai, and B. 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.
38. S. Kim and D. J. Wu. Multi-theorem preprocessing nizk from lwe. In *CRYPTO*. 2018.
39. D. Lapidot and A. Shamir. Publicly verifiable non-interactive zero-knowledge proofs. In *CRYPTO*, 1990.
40. A. B. Lewko and B. Waters. Decentralizing attribute-based encryption. In *EUROCRYPT*, 2011.
41. V. Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, 2012.
42. D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*. 2012.
43. P. Mukherjee and D. Wichs. Two round multiparty computation via multi-key fhe. In *EUROCRYPT*, 2016.
44. S. Myers, M. Sergi, and A. Shelat. Threshold fully homomorphic encryption and secure computation. *IACR Cryptology ePrint Archive*, 2011:454, 2011.
45. C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, 2009.
46. C. Peikert and S. Shiehian. Multi-key fhe from lwe, revisited. In *TCC*, 2016.
47. C. Peikert and B. Waters. Lossy trapdoor functions and their applications. *SIAM Journal on Computing*, 40(6):1803–1844, 2011.
48. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009.
49. A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
50. V. Shoup. Practical threshold signatures. In *EUROCRYPT*, 2000.
51. V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *J. Cryptology*, 15(2):75–96, 2002.
52. D. R. Stinson and R. Strobl. Provably secure distributed schnorr signatures and a (t, n) threshold scheme for implicit certificates. In *ACISP*, 2001.
53. L. G. Valiant. Short monotone formulae for the majority function. *J. Algorithms*, 1984.
54. X. Xie, R. Xue, and R. Zhang. Efficient threshold encryption from lossy trapdoor functions. In *PQCrypto*, 2011.