A Lattice-Based Universal Thresholdizer for Cryptographic Systems

Dan Boneh* Rosario Gennaro† Steven Goldfeder‡ Sam Kim§

Abstract

We develop a general approach to thresholdizing a large class of (non-threshold) cryptographic schemes. We show how to add threshold functionality to CCA-secure public-key encryption (PKE), signature schemes, pseudorandom functions, and others primitives. To do so, we introduce a general tool, called a *universal thresholdizer*, from which many threshold systems are possible. The tool builds upon a lattice-based fully-homomorphic encryption (FHE) system. Applying the tool to a (non-threshold) lattice-based signature, gives the first single-round threshold signature from the learning with errors problem (LWE). Applying the tool to a (non-threshold) lattice-base CCA-secure PKE, gives a single-round lattice-based threshold CCA-secure PKE.

1 Introduction

Threshold cryptography [DF89, Fra89, DDFY94] is a general technique used to protect a cryptographic secret by splitting it into N shares and storing each share on a different server. Any subset of t servers can use the secret, without re-constructing it. 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.

Moreover, the time to verify signatures or encrypt messages should be independent of N and t. In Section 6 we review the precise security properties that such systems must satisfy. 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. For now, we assume that the system is setup by a trusted entity that generates the secret key, and hands the key shares to the servers. We will often refer to the servers as *partial signers* or *partial decryptors*.

*Stanford University. Email: dabo@cs.stanford.edu.
†City College of New York. Email: rosario@cs.ccny.cuny.edu.
‡Princeton University. Email: stevenag@cs.princeton.edu.
§Stanford University. Email: skim13@cs.stanford.edu.

Many signature and encryption schemes have been thresholdized. For example, RSA signatures and encryption [Fra89, DDFY94, GRJK07, Sho00], Schnorr signatures [SS01], (EC)DSA signatures [GJKR01, GGN16], BLS signatures [BLS04, Bol03], Cramer-Shoup encryption [CG99], Regev encryption [BD10], and many more [SG02, BBH06].

Despite this great success, it is still an open problem to construct one-round threshold signatures from hard lattice problems, as discussed in the related work section (Section 1.2).

Our contributions. We present a general framework for universally thresholdizing many (non-threshold) cryptographic schemes. Specifically, we define a new primitive called a *universal thresholdizer*, which is a cryptographic scheme that can be composed with different types of systems to thresholdize them (Section 4). For example, we can take *any* signature or encryption scheme as a black box and construct from it a one-round threshold system. Similarly, we can thresholdize IBE key generation, ABE key generation, pseudorandom functions, and others. We demonstrate how to use the primitive by showing in detail how to apply the technique to any non-threshold signature and CCA-secure public key encryption schemes to obtain one-round threshold signatures and one-round CCA-secure threshold encryption (Section 6).

Our universal thresholdizer is built from a fully-homomorphic encryption (FHE) system, such as [GSW13]. Because this FHE is proven secure based on the learning with errors (LWE) problem, and because there are known secure (non-threshold) signature schemes based on LWE [GPV08, Boy10, Lyu12], we obtain the first one-round threshold signature scheme based on LWE that is both succinct and anonymous. This resolves a long-standing open problem in lattice-based cryptography.

Beyond signatures, our universal thresholdizer can be composed with an existing CCA-secure public key encryption (PKE) scheme [PW11, GPV08, Pei09, ABB10a, MP12] to obtain the first lattice-based (one-round) threshold CCA-secure PKE where the public key size and encryption time are independent of the number of servers. Similarly, we can compose the universal thresholdizer with an existing IBE scheme [GPV08, ABB10b, ABB10c, CHKP10] to obtain the first lattice-based IBE system with (one-round) threshold key generation. More generally, composing the universal thresholdizer with a functional encryption scheme gives a system with threshold key generation.

Threshold systems should also be robust [GRJK07]: a misbehaving server who sends an invalid partial signature or partial decryption, should be detected by the combiner. We construct our universal thresholdizer to provide robustness, thereby ensuring that all the derived threshold schemes are robust. Robustness is often achieved using non-interactive zero-knowledge (NIZK). However, because there are no known standard-model NIZKs from LWE, we instead show how to provide robustness using an LWE-based fully homomorphic signature scheme [GVW15]. In summary, we obtain a *robust* one-round LWE-based threshold signature, CCA-secure PKE, and other threshold primitives in the standard model.

1.1 Overview of the Main Construction

We begin with an overview of the main construction. While our approach gives a universal thresholdizer, in this section we give a concrete example showing how to apply our techniques to construct a threshold signature scheme. In Section 5 we proceed differently by first constructing universal thresholdizer scheme, and then presenting a threshold signature scheme as one of its immediate applications (Section 6).

MPC from FHE. Our starting point is the recent development of low-round multiparty computation (MPC) from LWE ([AJLA$^+$12, MW16, BP16, PS16]). For example, in [AJLA$^+$12] the N parties first generate a fully homomorphic encryption (FHE) public-key pk, where the secret key sk is shared among the parties in an N-out-of-N fashion. Next, every party encrypts its input using the FHE public-key pk, and broadcasts the ciphertext to all other parties. Each party receives the FHE encrypted inputs from its peers, and does the following: first, it locally and homomorphically computes the MPC function on the FHE encrypted inputs; second, it partially decrypts the resulting ciphertext using its share of sk; and third, it broadcasts the result to all other parties. The partial decryptions can then be combined into a full decryption so that all parties learn the result of the computation. Nothing else is revealed about the original inputs.

A threshold signature scheme can be viewed as a two-round MPC, but without a broadcast channel. Given a message to sign, every partial signer computes the partial signature individually without any interaction

with the other parties, and provides the partial signatures to the combiner. From this point of view, it is unclear how to extend the FHE technique above to the setting of threshold signatures, since the technique crucially relies on a broadcast channel to ensure that every party supplies the same encrypted input to all other parties. We bypass this limitation by observing that in the context of threshold signatures, all parties compute on a single input, namely the signature signing key, that can be provided to all parties at setup.

More precisely, the partial signers compute the public function $f_m(k) = \mathsf{Sign}(k, m)$, where k is a signing key and m is a message to be signed. Here the input k remains fixed, while the public function f_m to be computed changes for every signature. Our idea, then, is to provide each party at setup time with an FHE ciphertext ct_k which is an encryption of the signing key k, and also provide the party with one share of the FHE decryption key. Then, to sign, each party locally and homomorphically computes the signing function f_m on the shared ciphertext ct_k, to obtain an FHE encrypted signature. The party then partially decrypts this FHE ciphertext, and sends this partial decryption to the combiner. The combiner combines the partial decryptions, and obtains the signature $\mathsf{Sign}(k, m)$ in the clear.

Thresholdizing decryption. The construction described above gives an N-out-of-N threshold signature scheme where N valid partial signatures can be combined to produce a final signature. Our goal is to construct a scheme that supports arbitrary thresholds. To do this, we must delve into the specific properties of the LWE-based FHE constructions.

Recall that a ciphertext ct of an LWE-based FHE scheme (such as [GSW13]) is a matrix in $\mathbb{Z}_q^{n\times m}$. 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 particular column ct_m of the ciphertext, 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 $\mathsf{sk}_1, ..., \mathsf{sk}_N$, one for each user. Then, to sign, user i runs the signing function f_m on the FHE encryption of sk, to produce ct^* and then provides the inner product $\langle \mathsf{ct}^*, \mathsf{sk}_i\rangle$ to the combiner. The combiner can then compute the Lagrange coefficients λ_i^S for any set S of size t and recombine the shares as

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

Accounting for noise. Unfortunately, the construction described above is insecure. Every time a signer computes a partial decryption, it leaks information about its share of the decryption key sk_i by publishing its inner product with a publicly known vector ct^*. One way to resolve this issue is by adding some small additive noise term on the inner product

$$\langle \mathsf{ct}^*, \mathsf{sk}_i\rangle + \mathsf{noise}.$$

However, for a t-out-of-N threshold scheme, this additive error prevents correct reconstruction of the signature. Namely 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.

To resolve this, we use the technique of "clearing out the denominators" that was previously used in the works of [Sho00, ABV+12]. The idea 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. Therefore, we can modify the construction so that every signer first scales the noise that it adds by $(N!)^2$, so that when multiplied by the Lagrange coefficient, the quantity remains bounded as an integer. Now, with an upper bound on the FHE noise growth, and an upper bound on the scaled Lagrange coefficients, we can choose the FHE modulus to be large enough to preserve correct reconstruction. This has no impact on the final signature size, since the final combiner output is always just $\mathsf{Sign}(k, m)$.

Smudging out original noise. To prove the scheme secure, some additional complications arise. For instance, the noise produced by the FHE decryption (after computing the inner product $\langle \mathsf{ct}^*, \mathsf{sk}\rangle$) contains information about the underlying plaintext. In particular, every time a combiner reconstructs the final

signature, it learns some additional information about the underlying signing key. To prevent this, each signer must add on a large enough noise to overwhelm or "smudge out" the decryption noise. However, the noise that the signers add on are scaled by $(N!)^2$ and therefore, cannot be used to smudge out the decryption noise. To resolve this, we scale the noise in the FHE ciphertext appropriately such that the decryption noise is always a multiple of $(N!)^2$. In Appendix B, we show that this modification still preserves security of the underlying FHE constructions over a prime modulus.

Verifiability from homomorphic signatures. The final piece of the puzzle is enforcing robustness of the threshold signature scheme. Robustness means that a malicious partial signer cannot send an improperly generated partial signature without being caught. A natural approach to enforcing robustness is for the signers to include a non-interactive zero knowledge (NIZK) proof as part of the partial signatures. However, a NIZK construction based on LWE is not (yet) known. Therefore, to enforce robustness, we rely on the recent developments in homomorphic signature schemes [BF11b, GVW15] from LWE (more precisely based on the Short Integer Solutions problem). The idea of a homomorphic signature scheme is that given some data and a signature of this data, one can homomorphically compute on the data, and at the same time operate on the signatures to provide a proof of validity of the output of a computation. By providing each signer with a signature of their share of the decryption key, as well as a PRF key for generating the randomness in the protocol using a homomorphic signature scheme, we achieve robustness for our final construction.

1.2 Related Work

We survey some of the works on threshold cryptosystems based on lattices.

Threshold public-key encryption. Bendlin and Damgård [BD10] gave a threshold version of Regev's CPA-secure encryption scheme [Reg09], and Myers et al. [MSS11] applied the technique to fully homomorphic encryption. Xie et al. [XXZ11] gave a threshold CCA secure PKE scheme from lossy trapdoor functions, which can be instantiated from LWE [PW11] although the size of the public key and the ciphertext is at least linear in the number of decryptors. Finally, the threshold Gaussian sampling of Bendlin et al. [BKP13] gives a threshold (H)IBE, which can be converted to a CCA threshold PKE where the size of the public key and ciphertext is independent of the number of decryptors in the system. A limitation of this system is that the decrypting servers can only carry out an *a priori* bounded number of *online* non-interactive decryptions before they must perform an *offline* interactive phase.

Threshold Signatures. In the signatures front, there have been fewer works. Cayrel et al. [CLRS10] gave a lattice-based threshold ring signature scheme, in which at least t signers are needed to create an anonymous signature. In this system, each signers has its own public key, and verification time grows linearly with the number of signers. The threshold Gaussian sampling construction of [BKP13] gives a threshold signature where the public key and the size of the signatures are all independent of the number of signers. However, as in the encryption setting, the signers can only sign a bounded number of signatures non-interactively, before they must perform an interactive protocol.

Other Threshold Cryptosystems. Several other works construct threshold systems from lattices. Boneh et al. [BLMR13] give a threshold distributed PRF from key-homomorphic PRFs, where the key sizes are independent of the number of evaluators and the evaluation process is also non-interactive. Additional lattice-based threshold PRFs include [BP14, BV15].

2 Preliminaries

Basic Notations. For an integer n, we write $[n]$ to denote the set $\{1, ..., n\}$. 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 the 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 $\mathbf{x}$.

We write λ for the security parameter. We say that a function $\epsilon(\lambda)$ is negligible in λ if $\epsilon(\lambda) = o(1/\lambda^c)$ for every $c \in \mathbb{N}$, and we write $\mathsf{negl}(\lambda)$ to denote a negligible function in λ. We say that an event occurs with *negligible probability* if the probability of the event is $\mathsf{negl}(\lambda)$, and an event occurs with *overwhelming probability* if its complement occurs with negligible probability.

Statistical Distance. For two distributions X, Y over a finite domain Ω, the *statistical distance* between X and Y is defined by $\Delta(X,Y) \stackrel{\text{def}}{=} \frac{1}{2}\sum_{\omega\in\Omega} |X(\omega) - Y(\omega)|$. If X,Y are distribution ensembles parameterized by the security parameters, we write $X \stackrel{\text{stat}}{\approx} Y$ if the quantity $\Delta(X,Y)$ is negligible. Similarly, we write $X \stackrel{\text{comp}}{\approx} Y$ if they are computationally indistinguishable. We write $\omega \leftarrow X$ to denote that ω is sampled at random according to distribution X. For a finite domain Ω, we write $\omega \stackrel{\$}{\leftarrow} \Omega$ to denote that ω is sampled uniformly from Ω. For a distribution ensemble $\chi = \chi(\lambda)$ over the integers, and integer bounds $B = B(\lambda)$, we say that χ is B-*bounded* if $\Pr_{x\leftarrow\chi(\lambda)}[|x| \le B(\lambda)] = 1$.

For the proof of security of our main construction, we rely on the following lemma, which says that adding large noise "smudges out" any small values.

Lemma 2.1 ([AJLA+12, MW16])**.** *Let $B_1 = B_1(\lambda)$, and $B_2 = B_2(\lambda)$ be positive integers and let $e_1 \in [-B_1, B_1]$ be a fixed integer. Let $e_2 \stackrel{\$}{\leftarrow} [-B_2, B_2]$ be chosen uniformly at random. Then the distribution of e_2 is statistically indistinguishable from that of $e_2 + e_1$ as long as $B_1/B_2 = \mathsf{negl}(\lambda)$.*

2.1 Average-case Lattice Problems

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

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

where $\mathbf{A} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^{n\times m}$, $\mathbf{s} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \chi^m$, and $\mathbf{u} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^m$. For certain B-bounded error distributions χ, the $\mathsf{LWE}(n, m, q, \chi)$ problem is as hard as approximating certain worst-case lattice problems such as GapSVP and SIVP on n-dimensional lattices to within $\tilde{O}(n \cdot q/B)$ factor [Reg09, Pei09, ACPS09, MM11, MP12, BLP+13].

Short Integer Solutions. Let n, m, q, β be positive integers. In the $\mathsf{SIS}(n, m, q, \beta)$ problem, the adversary is given a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and its goal is to find a vector $\mathbf{u} \in \mathbb{Z}_q^m$ with $\mathbf{u} \neq \mathbf{0}$ and $\|\mathbf{u}\| \le \beta$ such that $\mathbf{A} \cdot \mathbf{u} = \mathbf{0}$.

For any $m = \mathsf{poly}(n)$, any $\beta > 0$, and any sufficiently large $q \ge \beta \cdot \mathsf{poly}(n)$, solving $\mathsf{SIS}(n, m, q, \beta)$ problem is as hard as approximating certain worst-case lattice problems such as GapSVP and SIVP on n-dimensional lattices to within $\beta \cdot \mathsf{poly}(n)$ factor [Ajt96, Mic04, MR07, MP13].

2.2 Threshold Secret Sharing

In this work, we work with a (t, N)-threshold secret sharing scheme [Sha79] over a secret k_0 in $\mathbb{Z}_q$. In such a scheme, one constructs key-shares by sampling a uniformly random polynomial $p(z) \in \mathbb{Z}_q[x]$ of degree $t-1$ such that $p(0) = k_0$, and the remaining coefficients are sampled uniformly at random from $\mathbb{Z}_q$. We then define shares $k_i = p(i)$ for $i \in [N]$. For secret shares constructed in this manner, it holds that for any $S \subset [N] \cup \{0\}$ of size t, we have that $p(j) = \sum_{i\in S} \lambda_{i,j}^S \cdot k_i$, where $\lambda_{i,j}^S \in \mathbb{Z}_q$ are efficiently computable Lagrange coefficients defined as follows:

$$\lambda_{i,j}^S = \prod_{\substack{m\in S \\ m\neq i}} \frac{j-m}{i-m}.$$

Furthermore, for any set S' with $|S'| < t$, the set of shares $\{k_i\}_{i\in S'}$ is distributed identically to a uniformly generated set $\{k'_i\}_{i\in S'}$ where $k'_i \stackrel{\$}{\leftarrow} \mathbb{Z}_q$ for all $i \in S'$.

For our purposes, we want the Lagrange coefficients to be "low-norm" values. However, interpreted as elements in $\mathbb{Z}_q$, there cannot be a bound on its norm. Therefore, for our construction, we use the technique

of "clearing out the denominator" [Sho00, ABV+12] where we use the fact that the term $(N!)^2 \cdot \lambda_{i,j}^S$ is an integer. The following lemma follows from a simple combinatorial argument.

Lemma 2.2. *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 $|(N!)^2 \cdot \lambda_{i,j}^S| \leq (N!)^3$.*

Proof. Note that $\lambda_{i,j}^S$ has denominator of the form

$$\prod_{\substack{m \in S \\ m \neq i}} (i - m).$$

The numbers $(i - m)$ lie in the interval $[-(\ell - 1), ..., (\ell - 1)]$, and a single number in $[-(\ell - 1), ..., (\ell - 1)]$ can repeat at most twice. Therefore, the denominator divides $(N!)^2$. For the upper bound, we note that

$$|(N!)^2 \cdot \lambda_{i,j}^S| \leq (N!)^2 |\prod_{\substack{m \in S \\ m \neq i}} (j - m)| \leq (N!)^3.$$

□

2.3 Basic Cryptographic Primitives

As building blocks for the constructions in this work, we use basic cryptographic primitives like PRFs, signature schemes, and public key encryption (PKE) schemes. For completeness, we provide precise definitions of these notions in Appendix A.

3 HE and HS fron LWE

In this section, we review the notion of homomorphic encryption scheme and homomorphic signature scheme from LWE.

3.1 Homomorphic Encryption

In this subsection, we recall the definition of a homomorphic encryption scheme (HE). A (leveled) homomorphic encryption scheme is a tuple of polynomial-time algorithms $\Pi_{\mathsf{HE}} = (\mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Eval}, \mathsf{HE.Dec})$ defined as follows:

- $\mathsf{HE.KeyGen}(1^\lambda, 1^d, 1^k) \to \mathsf{sk}$: On input the security parameter λ, a depth bound d, and a message length k, the key generation algorithm outputs a secret key sk.
- $\mathsf{HE.Enc}(\mathsf{sk}, \mu) \to \mathsf{ct}$: On input a secret key sk and a message $\mu \in \{0,1\}^k$, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{HE.Eval}(C, \mathsf{ct}) \to \mathsf{ct}'$: On input a circuit $C : \{0,1\}^k \to \{0,1\}$ of depth d and a ciphertext ct, the homomorphic evaluation algorithm outputs another ciphertext ct'.
- $\mathsf{HE.Dec}(\mathsf{sk}, \mathsf{ct}') \to b$: On input a secret key sk and a ciphertext ct', the decryption algorithm outputs a bit b.

Correctness. We require that for all $\lambda, d, k \in \mathbb{N}$, $\mathsf{sk} \leftarrow \mathsf{HE.KeyGen}(1^\lambda, 1^d, 1^k)$, $\mu \in \{0,1\}^k$, and boolean circuits $C : \{0,1\}^k \to \{0,1\}$ of depth at most d, we have that

$$\Pr\left[\mathsf{HE.Dec}(\mathsf{sk}, \mathsf{HE.Eval}(C, \mathsf{HE.Enc}(\mathsf{sk}, \mu))) = C(\mu)\right] = 1$$

where the probability is taken over $\mathsf{HE.Enc}$ and $\mathsf{HE.KeyGen}$.

Security. For security, we require standard semantic security. For any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, and for all $d, k = \mathsf{poly}(\lambda)$, there exists a negligible function negl such that

$$\Pr\left[b = b' : \begin{array}{l} \mathsf{sk} \leftarrow \mathsf{HE.KeyGen}(1^\lambda, 1^d, 1^k); \\ (\mu_0, \mu_1, \mathsf{st}_1) \leftarrow \mathcal{A}_1(1^\lambda, 1^d, 1^k); \\ b \xleftarrow{\$} \{0,1\}; \\ \mathsf{ct}_b \leftarrow \mathsf{HE.Enc}(\mathsf{sk}, \mu_b); \\ b' \leftarrow \mathcal{A}_2(\mathsf{ct}_b, \mathsf{st}_1) \end{array}\right] - \frac{1}{2} \leq \mathsf{negl}(\lambda).$$

There are a number of FHE constructions from LWE in the literature [BV14a, BGV12, GSW13, BV14b, ASP14, CM15, MW16, BP16, PS16]. Instead of focusing on a specific construction, we abstract out much of the specific details and work with properties that most of these constructions satisfy. However, for technical reasons in the security proof (Lemma 5.4), we do a simple modification on the existing constructions such that the noise in the FHE ciphertext is a multiple of some integer γ that the key generation algorithm $\mathsf{HE.KeyGen}$ takes in as an additional parameter. We give a high level description of a sample modification of the [GSW13] FHE construction in Appendix B and show that this modification does not break security. We summarize the properties that we need for our main construction in the following theorem.

Theorem 3.1 (HE from LWE)**.** *Fix the security parameter λ, depth bound $d(\lambda)$ and an integer $\gamma \in \mathbb{N}$. Let n, m, q, χ be LWE parameters with $q > 2^{\tilde{O}(d)}$. Then there is an HE scheme $\Pi_{\mathsf{HE}} = (\mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Eval}, \mathsf{HE.Dec})$ for circuits of depth d such that $2^{\tilde{O}(d)} < q$ with the following properties:*

- $\mathsf{HE.KeyGen}$ *takes in an extra parameter γ and outputs a secret key $\mathsf{sk} \in \mathbb{Z}_q^t$ where $t = \mathsf{poly}(\lambda)$.*
- $\mathsf{HE.Enc}$ *takes in a message $\mu \in \{0,1\}^k$ and outputs a ciphertext $\mathsf{ct} \in \{0,1\}^z$ where $z = \mathsf{poly}(k, \lambda, \log q)$.*
- $\mathsf{HE.Eval}$ *takes in a circuit C and a ciphertext $\mathsf{ct} \in \mathbb{Z}_q^z$ and outputs a ciphertext $\mathsf{ct}' \in \mathbb{Z}_q^t$.*
- $\mathsf{HE.Dec}$ *on input sk and ct, we have that*

$$\langle \mathsf{sk}, \mathsf{ct} \rangle \in [(q/2) \cdot \mu - E, (q/2) \cdot \mu + E]$$

  for some $E = \gamma \cdot 2^{\tilde{O}(d)}$ and $\langle \mathsf{sk}, \mathsf{ct} \rangle$ is an integer multiple of γ.
- *Security relies on $\mathsf{LWE}(n, m, q, \chi)$.*

3.2 Homomorphic Signatures

In this section, we recall the definition of homomorphic signature (HS) scheme. We present a simplified definition compared to the full definition as presented in [BF11a, GVW15], which is sufficient for this work.[1] A leveled homomorphic signature scheme is a tuple of polynomial-time algorithms $\Pi_{\mathsf{HS}} = (\mathsf{HS.KeyGen}, \mathsf{HS.Sign}, \mathsf{HS.SignEval}, \mathsf{HS.Verify})$ defined as follows:

- $\mathsf{HS.KeyGen}(1^\lambda, 1^d, 1^N) \to (\mathsf{sk}, \mathsf{vk})$: On input the security parameter λ, a depth bound d, and a date set bound N, the key generation algorithm outputs a signing key sk and a verification key vk.
- $\mathsf{HS.Sign}(\mathsf{sk}, \mathbf{m}) \to \sigma$: On input the signing key sk and a message $\mathbf{m} \in \{0,1\}^N$, the signing algorithm outputs a signature σ.
- $\mathsf{HS.SignEval}(C, \sigma) \to \sigma^*$: On input a circuit $C : \{0,1\}^N \to \{0,1\}$, the signature evaluation algorithm outputs a homomorphically computed signature σ^*.

[1] One omission is the compactness requirement, which states that the size of a homomorphically evaluated signature is independent of the size of the original data set. We note that homomorphic signatures without the compactness requirement and just the unforgeability security requirement is trivial to construct. However, we also require context-hiding security requirement, so our definition is not trivial to achieve. Nevertheless, the construction of [GVW15] achieves all of compactness, unforgeability, and context-hiding, and we leave out the compactness requirement just for simplicity and brevity of the definition.

- $\mathsf{HS.Verify}(\mathsf{vk}, C, y, \sigma^*) \to 0/1$: On input a verification key vk, a circuit C, an output value y and a signature σ^*, the verification algorithm accepts (outputs 1) or rejects (outputs 0).

Correctness. We require that for all $\lambda, N \in \mathbb{N}$, $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{HS.KeyGen}(1^\lambda, 1^N)$, $\mathbf{m} \in \{0,1\}^N$, $\sigma \leftarrow \mathsf{HS.Sign}(\mathsf{sk}, \mathbf{m})$, $C : \{0,1\}^N \to \{0,1\}$ of depth at most d, $y \leftarrow C(\mathbf{m})$,

$$\Pr[\mathsf{HS.Verify}(\mathsf{vk}, C, y, \mathsf{HS.SignEval}(C, \sigma)) = 1] = 1$$

Security. We require two security properties for homomorphic signatures. The first property is the unforgeability requirement, which roughly says that given a homomorphically signed data $\mathbf{m}$, the adversary cannot produce a circuit C and a valid signature $\sigma_{y'}$ for which $C(\mathbf{m}) \neq y'$.

Definition 3.2 (Unforgeability)**.** We say that a homomorphic signature scheme $\Pi_{\mathsf{HS}} = (\mathsf{HS.KeyGen}, \mathsf{HS.Sign}, \mathsf{HS.SignEval}, \mathsf{HS.Verify})$ satisfies unforgebability if for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{uf}}_{\Pi_{\mathsf{HS}},\mathcal{A}}(\lambda) = \Pr[\mathsf{Expt}^{\mathsf{uf}}_{\Pi_{\mathsf{HS}},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

where the experiment $\mathsf{Expt}^{\mathsf{uf}}_{\Pi_{\mathsf{HS}},\mathcal{A}}(\lambda)$ is defined as follows

1. $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{HS.KeyGen}(1^\lambda, 1^N, 1^d)$.
2. $(\mathbf{m}^*, \mathsf{st}_1) \leftarrow \mathcal{A}_1(\mathsf{vk})$.
3. $\sigma \leftarrow \mathsf{HS.Sign}(\mathsf{sk}, \mathbf{m}^*)$.
4. $(C^*, y^*, \sigma^*) \leftarrow \mathcal{A}_2(\sigma, \mathsf{st}_1)$.
5. Output $\mathsf{Verify}^*(\mathsf{vk}, C^*, y^*, \sigma^*)$.

where verification algorithm $\mathsf{Verify}^*(\mathsf{vk}, C^*, y^*, \sigma^*)$ accepts if all of the following conditions hold:

- C^* is a circuit of depth at most d.
- $C^*(\mathbf{m}^*) \neq y^*$.
- $\mathsf{HS.Verify}(\mathsf{vk}, C^*, y^*, \sigma^*)$ accepts.

The second security property that we require is the context-hiding requirement, which roughly says that given a homomorphically computed signature σ^*, an adversary does not learn any information about the original message $\mathbf{m}$ that was signed other than what is already implied by the output $C(\mathbf{m}) = y$.

Definition 3.3 (Context-Hiding)**.** We say that a homomorphic signature scheme $\Pi_{\mathsf{HS}} = (\mathsf{HS.KeyGen}, \mathsf{HS.Sign}, \mathsf{HS.SignEval}, \mathsf{HS.Verify})$ satisfies context-hiding if there exists a simulator $\mathcal{S}^{\mathsf{ch}}$ such that, for any choice of $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{HS.KeyGen}(1^\lambda, 1^d, 1^N)$, $\mathbf{m} \in \{0,1\}^N$, $\sigma \leftarrow \mathsf{HS.Sign}(\mathsf{sk}, \mathbf{m})$, and circuit C, we have that

$$\mathsf{HS.SignEval}(C, \sigma) \overset{\text{stat}}{\approx} \mathcal{S}^{\mathsf{ch}}(\mathsf{sk}, C, C(x)).$$

where the randomness is over the random coins of the simulator and the $\mathsf{HS.SignEval}$ procedure.

Constructions. There are a number of constructions on homomorphic signatures in the literature for various classes of functions that can be supported [BFKW09, GKKR10, BF11b, BF11a, CFW14, GVW15, FMNP16]. In particular, the construction of Gorbunov et al. [GVW15] supports the class of bounded depth circuits from the SIS problem, which is reducible to LWE.

Theorem 3.4 ([GVW15])**.** *Fix a securityu parameters λ and depth bound $d(\lambda)$. Let n, m, q be lattice parameters such that $q > 2^{\tilde{O}(d)}$. There is an* HS *scheme $\Pi_{\mathsf{HS}} = (\mathsf{HS.KeyGen}, \mathsf{HS.Sign}, \mathsf{HS.SignEval}, \mathsf{HS.Verify})$ whose security reduces to $\mathsf{SIS}(n, m, q, \beta)$ for $\beta = 2^{\tilde{O}(d)}$.*

4 Universal Thresholdizer

In this section, we define the notion of a universal thresholdizer scheme. At a high level, the setup of a universal thresholdizer scheme takes in some secret data x and splits it into a number of user shares. A computation can be performed on the secret value x only if enough users in the system compute their own share of the computation. Otherwise, the value x remains hidden. We define our notion to support bounded depth computation, which means that the setup algorithm takes in a bound on the maximum depth on the computation that can be performed on this secret value.

The secret data x can be an encoding of any type of private data in the form of a message, a canonical representation of a Turing machine, etc. and is left unspecified for generality. For intuition, however, the secret value x can be thought of as a key k for some cryptographic function, such as a signature scheme. The evaluation algorithm can be thought of as the evaluation of a cryptographic function $F_m(k) = F(k, m)$ with an input m hardwired inside the circuit. In fact, this will be how we use universal thresholdizers for our applications in Section 6.

Definition 4.1. Fix a security parameter λ and a data space $\mathcal{X}$. A universal thresholdizer scheme is a tuple of algorithms $\Pi_{\mathsf{UT}} = (\mathsf{UT.Setup}, \mathsf{UT.Eval}, \mathsf{UT.Verify}, \mathsf{UT.Combine})$ defined as follows:

- $\mathsf{UT.Setup}(1^\lambda, 1^N, 1^t, 1^d, x) \to (\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[N]})$: On input the security parameter λ, a number of users in the system N, a threshold $t \in [N]$, a bound on the depth d, and a secret $x \in \mathcal{X}$, the setup algorithm generates the public parameters pp and a set of secret keys $\mathsf{sk}_1, ..., \mathsf{sk}_N$ for each user in the system.
- $\mathsf{UT.Eval}(\mathsf{pp}, \mathsf{sk}_i, C) \to y_i$: On input the public parameters pp, a secret key sk_i, and a circuit C, the evaluation algorithm outputs a partial evaluation y_i.
- $\mathsf{UT.Verify}(\mathsf{pp}, C, y_i) \to 0/1$: On input the public parameters pp, a circuit C, and a partial evaluation y_i, the verification algorithm accepts (output 1) or rejects (output 0).
- $\mathsf{UT.Combine}(\mathsf{pp}, \{y_i\}_{i\in S}) \to y$: On input the public parameters pp, and a set of partial evaluations $\{y_i\}_{i\in S}$, the combining algorithm outputs the final evalaution y.

Evaluation Correctness. We say that a universal thresholdizer scheme $\Pi_{\mathsf{UT}} = (\mathsf{UT.Setup}, \mathsf{UT.Eval}, \mathsf{UT.Verify}, \mathsf{UT.Combine})$ satisfies evaluation correctness if the following conditions are true. For all $\lambda, N, t, d \in \mathbb{N}$, $x \in \mathcal{X}$, $(\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^N, 1^t, 1^d, x)$, $S \subseteq [N]$ of size $|S| = t$, and circuit $C : \mathcal{X} \to \{0, 1\}$ of depth at most d, we have that

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

Verification Correctness. We say that a universal thresholdizer scheme $\Pi_{\mathsf{UT}} = (\mathsf{UT.Setup}, \mathsf{UT.Eval}, \mathsf{UT.Verify}, \mathsf{UT.Combine})$ satisfies verification correctness if the following conditions are true. For all $\lambda, N, t, d \in \mathbb{N}$, $x \in \mathcal{X}$, $(\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^N, 1^t, 1^d, x)$, and circuit $C : \mathcal{X} \to \{0, 1\}$ of depth at most d, we have that

$$\Pr[\mathsf{UT.Verify}(\mathsf{pp}, C, \mathsf{UT.Eval}(\mathsf{pp}, \mathsf{sk}_i, C)) = 1] = 1 - \mathsf{negl}(\lambda)$$

for all $i \in [N]$.

Security. We require two security properties for a universal thresholdizer scheme. The first property is privacy, which states that any colluding set of corrupt users below the threshold cannot learn any information about the underlying secret value x. Furthermore, we require that even when given valid partial evaluations of f from the honest parties, the set of corrupt users below the threshold cannot learn information about x other than what is implied by the output $y = f(x)$.

We capture this intuition formally by requiring that there exists a simulator that can simulate all the partial evaluations of the honest users without given access to the actual data x, but given only the output values $f(x)$. Any adversary, given the set of secret keys of corrupt users below the threshold should not be able to distinguish the simulated partial evaluations from the real partial evaluations.

Definition 4.2 (Privacy)**.** We say that a universal thresholdizer scheme $\Pi_{\mathsf{UT}} = (\mathsf{UT.Setup}, \mathsf{UT.Eval}, \mathsf{UT.Verify}, \mathsf{UT.Combine})$ satisfies privacy if there exists a PPT simulator $\mathcal{S}$ such that for all PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\mathsf{Adv}_{\Pi_{\mathsf{UT}},\mathcal{A}}(\lambda) = \left|\Pr[\mathsf{Expt}^{\mathsf{Real}}_{\Pi_{\mathsf{UT}},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Expt}^{\mathsf{Rand}}_{\Pi_{\mathsf{UT}},\mathcal{A}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

where the experiments $\mathsf{Expt}^{\mathsf{Real}}_{\Pi_{\mathsf{UT}},\mathcal{A}}(\lambda)$ and $\mathsf{Expt}^{\mathsf{Rand}}_{\Pi_{\mathsf{UT}},\mathcal{A}}(\lambda)$ are defined as follows:

$\mathsf{Expt}^{\mathsf{REAL}}_{\Pi_{\mathsf{UT}},\mathcal{A}}(\lambda)$:

1. $(x^*, \mathsf{st}_1) \leftarrow \mathcal{A}_1(1^\lambda)$.
2. $(\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^N, 1^t, 1^d, \underline{x^*})$.
3. $(S^*, \mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{pp}, \mathsf{st}_1)$ where $|S^*| = t-1$.
4. $b \leftarrow \mathcal{A}_3^{\mathcal{O}_{\mathsf{Eval}}(\underline{\{\mathsf{sk}_i\}_{i\in[N]}},\cdot,\cdot)}(\{\mathsf{sk}_i\}_{i\in S^*}, \mathsf{st}_2)$.
5. Output b.

$\mathsf{Expt}^{\mathsf{RAND}}_{\Pi_{\mathsf{UT}},\mathcal{A}}(\lambda)$:

1. $(x^*, \mathsf{st}_1) \leftarrow \mathcal{A}_1(1^\lambda)$.
2. $(\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^N, 1^t, 1^d, \underline{0^{|x^*|}})$.
3. $(S^*, \mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{pp}, \mathsf{st}_1)$ where $|S^*| = t-1$.
4. $b \leftarrow \mathcal{A}_3^{\mathcal{S}^{\mathcal{O}_{\mathsf{Sim}}(\cdot)}(\underline{\{\mathsf{sk}_i\}_{i\in S^*}},\cdot,\cdot)}(\{\mathsf{sk}_i\}_{i\in S^*}, \mathsf{st}_2)$.
5. Output b.

where the oracles $\mathcal{O}_{\mathsf{Eval}}(\{\mathsf{sk}_i\}_{i\in[N]}, \cdot, \cdot)$ and $\mathcal{O}_{\mathsf{Sim}}(\cdot)$ are defined as follows

- $\mathcal{O}_{\mathsf{Eval}}(\{\mathsf{sk}_i\}_{i\in[N]}, C, j)$: On input the set of key $\{\mathsf{sk}_i\}_{i\in[N]}$, circuit C, and an index $j \in [N]\backslash S^*$, outputs $\mathsf{UT.Eval}(\mathsf{pp}, \mathsf{sk}_j, C)$.
- $\mathcal{O}_{\mathsf{Sim}}(C)$: On input a circuit C, if there exists a query (C, j) for some $j \in [N]\backslash S^*$ previously made by $\mathcal{A}_3$, the algorithm outputs $C(x^*)$. Otherwise, it outputs $\perp$.

We note that for the privacy notion, we restricted the set of corrupt set S^* to be size $t-1$. This is without loss of generality since the challenger itself can choose additional users to corrupt and bring the set of corrupt users to size $t-1$.

The second security property that we require is robustness, which states that any user in the system cannot convince a verifier with an improperly generated partial decryption.

Definition 4.3 (Robustness)**.** We say that a universal thresholdizer scheme $\Pi_{\mathsf{UT}} = (\mathsf{UT.Setup}, \mathsf{UT.Eval}, \mathsf{UT.Verify}, \mathsf{UT.Combine})$ satisfies robustness if for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{rb}}_{\Pi_{\mathsf{HS}},\mathcal{A}}(\lambda) = \Pr[\mathsf{Expt}^{\mathsf{rb}}_{\Pi_{\mathsf{HS}},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

where the experiment $\mathsf{Expt}^{\mathsf{rb}}_{\Pi_{\mathsf{HS}},\mathcal{A}}(\lambda)$ is defined as follows

1. $(x, \mathsf{st}_1) \leftarrow \mathcal{A}_1(1^\lambda)$.
2. $(\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^N, 1^t, 1^d, x)$.
3. $(C^*, y_i^*, i) \leftarrow \mathcal{A}_2(\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[N]}, \mathsf{st}_1)$.
4. Output $\mathsf{Verify}^*(\mathsf{pp}, C^*, y_i^*, i)$.

where the verification algorithm $\mathsf{Verify}^*(\mathsf{pp}, C, y_i, i)$ accepts if the following conditions hold

- $y_i \neq \mathsf{UT.Eval}(\mathsf{sk}_i, C)$.
- $\mathsf{UT.Verify}(\mathsf{pp}, C, y_i) = 1$.

Remarks. We note that for the definition above, we restricted the circuit C that the evaluation algorithm takes in to be a boolean function of binary output. We do this mainly for simplicity in the next section. This is, of course, without loss of generality because any circuit with output space $\{0,1\}^\alpha$ can be represented with α boolean circuits and α partial evaluations can be done in parallel. Our applications in Section 6 assumes a universal thresholdizer scheme that supports multiple bit outputs.

5 Main Construction

Here, we describe our main construction. As building blocks, we use a homomorphic encryption HE scheme $\Pi_{\mathsf{HE}} = (\mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Eval}, \mathsf{HE.Dec})$ and a homomorphic signature HS scheme $\Pi_{\mathsf{HS}} = (\mathsf{HS.KeyGen}, \mathsf{HS.Sign}, \mathsf{HS.SignEval}, \mathsf{HS.Verify})$. In addition, we use a PRF $F_k : \{0,1\}^* \to [-R, R]$ for some bound R to be specified in Section 5.1.

Fix a security parameter λ. We fix the data space $\mathcal{X}$ to be $\{0,1\}^*$. For simplicity of notation, we denote $\eta = (N!)^2$. We construct a universal thresholdizer scheme $\Pi_{\mathsf{UT}} = (\mathsf{UT.Setup}, \mathsf{UT.Eval}, \mathsf{UT.Verify}, \mathsf{UT.Combine})$ as follows:

- $\mathsf{UT.Setup}(1^\lambda, 1^N, 1^t, 1^d, \mathbf{x}) \to (\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[N]})$: On input the security parameter λ, a number of users N, a threshold $t \in [N]$, a depth bound d and data $\mathbf{x}$, the setup algorithm generates the HE secret key $\mathsf{hesk}_0 \leftarrow \mathsf{HE.KeyGen}(1^\lambda, 1^{d_{\mathsf{HE}}}, 1^k, \eta^2)$[2] and the HS key $(\mathsf{hsvk}, \mathsf{hssk}) \leftarrow \mathsf{HS.KeyGen}(1^\lambda, 1^{d_{\mathsf{HS}}}, 1^N)$. Then, it encrypts $\mathbf{x}$ with the HE scheme $\mathsf{ct} \leftarrow \mathsf{HE.Enc}(\mathsf{hesk}_0, \mathbf{x})$.

  Then, the setup algorithm divides the HE secret key $\mathsf{hssk}_0 \in \mathbb{Z}_q^n$ into N shares $\mathsf{hesk}_1, ..., \mathsf{hesk}_N$ using a (t, N)-threshold secret sharing scheme by secret sharing each components of the vector hssk_0 separately. It also samples N independently sampled PRF keys $\mathsf{prfk}_1, ..., \mathsf{prfk}_N$. Finally, for each $i = 1, ..., N$, the setup algorithm signs the tuple $\mathbf{m}_i = (\mathsf{hesk}_i, \mathsf{prfk}_i)$ using the HS scheme $\sigma_i \leftarrow \mathsf{HS.Sign}(\mathsf{hssk}, \mathbf{m}_i)$. It sets

$$\mathsf{pp} = (\mathsf{ct}, \mathsf{hsvk}) \qquad \mathsf{sk}_i = (\mathsf{hesk}_i, \mathsf{prfk}_i, \sigma_i) \quad \forall i \in [N].$$

- $\mathsf{UT.Eval}(\mathsf{pp}, \mathsf{sk}_i, C) \to y_i$: On input the public parameter pp, a secret key sk_i, and a circuit C, the evaluation algorithm homomorphically evaluates C on the ciphertext $\mathsf{ct}' \leftarrow \mathsf{HE.Eval}(C, \mathsf{ct})$. Then, it evaluates the PRF on the circuit C, producing $e_i \leftarrow F_{\mathsf{prfk}_i}(C)$, and computes

$$\tilde{y}_i = \langle \mathsf{ct}', \mathsf{hesk}_i \rangle + \eta \cdot e_i.$$

  Then, using the homomorphic signature scheme, the evaluation algorithm homomorphically evaluates the signature $\sigma_i \leftarrow \mathsf{HS.SignEval}(g_{C,\mathsf{ct}'}, \sigma_i)$ where the circuit $g_{C,\mathsf{ct}'}$ is defined as follows:

$$g_{C,\mathsf{ct}'}(\mathsf{hesk}_i, \mathsf{prfk}_i) = \langle \mathsf{ct}', \mathsf{hesk}_i \rangle + \eta \cdot F_{\mathsf{prfk}_i}(C).$$

  It outputs $y_i = (\tilde{y}_i, \sigma_i)$ as the partial evaluation.

- $\mathsf{UT.Verify}(\mathsf{pp}, C, y_i) \to 0/1$: On input the public parameters $\mathsf{pp} = (\mathsf{ct}, \mathsf{hsvk})$, a function C, and a partial evaluation $y_i = (\tilde{y}_i, \sigma_i)$, the verification algorithm outputs $\mathsf{HS.Verify}(\mathsf{hsvk}, g_{C,\mathsf{ct}'}, \tilde{y}_i, \sigma_i)$ where $\mathsf{ct}' \leftarrow \mathsf{HE.Eval}(C, \mathsf{ct})$.

- $\mathsf{UT.Combine}(\mathsf{pp}, \{y_i\}_{i\in S}) \to y$: On input the public parameters pp, and a set of partial evaluations $\{y_i = (\tilde{y}_i, \sigma_i)\}_{i\in S}$, the combining algorithm computes the Lagrange coefficients $\lambda_{i,0}^S$ for all $i \in S$ and then computes

$$\tilde{y} = \sum_{i\in S} \lambda_{i,0}^S \cdot \tilde{y}_i.$$

  It outputs 0 if $\tilde{y} \in [-q/4, q/4]$ and outputs 1 otherwise.

5.1 Parameters

Let d be the bound on the depth that the scheme supports. First, we set the PRF range R to be big enough to "smudge out" any information that is contained in the HE noise. For this, we note that for a circuit of depth d, the noise from HE decryption is at most $\eta^2 \cdot 2^{\tilde{O}(d)}$ (Theorem 3.1). We set $R = \eta^{1.5} \cdot 2^{\tilde{O}(d)+\omega(\log\lambda)}$. Next, we set the HE parameter d_{HE} so that the HE modulus is at least $8N \cdot \eta^{1.5} \cdot R$ for it to handle the growth

[2]Recall that the last parameter denotes the scalar multiplied to the noise (see Section 3.1).

of the noise. Finally, we set the homomorphic signatures d_{HS} to support the depth of the circuit $g_{C,\mathsf{ct}}$, which includes the inner product of the ciphertext ct^* with the secret keys hesk_i, as well as the depth of the PRF evaluation F_{prfk_i}.

Approximation Factors. The approximation factors for worst-case lattice problems that the parameters above translates to largely depends on the depth bound d on the circuit and the depth of the PRF used.[3] For most of the interesting applications of universal thresholdizers, the depth d is $\mathsf{poly}(\lambda)$ which results in subexponential approximation factors ($2^{O(n^{1/c})}$ for some constant c) due to the approximation factor implied by HE. For logarithmic depth (NC1 circuits), we can base our construction on super polynomial approximation factors ($n^{\omega(1)}$). This is done by using the HE construction of [BV14b], which results in polynomial approximation factor and the PRF constructions of [BPR12, BP14], which relies on super polynomial approxmation factors.[4] Note that a lattice-based GGM based PRF ([GGM86]) can be used, which relies on polynomial approximation factors, but now, since the depth of the PRF increases, the homomorphic signatures must be based on subexponential approximation factors.

5.2 Correctness

We now show correctness of the scheme above.

Theorem 5.1. *The universal thresholdizer scheme above with parameters instantiated as in Section 5.1 satisfies the evaluation correctness as defined in Section 4.*

Proof. The combining algorithm computes $\tilde{y}$ as follows

$$\begin{aligned}
\tilde{y} &= \sum_{i \in S} \lambda_{i,0}^S \cdot \tilde{y}_i \\
&= \sum_{i \in S} \lambda_{i,0}^S \cdot (\langle \mathsf{ct}^*, \mathsf{hesk}_i \rangle + \eta \cdot e_i) \\
&= \langle \mathsf{ct}^*, \mathsf{hesk}_0 \rangle + \sum_{i \in S} \eta \cdot \lambda_{i,0}^S \cdot e_i \\
&= (q/2) \cdot C(\mathbf{x}) + \tilde{e}_C + \sum_{i \in S} \eta \cdot \lambda_{i,0}^S \cdot e_i
\end{aligned}$$

where the last equality follows from the correctness of the FHE scheme. Now, it is sufficient to show that the error term $|\tilde{e}_C + \sum_{i \in S} \eta \cdot \lambda_{i,0}^S \cdot e_i| \leq 2 \cdot N \cdot \eta^{1.5} \cdot R$ is at most $q/4$. The parameters in Section 5.1, are set precisely to satisfy this bound and the theorem follows. □

The verification correctness follows immediately from the correctness of the homomorphic signature scheme.

Theorem 5.2. *The universal thresholdizer scheme above with parameters instantiated as in Section 5.1 satisfies the verification correctness as defined in Section 4.*

5.3 Security

In this section, we show security of the construction above.

Theorem 5.3. *The construction above with parameters instantiated as in Section 5.1 satisfies privacy as defined in Definition 4.2.*

Proof. We proceed through a series of hybrids. The first hybrid experiment H_0 represents the real experiment $\mathsf{Expt}^{\mathsf{REAL}}_{\Pi_{\mathsf{UT}},\mathcal{A}}(\lambda)$ and the hybrid experiment H_1 represents the ideal experiment $\mathsf{Expt}^{\mathsf{RAND}}_{\Pi_{\mathsf{UT}},\mathcal{A}}(\lambda)$ in Definition 4.2.

[3] This is assuming that we base the security of the construction solely on lattice assumptions.
[4] Note that these PRF constructions are conjectured to be hard even for polynomial approximation factors.

- **Hybrid** H_0: This is the real security game. Upon receiving the secret data $\mathbf{x}^*$ from the adversary, the challenger generates the public parameters pp and the secret keys $\{\mathsf{sk}_i\}_{i\in[N]}$ using $\mathsf{UT.Setup}$ and sends pp to the adversary. Upon the adversary's commitment to the corrupt set of users S^*, the challenger sends the secret keys $\{\mathsf{sk}_i\}_{i\in S^*}$. Then, for each query (C, j) that the adversary makes, the challenger computes the partial decryptions $\mathsf{UT.Eval}(\mathsf{sk}_j, C)$ and feeds it to the adversary.

- **Hybrid** H_1: This hybrid game is the same as H_0 except that now, the challenger simulates the homomorphic signatures instead of honestly evaluating on the original signatures. Specifically, for each query (C, j) that the adversary makes, the challenger computes the partial decryption of FHE honestly by computing
$$\tilde{y}_i = \langle \mathsf{ct}', \mathsf{hesk}_i \rangle + \eta \cdot F_{\mathsf{prfk}_i}(C)$$
as in the real evaluation $\mathsf{UT.Eval}(\mathsf{sk}_j, C)$, but to create the homomorphic signatures σ_i, the challenger runs the context-hiding simulator $\mathcal{S}^{\mathsf{ch}}$ of the homomorphic signature scheme (Definition 3.3) by computing $\sigma_i^* \leftarrow \mathcal{S}^{\mathsf{ch}}(\mathsf{hssk}, g_{C,\mathsf{ct}'}, \tilde{y}_i)$.

  We note that by the context-hiding property of the underlying homomorphic signature scheme, the hybrids H_0 and H_1 are statistically indistinguishable. We note that in H_1, the challenger does not use the PRF keys prfk_i or hesk_i to answer the adversary.

- **Hybrid** H_2: This hybrid game is the same as H_1 except that now, to answer the partial evaluation queries by the adversary the challenger samples a uniformly random value $u_i \xleftarrow{\$} [-R, R]$ and computes the partial evaluation as
$$\tilde{y}_i = \langle \mathsf{ct}', \mathsf{sk}_i \rangle + \eta \cdot u_i$$
instead of evaluating the PRF F_{prfk_i}. By the security of the underlying PRF F_{prfk_i}, we have that the hybrid experiments H_1 and H_2 are indistinguishable.

- **Hybrid** H_3: This hybrid game is the same as H_2 except that now, the challenger simulates the partial evaluations knowing only the $t-1$ FHE decryption keys $\{\mathsf{hesk}_i\}_{i\in S^*}$. Specifically, instead of generating the FHE secret key $\mathsf{hesk}_0 \leftarrow \mathsf{HE.KeyGen}(1^\lambda, 1^{d_{\mathsf{HS}}}, 1^k)$ and dividing it using a (t, N)-threshold secret sharing scheme, the challenger generates $t-1$ independently sampled secret keys $\mathsf{hesk}_i \leftarrow \mathsf{HS.KeyGen}(1^\lambda, 1^{d_{\mathsf{HS}}}, 1^N)$ for $i \in S^*$ and presents them to the adversary.

  Then, for each of the queries (C, j) that the adversary makes, the challenger answers the adversary as follows. The challenger first computes the coefficients $\lambda_{i,j}^{\tilde{S}^*}$ for $i \in \tilde{S}^*$ where $\tilde{S}^* = S^* \cup \{0\}$. Then, it samples a uniformly random value $u_j \leftarrow [-R, R]$ and computes
$$\tilde{y}_j = \lambda_{0,j}^{\tilde{S}^*} \cdot \frac{q}{2} \cdot C(\mathbf{x}) + \sum_{i\in S^*} (\lambda_{i,j}^{\tilde{S}^*} \cdot \langle \mathsf{ct}', \mathsf{hesk}_i \rangle) + \eta \cdot u_j.$$

  In Lemma 5.4 below, we show that the view of the adversary in hybrid experiments H_2 and H_3 are statistically instistinguishable. We note that in H_3, the challenger does not use decryption key hesk_0 in any of the simulation.

- **Hybrid** H_4: This hybrid game is the same as H_3, but the challenger runs the setup with respect to the all zeros string $0^{|\mathbf{x}|}$ instead of $\mathbf{x}$.

  Note that the challenger in both H_3 and H_4 does not use hesk_0 in any part of the simulation. The key shares $\{\mathsf{hesk}_i\}_{i\in S^*}$ are also sampled independently from hesk_0. Therefore, by semantic security of the HE scheme, the games H_3 and H_4 are computationally indistinguishable.

What remains to show is that the hybrid experiments H_2 and H_3 are indistinguishable.

Lemma 5.4. *The hybrid experiments* H_2 *and* H_3 *above are statistically indistinguishable.*

Proof. In H_3, the challenger answers an adversary's query (C, j) by computing

$$\begin{aligned}
\tilde{y}_j &= \lambda_{0,j}^{\tilde{S}^*} \cdot \frac{q}{2} \cdot C(\mathbf{x}) + \sum_{i \in S^*} (\lambda_{i,j}^{\tilde{S}^*} \cdot \langle \mathsf{ct}', \mathsf{hesk}_i \rangle) + \eta \cdot u_j \\
&= \lambda_{0,j}^{\tilde{S}^*} \cdot (\langle \mathsf{ct}', \mathsf{hesk}_0 \rangle + \tilde{e}) + \sum_{i \in S^*} (\lambda_{i,j}^{\tilde{S}^*} \cdot \langle \mathsf{ct}', \mathsf{hesk}_i \rangle) + \eta \cdot u_j \\
&= \langle \mathsf{ct}', \mathsf{hesk}_j \rangle + \lambda_{0,j}^{\tilde{S}^*} \cdot \tilde{e} + \eta \cdot u_j \\
&= \langle \mathsf{ct}', \mathsf{hesk}_j \rangle + \lambda_{0,j}^{\tilde{S}^*} \cdot (\eta^2 \cdot \tilde{e}') + \eta \cdot u_j \\
&= \langle \mathsf{ct}', \mathsf{hesk}_j \rangle + \eta \left(\underbrace{\eta \cdot \lambda_{0,j}^{\tilde{S}^*} \cdot \tilde{e}' + u_j}_{\tilde{u}_j} \right).
\end{aligned}$$

The second equality follows from the correctness of the HE scheme; namely, that $\langle \mathsf{ct}', \mathsf{hesk}_0 \rangle = \frac{q}{2} \cdot C(\mathbf{x}) - \tilde{e}$ for some decryption error $\tilde{e}$. We use the fact that this decryption error is an integer multiple of η^2 (Theorem 3.1) for the second last equality, where we denote $\tilde{e} = \eta^2 \cdot \tilde{e}'$. Now, the HE decryption noise $\tilde{e}'$ is bounded by $2^{\tilde{O}(d)}$ and therefore, by Lemma 2.2, the term $\eta \cdot \lambda_{0,j}^{S^*} \cdot \tilde{e}'$ is at most $\eta^{1.5} \cdot 2^{\tilde{O}(d)}$. Since the term u_j is sampled uniformly from $[-R, R]$ where $R = \eta^{1.5} \cdot 2^{\tilde{O}(d)+\omega(\log \lambda)}$, by Lemma 2.1, the component $\eta \cdot \lambda_{0,j}^{S^*} \cdot \tilde{e}'$ is "smudged out" by u_j and therefore, $\tilde{u}_j$ is statistically indistinguishable from uniform in $[-R, R]$. We note that this is precisely the distribution of $\tilde{y}_j$ in H_2. Therefore, H_2 and H_3 are statistically indistinguishable and this concludes the proof of the lemma. □

This concludes of the proof of Theorem 5.3. □

Theorem 5.5. *The construction above with parameters instantiated as in Section 5.1 satisfies robustness as defined in Definition 4.3.*

Proof. The robustness property follows from the unforgeability of the underlying homomorphic signatures in a very straightforward way. For an adversary $\mathcal{A}$ that breaks the robustness game of Definition 4.3, we build a simulator $\mathcal{B}$ that interacts with a homomorphic signatures challenger and breaks the unforgeability game of Definition 3.2.

The simulator $\mathcal{B}$ first receives the verification key hsvk from the HS challenger. Then, on the value $\mathbf{x}^*$ that the adversary $\mathcal{A}$ chooses to attack, the simulator $\mathcal{B}$ runs the thresholdizer setup $\mathsf{UT.Setup}(1^\lambda, 1^N, 1^t, 1^d, \mathbf{x}^*)$ honestly and generates $\mathsf{hesk}_1, ..., \mathsf{hesk}_N$ and PRF keys $\mathsf{prfk}_1, ..., \mathsf{prfk}_N$. However, to generate the signatures $\sigma_1, ..., \sigma_N$, $\mathcal{B}$ queries the homomorphic signatures challenger on the message $(\mathsf{hesk}_1, \mathsf{prfk}_1)|...|(\mathsf{hesk}_N, \mathsf{prfk}_N)$ to get $(\sigma_1, ..., \sigma_N)$. Now, when $\mathcal{A}$ outputs a fake partial decryption (C^*, y_i^*), the simulator $\mathcal{B}$ parses $y_i^* = (\tilde{y}_i^*, \sigma_i^*)$ and submits $(g_{C^*,\mathsf{ct}^*}, \tilde{y}_i^*, \sigma_i^*)$ as the forgery to the homomorphic signatures challenger where $\mathsf{ct}^* \leftarrow \mathsf{HE.Eval}(\mathsf{ct}, C^*)$.

We note that if $\mathcal{A}$ wins the robustness game, then we have that $\tilde{y}_i^* \neq g_{C^*,\mathsf{ct}^*}(\mathsf{hesk}_i, \mathsf{prfk}_i)$, but $\mathsf{UT.Verify}(\mathsf{pp}, C^*, \tilde{y}_i^*)$ accepts. Since $\mathsf{UT.Verify}$ simply checks the signature $\mathsf{HS.Verify}(\mathsf{hsvk}, g_{C,\mathsf{ct}^*}, y_i^*, \sigma_i^*)$ this is a valid forgery of the homomorphic signature scheme. Therefore, the simulator $\mathcal{B}$ breaks the unforgebability game of homomorphic signatures with the same advantage as $\mathcal{A}$.

□

6 Applications

In this section, we describe our applications of a universal thresholdizer scheme. We describe just two of the applications: threshold signatures and CCA secure threshold PKE. However, these applications, despite having been open in the lattice setting, serve only as proof of concept on how to use a universal thresholdizer scheme.[5] In fact, the concept of a universal thresholdizer is very general and can be used as a tool to construct

[5]By open, we mean constructions that are compact and non-interactive. For comparisons to previous works, we refer the readers to Section 1.2.

a variety of additional notions in threshold cryptography like *distributed PRFs* and *threshold identity based encryption* (IBE), and even thresholdize the key generation mechanisms of *attribute based encryption*, *predicate encryption*, and *functional encryption*. We note that a threshold IBE that satisfies *verifiable key generation* (robustness) implies both threshold signatures and CCA secure threshold PKE. However, we choose to describe threshold signatures and CCA secure threshold PKE separately for concreteness and for proof of concept.

For each subsection, we start by first defining the notion that we are constructing, followed by the description of the construction using a universal thresholdizer and then the security proof. We note that since the majority of the work is already done in proving the security of the universal thresholdizer scheme, the security proofs in each of the constructions can be made very simple. For precise definitions of basic primitives like PRFs, signature schemes, and PKEs, we refer the readers to Appendix A.

6.1 Threshold Signatures

In this section, we construct threshold signatures from universal thresholdizers. In a threshold signature scheme, the signing key of a signer is divided into different key shares and is distributed to multiple of signers. When signing a given 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.

6.1.1 Definition

A threshold signature (TS) scheme for a message space $\mathcal{M}$, is a tuple of efficient algorithms $\Pi_{\mathsf{TS}} = (\mathsf{TS.Setup}, \mathsf{TS.PartSign}, \mathsf{TS.PartSignVerify}, \mathsf{TS.Combine}, \mathsf{TS.Verify})$ defined as follows:

- $\mathsf{TS.Setup}(1^\lambda, 1^N, 1^t) \to (\mathsf{pp}, \mathsf{vk}_{\mathsf{ver}}, \{\mathsf{sk}_i\}_{i\in[N]})$: On input the security parameters λ, the number of signing authorities N, and a threshold $t \in [N]$, the setup algorithm outputs the public parameters pp, a signature verification key vk, and a set of partial signing keys $\{\mathsf{sk}_i\}_{i\in[N]}$.
- $\mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, \mathbf{m}) \to \sigma_i$: On input the public parameters pp, a partial signing key sk_i, and a message $\mathbf{m} \in \mathcal{M}$, the partial signing algorithm outputs a partial signature σ_i.
- $\mathsf{TS.PartSignVerify}(\mathsf{pp}, \mathbf{m}, \sigma_i) \to 0/1$: On input the public parameters pp, a message $\mathbf{m}$, and a partial signature σ_i, the partial signature verification algorithm accepts or rejects.
- $\mathsf{TS.Combine}(\mathsf{pp}, \{\sigma_i\}_{i\in S}) \to \sigma$: On input the public parameters pp, a set of partial signatures $\{\sigma_i\}_{i\in S}$, the signature combining algorithm outputs a full signature σ.
- $\mathsf{TS.Verify}(\mathsf{vk}_{\mathsf{ver}}, \mathbf{m}, \sigma) \to 0/1$: On input a signature verification key $\mathsf{vk}_{\mathsf{ver}}$, a message $\mathbf{m} \in \mathcal{M}$, and a signature σ, the verification algorithm accepts or rejects.

In the definition above, we decouple the public parameters pp and the signature verification key $\mathsf{vk}_{\mathsf{ver}}$. The public parameters pp is to be shared among the partial signers of the system and is not needed to verify the final signature. We do this mainly for compactness. In our construction in Section 6.1.2, the size of pp scales linearly in the number of partial signers while the size of the vk remains independent of the number of signers.

Combining Correctness. We require that for any $\lambda, N, t \in \mathbb{N}$ such that $t \le N$, $S \subset [N]$, $(\mathsf{pp}, \mathsf{vk}_{\mathsf{ver}}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{TS.Setup}(1^\lambda, 1^N, 1^t)$, $\mathbf{m} \in \mathcal{M}$, $\sigma_i \leftarrow \mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, \mathbf{m})$ for $i = 1, ..., N$, we have that

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

Partial Verification Correctness. We require that for any $\lambda, N, t \in \mathbb{N}$ such that $t \le N$, $(\mathsf{pp}, \mathsf{vk}_{\mathsf{ut}}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{TS.Setup}(1^\lambda, 1^N, 1^t)$, any $\mathbf{m} \in \mathcal{M}$, we have that

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

for all $i \in [N]$.

Security. There are three security requirements for a threshold signature scheme. The first is the standard unforgeability notion, which states that an adversary who holds less than t signing keys cannot forge a valid signature.

Definition 6.1 (Unforgeability)**.** Fix a security parameter λ. We say that a threshold signature scheme $\Pi_{\mathsf{TS}} = (\mathsf{TS.Setup}, \mathsf{TS.PartSign}, \mathsf{TS.PartSignVerify}, \mathsf{TS.Combine}, \mathsf{TS.Verify})$ satisfies unforgeability if for any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{uf}}_{\Pi_{\mathsf{TS}},\mathcal{A}}(\lambda) = \Pr[\mathsf{Expt}^{\mathsf{uf}}_{\Pi_{\mathsf{TS}},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

where the experiment $\mathsf{Expt}^{\mathsf{uf}}_{\Pi_{\mathsf{TS}},\mathcal{A}}(\lambda)$ is defined as follows:

1. $(\mathsf{pp}, \mathsf{vk}_{\mathsf{ver}}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{TS.Setup}(1^\lambda, 1^N, 1^t)$.
2. $(S^*, \mathsf{st}_1) \leftarrow \mathcal{A}_1(\mathsf{pp}, \mathsf{vk}_{\mathsf{ver}})$ where $|S^*| = t - 1$.
3. $(\mathbf{m}^*, \sigma^*) \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{Sign}}(\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[N]}, \cdot, \cdot)}(\{\mathsf{sk}_i\}_{i\in S^*}, \mathsf{st}_1)$.
4. Output $\mathsf{Verify}^*(\mathsf{vk}_{\mathsf{ver}}, \mathbf{m}^*, \sigma^*)$.

where the algorithms $\mathcal{O}_{\mathsf{Sign}}$ and Verify^* are defined as follows:

- $\mathcal{O}_{\mathsf{Sign}}(\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[N]}, \mathbf{m}, i)$: On input the set of signing keys $\{\mathsf{sk}_i\}_{i\in[N]}$, a message $\mathbf{m}$ and an index $i \in [N]\backslash S^*$, it outputs the partial signature $\sigma_i \leftarrow \mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, \mathbf{m})$.
- $\mathsf{Verify}^*(\mathsf{vk}_{\mathsf{ver}}, \mathbf{m}, \sigma)$: On input the signature verification key $\mathsf{vk}_{\mathsf{ver}}$, message $\mathbf{m}$, and a signature σ, it accepts if the following conditions are true:
  - $\mathsf{TS.Verify}(\mathsf{vk}_{\mathsf{ver}}, \mathbf{m}, \sigma) = 1$.
  - $\mathbf{m}$ was not previously queried to $\mathcal{O}_{\mathsf{Sign}}$.

The second security property is robustness, which says that an adversary cannot forge an improperly generated partial signature.

Definition 6.2 (Robustness)**.** Fix a security parameter λ. We say that a threshold signature scheme $\Pi_{\mathsf{TS}} = (\mathsf{TS.Setup}, \mathsf{TS.PartSign}, \mathsf{TS.PartSignVerify}, \mathsf{TS.Combine}, \mathsf{TS.Verify})$ satisfies robustness if for any efficient adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{rb}}_{\Pi_{\mathsf{TS}},\mathcal{A}} = \Pr[\mathsf{Expt}^{\mathsf{rb}}_{\Pi_{\mathsf{TS}},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

where the experiment $\mathsf{Expt}^{\mathsf{rb}}_{\Pi_{\mathsf{TS}},\mathcal{A}}(\lambda)$ is defined as follows:

1. $(\mathsf{pp}, \mathsf{vk}_{\mathsf{ver}}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{TS.Setup}(1^\lambda, 1^N, 1^t)$.
2. $(\mathbf{m}^*, \sigma_i^*, i) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{vk}_{\mathsf{ver}}, \{\mathsf{sk}_i\}_{i\in[N]})$.
3. Output $\mathsf{PartSignVerify}^*(\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[N]}, \mathbf{m}^*, \sigma_i^*, i)$.

where the algorithm $\mathsf{PartSignVerify}^*$ is defined as follows:

- $\mathsf{PartSignVerify}^*(\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[N]}, \mathbf{m}, \sigma_i, i)$: On input the public parameters pp, message $\mathbf{m}$, a partial signature σ_i, and an index i, it accepts if the following conditions hold:
  - $\mathsf{TS.PartSignVerify}(\mathsf{pp}, \mathbf{m}, \sigma_i) = 1$.
  - $\sigma_i \neq \mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, \mathbf{m})$.

The last security property that we require is the notion of anonymity, which says that a combined signature does not reveal any information about the set of signers that were involved in creating the final signature.

Definition 6.3 (Anonymity)**.** We say that a threshold signature scheme $\Pi_{\mathsf{TS}} = (\mathsf{TS.Setup}, \mathsf{TS.PartSign}, \mathsf{TS.PartSignVerify}, \mathsf{TS.Combine}, \mathsf{TS.Verify})$ satisfies anonymity if for all $\lambda, N, t \in \mathbb{N}$ such that $t \leq N$, $(\mathsf{pp}, \mathsf{vk}_{\mathsf{ver}}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{TS.Setup}(1^\lambda, N, t)$, $\mathbf{m} \in \mathcal{M}$, $S_0, S_1 \subseteq [N]$ such that $|S_0| = |S_1| = t$, we have that

$$\Pr\left[\begin{array}{c}\mathsf{TS.Combine}(\mathsf{pp}, \{\mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, \mathbf{m})\}_{i\in S_0}) \neq \\ \mathsf{TS.Combine}(\mathsf{pp}, \{\mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, \mathbf{m})\}_{i\in S_1})\end{array}\right] \leq \mathsf{negl}(\lambda)$$

Compactness. In addition to correctness and security, we require an additional property of compactness to threshold signatures. This means that the size of the verification key as well as the final signature produced by the combining algorithm is polynomial only in the security parameter and is independent of the number of users N or the threshold t.

Definition 6.4 (Compactness)**.** We say that a threshold signature scheme $\Pi_{\mathsf{TS}} = (\mathsf{TS.Setup}, \mathsf{TS.PartSign}, \mathsf{TS.PartSignVerify}, \mathsf{TS.Combine}, \mathsf{TS.Verify})$ satisfies compactness if there exist polynomials $s_1(\cdot)$ and $s_2(\cdot)$ such that for every $\lambda, N, t \in \mathbb{N}$ such that $t \leq N$, $(\mathsf{pp}, \mathsf{vk}_{\mathsf{ver}}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{TS.Setup}(1^\lambda, 1^N, 1^t)$, $\mathbf{m} \in \mathcal{M}$, $\sigma_i \leftarrow \mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, \mathbf{m})$ for $i = 1, ..., N$, $S \subset [N]$ of size t, $\sigma \leftarrow \mathsf{TS.Combine}(\mathsf{pp}, \{\sigma_i\}_{i\in S})$, we have that $|\sigma| \leq s_1(\lambda)$ and $|\mathsf{vk}_{\mathsf{ver}}| \leq s_2(\lambda)$.

6.1.2 Construction

Here, we describe our construction for a threshold signature scheme. As building blocks, we use a universal thresholdizer scheme $\Pi_{\mathsf{UT}} = (\mathsf{UT.Setup}, \mathsf{UT.Eval}, \mathsf{UT.Verify}, \mathsf{UT.Combine})$ and a regular signature scheme $\Pi_{\mathsf{S}} = (\mathsf{S.KeyGen}, \mathsf{S.Sign}, \mathsf{S.Verify})$. For the construction, we assume that the signing algorithm $\mathsf{S.Sign}$ is deterministic. This is without loss of generality since any randomized signature scheme can be derandomized (i.e. using PRFs).

Fix a security parameter λ. We construct a threshold signature scheme $\Pi_{\mathsf{TS}} = (\mathsf{TS.Setup}, \mathsf{TS.PartSign}, \mathsf{TS.PartSignVerify}, \mathsf{TS.Combine}, \mathsf{TS.Verify})$ as follows:

- $\mathsf{TS.Setup}(1^\lambda, 1^N, 1^t) \to (\mathsf{pp}, \mathsf{vk}_{\mathsf{ver}}, \{\mathsf{sk}_i\}_{i\in[N]})$: On input the security parameter λ, the number of signing authorities N, and a threshold $t \in [N]$, the setup algorithm first generates the keys for the signature scheme $(\mathsf{ssk}, \mathsf{svk}) \leftarrow \mathsf{S.KeyGen}(1^\lambda)$. Then, it instantiates a universal thresholdizer scheme with respect to the signing key ssk, $(\mathsf{utpp}, \{\mathsf{utsk}_i\}_{i\in[N]}) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^N, 1^t, 1^d, \mathsf{ssk})$ where d is the depth of the signing algorithm $\mathsf{S.Sign}$. Then, it sets
$$\mathsf{pp} = \mathsf{utpp}, \qquad \mathsf{vk}_{\mathsf{ver}} = \mathsf{svk}, \qquad \mathsf{sk}_i = \mathsf{utsk}_i \ \ \forall i \in [N]$$
- $\mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, \mathbf{m}) \to \sigma_i$: On input the public parameters $\mathsf{pp} = \mathsf{utpp}$, a partial signing key $\mathsf{sk}_i = \mathsf{utsk}_i$, and a message $\mathbf{m} \in \mathcal{M}$, the partial signing algorithm outputs $\sigma_i \leftarrow \mathsf{UT.Eval}(\mathsf{utpp}, \mathsf{utsk}_i, C_{\mathbf{m}})$ where the circuit $C_{\mathbf{m}}$ is defined as follows
$$C_{\mathbf{m}}(\mathsf{ssk}) = \mathsf{S.Sign}(\mathsf{ssk}, \mathbf{m}).$$
- $\mathsf{TS.PartSignVerify}(\mathsf{pp}, \mathbf{m}, \sigma_i) \to 0/1$: On input the public parameters $\mathsf{pp} = \mathsf{utpp}$, a message $\mathbf{m}$, and a partial signature σ_i, the partial signature verification algorithm outputs $\mathsf{UT.Verify}(\mathsf{utpp}, C_{\mathbf{m}}, \sigma_i)$.
- $\mathsf{TS.Combine}(\mathsf{pp}, \{\sigma_i\}_{i\in S}) \to \sigma$: On input the public parameters $\mathsf{pp} = \mathsf{utpp}$, and a set of signatures $\{\sigma_i\}_{i\in S}$, the signature combining algorithm outputs $\sigma \leftarrow \mathsf{UT.Combine}(\mathsf{utpp}, \{\sigma_i\}_{i\in S})$.
- $\mathsf{TS.Verify}(\mathsf{vk}_{\mathsf{ver}}, \mathbf{m}, \sigma) \to 0/1$: On input a signature verification key $\mathsf{vk}_{\mathsf{ver}} = \mathsf{svk}$, a message $\mathbf{m} \in \mathcal{M}$, and a signature σ, the verification algorithm outputs $\mathsf{S.Verify}(\mathsf{svk}, \mathbf{m}, \sigma)$.

Correctness. The combining correctness of the scheme Π_{TS} above follows directly from the evaluation correctness of the universal thresholdizer scheme Π_{UT} and the correctness of the underlying signature scheme Π_{S}. The partial verification correctness of Π_{TS} follows directly from the verification correctness of Π_{UT}.

6.1.3 Security

We now show security of the threshold signature scheme above.

Theorem 6.5. *The threshold signature scheme above satisfies the unforgeability notions as defined in Definition 6.1 assuming that the underlying universal thresholdizer scheme* Π_{UT} *satisfies privacy (Definition 4.2) and the signature scheme* Π_{S} *satisfies unforgeability.*

Proof. We start with the real game **Hybrid** H_0 and proceed to an altered game **Hybrid** H_1 where we use the simulator $\mathcal{S}$ for the universal thresholdizer scheme (Definition 4.2) to answer the adversary's queries. We first show that the hybrid experiments H_0 and H_1 are indistinguishable to an adversary, which shows that the difference in the adversary's advantage in producing a valid forgery in the two experiments is negligible. Then, in **Hybrid** H_1, we show that a forgery by the adversary can be translated directly to a forgery of the underlying signature scheme and therefore, is negligible. This shows that the adversary's advantage in producing a valid forgery in the real game **Hybrid** H_0 is also negligible.

- **Hybrid** H_0: This is the real unforgeability game instantiated with the Π_{TS} scheme above. The challenger runs setup with $\mathsf{TS.Setup}$, and provides $\mathsf{pp}, \mathsf{vk}_{\mathsf{ver}}$ to the adversary. The adversary then commits to the set of signers S^* of size $t-1$ to corrupt. The challenger provides the partial signing keys corresponding to S^* to the adversary. For each of the signing queries to $\mathcal{O}_{\mathsf{Sign}}$ that the adversary makes, the challenger computes the partial signatures honestly using $\mathsf{TS.PartSign}$ and provides it to the adversary. At the end of the game, the adversary returns an attempted forgery.
- **Hybrid** H_1: In this hybrid, the challenger simulates the partial signatures without the signing key ssk using the simulator $\mathcal{S}$ of the universal thresholdizer scheme (Definition 4.2). We first change the way the challenger runs the setup $\mathsf{TS.Setup}$. Namely, the challenger still generates the signature scheme honestly $(\mathsf{ssk}, \mathsf{svk}) \leftarrow \mathsf{S.KeyGen}(1^\lambda)$, but now, instead of instantiating the universal thresholdizer with respect to ssk, the challenger instantiates the universal signature scheme with respect to the all zeros string $(\mathsf{utpp}, \{\mathsf{utsk}_i\}_{i\in[N]}) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^N, 1^t, 1^d, 0^{|\mathsf{ssk}|})$. Then, for each of the partial signing query that the adversary makes, the challenger answers by running the simulator $\mathcal{S}$. It provides the simulator with the signing keys $\{\mathsf{utsk}_i\}_{i\in S^*}$ and for each call to $\mathcal{O}_{\mathsf{Sim}}$ that $\mathcal{S}$ makes, the challenger feeds $\mathcal{S}$ with $\sigma \leftarrow \mathsf{S.Sign}(\mathsf{ssk}, \mathbf{m})$.

  It is easy to see that the view of the adversary in H_0 is exactly the view of the adversary in $\mathsf{Expt}^{\mathsf{Real}}_{\Pi_{\mathsf{UT}},\mathcal{A}}$ (restricted to the signing circuit) and the view of the adversary in H_1 is exactly the view of the adversary in $\mathsf{Expt}^{\mathsf{RAND}}_{\Pi_{\mathsf{UT}},\mathcal{A}}$ (Definition 4.2). Therefore, by the privacy security of Π_{UT}, the two hybrids are indistinguishable.

To conclude the proof, we show that a forgery in H_1 leads to a forgery in the underlying signature scheme. For an adversary $\mathcal{A}$, denote by $\mathsf{H}_1(\mathcal{A})$ the indicator random variable that $\mathcal{A}$ returns a valid forgery at the end of the hybrid experiment H_1. Then we have the following lemma.

Lemma 6.6. *For any efficient adversary* $\mathcal{A}$*, we have that*

$$\Pr[\mathsf{H}_1(\mathcal{A}) = 1] \leq \mathsf{negl}(\lambda)$$

assuming that Π_{S} *is unforgeable.*

Proof. Since the signing key of the signature scheme ssk is not used by the challenger in H_1 other than generating the signatures σ to feed to $\mathcal{S}$, we can construct a straightforward reduction to the underlying signature scheme.

Formally, for adversary $\mathcal{A}$, we construct a simulator $\mathcal{B}$ that interacts with the unforgeability challenger of Π_{S}. $\mathcal{B}$ first receives a verification key svk from the signatures challenger for Π_{S}. Then it generates the thresholdizer parameters and keys $(\mathsf{utpp}, \{\mathsf{utsk}_i\}_{i\in[N]}) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^N, 1^t, 1^d, 0^{|\mathsf{ssk}|})$ and provides $\mathsf{pp} = \mathsf{utpp}$, $\mathsf{vk}_{\mathsf{ver}} = \mathsf{svk}$ to the adversary. Upon receiving the challenge set S^* from $\mathcal{A}$, the simulator $\mathcal{B}$ provides

$\{\mathsf{utsk}_i\}_{i\in S^*}$ to the adversary. Then, for each partial signing query $(\mathbf{m}, i)$ that $\mathcal{A}$ makes, $\mathcal{B}$ answers by running the thresholdizer simulator $\sigma_i \leftarrow \mathcal{S}^{\mathcal{O}_{\mathsf{Sim}}(\cdot)}(\mathbf{m}, i)$. For each $\mathcal{S}$'s oracle call to $\mathcal{O}_{\mathsf{Sim}}(C_{\mathbf{m}})$,[6] $\mathcal{B}$ submits $\mathbf{m}$ to the Π_{S} signatures challenger to receive $\sigma_{\mathbf{m}}$ and feeds it to $\mathcal{S}$. We note that by definition of the challenger, we have that $\sigma_{\mathbf{m}} = C_{\mathbf{m}}(\mathsf{ssk})$.

Now, the adversary $\mathcal{A}$ submits a forgery $(\mathbf{m}^*, \sigma^*)$. If this is a valid forgery, then $\mathbf{m}^*$ was not previously queried to $\mathcal{O}_{\mathsf{Sign}}$, which means that the simulator $\mathcal{B}$ did not submit $\mathbf{m}^*$ as a signing query to the Π_{S} challenger. Furthermore, if $(\mathbf{m}^*, \sigma^*)$ is a valid forgery, then $\mathsf{S.Verify}(\mathbf{m}^*, \sigma^*)$ accepts and therefore $\mathcal{B}$ can submit $(\mathbf{m}^*, \sigma^*)$ as its own valid forgery to the Π_{S} challenger. Therefore, with the winning advantage of $\mathcal{A}$, the simulator $\mathcal{B}$ can construct a valid forgery and break the security of Π_{S}. This concludes the proof of the lemma. □

□

We now show that our threshold signatures scheme above is robust.

Theorem 6.7. *The threshold signature scheme above satisfies robustness as defined in Definition 6.2 assuming that the underlying universal thresholdizer scheme Π_{UT} satisfies robustness (Definition 4.3).*

Outline. The robustness property of the scheme follows very directly from the robustness property of the universal thresholdizer and we provide just an outline of the proof.

In the robustness security game of threshold signatures, an adversary wins if it can generate a message, a partial signature pair $(\mathbf{m}^*, \sigma_i^*)$, and an index i such that the partial signature verification algorithm accepts $\mathsf{TS.PartSignVerify}(\mathsf{pp}, \mathbf{m}, \sigma_i) = 1$ and the partial signature σ_i^* is different from an honestly generated partial signature $\sigma_i^* \neq \mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, \mathbf{m})$. We note, however, that $\mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, \mathbf{m})$ is exactly the universal thresholdizer partial evaluation algorithm $\mathsf{UT.Eval}(\mathsf{utpp}, \mathsf{sk}_i, C_{\mathbf{m}})$ where $C_{\mathbf{m}}(\mathsf{ssk}) = \mathsf{S.Sign}(\mathsf{ssk}, \mathbf{m})$, and the partial signature verification algorithm is exactly the universal thresholdizer verification algorithm $\mathsf{UT.Verify}(\mathsf{utpp}, C_{\mathbf{m}}, \sigma_i)$. Therefore, an improperly generated partial signature for which the partial signature verification algorithm accepts directly violates the robustness property of the underlying universal thresholdizer. This concludes the proof outline. □

Finally, we show that the construction satisfies anonymity, which follows from the evaluation correctness of the underlying universal thresholdizer scheme in a straightforward way.

Theorem 6.8. *The threshold signature scheme above satisfies anonymity as defined in Definition 6.3 assuming that the underlying universal thresholdizer scheme Π_{UT} satisfies evaluation correctness.*

Outline. We note that the evaluation correctness of the universal thresholdizer scheme states that for any set $S \subset [N]$ of size $|S| = t$ and an admissible circuit C, we have that

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

For the construction, the signers evaluate the circuit $C_{\mathbf{m}}(\mathsf{ssk}) = \mathsf{S.Sign}(\mathsf{ssk}, \mathbf{m})$. Therefore for any set S, the combined signature equals $\mathsf{S.Sign}(\mathsf{ssk}, \mathbf{m})$ for any message $\mathbf{m}$ with overwhelming probability, which shows that any two combined signatures signed by two distinct sets of signers must be equal with overwhelming probability. □

Compactness. In the construction above, the combined signature is equal to a signature generated by a single authority signature scheme Π_{S} by evaluation correctness. Therefore, the size of the combined signature is independent of t and N. Furthermore, the signature verification key $\mathsf{vk}_{\mathsf{ver}}$ is defined to be svk, which has size independent of t and N. Therefore, the compactness of the construction easily follows.

[6] Recall that $C_{\mathbf{m}}(\mathsf{ssk}) = \mathsf{S.Sign}(\mathsf{ssk}, \mathbf{m})$.

6.2 CCA Threshold PKE

In this section, we construct a CCA secure threshold PKE from universal thresholdizers. In a threshold PKE scheme, the decryption key is divided into different key shares and is distributed to multiple of decryption servers. When decrypting a message, each of the decryption server creates its own decryption share, and the decryption shares can be publicly combined to result in a full decryption.

Definition 6.9. A threshold PKE (TPKE) scheme for a message space $\mathcal{M}$ is a tuple of efficient algorithms $\Pi_{\mathsf{TPKE}} = (\mathsf{TPKE.Setup}, \mathsf{TPKE.Enc}, \mathsf{TPKE.PartDec}, \mathsf{TPKE.PartDecVerify}, \mathsf{TPKE.Combine})$ defined as follows:

- $\mathsf{TPKE.Setup}(1^\lambda, 1^N, 1^t) \to (\mathsf{pp}, \mathsf{pk_{enc}}, \{\mathsf{sk}_i\}_{i\in[N]})$: On input the security parameter λ, the number of decryption servers N, and a threshold $t \in [N]$, the setup algorithm outputs the public parameters pp, public encryption key $\mathsf{pk_{enc}}$, and a set of partial decryption keys $\{\mathsf{sk}_i\}_{i\in[N]}$.
- $\mathsf{TPKE.Enc}(\mathsf{pk_{enc}}, \mathbf{m}) \to \mathsf{ct}$: On input the encryption key $\mathsf{pk_{enc}}$, and a message $\mathbf{m} \in \mathcal{M}$, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{TPKE.PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct}) \to \mathsf{ct}_i$: On input the public parameters pp, a partial decryption key sk_i, and a ciphertext ct, the partial decryption algorithm outputs a decryption share ct_i.
- $\mathsf{TPKE.PartDecVerify}(\mathsf{pp}, \mathsf{ct}, \mathsf{ct}_i) \to \{0, 1\}$: On input the public parameters pp, a ciphertext ct, and a decryption share ct_i, the decryption verification algorithm accepts or rejects.
- $\mathsf{TPKE.Combine}(\mathsf{pp}, \{\mathsf{ct}_i\}_{i\in S}) \to \mathbf{m}'$: On input the public parameters pp, and a set of decryption shares $\{\mathsf{ct}_i\}_{i\in S}$, the combining algorithm outputs a message $\mathbf{m}'$.

Decryption Correctness. We require that for any $\lambda, N, t \in \mathbb{N}$ such that $t \le N$, $(\mathsf{pp}, \mathsf{pk_{enc}}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{TPKE.Setup}(1^\lambda, 1^N, 1^t)$, $\mathbf{m} \in \mathcal{M}$, $\mathsf{ct} \leftarrow \mathsf{TPKE.Enc}(\mathsf{pk_{enc}}, \mathbf{m})$, $\mathsf{ct}_i \leftarrow \mathsf{TPKE.PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct})$ for $i = 1, ..., N$, $S \subset [N]$ of size t, we have that

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

Partial Decryption Verification Correctness. We require that for any $\lambda, N, t \in \mathbb{N}$ such that $t \le N$, $(\mathsf{pp}, \mathsf{pk_{enc}}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{TPKE.Setup}(1^\lambda, 1^N, 1^t)$, $\mathbf{m} \in \mathcal{M}$, $\mathsf{ct} \leftarrow \mathsf{TPKE.Enc}(\mathsf{pk_{enc}}, \mathbf{m})$ we have that

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

for all $i \in [N]$.

Security. There are two security requirements for a threshold PKE scheme. The first is the threshold CCA security, which states that an adversary who holds less than t shares of the decryption key cannot learn any information about the encrypted message.

Definition 6.10 (CCA Security)**.** We say that a threshold PKE scheme $\Pi_{\mathsf{TPKE}} = (\mathsf{TPKE.Setup}, \mathsf{TPKE.PartDec}, \mathsf{TPKE.PartDecVerify}, \mathsf{TPKE.Combine})$ satisfies CCA security if for any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{cca}}_{\Pi_{\mathsf{TPKE}},\mathcal{A}} = \left|\Pr[\mathsf{Expt}^{\mathsf{cca},(0)}_{\Pi_{\mathsf{TPKE}},\mathcal{A}}(\lambda)] - \Pr[\mathsf{Expt}^{\mathsf{cca},(1)}_{\Pi_{\mathsf{TPKE}},\mathcal{A}}(\lambda)]\right| \le \mathsf{negl}(\lambda)$$

where the experiment $\mathsf{Expt}^{\mathsf{cca},(b)}_{\Pi_{\mathsf{TPKE}},\mathcal{A}}(\lambda)$ is defined as follows:

1. $(\mathsf{pp}, \mathsf{pk_{enc}}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{TPKE.Setup}(1^\lambda, 1^N, 1^t)$.
2. $(S^*, \mathsf{st}_1) \leftarrow \mathcal{A}_1(\mathsf{pp}, \mathsf{pk_{enc}})$ where $|S^*| = t - 1$.
3. $(\mathbf{m}_0, \mathbf{m}_1, \mathsf{st}_2) \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{Dec}}(\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[N]}, \cdot, \cdot)}(\{\mathsf{sk}_i\}_{i\in S^*}, \mathsf{st}_1)$.

4. $\mathsf{ct}_b \leftarrow \mathsf{TPKE.Enc}(\mathsf{pk}_{\mathsf{enc}}, \mathbf{m}_b)$.

5. Output $\mathcal{A}_3^{\mathcal{O}_{\mathsf{Dec}}(\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[N]}, \cdot, \cdot)}(\mathsf{ct}_b, \mathsf{st}_2)$.

where the algorithms $\mathcal{O}_{\mathsf{Dec}}$ is defined as follows:

- $\mathcal{O}_{\mathsf{Dec}}(\mathsf{pp}, \{\mathsf{sk}_i\}_{i\in[N]}, \mathsf{ct}, i)$: On input the public parameters pp, a set of signing keys $\{\mathsf{sk}_i\}_{i\in[N]}$, a ciphertext ct, and an index $i \in [N]\backslash S^*$, it outputs the decryption share $\mathsf{ct}_i \leftarrow \mathsf{TPKE.PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct})$.

For the admissibility condition, we require that the adversary cannot query the challenge ciphertext ct_b to the partial decryption oracle $\mathcal{O}_{\mathsf{Dec}}$ to prevent the adversary from trivially winning the game.

As in the setting of threshold signatures, the second security property that we require is robustness, which says that an adversary cannot convince the partial decryption verifier with an improperly generated decryption share.

Definition 6.11 (Robustness)**.** We say that a threshold PKE scheme $\Pi_{\mathsf{TPKE}} = (\mathsf{TPKE.Setup}, \mathsf{TPKE.PartDec}, \mathsf{TPKE.PartDecVerify}, \mathsf{TPKE.Combine})$ satisfies robustness if for any efficient adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{rb}}_{\Pi_{\mathsf{TPKE}},\mathcal{A}} = \Pr[\mathsf{Expt}^{\mathsf{rb}}_{\Pi_{\mathsf{TPKE}},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

where the experiment $\mathsf{Expt}^{\mathsf{rb}}_{\Pi_{\mathsf{TPKE}},\mathcal{A}}(\lambda)$ is defined as follows:

1. $(\mathsf{pp}, \mathsf{pk}_{\mathsf{enc}}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{TPKE.Setup}(1^\lambda, 1^N, 1^t)$.

2. $(\mathsf{ct}^*, \mathsf{ct}^*_i, i) \leftarrow \mathcal{A}_1(\mathsf{pp}, \mathsf{pk}_{\mathsf{enc}}, \{\mathsf{sk}_i\}_{i\in[N]})$.

3. Output $\mathsf{Verify}^*(\mathsf{pp}, \mathsf{ct}^*, \mathsf{ct}^*_i, i)$.

where the algorithm $\mathsf{Verify}^*(\mathsf{pk}_{\mathsf{ver}}, \mathsf{ct}, \mathsf{ct}_i, i)$ is defined as follows:

- $\mathsf{Verify}^*(\mathsf{pp}, \mathsf{ct}, \mathsf{ct}_i, i)$: On input the public parameters pp, a ciphertext ct, a decryption share ct_i, and an index i, it accepts if the following conditions hold:
  - $\mathsf{TPKE.PartDecVerify}(\mathsf{pp}, \mathsf{ct}, \mathsf{ct}_i) = 1$.
  - $\mathsf{ct}_i \neq \mathsf{TPKE.PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathbf{m})$.

Compactness. As in the case of threshold signatures, we require an additional property of compactness for threshold PKE's. Namely, we require that the size of the ciphertext and the public encryption key $\mathsf{pk}_{\mathsf{enc}}$ to be polynomial in the security parameters and independent of the number of decryption servers N or the threshold t.

Definition 6.12 (Compactness)**.** We say that a threshold PKE scheme $\Pi_{\mathsf{TPKE}} = (\mathsf{TPKE.Setup}, \mathsf{TPKE.PartDec}, \mathsf{TPKE.PartDecVerify}, \mathsf{TPKE.Combine})$ satisfies compactness if there exist polynomials $s_1(\cdot), s_2(\cdot)$ such that for every $\lambda, N, t \in \mathbb{N}$ such that $t \leq N$, any $(\mathsf{pp}, \mathsf{pk}_{\mathsf{enc}}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{TPKE.Setup}(1^\lambda, 1^N, 1^t)$, $\mathbf{m} \in \mathcal{M}$, $\mathsf{ct} \leftarrow \mathsf{TPKE.Enc}(\mathsf{pk}_{\mathsf{enc}}, \mathbf{m})$ we have that $|\mathsf{ct}| \leq s_1(\lambda)$, $|\mathsf{pk}_{\mathsf{enc}}| \leq s_2(\lambda)$.

6.2.1 Construction

In this section, we describe our construction for a threshold PKE scheme. As building blocks, we use a universal thresholdizer scheme $\Pi_{\mathsf{UT}} = (\mathsf{UT.Setup}, \mathsf{UT.Eval}, \mathsf{UT.Verify}, \mathsf{UT.Combine})$ and a CCA secure PKE scheme $\Pi_{\mathsf{PKE}} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$.

Fix a security parameter λ. We construct a threshold PKE scheme $\Pi_{\mathsf{TPKE}} = (\mathsf{TPKE.Setup}, \mathsf{TPKE.Enc}, \mathsf{TPKE.PartDec}, \mathsf{TPKE.PartDecVerify}, \mathsf{TPKE.Combine})$ as follows:

- $\mathsf{TPKE.Setup}(1^\lambda, 1^N, 1^t) \to (\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[N]})$: On input the security parameter λ, the number of signing authorities N, and a threshold $t \in [N]$, the setup algorithm first generates the keys for the PKE $(\mathsf{pkepk}, \mathsf{pkesk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$. Then, it instantiates a universal thresholdizer scheme with respect to the secret key pkesk, $(\mathsf{utpp}, \{\mathsf{utsk}_i\}_{i\in[N]}) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^N, 1^t, 1^d, \mathsf{pkesk})$ where d is the depth of the decryption algorithm $\mathsf{PKE.Dec}$. Then, it sets

$$\mathsf{pp} = \mathsf{utpp}, \qquad \mathsf{pk}_{\mathsf{enc}} = \mathsf{pkepk}, \qquad \mathsf{sk}_i = \mathsf{utsk}_i \ \ \forall i \in [N]$$

- $\mathsf{TPKE.Enc}(\mathsf{pk}_{\mathsf{enc}}, \mathbf{m}) \to \mathsf{ct}$: On input the public key $\mathsf{pk}_{\mathsf{enc}} = \mathsf{pkepk}$, and a message $\mathbf{m}$, the the encryption algorithm outputs $\mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pkepk}, \mathbf{m})$.

- $\mathsf{TPKE.PartDec}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct}) \to \mathsf{ct}_i$: On input the public parameters $\mathsf{pp} = \mathsf{utpp}$, a partial decryption key $\mathsf{sk}_i = \mathsf{utsk}_i$, and a ciphertext ct, the partial decryption algorithm outputs $\mathsf{ct}_i \leftarrow \mathsf{UT.Eval}(\mathsf{utpp}, \mathsf{utsk}_i, C_{\mathsf{ct}})$ where the circuit C_{ct} is defined as follows

$$C_{\mathsf{ct}}(\mathsf{sk}) = \mathsf{PKE.Dec}(\mathsf{sk}, \mathsf{ct}).$$

- $\mathsf{TPKE.PartDecVerify}(\mathsf{pp}, \mathsf{ct}, \mathsf{ct}_i) \to \{0, 1\}$: On input the public parameters $\mathsf{pp} = \mathsf{utpp}$, a ciphertext ct, and a decryption share ct_i, the decryption verification algorithm outputs $\mathsf{UT.Verify}(\mathsf{utpp}, C_{\mathsf{ct}}, \mathsf{ct}_i)$.

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

Correctness. The combining correctness of the scheme Π_{PKE} above follows directly from the evaluation correctness of the universal thresholdizer scheme Π_{UT} and the correctness of the underlying PKE scheme Π_{PKE}. The partial decryption verification correctness follows directly from the verification correctness of Π_{UT}.

6.2.2 Security

We now show security of the threshold PKE scheme above.

Theorem 6.13. *The threshold PKE scheme above satisfies the CCA security notion as defined in Definition 6.10 assuming that the underlying universal thresholdizer scheme Π_{UT} satisfies privacy (Definition 4.2) and the PKE scheme Π_{PKE} satisfies CCA security.*

Theorem 6.13. We proceed through a series of hybrid experiments where the experiment H_0 corresponds to $\mathsf{Expt}^{\mathsf{cca},(0)}_{\Pi_{\mathsf{TPKE}},\mathcal{A}}$ and the experiment H_3 corresponds to $\mathsf{Expt}^{\mathsf{cca},(1)}_{\Pi_{\mathsf{TPKE}},\mathcal{A}}$ in Definition 6.10.

- **Hybrid** H_0: This experiment corresponds to the experiment $\mathsf{Expt}^{\mathsf{cca},(0)}_{\Pi_{\mathsf{TPKE}},\mathcal{A}}$. The challenger runs the honest $\mathsf{TPKE.Setup}$, and provides the keys $\mathsf{pp}, \mathsf{pk}_{\mathsf{enc}}$ to the adversary. The adversary then commits to the set of decryptors S^* of size $t-1$ to corrupt. The challenger provides the partial decryption keys to the adversary. For each partial decryption queries $\mathcal{O}_{\mathsf{Dec}}$ that the adversary makes, the challenger computes the decryption shares honestly using $\mathsf{TPKE.PartDec}$ and provides it to the adversary. On the challenge $(\mathbf{m}_0, \mathbf{m}_1)$ that the adversary submits, the challenger encrypts $\mathbf{m}_0$, $\mathsf{ct}^* \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{enc}}, \mathbf{m}_0)$ and sends it to the adversary.

- **Hybrid** H_1: In this hybrid experiment, the challenger simulates the decryption shares without the decryption key pkesk using the simulator $\mathcal{S}$ of the universal thresholdizer scheme. We first change the way the challenger runs the setup $\mathsf{UT.Setup}$. The challenger still generates the PKE keys honestly $(\mathsf{pkepk}, \mathsf{pkesk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, but now, instead of instantiating the universal thresholdizer scheme with respect to pkesk, the challenger instantiates it with respect to the all zeros string $(\mathsf{utpp}, \{\mathsf{utsk}_i\}_{i\in[N]}) \leftarrow \mathsf{UT.Setup}(1^\lambda, 1^N, 1^t, 1^d, 0^{|\mathsf{pkesk}|})$. Then, for each of the partial decryption queries that the adversary makes, the challenger answers by running the simulator $\mathcal{S}$. It provides the simulator with the signing keys $\{\mathsf{sk}_i\}_{i\in S^*}$ and for each call to $\mathcal{O}_{\mathsf{Sim}}$ that $\mathcal{S}$ makes, the challenger feeds $\mathcal{S}$ with

$\mathbf{m} \leftarrow \mathsf{PKE.Dec}(\mathsf{pkesk}, \mathsf{ct})$. For the challenge $(\mathbf{m}_0, \mathbf{m}_1)$ that the adversary submits, the challenger encrypts $\mathbf{m}_0$, $\mathsf{ct}^* \leftarrow \mathsf{PKE.Enc}(\mathsf{pkepk}, \mathbf{m}_0)$ and sends it to adversary.

We note that the view of the adversary in H_0 is exactly the view of the adversary in $\mathsf{Expt}^{\mathsf{Real}}_{\Pi_{\mathsf{UT}},\mathcal{A}}$ and the view of the adversary in H_1 is exactly the view of the adversary in $\mathsf{Expt}^{\mathsf{Rand}}_{\Pi_{\mathsf{UT}},\mathcal{A}}$ in Definition 4.2. Therefore, by the privacy security of Π_{UT}, the two hybrids are indistinguishable. We note that in H_1, the challenger does not use the decryption key pkesk other than to feed the simulator $\mathcal{S}$ with the correct decryption of ciphertexts that the adversary sends for partial decryption.

- **Hybrid** H_2: This hybrid experiment is the same as H_1 except that now, the challenger changes the way it generates the challenge ciphertext. Specifically, upon receiving the challenge messages $(\mathbf{m}_0, \mathbf{m}_1)$, the challenger encrypts $\mathbf{m}_1$, $\mathsf{ct}^* \leftarrow \mathsf{PKE.Enc}(\mathsf{pkepk}, \mathbf{m}_1)$ instead of $\mathbf{m}_0$, and sends ct^* to the adversary.

  We note that by the CCA security of the underlying PKE scheme Π_{PKE}, the adversary cannot distinguish H_1 and H_2. This follows from the fact that the challenger does not use the decryption key pkesk other than computing the full decryptions to answer the partial decryption queries, which a simulator can simulate using the decryption oracle for CCA security of PKE.

- **Hybrid** H_3: This hybrid experiment corresponds to the experiment $\mathsf{Expt}^{\mathsf{cca},(1)}_{\Pi_{\mathsf{TPKE}},\mathcal{A}}$. This experiment is identical to H_2 except that now, the challenger provides the honest partial decryptions instead of the simulated partial decryptions for the decryption queries that the adversary makes.

  By the same indistinguishability argument for the experiments H_0 and H_1, we can argue that the experiments H_2 and H_3 are indistinguishable by the privacy security of Π_{UT}.

□

The robustness of the threshold PKE scheme above follows very directly from the robustness of the underlying universal thresholdizer. The proof of the following theorem follows from the same proof outline of Theorem 6.7.

Theorem 6.14. *The threshold PKE scheme above satisfies robustness as defined in Definition 6.11 assuming that the underlying universal thresholdizer scheme* Π_{UT} *satisfies robustness (Definition 4.3).*

Compactness. As in the case of threshold signatures, the ciphertexts and public encryption keys correspond to the regular ciphertexts and public keys for the underlying PKE scheme Π_{PKE}. Therefore, it is easy to see that the threshold PKE construction above satisfies the notion of compactness.

7 Conclusion and open problems

We introduced a framework for thresholdizing a variety of cryptographic tasks using a new primitive we call a *universal thresholdizer*. We show that this primitive can be constructed from standard lattice assumptions. By composing a universal thresholdizer with existing cryptographic constructions, such as digital signatures, CCA-secure PKE, and pseudorandom functions, we obtain threshold equivalents of these notions. In particular, our construction gives rise to a one-round *lattice-based threshold signatures*, *threshold CCA-secure PKE*, and *thresholdized functional encryption*, all of which were long standing open problems from lattices.

Open problems. Our work gives rise to a number of open problems. Although our framework is very general, and proves feasibility for a variety of primitives, our reliance on fully homomorphic encryption causes the resulting schemes to be slow in practice. Constructing *practical* one-round threshold cryptosystems, such as threshold signatures, that are resistant to quantum attacks is an important area for future work. In addition, our construction from universal thresholdizer assumes a trusted setup, where a trusted authority provides each user in the system with a share of the FHE decryption key. Is there a universal thresholdizer without trusted setup? Finally, can we construct universal thresholdizers from other standard assumptions? Even for more restricted functionalities, such a construction may provide new insights for constructing threshold systems.

Acknowledgements

D. Boneh and S. Kim are supported by NSF, DARPA, the Simons foundation, and a grant from ONR. R. Gennaro is supported by NSF grant 1545759. S. Goldfeder is supported by NSF Graduate Research Fellowship under grant number DGE 1148900. Opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA.

References

[ABB10a] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, 2010.

[ABB10b] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, 2010.

[ABB10c] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter ciphertext hierarchical ibe. In *CRYPTO*, 2010.

[ABV+12] Shweta Agrawal, Xavier Boyen, Vinod Vaikuntanathan, Panagiotis Voulgaris, and Hoeteck Wee. Functional encryption for threshold functions (or fuzzy ibe) from lattices. In *PKC*, 2012.

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

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

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems. In *STOC*, 1996.

[ASP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In *CRYPTO*, 2014.

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

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

[BF11a] Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In *EUROCRYPT*, 2011.

[BF11b] Dan Boneh and David Mandell Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In *PKC*, 2011.

[BFKW09] Dan Boneh, David Freeman, Jonathan Katz, and Brent Waters. Signing a linear subspace: Signature schemes for network coding. In *PKC*, 2009.

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

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

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

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

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

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

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

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

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

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *EUROCRYPT*, 2012.

[BV14a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *SIAM Journal on Computing*, 43(2):831–871, 2014.

[BV14b] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based fhe as secure as pke. In *ITCS*, 2014.

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

[CFW14] Dario Catalano, Dario Fiore, and Bogdan Warinschi. Homomorphic signatures with efficient verification for polynomial functions. In *CRYPTO*, 2014.

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

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, 2010.

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

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

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

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

[FMNP16] Dario Fiore, Aikaterini Mitrokotsa, Luca Nizzardo, and Elena Pagnin. Multi-key homomorphic authenticators. In *ASIACRYPT*, 2016.

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

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

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

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

[GKKR10] Rosario Gennaro, Jonathan Katz, Hugo Krawczyk, and Tal Rabin. Secure network coding over the integers. In *PKC*, 2010.

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

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

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

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

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

[Mic04] Daniele Micciancio. Almost perfect lattices, the covering radius problem, and applications to ajtai's connection factor. *SIAM Journal on Computing*, 34(1):118–169, 2004.

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

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

[MP13] Daniele Micciancio and Chris Peikert. Hardness of sis and lwe with small parameters. In *CRYPTO*. 2013.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM Journal on Computing*, 37(1):267–302, 2007.

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

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

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

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

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

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

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

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

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

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

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

A Basic Cryptographic Primitives

In this section, we recall the definitions of basic cryptographic primitives that we use throughout the paper.

A.1 Pseudorandom Functions

We recall the definition of pseudorandom functions (PRF) [GGM86].

Definition A.1 (PRF)**.** Fix the security parameter λ. A PRF $F : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^m$ with key space $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$ is secure if for all efficient algorithms $\mathcal{A}$,

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

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

A.2 Signature Scheme

Fix a security parameter λ. A signature scheme $\Pi_\mathsf{S} = (\mathsf{S.KeyGen}, \mathsf{S.Sign}, \mathsf{S.Verify})$ with message space $\mathcal{M}$ consists of the following algorithms

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

Correctness. For correctness, we require that for all $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$ $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{S.KeyGen}(1^\lambda)$, we have that

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

Security. For security, we require that the signature scheme is unforgeable.

Definition A.2 (Unforgebability)**.** We say that a signature scheme $\Pi_\mathsf{S} = (\mathsf{S.KeyGen}, \mathsf{S.Sign}, \mathsf{S.Verify})$ satisfies unforgeability if for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{uf}}_{\Pi_\mathsf{S},\mathcal{A}}(\lambda) = \Pr[\mathsf{Expt}^{\mathsf{uf}}_{\Pi_\mathsf{S},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

where the experiment $\mathsf{Expt}^{\mathsf{uf}}_{\Pi_\mathsf{S},\mathcal{A}}(\lambda)$ is defined as follows:

1. $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{S.KeyGen}(1^\lambda)$.
2. $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{S.Sign}(\mathsf{sk},\cdot)}(\mathsf{vk})$.
3. Output $\mathsf{Verify}^*(\mathsf{vk}, m^*, \sigma^*)$.

where the algorithm $\mathsf{Verify}^*(\cdot,\cdot,\cdot)$ accepts if the following conditions hold:

- $\mathsf{S.Verify}(\mathsf{vk}, m^*, \sigma^*) = 1$.
- m^* was not previously queried to the $\mathsf{S.Sign}(\mathsf{sk},\cdot)$ oracle.

A.3 Public Key Encryptions

Fix a security parameter λ. A public key encryption (PKE) scheme $\Pi_{\mathsf{PKE}} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ with message space $\mathcal{M}$ consists of the following algorithms

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

Correctness. For correctness, we require that for all $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$, $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, we have that

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

Security. For this work, we deal with the standard chosen ciphertext attack (CCA) security of PKEs. Formally, we define the following security definition

Definition A.3 (CCA Security)**.** We say that a PKE scheme $\Pi_{\mathsf{PKE}} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ satisfies CCA security if for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{cca}}_{\Pi_{\mathsf{PKE}},\mathcal{A}}(\lambda) = \left|\Pr[\mathsf{Expt}^{\mathsf{cca},(0)}_{\Pi_{\mathsf{PKE}},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Expt}^{\mathsf{cca},(1)}_{\Pi_{\mathsf{PKE}},\mathcal{A}} = 1]\right| \leq \mathsf{negl}(\lambda)$$

where the experiments $\mathsf{Expt}^{\mathsf{cca},(b)}_{\Pi_{\mathsf{PKE}},\mathcal{A}}(\lambda)$ is defined as follows:

- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$.
- $(m_0, m_1, \mathsf{st}_1) \leftarrow \mathcal{A}_1^{\mathsf{PKE.Dec}(\mathsf{sk},\cdot)}(\mathsf{pk})$.
- $\mathsf{ct}^* \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, m_b)$.
- Output $\mathcal{A}_2^{\mathsf{PKE.Dec}(\mathsf{sk},\cdot)}(\mathsf{ct}^*, \mathsf{st}_1)$.

where we require that $\mathcal{A}$ never queries the decryption oracle $\mathsf{PKE.Dec}(\mathsf{sk}, \cdot)$ on the challenge ciphertext ct^*.

B FHE Modification

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

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

We define the GSW encryption scheme as follows:

- $\mathsf{HE.KeyGen}(1^\lambda, 1^d, 1^k) \to \mathsf{sk}$: The key generation algorithm generates a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. It also samples a uniformly random vector $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and an error vector $\mathbf{e} \leftarrow \chi$ and defines $\tilde{\mathbf{A}} = \begin{pmatrix} \mathbf{A} \\ \mathbf{s}^T\mathbf{A} + \mathbf{e}^T \end{pmatrix}$. Then, it sets

$$\mathsf{pp} = \tilde{\mathbf{A}} \qquad \mathsf{sk} = \begin{pmatrix} -\mathbf{s} & 1 \end{pmatrix}$$

- $\mathsf{HE.Enc}(\mathsf{sk}, \mu) \to \mathsf{ct}$: The encryption algorithm generates a uniform matrix $\mathbf{R} \leftarrow \{0,1\}^{m \times m}$ and outputs

$$\mathsf{ct} = \tilde{\mathbf{A}} \cdot \mathbf{R} + \mu \cdot \mathbf{G}.$$

- $\mathsf{HE.Dec}(\mathsf{sk}, \mathsf{ct}') \to b$: The decryption algorithm computes $\mathbf{y} = \langle \mathsf{sk}, \mathsf{ct}' \cdot \mathbf{w}_m \rangle$ and outputs 0 if $\mathbf{y} \in [-q/4, q/4]$ and outputs 1 otherwise.

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

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

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

For a ciphertext $\mathsf{ct}' = \tilde{\mathbf{A}}\mathbf{R}^* + \mu^* \cdot \mathbf{G}$, decryption procedure can be described as computing the inner product

$$\langle \mathsf{sk}, \mathsf{ct}' \cdot \mathbf{w}_m \rangle = (q/2) \cdot \mu^* + \mathbf{e}^T \cdot \mathbf{R}^* \cdot \mathbf{w}_m.$$

Here, the decryption noise is of the form $\mathbf{e}^T \cdot \mathbf{r}^*$ for some low-norm integer vector $\mathbf{r}^* = \mathbf{R}^* \cdot \mathbf{w}_m$.

Tweak: We modify the construction by simply multiplying the error vector $\mathbf{e}$ by γ. Namely, given 1^γ as part of the input, the key generation algorithm defines

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

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

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

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

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