Towards Practical and Round-Optimal Lattice-Based Threshold and Blind Signatures

Shweta Agrawal* Damien Stehlé† Anshu Yadav‡

Abstract

Threshold and blind signature schemes have found numerous applications in cryptocurrencies, e-cash, e-voting and other privacy-preserving technologies. In this work, we make advances in bringing *lattice-based* constructions for these primitives closer to practice.

1. **Threshold Signatures.** For round optimal threshold signatures, we improve the only known construction by Boneh *et al* [CRYPTO'18] as follows:
   - *Efficiency.* We reduce the amount of noise flooding from $2^{\Omega(\lambda)}$ down to $\sqrt{Q_S}$, where Q_S is the bound on the number of generated signatures and λ is the security parameter. By using lattice hardness assumptions over polynomial rings, this allows to decrease signature bit-lengths from $\widetilde{O}(\lambda^3)$ to $\widetilde{O}(\lambda)$.
   - *Towards Adaptive Security.* The construction of Boneh *et al* satisfies only selective security, where all the corrupted parties must be announced before any signing queries are made. We improve this in two ways: in the ROM, we obtain *partial adaptivity* where signing queries can be made before the corrupted parties are announced but the set of corrupted parties must be announced *all at once*. In the standard model, we obtain full adaptivity, where parties can be corrupted at any time but this construction is in a weaker *pre-processing* model where signers must be provided correlated randomness of length proportional to the number of signatures, in an offline pre-processing phase.
2. **Blind Signatures.** For blind signatures, we improve the state of art lattice-based construction by Hauck *et al* [CRYPTO'20] as follows:
   - *Round Complexity.* We improve the round complexity from three to two – this is optimal.
   - *Efficiency.* Again, we reduce the amount of noise flooding from $2^{\Omega(\lambda)}$ down to $\sqrt{Q_S}$, where Q_S is the bound on the number of signatures and λ is the security parameter.
   - *Number of Signing Queries.* Unlike the scheme from Hauck *et al*, our construction enjoys a proof that is not restricted to a polylogarithmic number of signatures. Using lattice hardness assumptions over rings, we obtain signatures of bit-lengths bounded as $\widetilde{O}(\lambda)$. In contrast, the signature bit-length in the scheme from Hauck *et al* is $\Omega(\lambda^3 + Q_S \cdot \lambda)$.

Concretely, we can obtain blind/threshold signatures of size $\approx$ 3 KB using a variant of Dilithium-G with $\approx$ 128 bit-security, for adversaries limited to getting 256 signatures. In contrast, parameters provided by Hauck *et al* lead to blind signatures of $\approx$ 7.73 MB, for adversaries limited to getting 7 signatures, while concrete parameters are not provided for the construction of threshold signatures by Boneh *et al*.

*IIT Madras, shweta.a@cse.iitm.ac.in.

†ENS de Lyon and Institut Universitaire de France, damien.stehle@ens-lyon.fr.

‡IIT Madras, anshu.yadav06@gmail.com

Contents

1 Introduction

Threshold and blind signatures are fundamental cryptographic primitives with numerous applications in cryptocurrencies and blockchains [74], e-cash [21], e-voting [46], and other privacy-preserving technologies [75]. In a threshold signature scheme [25], the signature issuing capacity is shared among several users, and a signature can be generated only if a sufficient number of users collaborate to sign a message. In a blind signature scheme [21], a user may request a signature from a signer, with the signer not being able to link a message-signature pair with a protocol execution, and the user not being able to forge signatures even after several interactions with the signer.

While there exist many practical realizations of threshold [50, 38, 26, 19, 20] and blind signatures [61, 62, 63, 64, 68, 57] under number-theoretic assumptions, the situation is very unsatisfactory in the post-quantum regime, especially for optimal round complexity. Below, we summarize the state of the art from lattice assumptions, which is the focus of our work.

1.1 State of the Art from Lattices

Threshold Signatures. The thresholdisation of lattice-based signatures from the NIST post-quantum cryptography project has been investigated in [22] but this resulted in several rounds of communication, as well as slow runtime estimates. To the best of our knowledge, the only lattice-based, round-optimal threshold signature construction is by Boneh *et al* [12]. However, while this construction provided the first feasibility result for a long-standing open problem, it is far from being deployable in practice for the following reasons:

1. *Noise Flooding and Impact on Parameters.* It makes use of the so-called "noise flooding" technique [39, 9, 42], which aims to hide a noise term $e \in \mathbb{Z}$ that possibly contains sensitive information, by adding to it a fresh noise term e' whose distribution has a standard deviation that is much larger than an a priori upper bound on $|e|$. To get security against attackers with advantage $2^{-o(\lambda)}$, the standard deviation of e' must be a factor $2^{\Omega(\lambda)}$ larger than the upper bound on $|e|$.

   Unfortunately, this precludes the use of an efficient parametrisation of the Learning With Errors problem (LWE). Concretely, one has to set the LWE noise rate α as $2^{-\Omega(\lambda)}$ so that $|e'|$ remains small compared to the working modulus q. As the best known algorithms for attacking LWE with (typical) parameters n, q, α have run-times that grow as $\exp(\widetilde{O}(n \log q / \log^2 \alpha))$ (see, e.g., [44]) this leads to setting $n \log q = \widetilde{\Omega}(\lambda^3)$. As the signature shares have bit-sizes that grow as $\Omega(n \log q)$, this degrades signature size to $\widetilde{\Omega}(\lambda^3)$ which is prohibitively expensive in practice.

2. *Selective Security.* It only achieves a very restricted *selective* notion of security, where all the corrupted parties must be announced before any partial signing queries are made. To be secure in the more realistic *adaptive* setting, we must rely on complexity leveraging, which further degrades the parameters.

3. *Lack of Instantiation for Building Blocks.* It evaluates a generic signature scheme within a homomorphic encryption (HE) scheme but does not instantiate either the signature or the encryption scheme. In general, it is nontrivial to find the right HE-compatible building blocks – for instance, a natural choice for a lattice-based signature would be that by Gentry, Peikert and Vaikuntanathan [40] or Lyubashevsky [52, 53] but both constructions use rejection sampling.

When converted to a circuit, rejection sampling leads to very large circuit depth, which is not suited to homomorphic evaluation. It is not clear whether using these signatures within a homomorphic encryption scheme is even feasible.

Blind Signatures. The first proposal of a lattice-based blind signature was by Rückert [67], which has been investigated in a series of works, all of which have been shown to have incomplete security proofs and sometimes even attacks [67, 4, 3, 14, 49, 59] (we refer to [43] for a discussion). The state of art construction in this regime is due to Hauck *et al* [43], who state their main contribution as providing a scheme with a correct proof. However, this construction suffers from some significant drawbacks impacting its efficiency:

1. *Noise Flooding.* Similarly to the above one-round threshold signature, it relies on a variant of noise flooding, used this time with the Short Integer Solution problem (SIS). For λ-bit security, it leads to setting the SIS parameters such that $n \log q = \widetilde{\Omega}(\lambda^3)$, which is a lower bound on the signature bit-length.

2. *Loss in Security Proof.* The scheme is an adaptation of the Okamoto-Schnorr blind signature [57] from the discrete logarithm setting to the lattice setting. The best known security proof for the Okamoto-Schnorr blind signature under standard assumptions [64] suffers from a loss in advantage larger than $2^{Q_S}/|\mathcal{C}|$. Here Q_S denotes an upper bound on the number of generated signatures and $\mathcal{C}$ denotes the challenge space. This drastically limits Q_S or forces signature bit-lengths to grow at least linearly in Q_S.

3. *Suboptimal Round Complexity.* Finally, the scheme of [43] uses three rounds, while optimal round complexity is two.

We emphasize that the above blind signature constructions are in the random oracle model. There does exist a round-optimal construction in the *standard* model [37], but this construction is primarily a feasibility result, since it relies heavily on expensive complexity leveraging and general purpose witness indistinguishable proofs. Indeed, the subsequent work of [36] was dedicated to mitigating the impact of complexity leveraging and instantiating the proofs to minimize costs, but this work is restricted to the number-theoretic setting.

1.2 Our Contributions

We make several advances on round-optimal lattice-based threshold and blind signatures to bring them closer to practice.

1. **Threshold Signatures.** For round-optimal threshold signatures, we improve the construction from [12] as follows:

   - *Efficiency.* We decrease the noise flooding ratio from $2^{\Omega(\lambda)}$ down to $\sqrt{Q_S}$, where Q_S is the bound on the number of generated signatures. This gives a one-round threshold signature of bit-length growing as $\widetilde{O}(\lambda \log^2 Q_S)$, which is $\widetilde{O}(\lambda)$ for any polynomially bounded Q_S,[1] in

[1] For many applications, the bound Q_S is quite limited and can be considered to be a small polynomial in λ. For example, for applications pertaining to cryptocurrencies, the bound Q_S captures the total number of transactions made with a user's wallet during the lifetime of a signing key. According to statistics available at the URLs

contrast with $\widetilde{O}(\lambda^3)$ for the construction from [12]. These bit-lengths are obtained when relying on the ring variants of SIS and LWE [54, 60, 71, 55]. Additionally, we show that the amount of noise flooding used in our construction is *optimal*, by exhibiting an attack when smaller noise is used.

- *Selective versus Adaptive.* The construction of [12] satisfies only selective security, where all the corrupted parties must be announced before any signing queries are made. We improve this in two ways: in the random oracle model, we obtain a notion of *partial adaptivity* where signing queries can be made before the corrupted parties are announced. However, the set of corrupted parties must be announced *all at once*. In the standard model, we obtain a construction with full adaptivity, where parties can be corrupted at any stage in the protocol. However, this construction is in a weaker *pre-processing* model[2] where signers must be provided correlated randomness of length proportional to the number of signing queries.

2. **Blind Signatures.** For blind signatures, we improve the state of art (in the ROM) lattice-based construction [43] in several ways.

   - *Round Complexity.* We improve the round complexity of [43] from three to *two* rounds – this is optimal.
   - *Efficiency.* We reduce the amount of noise flooding from $2^{\Omega(\lambda)}$ down to $\sqrt{Q_S}$, where Q_S is the bound on the number of generated signatures and λ is the security parameter.
   - *Number of Signing Queries.* Unlike the scheme from Hauck *et al*, our construction enjoys a proof that is not restricted to a polylogarithmic number of signatures. Using lattice hardness assumptions over rings, we obtain signatures of bit-lengths bounded as $\widetilde{O}(\lambda)$. In contrast, the signature bit-length in the scheme from Hauck *et al* is $\Omega(\lambda^3 + Q_S \cdot \lambda)$.

1.3 Technical Overview

The starting point of our work is to observe that homomorphic encryption (HE) can been used to achieve optimal round complexity for both threshold and blind signatures in the lattice regime [12, 37]. Historically, in both cases, these solutions were treated primarily as feasibility results, and an independent line of work investigated practical, "direct" constructions with limited success, as discussed above. In this work, we argue that the HE based approach can in fact be made to outperform direct constructions where they exist, in several important parameters of interest. Thus, we contend that this approach may be the most viable design strategy for practical constructions of threshold and blind signatures with low round complexity. However, making this approach work requires several new ideas, which we describe next.

below, one transaction per day and per user is a generous upper bound. This suggests that number of signing queries in the lifecycle of the key will be quite small. https://www.blockchain.com/charts/n-transactions, https://www.statista.com/statistics/647374/worldwide-blockchain-wallet-users/

[2]The informed reader may notice similarities with the "MPC with Preprocessing" model (please see [35] and references therein).

Circuitizing the Underlying Signature. Both the threshold and blind signature constructions require to homomorphically evaluate a signing algorithm. For this to be efficient, we need a lattice-based signature scheme that can be expressed as a relatively simple circuit. The GPV signature scheme [40] and its practical versions [29, 65, 34] seem ill-suited to our needs, as the signing algorithm is very sequential, and the requisite 1-dimensional Gaussian samples are obtained via algorithms based on rejection sampling (see [45, 76] and the references therein) that are costly to transform into circuits. The other candidate is Lyubashevsky's signature scheme [52, 53]. It has the advantage of being far less sequential, but it also relies on rejection sampling: when some rejection test does not pass, then one needs to restart the signing process. The rejection probability is typically a non-zero constant: for example, it is ≈ 0.85 for the recommended parameter set of [27].

We show that Lyubashevsky's signature is suitable for transformation into a circuit using two modifications which enjoy different performance properties. The first possibility is to run $Q_S \cdot \omega(\log \lambda)$ executions of Lyubashevsky's signing algorithm in parallel and keep one that passes the rejection test (return $\perp$ if they all fail). If no more than Q_S signatures are generated, then for each of them the signing algorithm succeeds with probability $\geq 1 - \lambda^{-\omega(1)}$. However, this approach requires parallel repetition, implementing the rejection test and choosing the right output which may be cumbersome to implement homomorphically, and additionally results in some error of correctness. Another possibility is to remove the rejection test from Lyubashevsky's signature altogether, which we describe next. Observe that Lyubashevsky's signature consists of a pair of matrices or polynomials over the integers (depending whether one relies on the SIS assumption or its ring variant). It is well-known that when the magnitudes of these integers is bounded by $\lambda^{O(1)}$, then removing the rejection test leads to polynomial-time attacks. Oppositely, taking exponential magnitudes allows, via the noise flooding technique, to prove unforgeability of the rejection-free variant [23], but this leads to setting $n \log q = \widetilde{\Omega}(\lambda^3)$. In this work, we show that limited magnitudes of the order of $\sqrt{Q_S}$ suffice to ensure unforgeability. The proof is based on the Rényi divergence technique [55, 48, 7] and results in a sufficient condition $n \log q = \widetilde{\Omega}(\lambda \log^2 Q_S)$. Since Q_S is often a small polynomial in practice as discussed above, this improvement is significant. Furthermore, we show that this amount of noise flooding is optimal by exhibiting a statistical attack if smaller noise is used. We believe this contribution could have other applications, for instance in evaluating this signature within an MPC protocol. Please see Section 3 for more details.

Threshold Signatures. We improve the threshold signature scheme by Boneh *et al* [12] in two ways:

Efficiency. For threshold signatures, the main efficiency bottleneck in the construction from [12] is the exponential noise flooding as described above. Again, we show that a limited flooding growing as $\sqrt{Q_S}$ suffices by using Rényi divergence instead of statistical distance in the analysis. This leads to a significant improvement in the signature bit size. Moreover, we show that this magnitude of flooding is necessary for this construction, by exhibiting a statistical attack when smaller noise is used. At a high level, our attack proceeds as follows. First we show that using legitimate information available to her, the adversary can compute $\mathsf{err}_M + e_{1,M}$ where err_M is the error that results from homomorphically evaluating the signing algorithm for message M and $e_{1,M}$ is the flooding noise that is used in the partial signature of the first party. As a warmup, consider the setting where the flooding noise is randomized. Now, since the signature scheme is deterministic, the term err_M depends only on M and remains fixed across multiple queries for the same message. On the other hand, the term $e_{1,M}$ keeps changing. Using Hoeffding's bound, it is possible to estimate the average

of $e_{1,M}$ across multiple queries and use this to recover err_M, leading to an attack.

However, the above attack may be easily fixed by making the flooding noise deterministic for a given message, say by using a PRF evaluated on the message to generate the noise. In this setting, we devise a new signature scheme which includes "useless" encryptions of 0 in the signature which do not add to any functionality but allow us to recreate the attack from the randomized setting. In more detail, these encryptions are designed to be a deterministic function of the secret key so that across multiple messages, the error term recovered by the adversary contains a fixed term dependent on the secret key and a fresh term dependent on the message. This allows to launch the attack described previously. Please see Section 4 for more details.

Security. Another limitation of the construction of Boneh *et al* [12] is that security is proved in the weak "selective" model where the adversary must announce all corrupted users before receiving the public parameters and verification key. In contrast, the more reasonable adaptive model allows the adversary to corrupt users based on the public parameters, the verification key and previous user corruptions it may have made. We briefly describe the difficulty in achieving adaptive security. Intuitively, in the selective game, the challenger proceeds by simulating the partial keys corresponding to the honest parties in a "special way". The challenge in the adaptive setting is that without knowing who are the honest/corrupted parties, the challenger does not know which partial keys to program thus.

We overcome this hurdle in the ROM by having the challenger simulate all partial keys as though corresponding to a corrupt user and when later, the list of corrupted parties becomes available, "program" the ROM to "explain" the returned keys in a consistent way. This yields an intermediate notion of "partial adaptivity", in which the attacker can make signing queries before corruption, but must announce its corrupted users all at once. We also provide a construction in the standard model which achieves full adaptivity, but in a weaker pre-processing model where the signers must be provided correlated randomness of length proportional to the number of signing queries, in an offline pre-processing phase. We emphasize that the correlated randomness is independent of the messages to be signed later. This model is reminiscent of the "MPC with Preprocessing" model (please see [35] and references therein). We refer the reader to Section 5 for more details.

Blind Signatures. For blind signatures, we start by observing that i) all direct constructions of lattice-based blind signatures are only secure in the ROM, ii) the two round construction of Garg *et al* [37] has been considered unrealistic for practice primarily because it is in the *standard* model and therefore must use complexity leveraging due to lower bounds [33]. Because of this, no effort has been made in the last decade to instantiate this construction and it has been considered strictly a feasibility result.

We revisit the construction of Garg *et al* [37] and observe that if it is *degraded* to rely on the random oracle (its primary objective was to provide a standard model construction), then the primary source of inefficiency, namely complexity leveraging, can be removed. Similarly, expensive witness indistinguishable ZAPs can be replaced by simple and efficient zero-knowledge proofs for specific algebraic statements [52, 53]. This yields a simple and efficient blind signature in two rounds, that outperforms the state of the art [43] in almost all measures. To take a decade old, unrealistic protocol that uses complexity leveraging and use ROM to convert it to a simple, practical protocol which can be suitably instantiated, is one of the main conceptual contributions of this work. We proceed to discuss the technical challenges in making this template work, and our approaches to handling these.

Circuit Private Homomorphic Encryption. Having removed complexity leveraging, the main difficulty in instantiating the generic blind signature construction of [37] is the need of a homomorphic encryption scheme that enjoys circuit privacy. Informally, circuit privacy requires that the ciphertext distribution obtained by homomorphically evaluating a circuit depends only on the size of the circuit and the underlying plaintext, but not on the circuit itself. This property should hold even if the decryption key is known. A first proposal to get circuit privacy was described in [39, Chapter 20] but is based on the noise flooding technique. More efficient strategies have been designed later in [30] and [15]. The latter works for all homomorphic encryption schemes but requires bootstrapping, whereas the former does not have this limitation but is limited to the homomorphic encryption scheme from [41]. Unfortunately, none of these three approaches fulfills our needs as they only achieve a weak notion of circuit privacy in which the secret and public keys and the input ciphertexts are assumed to be honestly generated. To prove that our blind signature is unforgeable, we need a stronger notion of circuit privacy in which the keys and ciphertexts may be maliciously generated. Such notion was studied in [58] but, being in the standard model, required an inefficient simulator due to lower bounds. Again, we leverage the random oracle model to upgrade the homomorphic encryption scheme from [15] from weak to malicious circuit privacy, in an efficient way. For this, we rely on the specific proof of circuit privacy in [15], which shows that it suffices to ensure that some component of the public key is uniform and that the rest of the public key and the ciphertexts are well-formed (but not necessarily well-distributed). We achieve the former by letting it be output by the random oracle, and the latter with non-interactive zero-knowledge (NIZK) proofs.

Suitable Non-Interactive Zero Knowledge Proofs. It remains to instantiate suitable NIZK proofs. This requires care since NIZK proofs can often be expensive. In recent years, there has been significant progress in constructing efficient lattice-based proofs for *specific* statements, in particular for linear statements with small unknowns[73, 13, 31]. These constructions greatly improve over older zero-knowledge proofs such as those based on [51]. We show that the algebraic structure of the [15] ciphertexts is compatible with these efficient zero-knowledge proofs, leading to an efficient lattice instantiation. Please see Section 6 for details.

Note that we make heavy use of the random oracle – not only to bypass the (partial) impossibility result [33] on the number of rounds of blind signatures and get rid of complexity leveraging, but also to upgrade FHE with a special form [15] to satisfy malicious circuit privacy in a simple and efficient way. We also carefully "stitch together" the specific algebraic structure of the FHE [15] with efficient proofs for specific languages [73, 13, 31] to obtain an overall simple and efficient instantiation from lattices. We are optimistic that these techniques may find other applications.

1.4 Perspective and Open Problems

Since threshold and blind signatures are practice oriented primitives, the final goal in this line of work would be an implementation with concrete parameter sets to provide a benchmark. We do not claim to provide a full solution towards this goal and hence do not provide a benchmark, since several highly non-trivial issues still remain to be tackled. We view our work as taking a first, important step in this direction and discuss here some remaining challenges. As an example, both our constructions rely on the homomorphic evaluation of a signing algorithm. While we designed an "HE compatible" signature scheme in Section 3, our construction, along with all practical lattice-based signature schemes (in the NIST competition for instance), relies on the random oracle. In practice, this must

be instantiated by a hash function, which implies homomorphic evaluation of a hash function by the signer. Choosing a hash function that is suitable for homomorphic evaluation is a serious research question – while there has been some effort in the community to design block ciphers that minimize multiplicative size and depth of description, so as to be more "HE compatible" (see, e.g., [2, 1]), this question has not been studied sufficiently for the case of hash functions that must model random oracles. We discuss some approaches towards this instantiation, but a full solution is beyond the scope of the present work. Please see Section 6.3 for a detailed discussion on this and other issues.

2 Preliminaries

In this section, we define some preliminaries used in our work. Additional preliminaries may be found in Appendix A.

Notation. We write vectors with bold small letters and matrices with bold capital letters. For any vector $\mathbf{v}$, we denote its ith element by $\mathbf{v}[i]$ or $\mathbf{v}_i$. Similarly, for any matrix $\mathbf{M}$, $\mathbf{M}[i][j]$ or $\mathbf{M}_{ij}$ represents the element in the jth column of ith row. Let S be any set, then $|S|$ represents the cardinality of S, while in case of any $x \in \mathbb{R}$, $|x|$ represents absolute value of x. For any $n \in \mathbb{N}$, we let the set $\{1, 2, \ldots, n\}$ be denoted by $[n]$. For any set S, we let $\mathcal{P}(S)$ denote the power set of S, i.e., the set of all subsets of S. $\mathcal{D}_{\Lambda,s,\mathbf{c}}$ represents discrete Gaussian distribution over lattice Λ, with center $\mathbf{c}$ and standard deviation parameter s. When $\mathbf{c} = 0$, we omit it. Similarly, we omit Λ, if $\Lambda = \mathbb{Z}$.

2.1 Threshold Signatures

Definition 2.1 (Threshold Signatures)**.** Let $P = \{P_1, \ldots, P_N\}$ be a set of N parties. A threshold signature scheme for a class of efficient access structures $\mathbb{S}$ on P is a tuple of PPT algorithms denoted by $\mathsf{TS} = (\mathsf{TS.KeyGen}, \mathsf{TS.PartSign}, \mathsf{TS.PartSignVerify}, \mathsf{TS.Combine}, \mathsf{TS.Verify})$ defined as follows:

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

A TS scheme should satisfy the following compactness, correctness and security requirements.

Definition 2.2 (Compactness)**.** A TS scheme for $\mathbb{S}$ satisfies compactness if there exist polynomials $\text{poly}_1(\cdot), \text{poly}_2(\cdot)$ such that for all λ, $\mathsf{A} \in \mathbb{S}$ and $S \in \mathsf{A}$, the following holds. For $(\mathsf{pp}, \mathsf{vk}, \{\mathsf{sk}_i\}_{i=1}^N) \leftarrow \mathsf{TS.KeyGen}(1^\lambda, \mathsf{A})$, $\sigma_i \leftarrow \mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m)$ for $i \in S$, and $\sigma \leftarrow \mathsf{TS.Combine}(\mathsf{pp}, \{\sigma_i\}_{i \in S})$, we have that $|\sigma| \leq \text{poly}_1(\lambda)$ and $|\mathsf{vk}| \leq \text{poly}_2(\lambda)$.

Definition 2.3 (Evaluation Correctness)**.** A signature scheme TS for $\mathbb{S}$ satisfies evaluation correctness if for all $\lambda, \mathsf{A} \in \mathbb{S}$ and $S \in \mathsf{A}$, the following holds. For $(\mathsf{pp}, \mathsf{vk}, \{\mathsf{sk}_i\}_{i=1}^N) \leftarrow \mathsf{TS.KeyGen}(1^\lambda, \mathsf{A})$, $\sigma_i \leftarrow \mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m)$ for $i \in [N]$ and $\sigma_m \leftarrow \mathsf{TS.Combine}(\mathsf{pp}, \{\sigma_i\}_{i \in S})$, we have:

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

Definition 2.4 (Partial Verification Correctness)**.** A signature scheme TS for $\mathbb{S}$ satisfies partial verification correctness if for all λ and $\mathsf{A} \in \mathbb{S}$, the following holds. For $(\mathsf{pp}, \mathsf{vk}, \{\mathsf{sk}_i\}_{i=1}^N) \leftarrow \mathsf{TS.KeyGen}(1^\lambda, \mathsf{A})$,

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

Definition 2.5 (Unforgeability)**.** A TS scheme is unforgeable if for any adversary $\mathcal{A}$ with run-time $2^{o(\lambda)}$, the output of the following experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{TS},uf}(1^\lambda)$ is 1 with probability $2^{-\Omega(\lambda)}$:

1. On input the security parameter λ, the adversary outputs an access structure $\mathsf{A} \in \mathbb{S}$.
2. Challenger runs the $\mathsf{TS.KeyGen}(1^\lambda)$ algorithm and generates public parameters pp, verification key vk and set of N key shares $\{\mathsf{sk}_i\}_{i=1}^N$. It sends pp and vk to $\mathcal{A}$.
3. Adversary $\mathcal{A}$ then outputs a maximally invalid party set $S \subseteq [N]$ to get key shares sk_i for $i \in S$.
4. Challenger provides the set of keys $\{\mathsf{sk}_i\}_{i \in S}$ to $\mathcal{A}$.
5. Adversary $\mathcal{A}$ issues polynomial number of adaptive queries of the form (m, i), where $i \in [N] \setminus S$ to get partial signature σ_i for m. For each query the challenger computes σ_i as $\mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m)$ and provides it to $\mathcal{A}$.
6. At the end of the experiment, adversary $\mathcal{A}$ outputs a message-signature pair (m^*, σ^*). The experiment outputs 1 if m^* was not queried previously as a signing query and $\mathsf{TS.Verify}(\mathsf{vk}, m^*, \sigma^*) = \mathit{accept}$.

Robustness. A TS scheme for $\mathbb{S}$ satisfies robustness if for all λ, the following holds. For all adversary $\mathcal{A}$ with run-time $2^{o(\lambda)}$, the following experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{TS},rb}(1^\lambda)$ outputs 1 with probability $2^{-\Omega(\lambda)}$:

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

2.2 Blind Signatures

To begin, we introduce some notation for interactive executions between algorithms $\mathcal{X}$ and $\mathcal{Y}$. By $(a, b) \leftarrow \langle \mathcal{X}(x), \mathcal{Y}(y) \rangle$, we denote the joint execution of $\mathcal{X}$ and $\mathcal{Y}$ where $\mathcal{X}$ has private input x, $\mathcal{Y}$ has private input y and $\mathcal{X}$ receives private output a while $\mathcal{Y}$ receives private output b.

Definition 2.6 (Blind Signature)**.** A blind signature scheme BS consists of PPT algorithms Gen, Vrfy along with interactive PPT algorithms $\mathcal{S}, \mathcal{U}$ such that for any λ:

- $\mathsf{Gen}(1^\lambda)$ generates a key pair $(\mathsf{Sig.sk}, \mathsf{Sig.vk})$.
- The joint execution of $\mathcal{S}(\mathsf{Sig.sk})$ and $\mathcal{U}(\mathsf{Sig.vk}, m)$, where $m \in \{0,1\}^*$, generates an output σ for the user and no output for the signer. This is denoted as $(\perp, \sigma) \leftarrow \langle \mathcal{S}(\mathsf{Sig.sk}), \mathcal{U}(\mathsf{Sig.vk}, m) \rangle$.
- Algorithm $\mathsf{Vrfy}(\mathsf{Sig.vk}, m, \sigma)$ outputs a bit b.

We assume completeness: for any $m \in \{0,1\}^*$, $(\mathsf{Sig.sk}, \mathsf{Sig.vk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and σ output by $\mathcal{U}$ in the joint execution of $\mathcal{S}(\mathsf{Sig.sk})$ and $\mathcal{U}(\mathsf{Sig.vk}, m)$, it holds that $\mathsf{Vrfy}(\mathsf{Sig.vk}, m, \sigma) = 1$ with probability $1 - \lambda^{-\omega(1)}$.

Blind signatures must satisfy two properties: one more unforgeability and blindness [47].

Definition 2.7 (One More Unforgeability)**.** The blind signature $\mathsf{BS} = (\mathsf{Gen}, \mathcal{S}, \mathcal{U}, \mathsf{Vrfy})$ is one more unforgeable if for any polynomial ℓ, and any algorithm $\mathcal{U}^*$ with run-time $2^{o(\lambda)}$, the success probability of $\mathcal{U}^*$ in the following game is $2^{-\Omega(\lambda)}$:

1. $\mathsf{Gen}(1^\lambda)$ outputs $(\mathsf{ssk}, \mathsf{svk})$, and $\mathcal{U}^*$ is given svk.
2. Algorithm $\mathcal{U}^*$ interacts concurrently with ℓ instances $\mathcal{S}^1_{\mathsf{ssk}}, \ldots, \mathcal{S}^\ell_{\mathsf{ssk}}$.
3. Algorithm $\mathcal{U}^*$ outputs $(m_1, \sigma_1, \ldots, m_{\ell+1}, \sigma_{\ell+1})$.

Algorithm $\mathcal{U}^*$ succeeds if the m_i's are distinct and $\mathsf{Vrfy}(\mathsf{svk}, m_i, \sigma_i) = 1$ for all $i \in [\ell + 1]$.

The blindness condition says that it should be infeasible for any malicious signer $\mathcal{S}^*$ to decide which of two messages m_0 and m_1 of its choice has been signed first in two executions with a honest user $\mathcal{U}$. If one of these executions has returned $\perp$, then the signer is not informed about the other signature either. We will focus on the following notion of honest signer blindness.

Definition 2.8 (Honest Signer Blindness)**.** The blind signature $\mathsf{BS} = (\mathsf{Gen}, \mathcal{S}, \mathcal{U}, \mathsf{Vrfy})$ satisfies honest signer blindness if for any algorithm $\mathcal{S}^*$ with run-time $2^{o(\lambda)}$, the advantage of $\mathcal{S}^*$ in the following game is $2^{-\Omega(\lambda)}$:

1. $\mathsf{Gen}(1^\lambda)$ outputs $(\mathsf{ssk}, \mathsf{svk})$, and $\mathcal{S}^*$ is given $(\mathsf{ssk}, \mathsf{svk})$.
2. A random bit b is chosen and $\mathcal{S}^*$ interacts concurrently with $\mathcal{U}_1 := \mathcal{U}(\mathsf{svk}, m_b)$ and $\mathcal{U}_2 := \mathcal{U}(\mathsf{svk}, m_{\bar{b}})$ by honestly following the protocol. When $\mathcal{U}_1, \mathcal{U}_2$ have completed their executions, the values $\sigma_b, \sigma_{\bar{b}}$ are defined as follows:
   - If either $\mathcal{U}_1$ or $\mathcal{U}_2$ abort, then $(\sigma_b, \sigma_{\bar{b}}) := (\perp, \perp)$.
   - Otherwise, let σ_b (resp. $\sigma_{\bar{b}}$) be the output of $\mathcal{U}_1$ (resp. $\mathcal{U}_2$).

Algorithm $\mathcal{S}^*$ is given (σ_0, σ_1).

3. Algorithm $\mathcal{S}^*$ outputs a bit b'.

Algorithm $\mathcal{S}^*$ succeeds if $b' = b$. If succ denotes the latter event, then the advantage of $\mathcal{S}^*$ is defined as $|\Pr[\mathsf{succ}] - \frac{1}{2}|$.

Full-fledged blindness lets the adversary $\mathcal{S}^*$ sample its own pair $(\mathsf{ssk}, \mathsf{svk})$ at Step 1 (possibly maliciously), and gives svk to the challenger. At Step 2, the adversary may decide not to follow the specified protocol. A blind signature scheme is secure if it is unforgeable and blind.

2.3 Lattices and Discrete Gaussians

In this section we provide definitions of lattices and discrete Gaussian distributions.

2.3.1 Lattices

: An n-dimensional lattice Λ is a discrete additive subgroup of $\mathbb{R}^n$. For an integer $k < n$ and a rank k matrix $\mathbf{B} \in \mathbb{R}^{n \times k}$, $\Lambda(\mathbf{B}) = \{\mathbf{Bx} : \mathbf{x} \in \mathbb{Z}^k\}$ is the lattice generated by integer linear combinations of columns of matrix $\mathbf{B}$. The matrix $\mathbf{B}$ is called a *basis* of the lattice.

Dual of a lattice: The dual of a lattice Λ is defined by $\Lambda^* = \{\mathbf{y} \in \mathbb{R}^n : \mathbf{y}^T \Lambda \subseteq \mathbb{Z}\}$.

2.3.2 Gaussian distribution

For any vector $\mathbf{c} \in \mathbb{R}^n$ and any real $s > 0$, the (spherical) Gaussian function with standard deviation parameter s and center $\mathbf{c}$ is defined as:

$$\forall \mathbf{x} \in \mathbb{R}^n, \rho_{s,\mathbf{c}}(\mathbf{x}) = \exp(-\frac{\pi \|\mathbf{x} - \mathbf{c}\|^2}{s^2}).$$

The Gaussian distribution is $\mathcal{D}_{s,\mathbf{c}}(\mathbf{x}) = \rho_{s,\mathbf{c}}(\mathbf{x})/s^n$.

The (spherical) *discrete Gaussian distribution* over a lattice $\Lambda \subseteq \mathbb{R}^n$, with standard deviation parameter $s > 0$ and center $\mathbf{c}$ is defined as:

$$\forall \mathbf{x} \in \Lambda, \mathcal{D}_{\Lambda,s,\mathbf{c}} = \frac{\rho_{s,\mathbf{c}}(\mathbf{x})}{\rho_{s,\mathbf{c}}(\Lambda)},$$

where $\rho_{s,\mathbf{c}}(\Lambda) = \sum_{\mathbf{x} \in \Lambda} \rho_{s,\mathbf{c}}(\mathbf{x})$. When $\mathbf{c} = \mathbf{0}$, we omit the subscript $\mathbf{c}$.

2.3.3 Smoothing parameter

The smoothing parameter of an n-dimensional lattice Λ with respect to $\epsilon > 0$, denoted by $\eta_\epsilon(\Lambda)$, is the smallest $s > 0$, such that $\rho_{1/s}(\Lambda^* \setminus \{0\}) \leq \epsilon$.

2.4 Hardness Assumptions

We will need the Learning With Errors (LWE) problem, which is known to be at least as hard as certain standard lattice problems in the worst case [66, 17].

Definition 2.9 (Learning With Errors (LWE))**.** Let q, α, m be functions of a parameter n. For a secret $\mathbf{s} \in \mathbb{Z}_q^n$, the distribution $A_{q,\alpha,\mathbf{s}}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ is obtained by sampling $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ and an $e \leftarrow \mathcal{D}_{\mathbb{Z},\alpha q}$, and returning $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^{n+1}$. The Learning With Errors problem $\mathsf{LWE}_{q,\alpha,m}$ is as follows: For $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, the goal is to distinguish between the distributions:

$$D_0(\mathbf{s}) := U(\mathbb{Z}_q^{m \times (n+1)}) \quad \text{and} \quad D_1(\mathbf{s}) := (A_{q,\alpha,\mathbf{s}})^m.$$

We say that a PPT algorithm $\mathcal{A}$ solves $\mathsf{LWE}_{q,\alpha}$ if it distinguishes $D_0(\mathbf{s})$ and $D_1(\mathbf{s})$ with non-negligible advantage (over the random coins of $\mathcal{A}$ and the randomness of the samples), with non-negligible probability over the randomness of $\mathbf{s}$.

Definition 2.10 (Short Integer Solution ($\mathsf{SIS}_{q,n,m,d}$) problem)**.** Let $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \leftarrow \{-d, \ldots, 0, \ldots, d\}^m$ and $\mathbf{t} = \mathbf{A}\mathbf{s}$. Then given $\mathbf{A}, \mathbf{t}$, the task is to find $\mathbf{s}' \in \{-d, \ldots, 0, \ldots, d\}^m$ such that $\mathbf{A}\mathbf{s}' = \mathbf{t}$.

The problem can be defined with respect to different norms as well. Below we give the definition for ℓ_2-SIS.

Definition 2.11 (Short Integer Solution (ℓ_2-$\mathsf{SIS}_{q,n,m,\beta}$))**.** Given a random matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$, find a vector $\mathbf{v} \in \mathbb{Z}^m \setminus \{0\}$ such that $\mathbf{A}\mathbf{v} = 0$ and $\|\mathbf{v}\| \leq \beta$.

In order for the above problem to not be vacuously hard, we need to have $\beta \geq \sqrt{m} q^{n/m}$. This ensures that there exists a solution $\mathbf{v}$.

2.5 Rényi Divergence

The Rényi Divergence (RD) is a measure of closeness of any two probability distributions. In certain cases, especially in proving the security of cryptographic primitives where the adversary is required to solve a search-based problem, the RD can be used as an alternative to the statistical distance [7], which may help obtain security proofs for smaller scheme parameters and may sometimes lead to simpler proofs.

Definition 2.12 (Rényi Divergence)**.** Let P and Q be any two discrete probability distributions such that $Supp(P) \subseteq Supp(Q)$. Then for $a \in (1, \infty)$, the Rényi Divergence of order a is defined by

$$R_a(P||Q) = \left(\sum_{x \in Supp(P)} \frac{P(x)^a}{Q(x)^{a-1}} \right)^{\frac{1}{a-1}}.$$

For $a = 1$ and $a = \infty$, the RD is defined as

$$R_1(P||Q) = \exp\left(\sum_{x \in Supp(P)} P(x) \log \frac{P(x)}{Q(x)} \right) \quad \text{and} \quad R_\infty(P||Q) = \max_{x \in Supp(P)} \frac{P(x)}{Q(x)}.$$

For any fixed distributions P and Q, the function $f(a) = R_a(P\|Q)$ is non decreasing, continuous over $(1, \infty)$ and tends to $R_\infty(P\|Q)$ as a goes to infinity. Further, if $R_a(P\|Q)$ is finite for some a, then it tends to $R_1(P\|Q)$ as a tends to 1.

Lemma 2.13 ([7, Lemma 2.9])**.** *Let $a \in [1, \infty]$. Let P and Q denote distributions with $Supp(P) \subseteq Supp(Q)$. Then the following properties hold*

- ***Log Positivity:*** $R_a(P||Q) \geq R_a(P||P) = 1$.
- ***Data Processing Inequality:*** $R_a(P^f||Q^f) \leq R_a(P||Q)$ *for any function* f*, where* P^f *(resp.* Q^f*) denotes the distribution of* $f(y)$ *induced by sampling* $y \leftarrow P$ *(resp.* $y \leftarrow Q$*).*
- ***Probability preservation:*** *Let* $E \subseteq Supp(Q)$ *be an arbitrary event. If* $a \in (1, \infty)$*, then*

$$Q(E) \geq P(E)^{\frac{a}{a-1}} / R_a(P\|Q).$$

For $a = \infty$*,*

$$Q(E) \geq P(E) / R_\infty(P\|Q).$$

For $a = 1$*, Pinsker's inequality gives the following analogue property:*

$$Q(E) \geq P(E) - \sqrt{\ln R_1(P||Q)/2}.$$

- ***Multiplicativity:*** *Assume that* P *and* Q *are two distributions of a pair of random variables* (Y_1, Y_2)*. For* $i \in \{1, 2\}$*, let* P_i *(resp.* Q_i*) denote the marginal distribution of* Y_i *under* P *(resp.* Q*), and let* $P_{2|1}(\cdot|y_1)$ *(resp.* $Q_{2|1}(\cdot|y_1)$*) denote the conditional distribution of* Y_2 *given that* $Y_1 = y_1$*. Then we have:*
  - $R_a(P||Q) = R_a(P_1||Q_1) \cdot R_a(P_2||Q_2)$ *if* Y_1 *and* Y_2 *are independent for* $a \in [1, \infty]$*.*
  - $R_a(P||Q) \leq R_\infty(P_1||Q_1) \cdot \max_{y_1 \in Y_1} R_a(P_{2|1}(\cdot|y_1)||Q_{2|1}(\cdot|y_1))$*.*
- ***Weak Triangle Inequality:*** *Let* P_1, P_2, P_3 *be three distributions with* $Supp(P_1) \subseteq Supp(P_2) \subseteq Supp(P_3)$*. Then we have*

$$R_a(P_1||P_3) \leq \begin{cases} R_a(P_1||P_2) \cdot R_\infty(P_2||P_3), \\ R_\infty(P_1||P_2)^{\frac{a}{a-1}} \cdot R_a(P_2||P_3) \quad \text{if } a \in (1, +\infty). \end{cases} \tag{2.1}$$

We will use the following RD bounds. Note that proof tightness can often be improved by optimizing over a, as suggested in [72].

Lemma 2.14 ([7])**.** *For any* n*-dimensional lattice,* $\Lambda \subseteq \mathbb{R}^n$ *and* $s > 0$*, let* P *be the distribution* $\mathcal{D}_{\Lambda,s,\mathbf{c}}$ *and* Q *be the distribution* $\mathcal{D}_{\Lambda,s,\mathbf{c}'}$ *for some fixed* $\mathbf{c}, \mathbf{c}' \in \mathbb{R}^n$*. If* $\mathbf{c}, \mathbf{c}' \in \Lambda$*, let* $\epsilon = 0$*. Otherwise fix* $\epsilon \in (0, 1)$ *and assume that* $s > \eta_\epsilon(\Lambda)$*. Then for any* $a \in (1, +\infty)$

$$R_a(P||Q) \in \left[\left(\frac{1-\epsilon}{1+\epsilon} \right)^{\frac{2}{a-1}}, \left(\frac{1+\epsilon}{1-\epsilon} \right)^{\frac{2}{a-1}} \right] \cdot \exp\left(a\pi \frac{||\mathbf{c} - \mathbf{c}'||^2}{s^2} \right).$$

2.6 Homomorphic Encryption (HE).

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

Definition 2.15 (Homomorphic Encryption)**.** A homomorphic encryption scheme HE is a tuple of PPT algorithms $\mathsf{HE} = (\mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Eval}, \mathsf{HE.Dec})$ defined as follows:

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

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

Definition 2.16 (Compactness)**.** We say that an HE scheme is compact if there exists a polynomial function $f(\cdot, \cdot)$ such that for all λ, depth bound d, circuit $\mathcal{C} : \{0,1\}^k \to \{0,1\}$ of depth at most d, and $\mu_i \in \{0,1\}$ for $i \in [k]$, the following holds: for $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.KeyGen}(1^\lambda, 1^d)$, $\mathsf{ct}_i \leftarrow \mathsf{HE.Enc}(\mathsf{pk}, \mu_i)$ for $i \in [k]$, $\hat{\mathsf{ct}} \leftarrow \mathsf{HE.Eval}(\mathsf{pk}, \mathcal{C}, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$, the bit-length of $\hat{ct}$ is at most $f(\lambda, d)$.

Definition 2.17 (Correctness)**.** We say that an HE scheme is correct if for all λ, depth bound d, circuit $\mathcal{C} : \{0,1\}^k \to \{0,1\}$ of depth at most d, and $\mu_i \in \{0,1\}$ for $i \in [k]$, the following holds: for $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.KeyGen}(1^\lambda, 1^d)$, $\mathsf{ct}_i \leftarrow \mathsf{HE.Enc}(\mathsf{pk}, \mu_i)$ for $i \in [k]$, $\hat{\mathsf{ct}} \leftarrow \mathsf{HE.Eval}(\mathsf{pk}, \mathcal{C}, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$, we have

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

Definition 2.18 (Security)**.** We say that an HE scheme is secure if for all λ and depth bound d, the following holds: for any adversary $\mathcal{A}$ with run-time $2^{o(\lambda)}$, the following experiment outputs 1 with probability $2^{-\Omega(\lambda)}$:

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

In this work, our constructions use a *special* HE having some additional properties as described in [12]. These properties are satisfied by direct adaptations of typical HE schemes such as [18, 41] (see, e.g., [12, Appendix B]).

Definition 2.19 (Special HE)**.** An HE scheme is a special HE scheme if it satisfies the following properties:

1. On input $(1^\lambda, 1^d)$, the key generation algorithm $\mathsf{HE.KeyGen}$ outputs $(\mathsf{pk}, \mathsf{sk})$, where the public key contains a prime q and the secret key is a vector $\mathsf{sk} \in \mathbb{Z}_q^m$ for some $m = \mathrm{poly}(\lambda, d)$.
2. The decryption algorithm $\mathsf{HE.Dec}$ consists of two functions $(\mathsf{HE.decode}_0, \mathsf{HE.decode}_1)$ defined as follows:

- $\mathsf{HE.decode}_0(\mathsf{sk}, \mathsf{ct})$: On input an encryption of a message $\mu \in \{0,1\}$ and a secret key vector sk, it outputs $p = \mu \lfloor q/2 \rceil + e \in \mathbb{Z}_q$ for $e \in [-cB, cB]$ with $B = B(\lambda, d, q)$ and e is an integer multiple of c. This algorithm must be a linear operation over $\mathbb{Z}_q$ in the secret key sk.
- $\mathsf{HE.decode}_1(p)$: On input $p \in \mathbb{Z}$, it outputs 1 if $p \in [-\lfloor q/4 \rceil, \lfloor q/4 \rceil]$, and 0 otherwise.

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

For the blind signature construction, we will also need the HE scheme to satisfy an additional security property called circuit privacy.

Definition 2.20 (Circuit Privacy)**.** We say that the homomorphic encryption scheme HE is semi-honest circuit private if for $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HE.KeyGen}(1^\lambda, 1^d)$, any circuit $\mathcal{C} : \{0,1\}^k \to \{0,1\}$ of depth at most d, $\mu_i \in \{0,1\}$ for $i \in [k]$, and $\mathsf{ct}_i \leftarrow \mathsf{HE.Enc}(\mathsf{pk}, \mu_i)$ for $i \in [k]$, the statistical distance between the distributions:

$$\Big(\mathsf{HE.Eval}(\mathsf{pk}, \mathcal{C}, \{\mathsf{ct}_i\}_{i\le k}), \{\mathsf{ct}_i\}_{i\le k}, \mathsf{pk}, \mathsf{sk}\Big)$$
$$\text{and } \Big(\mathsf{HE.Eval}(\mathsf{pk}, \mathcal{C}^0, \{\mathsf{ct}'_i\}_{i\le k}), \{\mathsf{ct}_i\}_{i\le k}, \mathsf{pk}, \mathsf{sk}\Big)$$

is $2^{-\Omega(\lambda)}$, where $\mathsf{ct}'_1 = \mathsf{HE.Enc}(\mathsf{pk}, \mathcal{C}(\mu_1, \ldots, \mu_k))$, $\mathsf{ct}'_i = \mathsf{HE.Enc}(\mathsf{pk}, 0)$ for $1 < i \le k$ and $\mathcal{C}^0 : \{0,1\}^k \to \{0,1\}$ is the trivial circuit of depth d that outputs its first input and ignores the others.

We say that HE is maliciously circuit private if the above holds even if the keys $(\mathsf{pk}, \mathsf{sk})$ and ciphertexts ct_i for $i \in [k]$ are not necessarily generated honestly.

2.7 Secret Sharing.

We now recall some standard definitions related to secret sharing.

Definition 2.21 (Monotone Access Structure)**.** Let $P = \{P_i\}_{i\in[N]}$ be a set of parties. A collection $\mathsf{A} \subseteq \mathcal{P}(P)$ is monotone if for any two sets $B, C \subseteq P$, if $B \in \mathsf{A}$ and $B \subseteq C$, then $C \in \mathsf{A}$. A monotone access structure on P is a monotone collection $\mathsf{A} \subseteq \mathcal{P}(P) \setminus \emptyset$. The sets in A are called *valid sets* and the sets in $\mathcal{P}(P) \setminus \mathsf{A}$ are called *invalid sets*.

Let $S \subseteq P$ be a subset of parties in P. S is called maximal invalid party set if $S \notin \mathsf{A}$, but for any $P_i \in P \setminus S$, we have $S \cup \{P_i\} \in \mathsf{A}$. S is called minimal valid party set if $S \in \mathsf{A}$, but for any $S' \subsetneq S$, we have $S' \notin \mathsf{A}$.

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

Definition 2.22 (Threshold Access Structure)**.** Let $P = \{P_i\}_{i\in[N]}$ be a set of N parties. An access structure A_t is called a threshold access structure, if for all $S \subseteq P$, we have $S \in \mathsf{A}$ iff $|S| \ge t$. We let TAS denote the class of all access structures A_t for all $t \in \mathbb{N}$.

For any set of parties $S \subseteq P$, we define $\mathbf{x}_S = (x_1, \ldots, x_N) \in \{0,1\}^N$ with $x_i = 1$ iff $P_i \in S$.

Definition 2.23 (Efficient Access Structure)**.** An access structure A on set P as defined above is called an efficient access structure if there exists a polynomial size circuit $f_\mathsf{A} : \{0,1\}^N \to \{0,1\}$, such that for all $S \subseteq P$, $f_\mathsf{A}(\mathbf{x}_S) = 1$ iff $S \in \mathsf{A}$.

Definition 2.24 (Secret sharing)**.** Let $P = \{P_1, \ldots, P_N\}$ be a set of parties and $\mathbb{S}$ be a class of efficient access structures on P. A secret sharing scheme SS for a secret space $\mathcal{K}$ is a tuple of PPT algorithms $\mathsf{SS} = (\mathsf{SS.Share}, \mathsf{SS.Combine})$ defined as follows:

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

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

Definition 2.25 (Correctness)**.** For all $S \in \mathsf{A}$ and $\mathsf{k} \in \mathcal{K}$, if $(s_1, \ldots, s_N) \leftarrow \mathsf{SS.Share}(\mathsf{k}, \mathsf{A})$, then

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

Definition 2.26 (Privacy)**.** For all $S \notin \mathsf{A}$ and $\mathsf{k}_0, \mathsf{k}_1 \in \mathcal{K}$, if $(s_{b,1}, \ldots, s_{b,N}) \leftarrow \mathsf{SS.Share}(\mathsf{k}_b, \mathsf{A})$ for $b \in \{0, 1\}$, then the distributions $\{s_{0,i}\}_{i \in S}$ and $\{s_{1,i}\}_{i \in S}$ are identical.

Definition 2.27 (Linear Secret Sharing (LSSS))**.** Let $P = \{P_i\}_{i \in [N]}$ be a set of parties and $\mathbb{S}$ be a class of efficient access structures. A secret sharing scheme SS with secret space $\mathcal{K} = \mathbb{Z}_p$ for some prime p is called a linear secret sharing scheme if it satisfies the following properties:

- $\mathsf{SS.Share}(\mathsf{k}, \mathsf{A})$: There exists a matrix $\mathbf{M} \in \mathbb{Z}_p^{\ell \times N}$ called the *share matrix*, and each party P_i is associated with a partition $T_i \subseteq [\ell]$. To create the shares on a secret k, the sharing algorithm first samples uniform values $r_2, \ldots, r_N \leftarrow \mathbb{Z}_p$ and defines a vector $\mathbf{w} = \mathbf{M} \cdot (\mathsf{k}, r_2, \ldots, r_N)^T$. The share for P_i consists of the entries $\{w_j\}_{j \in T_i}$.
- $\mathsf{SS.Combine}(B)$: For any valid set $S \in \mathsf{A}$, we have

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

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

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

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

Definition 2.28. Let $P = \{P_1, \ldots, P_N\}$ be a set of parties, $\mathbb{S}$ a class of efficient structures on P, and SS a linear secret sharing scheme with share matrix $\mathbf{M} \in \mathbb{Z}_q^{\ell \times N}$. For a set of indices $T \subseteq [\ell]$, T is said to be a valid share set if $(1, 0, \ldots, 0) \in \mathsf{span}(\{\mathbf{M}[j]\}_{j \in T})$, and an invalid share set otherwise. We also use following definitions:

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

The class of access structures that can be supported by a linear secret sharing scheme on N parties is represented by LSSS_N. When the context is clear LSSS_N is simply written as LSSS. We let $\{\mathbf{0},\mathbf{1}\}$-$\mathsf{LSSS}$ denote the class of access structures that can be supported by a LSSS where the recovery coefficients are binary: for a set P of N parties, let k be the shared secret and $\{w_j\}_{j\in T_i}$ be the share of party P_i for $i \in [N]$; then for every set $S \in \mathsf{A}$, there exists a subset $T \subseteq \bigcup_{i\in S} T_i$ such that $\mathsf{k} = \sum_{j\in T} w_j$. It was shown in [12] that such a set $T \subseteq \bigcup_{i\in S} T_i$ can be computed efficiently, and that TAS belongs to $\{0,1\}$-LSSS. Hence, we can use $\{0,1\}$-LSSS for secret sharing in TAS.

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

2.8 Threshold Homomorphic Encryption

Definition 2.29 (Threshold Homomorphic Encryption)**.** A threshold homomorphic encryption for a class of efficient access structures $\mathbb{S}$, defined on a set $P = \{P_1, P_2, \ldots, P_N\}$ of parties is defined by a tuple of five algorithms $\mathsf{THE} = (\mathsf{THE.KeyGen}, \mathsf{THE.Enc}, \mathsf{THE.Eval}, \mathsf{THE.PartDec}, \mathsf{THE.FinDec})$ with the following specifications:

- $\mathsf{THE.KeyGen}(1^\lambda, 1^d, \mathsf{A}) \to (\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N)$: On input the security parameter λ, a depth bound d and an access structure $\mathsf{A} \in \mathbb{S}$, the KeyGen algorithm outputs a public key pk and a set of secret key shares $\{\mathsf{sk}_i\}_{i=1}^N$.
- $\mathsf{THE.Enc}(\mathsf{pk}, \mu) \to \mathsf{ct}$: On input a public key pk and a single bit message $\mu \in \{0,1\}$, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{THE.Eval}(\mathsf{pk}, \mathcal{C}, \mathsf{ct}_1, \mathsf{ct}_2, \ldots, \mathsf{ct}_k) \to \hat{\mathsf{ct}}$: On input a public key pk, a circuit $\mathcal{C} : \{0,1\}^k \to \{0,1\}$ of depth at most d and a set of ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$, the evaluation algorithm outputs an evaluated ciphertext $\hat{\mathsf{ct}}$.
- $\mathsf{THE.PartDec}(\mathsf{pk}, \mathsf{sk}_i, \mathsf{ct}) \to p_i$: On input a public key pk, a secret key share sk_i and a ciphertext ct, the partial decryption algorithm outputs a partial decryption p_i corresponding to the party P_i.
- $\mathsf{THE.FinDec}(\mathsf{pk}, \{p_i\}_{i\in S}) \to \hat{\mu}$: On input a public key pk and a set of partial decryptions corresponding to parties in some set $S \subseteq [N]$, the final decryption algorithm outputs a message $\hat{\mu} \in \{0, 1, \perp\}$.

Correctness. A THE scheme for $\mathbb{S}$ is said to satisfy evaluation correctness if for all λ, depth bound d, access structure $\mathsf{A} \in \mathbb{S}$, circuit $\mathcal{C} : \{0,1\}^k \to \{0,1\}$ of depth at most d, $S \in \mathsf{A}$, and $\mu_i \in \{0,1\}$ for $i \in [k]$, the following condition holds. For $(\mathsf{pp}, sk_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{THE.KeyGen}(1^\lambda, 1^d, \mathsf{A})$, $\mathsf{ct}_i \leftarrow \mathsf{THE.Enc}(\mathsf{pk}, \mu_i)$ for $i \in [k]$, $\hat{\mathsf{ct}} \leftarrow \mathsf{THE.Eval}(\mathsf{pk}, \mathcal{C}, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$:

$$\Pr[\mathsf{THE.FinDec}(\mathsf{pk}, \{\mathsf{THE.PartDec}(\mathsf{pk}, \mathsf{sk}_i, \hat{\mathsf{ct}})\}_{i\in S}) = \mathcal{C}(\{\mu_i\}_{i\in[k]})] = 1 - \lambda^{-\omega(1)}.$$

Semantic security. A THE scheme is said to satisfy semantic security if for all λ and depth bound d, the following holds. For any adversary $\mathcal{A}$ with run-time bounded as $2^{o(\lambda)}$, the experiment below outputs 1 with probability $2^{-\Omega(\lambda)}$:

1. On input the security parameter λ, and a circuit depth d, the adversary $\mathcal{A}$ outputs an access structure $\mathsf{A} \in \mathbb{S}$.
2. The challenger runs $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{THE.KeyGen}(1^\lambda, 1^d, \mathsf{A})$ and provides pk to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a set S of participants, such that $S \notin \mathsf{A}$.
4. The challenger provides $\{\mathsf{sk}_i\}_{i \in S}$ and $\mathsf{THE.Enc}(\mathsf{pk}, b)$, where $b \leftarrow \{0,1\}$ to $\mathcal{A}$.
5. $\mathcal{A}$ outputs a guess bit b'. The experiment outputs 1 if $b = b'$.

Simulation security. A THE scheme for $\mathbb{S}$ is said to satisfy simulation security if for all λ, depth bound d and access structure A, the following holds: there exists a stateful PPT algorithm $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for any adversary $\mathcal{A}$ with run-time bounded as $2^{o(\lambda)}$, the following two experiments are indistinguishable.

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

1. On input the security parameter λ and a circuit depth d, the adversary $\mathcal{A}$ outputs an access structure $\mathsf{A} \in \mathbb{S}$.
2. The challenger runs $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{THE.KeyGen}(1^\lambda, 1^d, \mathsf{A})$ and provides pk to $\mathcal{A}$.
3. Adversary $\mathcal{A}$ outputs a maximal invalid party set $S^* \subseteq [N]$ and a set of message bits, $\mu_1, \mu_2, \ldots, \mu_k \in \{0,1\}$.
4. The challenger provides $\{\mathsf{sk}_i\}_{i \in S^*}$ and $\{\mathsf{ct}_i = \mathsf{THE.Enc}(\mathsf{pk}, \mu_i)\}_{i \in [k]}$ to $\mathcal{A}$.
5. Adversary $\mathcal{A}$ issues a polynomial number of adaptive queries of the form $(S \subseteq \{P_1, \ldots, P_N\}, \mathcal{C})$ for circuits $\mathcal{C} : \{0,1\}^k \to \{0,1\}$ of depth at most d.
6. For each query, the challenger computes $\hat{\mathsf{ct}} \leftarrow \mathsf{THE.Eval}(\mathsf{pk}, \mathcal{C}, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ and sends $\{\mathsf{THE.PartDec}(\mathsf{pk}, \mathsf{sk}_i, \hat{\mathsf{ct}})\}_{i \in$ to $\mathcal{A}$.
7. At the end of the experiment, adversary $\mathcal{A}$ outputs a bit b.

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

1. On input the security parameter λ and circuit depth d, the adversary $\mathcal{A}$ outputs an access structure $\mathsf{A} \in \mathbb{S}$.
2. The challenger runs $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N, \mathsf{st}) \leftarrow \mathcal{S}_1(1^\lambda, 1^d, \mathsf{A})$ and provides pk to $\mathcal{A}$.
3. Adversary $\mathcal{A}$ outputs a maximal invalid party set $S^* \subseteq P$ and a set of message bits, $\mu_1, \mu_2, \ldots, \mu_k \in \{0,1\}$.
4. The challenger provides $\{\mathsf{sk}_i\}_{i \in S^*}$ and $\{\mathsf{ct}_i = \mathsf{THE.Enc}(\mathsf{pk}, \mu_i)\}_{i \in [k]}$ to $\mathcal{A}$.
5. Adversary $\mathcal{A}$ issues a polynomial number of adaptive queries of the form $(S \subseteq [N], \mathcal{C})$ for circuits $\mathcal{C} : \{0,1\}^k \to \{0,1\}$ of depth at most d.
6. For each query, the challenger runs the simulator $\mathcal{S}_2$ to compute partial decryptions as $\{\mathsf{p}_i\}_{i \in S} \leftarrow \mathcal{S}_2(\mathcal{C}, \mathsf{ct}_1, \ldots, \mathsf{ct}_k, \mathcal{C}(\mu_1, \ldots, \mu_k), S, \mathsf{st})$ and sends $\{\mathsf{p}_i\}_{i \in S}$ to $\mathcal{A}$.
7. At the end of the experiment, adversary $\mathcal{A}$ outputs a bit b.

3 Lyubashevsky's Signature Without Aborts

As discussed in Section 1, we provide a rejection-free variant of Lyubashevsky's signature scheme from [53]. Since our applications use this signature within an HE scheme, we also need to derandomize the signing algorithm. To achieve this, we use a pseudorandom function (PRF) to generate the randomness.

3.1 Construction

We use the following building blocks:

- A hash function $H : \{0,1\}^* \to \{\mathbf{v} : \mathbf{v} \in \{-1,0,1\}^k; \|\mathbf{v}\|_1 \leq \alpha\}$, modeled as a random oracle. Here α is a parameter to the scheme.
- A PRF $F : \{0,1\}^r \times \{0,1\}^* \to \{0,1\}^r$.

Our signature scheme is described in Figure 1.

$\mathsf{KeyGen}(1^\lambda)$: Upon input the security parameter λ, do the following:

1. Sample the key of PRF F as $\mathsf{k}_{\mathsf{prf}} \leftarrow \{0,1\}^r$.
2. Sample $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ and $\mathbf{S} \leftarrow \{-d, \ldots, 0, \ldots, d\}^{m \times k}$.
3. Set $\mathbf{T} = \mathbf{AS}$.
4. Output $\mathsf{vk} = (\mathbf{A}, \mathbf{T})$, $\mathsf{sk} = (\mathsf{k}_{\mathsf{prf}}, \mathbf{S})$.

$\mathsf{Sign}(\mathsf{sk}, \mu)$: Upon input the signing key sk and a message μ, do the following:

1. Generate message specific randomness by setting $u = F(\mathsf{k}_{\mathsf{prf}}, \mu)$.
2. Sample $\mathbf{y} \leftarrow \mathcal{D}_\sigma^m$ using randomness u computed in the previous step.
3. Set $\mathbf{c} = H(\mathbf{Ay}, \mu)$.
4. Set $\mathbf{z} = \mathbf{y} + \mathbf{Sc}$.
5. Output $(\mathbf{z}, \mathbf{c})$.

$\mathsf{Verify}(\mathsf{vk}, \mu, (\mathbf{z}, \mathbf{c}))$: Upon input the verification key vk, a message μ, and a signature $(\mathbf{z}, \mathbf{c})$, do the following:

1. Check if $\|\mathbf{z}\| \leq \gamma$, where $\gamma = (\sigma + \alpha d)\sqrt{m}$.
2. Check if $H(\mathbf{Az} - \mathbf{Tc}, \mu) = \mathbf{c}$.
3. If both checks pass, then accept.

Figure 1 Lyubashevsky's Signature Without Aborts

Correctness. Since $\mathbf{z} = \mathbf{y} + \mathbf{Sc}$, where $\mathbf{y} \leftarrow \mathcal{D}_\sigma^m$, we get $\mathbf{z} \leftarrow \mathcal{D}_{\sigma, \mathbf{Sc}}^m$. Hence, we have $\|\mathbf{z}\| \leq \sigma\sqrt{m} + \|\mathbf{Sc}\|$ with probability $1 - 2^{-\Omega(m)}$, using standard Gaussian tail bounds. Since $\|\mathbf{S}\|_\infty \leq d$ and $\|\mathbf{c}\|_1 \leq \alpha$, we have $\|\mathbf{Sc}\| \leq d\alpha\sqrt{m}$. This gives us $\|\mathbf{z}\| \leq (\sigma + d\alpha)\sqrt{m}$ with overwhelming probability. Finally, note that

$$H(\mathbf{Az} - \mathbf{Tc}, \mu) = H(\mathbf{A}(\mathbf{y} + \mathbf{Sc}) - \mathbf{ASc}, \mu) = H(\mathbf{Ay}, \mu) = \mathbf{c}.$$

Security. Next, we establish security via the following theorem.

Theorem 3.1. *Assume that ℓ_2-$\mathsf{SIS}_{q,n,m,\beta}$ for $\beta = 2\gamma + 2d\alpha\sqrt{m}$ is hard, that F is a secure PRF, and that $\sigma \geq \alpha d\sqrt{mQ}$, where Q is the maximum number of signing queries an attacker can make. Then the above construction of signature scheme satisfies unforgeability in the ROM.*

We prove the security via the following hybrids:

Hybrid_0: This is the real world, i.e., with honest executions of the Sign algorithm on signing queries by the adversary.

Hybrid_1: In this hybrid, the vector $\mathbf{y}$ is sampled directly from $\mathcal{D}_\sigma^m$.

Hybrid_2: In this hybrid the challenger responds to the signing query for any message μ as follows:

1. Sample $\mathbf{y} \leftarrow \mathcal{D}_\sigma^m$ (using genuine randomness u).
2. Sample $\mathbf{c} \leftarrow \{\mathbf{v} : \mathbf{v} \in \{-1, 0, 1\}^k, \|\mathbf{v}\|_1 \leq \alpha\}$.
3. Set $\mathbf{z} = \mathbf{y} + \mathbf{Sc}$.
4. Set $H(\mathbf{Az} - \mathbf{Tc}, \mu) = \mathbf{c}$.
5. Output $(\mathbf{z}, \mathbf{c})$.

Hybrid_3: In this hybrid the challenger responds to the signing query for any message μ as follows:

1. Sample $\mathbf{c} \leftarrow \{\mathbf{v} : \mathbf{v} \in \{-1, 0, 1\}^k, \|\mathbf{v}\|_1 \leq \alpha\}$.
2. Sample $\mathbf{z} \leftarrow \mathcal{D}_\sigma^m$.
3. Set $H(\mathbf{Az} - \mathbf{Tc}, \mu) = \mathbf{c}$.
4. Output $(\mathbf{z}, \mathbf{c})$.

Claim 3.2. *Assume that F is a secure PRF. Then Hybrid_0 is indistinguishable from Hybrid_1.*

Proof. The only difference between the two hybrids is how u is sampled: using a PRF in Hybrid_0, while sampled uniformly in Hybrid_1. Hence the two hybrids are indistinguishable because of the PRF security. □

Claim 3.3 ([53, Lemma 5.3])**.** *Let $\mathcal{D}$ be any distinguisher who distinguishes Hybrid_1 and Hybrid_2 and makes Q_H oracle queries for H and Q_S signing queries. Then for $(1 - e^{-\Omega(n)})$ fraction of all possible matrices $\mathbf{A}$, the distinguishing advantage of $\mathcal{D}$ is at most $Q_S(Q_S + Q_H)2^{-n}$.*

Claim 3.4. *If there is an adversary who makes at most Q_S signing queries and can win the game in Hybrid_2 with probability δ, then the probability of winning in Hybrid_3 is polynomial in δ, if $\sigma \geq \alpha d\sqrt{mQ_S}$.*

Proof. The only difference between the two hybrids is in the value of $\mathbf{z}$. For $1 \leq i \leq Q_S$, in Hybrid_2, we have $\mathbf{z}_i = \mathbf{y}_i + \mathbf{Sc}_i$ with $\mathbf{y}_i \leftarrow \mathcal{D}_\sigma^m$, while in Hybrid_3, we have $\mathbf{z}_i \leftarrow \mathcal{D}_\sigma^m$. Let us refer to the joint distribution of all $\mathbf{z}$'s in Hybrid_2 as D_1 and that in Hybrid_3 as D_2. Let E denote the event that the

adversary wins the game. Then by our assumption, we have $D_1(E) = \delta$. From the probability preservation property (Lemma 2.13) of the Rényi Divergence, we get:

$$D_2(E) \geq \frac{\delta^{\frac{a}{a-1}}}{R_a(D_1 \| D_2)}, \text{ for any } a \in (1, \infty). \tag{3.1}$$

Computing $R_a(D_1\|D_2)$: For $1 \leq i \leq Q_S$, the vector $\mathbf{z}_i$ is from distribution $D_{1i} := \mathcal{D}_{\sigma,\mathbf{Sc}_i}$ in Hybrid_2 and from distribution $D_{2i} = \mathcal{D}_\sigma^m$ in Hybrid_3. Note that $D_1 = (D_{11}, \ldots, D_{1Q_S})$ and $D_2 = (D_{21}, \ldots, D_{2Q_S})$. By Lemma 2.14, we have

$$R_a[D_{1i}\|D_{2i}] = \exp\left(a\pi \frac{\|\mathbf{Sc}_i\|^2}{\sigma^2}\right) \quad \text{for any } a \in (1, \infty).$$

As seen in the correctness proof, we have $\|\mathbf{Sc}_i\| \leq d\alpha\sqrt{m}$. By using the multiplicativity property of the Rényi divergence (Lemma 2.13), we get:

$$R_a(D_1\|D_2) \leq \exp\left(a\pi \frac{Q_S(d\alpha\sqrt{m})^2}{\sigma^2}\right), \quad \text{for any } a \in (1, \infty). \tag{3.2}$$

Using the assumption $\sigma \geq d\alpha\sqrt{m} \cdot \sqrt{Q_S}$, we get $R_a(D_1\|D_2) \leq \exp(a\pi)$. Using Equation (3.1), we obtain that $D_2(E) \geq \delta^{\frac{a}{a-1}} \exp(-a\pi)$. Taking any value of $a > 1$ provides the result.

□

Claim 3.5. *If there is a forger $\mathcal{F}$ who makes at most Q_S signing queries and Q_H queries to the random oracle and succeeds in forging a valid signature with probability δ in Hybrid_3, then we can define an algorithm $\mathcal{B}$ which given $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$, finds a non-zero $\mathbf{v}$ such that $\|\mathbf{v}\| \leq (2\gamma + 2d\alpha\sqrt{m})$ and $\mathbf{Av} = \mathbf{0}$ with probability at least*

$$\left(\frac{1}{2} - \frac{\epsilon}{2}\right)\left(\delta - \frac{1}{|D_H|}\right)\left(\frac{\delta - 1/|D_H|}{Q_H + Q_S} - \frac{1}{|D_H|}\right).$$

Here D_H denotes the range of the hash function and $\epsilon = \frac{q^n}{(2d+1)^m}$, which for $m > \lambda + (n \log q)/(\log(2d+1)$, is below $2^{-\lambda}$.

This claim and its proof are identical to [53, Lemma 5.4]. We provide it in Appendix B.1 for the sake of completeness. In Appendix B.2, we show that the flooding noise used in the above construction is essentially optimal, and in particular that the dependence on $\sqrt{Q_S}$ is necessary, by exhibiting an attack when flooding noise is smaller.

3.2 From Gaussian to Uniform

In some applications, it may be preferable to use a vector $\mathbf{y}$ whose coordinates are uniform in some interval $[-B, B]$ rather than Gaussian (at Step 2 of the `Sign` algorithm). This is the choice made for the regular Dilithium signature scheme [27]. Looking ahead, if the sampling of $\mathbf{y}$ is done under a homomorphic encryption layer, this may be significantly simpler to implement.

To adapt the current proof, the only step that needs to be modified is in the transition between Hybrid 2 and Hybrid 3. A difficulty is that the support of the distribution of $\mathbf{z}$ in Hybrid 3 has to contain the support of the distribution of $\mathbf{z}$ in Hybrid 2, for the Rényi divergence to be defined. For this

purpose, we consider a wider interval in Hybrid 3, which contains all possible intervals $[-B, B]^m + \mathbf{Sc}$ of Hybrid 2. Concretely, in Hybrid 2, the vector $\mathbf{z}$ is sampled from $D_1 = U([-B, B]^m) + \mathbf{Sc}$, whereas in Hybrid 3, the vector $\mathbf{z}$ is sampled from $D_2 = U([-B', B']^m)$. As $\|\mathbf{Sc}\|_\infty \leq \alpha d$, we can take $B' = B + \alpha d$. Assuming that both B and B' are integers, we have

$$R_a(D_1 \| D_2) = \left(\frac{2B'+1}{2B+1}\right)^{Q_S m} \leq \left(1 + \frac{2\alpha d}{B}\right)^{Q_S m}, \quad \text{for any } a \in (1, \infty).$$

We obtain that for transiting from Hybrid 2 to Hybrid 3, it suffices to set $B \geq \Omega(md\alpha Q_S)$.

3.3 Instantiation

In this section, we analyze the concrete size of the above signature. For efficiency, it is preferable to consider variants over polynomial rings [52] (note that the above proofs carry over directly to the ring and module settings). We consider the Dilithium-G signature scheme from [28, Appendix B]. It is a variant of Dilithium [27], a concrete implementation of Lyubashevsky's signature scheme designed to reach a compromise between efficiency and simplicity. Dilithium-G has slightly smaller signature sizes than Dilithium, but requires that $\mathbf{y}$ has Gaussian coordinates rather than having them uniform in an interval. This makes it more cumbersome to implement, but our analysis is directly applicable. We note that for the application to threshold signatures described in Section 4, regular Dilithium would be more suitable, as the sampling of $\mathbf{y}$ happens below the HE layer. Note that for the blind signature application described in Section 6, the sampling of $\mathbf{y}$ is done in the clear.

Rather than defining new parameter sets for Dilithium-G (which would most likely be preferable for target concrete applications), we show that Dilithium-G already offers significant security when removing rejection sampling. We consider two variants of Dilithium-G: with the Bai-Galbraith [6] signature truncation technique as in [28], and without it.[3] Indeed, a variant without the truncation technique may be more convenient to evaluate homomorphically.

For security, we place ourselves in the (classical/quantum) core-SVP hardness model, like in [28]. State of the art cryptanalysis suggests that bit security is expected to be at least 10-15 bits higher than core-SVP security (see [69, Section 5.2]). We consider the 'medium', 'recommended' and 'very high' parameter sets from [28]. For a given value of Q_S, and use the analysis above to derive core-SVP hardness estimates for this number of signature queries. Concretely, we use Equations (3.1) and (3.2) with $\delta \approx 1$ (which can be assumed in the multi-user setting, and can also be justified by considering known attack strategies [28]) and $a \approx 1$ (which minimizes the upper bound on the Rényi divergence). By adapting the figures from [28, Table 2], we obtain Table 1.
For example, using the 'very high' parameter set, we lose no more than 100 bits of security by removing rejection sampling, when the number of signature queries is limited to 256. In this setting, the overall classical bit security remains close to 128 bits (more precisely, 118 bits of classical core-SVP hardness). And the signature size is below 3KB.

4 More Efficient Threshold Signatures from Lattices

In this section, we show how to drastically decrease the exponential flooding used in the scheme by Boneh *et al* [12]. We also show that the limited flooding that we use is in fact optimal, and

[3] Concretely, Algorithm 14 from [28] is modified by removing Step 6, replacing $\mathbf{w_1}$ by $\mathbf{w}$ in Step 7, removing Step 13 and replacing $\mathbf{h}$ by $\mathbf{z_2}$ in Step 14. Algorithm 15 is updated accordingly.

	medium	recommended	very high
Standard deviation σ of $\mathbf{z}$	48127	44868	31082
Dimension m of $\mathbf{z}$	1536	2048	2560
Secret key magnitude bound d	6	5	3
Hamming weight α of $\mathbf{c}$	60	60	60
pk size	1184	1568	1952
sig size with truncation	1850	2435	2950
sig size without truncation	3200	4225	5105
Security with rejection	109 / 99	162 / 147	228 / 206
Security loss with $Q_S = 64$	25	27	25
Security loss with $Q_S = 128$	50	53	50
Security loss with $Q_S = 256$	100	106	100

Table 1: Security of Dilithium-G without rejection sampling. Sizes are given in bytes, security corresponds to classical/quantum core-SVP hardness, security losses are in bits. Note that due to a different Gaussian normalization, our Gaussian standard deviation is $\sqrt{2\pi}$ times higher than in [28, Table 2]. Adapting the methodology from [28, Appendix B.5], we bound the no-truncation signature size by $32 + (2.25 + \log_2(\sigma/\sqrt{2\pi})) \cdot m/8$ bytes.

smaller noise would lead to an attack. For ease of exposition, the construction below is for the special case of N out of N threshold (where all the N members need to collaborate to construct a valid signature on any message M) and restricted to selective security. We extend it to t out of N threshold in Appendix D and to adaptive security in Section 5.

4.1 Optimizing the Boneh *et al* scheme using the Rényi Divergence

Our scheme is similar to the one in [12]. We improve it in terms of flooding amount for which we use the Rényi Divergence as a measure of distance between two distributions. The construction uses the following building blocks:

- A PRF $F : \mathcal{K} \times \{0,1\}^* \to \{0,1\}^r$, where $\mathcal{K}$ is the PRF key space and r is the bit-length of randomness used in sampling from discrete Gaussian $\mathcal{D}_s$.
- A homomorphic encryption scheme $\mathsf{HE} = (\mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Dec}, \mathsf{HE.Eval})$. As in [12], we also assume that the $\mathsf{HE.Dec}$ can be divided into two sub-algorithms: $\mathsf{HE.decode}_0$ and $\mathsf{HE.decode}_1$ as defined in Appendix 2.6.
- A UF-CMA signature scheme $\mathsf{Sig} = (\mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$.
- A context hiding homomorphic signature scheme $\mathsf{HS} = (\mathsf{HS.PrmsGen}, \mathsf{HS.KeyGen}, \mathsf{HS.Sign}, \mathsf{HS.SignEval}, \mathsf{HS.Process}, \mathsf{HS.Verify}, \mathsf{HS.Hide}, \mathsf{HS.HVerify})$ to provide robustness.
- An N out of N secret sharing scheme Share.

4.1.1 Construction.

We proceed to describe our construction.

$\mathsf{TS.KeyGen}(1^\lambda)$: Upon input the security parameter λ, do the following.

1. For each party P_i, sample a PRF key $\mathsf{sprf}_i \leftarrow \mathcal{K}$.
2. Generate the signature scheme keys $(\mathsf{Sig.vk}, \mathsf{Sig.sk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$.
3. Generate the HE keys $(\mathsf{HE.PK}, \mathsf{HE.SK}) \leftarrow \mathsf{HE.KeyGen}(1^\lambda)$ and compute an HE encryption of $\mathsf{Sig.sk}$ as $\mathsf{CT}_{\mathsf{Sig.sk}} \leftarrow \mathsf{HE.Enc}(\mathsf{HE.PK}, \mathsf{Sig.sk})$.
4. Generate the HS public parameters $\mathsf{HS.pp} \leftarrow \mathsf{HS.PrmsGen}(1^\lambda, 1^n)$ and the public and the signing keys $(\mathsf{HS.pk}, \mathsf{HS.sk}) \leftarrow \mathsf{HS.KeyGen}(1^\lambda, \mathsf{HS.pp})$. Here n is the bit-length of $(\mathsf{HE.SK}, \mathsf{sprf}_i)$.
5. Share $\mathsf{HE.SK}$ as $\{\mathsf{sk}_i\}_{i=1}^N \leftarrow \mathsf{Share}(\mathsf{HE.SK})$ such that $\sum_{i=1}^N \mathsf{sk}_i = \mathsf{HE.SK}$.
6. For each party P_i, randomly choose a tag $\tau_i \in \{0,1\}^*$ and compute $(\pi_{\tau_i}, \pi_i) \leftarrow \mathsf{HS.Sign}(\mathsf{HS.sk}, (\mathsf{sk}_i, \mathsf{sprf}_i), \tau_i)$.
7. Output $\mathsf{TSig.pp} = \{\mathsf{HE.PK}, \mathsf{CT}_{\mathsf{Sig.sk}}, \mathsf{HS.pp}, \mathsf{HS.pk}, \{\tau_i, \pi_{\tau_i}\}_{i=1}^N\}$, $\mathsf{TSig.vk} = \mathsf{Sig.vk}$, $\mathsf{TSig.sk} = \{\mathsf{sk}_i, \mathsf{sprf}_i, \pi_i, \}_{i=1}^N$.

$\mathsf{TS.PartSign}(\mathsf{TSig.pp}, \mathsf{TSig.sk}_i, M)$: Upon input the public parameters $\mathsf{TSig.pp}$, a partial signing key $\mathsf{TSig.sk}_i$ and a message M, parse $\mathsf{TSig.pp}$ as $(\mathsf{HE.PK}, \mathsf{CT}_{\mathsf{Sig.sk}}, \mathsf{HS.pp}, \mathsf{HS.pk}, \{\tau_i, \pi_{\tau_i}\}_{i=1}^N\})$, parse $\mathsf{TSig.sk}_i$ as $(\mathsf{sk}_i, \mathsf{sprf}_i, \pi_i)$ and do the following.

1. Compute $u = F(\mathsf{sprf}_i, M)$ and sample $e_i' \leftarrow \mathcal{D}_s(u)$, where $\mathcal{D}_s(u)$ represents sampling from $\mathcal{D}_s$ using u as the randomness.
2. Let $\mathcal{C}_M$ be the signing circuit, with message M being hardwired. Compute $\mathsf{CT}_\sigma = \mathsf{HE.Eval}(\mathsf{HE.PK}, \mathcal{C}_M, \mathsf{CT}_{\mathsf{Sig.sk}})$.
3. Compute $\sigma_i = \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_\sigma) + e_i'$.
4. This step computes a homomorphic signature $\tilde{\pi}_i$ on partial signature σ_i to provide robustness).
   Let $\mathcal{C}_{\mathsf{PS}}$ be the circuit to compute $\mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_\sigma) + e_i'$ in which CT_σ is hardcoded and the HE key share sk_i and the PRF key sprf_i are given as inputs.
   - Compute $\pi_i^* = \mathsf{HS.SignEval}(\mathsf{HS.pp}, \mathcal{C}_{\mathsf{PS}}, \pi_{\tau_i}, (\mathsf{sk}_i, \mathsf{sprf}_i), \pi_i)$.
   - Compute $\tilde{\pi}_i = \mathsf{HS.Hide}(\mathsf{HS.pk}, \sigma_i, \pi_i^*)$.
5. Output $y_i = (\sigma_i, \tilde{\pi}_i)$.

$\mathsf{TS.Combine}(\mathsf{TSig.pp}, \{y_i\}_{i\in[N]})$: Upon input the public parameters $\mathsf{TSig.pp}$ and a set of partial signatures $\{y_i\}_{i\in[N]}$, parse y_i as $(\sigma_i, \tilde{\pi}_i)$ and output $\sigma_M = \mathsf{HE.decode}_1(\sum_{i=1}^N \sigma_i)$.

$\mathsf{TS.PartSignVerify}(\mathsf{TSig.pp}, M, y_i)$: On input the public parameters $\mathsf{TSig.pp}$, message M, and a partial signature y_i, parse y_i as $(\sigma_i, \tilde{\pi}_i)$ and do the following.

1. Compute $\mathsf{CT}_\sigma = \mathsf{HE.Eval}(\mathsf{HE.PK}, \mathcal{C}_M, \mathsf{CT}_{\mathsf{Sig.sk}})$.
2. Compute $\alpha = \mathsf{HS.Process}(\mathsf{HS.pp}, \mathcal{C}_{\mathsf{PS}})$, where $\mathcal{C}_{\mathsf{PS}}$ is as described above.
3. Output $\mathsf{HS.HVerify}(\mathsf{HS.pk}, \alpha, \sigma_i, \tau_i, (\pi_{\tau_i}, \tilde{\pi}_i))$.

$\mathsf{TS.Verify}(\mathsf{TSig.vk}, M, \sigma_M)$: Upon input the verification key $\mathsf{TSig.vk}$, a message M and a signature σ_M, output $\mathsf{Sig.Verify}(\mathsf{TSig.vk}, M, \sigma_M)$.

In the above, we set $s = B_{eval} \cdot \sqrt{Q\lambda}$, where where $B_{eval} \leq \mathrm{poly}(\lambda)$ is a bound on the HE decryption noise after homomorphic evaluation of the signing circuit $\mathcal{C}_M$, and Q is the bound on the number of signatures.

Correctness. From the correctness of $\mathsf{HE.Eval}$ algorithm, CT_σ is an encryption of $\mathcal{C}_M(\mathsf{Sig.sk}) = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, M) = \sigma_M$, which decrypts with the HE secret key $\mathsf{HE.SK}$. So, $\mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_\sigma) = \sigma_M \lfloor q/2 \rceil + e$. The signature computed by the $\mathsf{TS.Combine}$ algorithm is

$$\begin{aligned}
\mathsf{HE.decode}_1(\sum_{i=1}^{N} \sigma_i) &= \mathsf{HE.decode}_1(\sum_{i=1}^{N} \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_\sigma) + \sum_{i=1}^{N} e'_i) \\
&= \mathsf{HE.decode}_1(\mathsf{HE.decode}_0(\sum_{i=1}^{N} \mathsf{sk}_i, \mathsf{CT}_\sigma) + \sum_{i=1}^{N} e'_i) \\
&= \mathsf{HE.decode}_1(\mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_\sigma) + \sum_{i=1}^{N} e'_i) \\
&= \mathsf{HE.decode}_1(\sigma_M \lfloor q/2 \rceil + e + \sum_{i=1}^{N} e'_i) = \sigma_M
\end{aligned}$$

Security. For security, we prove the following theorem.

Theorem 4.1. *Assume F is a secure PRF, Sig is a UF-CMA secure signature scheme, HE is a secure homomorphic encryption scheme (Definition 2.18), Share is a secret sharing scheme that satisfies privacy (Definition 2.26) and HS is a context hiding secure homomorphic signature scheme (Definitions A.4 and A.3). Then the construction for threshold signatures satisfies unforgeablity if the flooding noise is of the size $\mathrm{poly}(\lambda) \cdot \sqrt{Q}$, where Q is the number of the signing queries.*

The security of the construction can be argued using a sequence of hybrids. We assume w.l.o.g. that the adversary queries for all but the first key share, i.e., $S = [N] \setminus \{1\}$.

Hybrid_0: This is the real world.

Hybrid_1: Same as Hybrid_0, except that now $\mathsf{PartSign}$ differs in the computation of $\tilde{\pi}_1$ in $y_1 = (\sigma_1, \tilde{\pi}_1)$. To generate $\tilde{\pi}_1$, the challenger now uses the simulator for HS as follows:

$$\tilde{\pi}_1 = \mathsf{Sim}(\mathsf{HS.sk}, \alpha, \sigma_1, \tau_1, \pi_{\tau_1}),$$

where $\alpha = \mathsf{HS.Process}(\mathsf{HS.pp}, \mathcal{C}_{\mathsf{PS}})$.

Hybrid_2: Same as Hybrid_1 except that to compute $\sigma_1 = \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_\sigma) + e'_1$, the randomness u used to sample $e'_1 \leftarrow \mathcal{D}_s(u)$ is chosen uniformly randomly instead of computing it using the PRF.

Hybrid_3: Same as Hybrid_2, except that now, for signing query for $(m, 1)$, the challenger simulates σ_1 as follows:

1. Compute $\mathsf{CT}_\sigma = \mathsf{HE.Eval}(\mathsf{HE.PK}, \mathcal{C}_m, \mathsf{CT}_{\mathsf{Sig.sk}})$.
2. For $i \in [2, n]$, compute $\sigma'_i = \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_\sigma)$.
3. Compute $\sigma_m = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, m)$.
4. Set $\sigma_1 = \sigma_m \cdot \lfloor q/2 \rceil - \sum_{i=2}^{N} \sigma'_i + e'_1$, where $e'_1 \leftarrow \mathcal{D}_s$.

Hybrid_4: Same as Hybrid_3 except that instead of sharing $\mathsf{HE.SK}$, now the challenger generates the HE key shares as $\{\mathsf{sk}_i\}_{i=1}^{N} \leftarrow \mathsf{Share}(\mathbf{0})$.

Hybrid_5: Same as Hybrid_4, except that $\mathsf{CT}_{\mathsf{Sig.sk}}$ in $\mathsf{TSig.pp}$ is replaced by $\mathsf{CT}_{\mathbf{0}}$, i.e., a ciphertext of $\mathbf{0}$.

Indistinguishability of Hybrids. Next, we show that consecutive hybrids are indistinguishable.

Claim 4.2. *Assume* HS *is a context hiding homomorphic signature scheme. Then,* Hybrid_0 *and* Hybrid_1 *are indistinguishable.*

Proof. The two hybrids differ only in the way $\tilde{\pi}_1$ is computed. In Hybrid_0, $\tilde{\pi}_1 = \mathsf{HS.Hide}(\mathsf{HS.pk}, \sigma_1, \pi_1^*)$, where $\pi_1^* = \mathsf{HS.SignEval}(\mathsf{HS.pp}, \mathcal{C}_{\mathsf{PS}}, \pi_{\tau_1}, (\mathsf{sk}_1, \mathsf{sprf}_1), \pi_1)$. In Hybrid_1, $\tilde{\pi}_1 = \mathsf{Sim}(\mathsf{HS.sk}, \alpha, \sigma_1, \tau_1, \pi_{\tau_1})$, where $\alpha = \mathsf{HS.Process}(\mathsf{HS.pp}, \mathcal{C}_{\mathsf{PS}})$. Hence, the two hybrids are indistinguishable because of context hiding property of HS which ensures that $\mathsf{HS.Hide}(\mathsf{HS.pk}, \sigma_1, \pi_1^*) \approx \mathsf{Sim}(\mathsf{HS.sk}, \alpha, \sigma_1, \tau_1, \pi_{\tau_1})$.

□

Claim 4.3. *Assume* F *is a secure PRF. Then* Hybrid_1 *and* Hybrid_2 *are indistinguishable.*

The proof follows via a standard reduction to PRF security and is omitted.

Claim 4.4. *If there is an adversary that can win the game in* Hybrid_2 *with probability* ϵ*, then its probability of winning in* Hybrid_3 *is at least* $\epsilon^2/2$*.*

Proof. Let the number of signing queries that the adversary can make be bounded by Q. The two hybrids differ only in the error term in σ_1, as shown below. In Hybrid_2, we have $\sigma_1 = \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_\sigma) + e'_1$, for $e'_1 \leftarrow \mathcal{D}_s$. In Hybrid_3, we have:

$$
\begin{aligned}
\sigma_1 &= \sigma_m . \lfloor q/2 \rceil - \sum\nolimits_{i=2}^{N} \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_\sigma) + e'_1 \\
&= \sigma_m . \lfloor q/2 \rceil - \sum\nolimits_{i=1}^{N} \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_\sigma) + \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_\sigma) + e'_1 \\
&= \sigma_m . \lfloor q/2 \rceil - \mathsf{HE.decode}_0(\sum\nolimits_{i=1}^{N} \mathsf{sk}_i, \mathsf{CT}_\sigma) + \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_\sigma) + e'_1 \\
&= \sigma_m . \lfloor q/2 \rceil - \mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_\sigma) + \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_\sigma) + e'_1 \\
&= \sigma_m . \lfloor q/2 \rceil - \sigma_m . \lfloor q/2 \rceil + e + \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_\sigma) + e'_1 \\
&= \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_\sigma) + (e'_1 + e),
\end{aligned}
$$

for some e satisfying $|e| \leq B_{eval}$. Thus, the difference in the two hybrids is in the error terms in σ_1. In Hybrid_2, the error is e'_1, while in Hybrid_3, it is $e'_1 + e$. Since e'_1 is sampled from a discrete Gaussian with standard deviation parameter s, we consider the distributions $\mathcal{D}_{s,0}$ and $\mathcal{D}_{s,e}$. For simplicity, we refer to the former as D_0 and the latter as D_1.

To begin, consider an adversary that makes a single query. In this case, let E represents the event that the adversary wins the game. We assumed that $D_0(E) = \epsilon$. From the probability preservation property of the Rényi divergence (Lemma 2.13), we have:

$$D_1(E) \geq \frac{D_0(E)^{\frac{a}{a-1}}}{R_a(D_0\|D_1)}, \text{ for any } a \in (1,\infty). \tag{4.1}$$

By Lemma 2.14, we have that $R_a(D_0\|D_1) = \exp(a\pi\frac{e^2}{s^2})$.

The error term e in the above discussion is the error in HE ciphertext CT_σ and thus depends on the evaluation circuit $\mathcal{C}_m$. Notice that the adversary can make adaptive queries based on the response obtained for previous queries. Therefore, the centers of discrete Gaussian distributions in the second distribution can be correlated. Thus, for Q queries,

$$R_a(D_0\|D_1) = R_a(\underbrace{\mathcal{D}_s,\ldots,\mathcal{D}_s}_{Q}\|\mathcal{D}_{s,e_1},\ldots,\mathcal{D}_{s,e_Q}),$$

where $e_1,\ldots,e_Q$ can be correlated. We can replace the Q Gaussian distributions with standard deviation parameter s and centers $e_1,\ldots,e_Q$ with a single Q-dimensional Gaussian distribution with center $\mathbf{e} = (e_1,\ldots,e_Q)$ and standard deviation parameter s. Thus, we get $R_a(D_0\|D_1) = R_a(\mathcal{D}_{\mathbb{Z}^Q,s,\mathbf{0}}\|\mathcal{D}_{\mathbb{Z}^Q,s,\mathbf{e}})$, for some vector $\mathbf{e}$ satisfying $\|\mathbf{e}\| \leq \sqrt{Q}\cdot B_{eval}$. Using Lemma 2.14, we have $R_a(\mathcal{D}_{\mathbb{Z}^Q,s,\mathbf{0}}\|\mathcal{D}_{\mathbb{Z}^Q,s,\mathbf{e}}) = \exp(a\pi\|\mathbf{e}\|^2/s^2)$. Hence, we obtain $R_a(D_0\|D_1) \leq \exp(a\pi Q\cdot B_{eval}^2/s^2)$, for all $a \in (1,\infty)$. As $s = B_{eval}\cdot\sqrt{Q\lambda}$, we get $R_a(D_0\|D_1) \leq \exp(a\pi/\lambda)$. Therefore, from Equation (4.1), we have

$$D_1(E) \geq \frac{D_0(E)^{\frac{a}{a-1}}}{R_a(D_0\|D_1)} \geq D_0(E)^{\frac{a}{a-1}}\exp(-\frac{a\pi}{\lambda}).$$

The result is obtained by setting $a = 2$. □

Claim 4.5. *Assume that* Share *is a secret sharing scheme that satisfies privacy (Definition 2.26). Then,* Hybrid_3 *and* Hybrid_4 *are indistinguishable.*

Proof. The only difference between Hybrid_3 and Hybrid_4 is in the way the key shares $\mathsf{sk}_1,\mathsf{sk}_2,\ldots,\mathsf{sk}_N$ are generated. In Hybrid_3 $(\mathsf{sk}_1,\mathsf{sk}_2,\ldots,\mathsf{sk}_N)\leftarrow\mathsf{Share}(\mathsf{HE.SK})$, while in Hybrid_4, we have that $(\mathsf{sk}_1,\mathsf{sk}_2,\ldots,\mathsf{sk}_N)\leftarrow\mathsf{Share}(\mathbf{0})$. Since, the adversary is given secret shares for an invalid set of parties, distribution in the two hybrids are identical. □

Claim 4.6. *Assume the* HE *is a secure homomorphic encryption (Definition 2.18). Then* Hybrid_4 *and* Hybrid_5 *are indistinguishable.*

Proof. Let $\mathcal{A}$ be an adversary who can distinguish Hybrid_4 and Hybrid_5. Then we construct an adversary $\mathcal{B}$ against the HE scheme as follows.

1. After receiving $\mathsf{HE.PK}$ from the HE challenger, adversary $\mathcal{B}$ generates $(\mathsf{Sig.sk},\mathsf{Sig.vk})\leftarrow\mathsf{Sig.KeyGen}(1^\lambda)$ and HS keys.

2. It generates secret shares of $\mathbf{0}$ as $(\mathsf{sk}_1,\mathsf{sk}_2,\ldots,\mathsf{sk}_N)\leftarrow\mathsf{Share}(\mathbf{0})$.

3. It sends the challenge messages $m_0 = \mathsf{Sig.sk}$ and $m_1 = 0$ to the HE challenger.

4. After receiving the challenge ciphertext CT_b from the HE challenger, adversary $\mathcal{B}$ constructs $\mathsf{TSig.pp}$ using CT_b. It also generates $\mathsf{TSig.vk}$ and $\mathsf{TSig.sk}$ as defined for the hybrid.

5. To answer a $\mathsf{PartSign}$ query for a message m made by adversary $\mathcal{A}$, adversary $\mathcal{B}$ computes σ_1 as follows. It computes $\sigma = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, m)$; for $i \in [2, N]$, it computes $\sigma_i = \mathsf{TS.decode}_0(\mathsf{TSig.sk}_i, \mathsf{CT}_b)$ and samples a flooding noise $e'_1 \leftarrow \mathcal{D}_{s,0}$. It sets $\sigma_1 = \sigma - \sum_{i=2}^{N} \sigma_i + e'_1$.

6. Finally, if $\mathcal{A}$ outputs the guess as Hybrid_4, then $\mathcal{B}$ sends $b' = 0$, else $b' = 1$ to the HE challenger.

□

Claim 4.7. *If the underlying signature scheme* Sig *is unforgeable, then an attacker has negligible probability of winning the unforgeability game in* Hybrid_5.

Proof. Let $\mathcal{A}$ be an adversary who wins the unforgeability game in Hybrid_5. Then we can construct an adversary $\mathcal{B}$ against the signature scheme Sig as follows:

1. On receiving verification key $\mathsf{Sig.vk}$ from Sig challenger, adversary $\mathcal{B}$ runs $\mathsf{HE.KeyGen}$ to generate $\mathsf{HE.SK}, \mathsf{HE.PK}$, and $\mathsf{HS.PrmsGen}$ and $\mathsf{HS.KeyGen}$ to generate $\mathsf{HS.pp}$, $\mathsf{HS.pk}$ and $\mathsf{HS.sk}$. Then $\mathcal{B}$ computes TS public parameters, $\mathsf{TSig.pp}$, key shares $\{\mathsf{TSig.sk}_i\}_{i=1}^{N}$ as defined for the hybrid. Note that since in Hybrid_5, $\mathsf{TSig.pp}$ contains $\mathsf{CT_0}$ instead of $\mathsf{CT}_{\mathsf{Sig.sk}}$, $\mathcal{B}$ can generate a valid $\mathsf{TSig.pp}$. It sends $\{\mathsf{TSig.vk} = \mathsf{Sig.vk}, \mathsf{TSig.pp}, \{\mathsf{TSig.sk}_i\}_{i=2}^{N}\}$ to $\mathcal{A}$.

2. To simulate $\mathsf{PartSign}$ query σ_1 for any message m, adversary $\mathcal{B}$ needs σ_m. For this, it issues a signing query on message m to the Sig challenger and receives σ_m.

3. Let (m^*, σ^*) be a message-signature pair returned by $\mathcal{A}$. Then $\mathcal{B}$ also returns the same pair to the Sig challenger.

Since $\mathcal{B}$ issues signing queries on only those messages m for which $\mathcal{A}$ also issues signing queries to $\mathcal{B}$, if (m^*, σ^*) is a valid forgery for $\mathcal{A}$, then it is a valid forgery for $\mathcal{B}$ as well. □

Robustness.

Claim 4.8. *If* HS *is multi data secure (Definition A.3) homomorphic signature, then the construction of* TS *satisfies robustness.*

Proof. In the robustness security experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{TS},rb}(1^\lambda)$, the adversary wins if $\mathcal{A}$ outputs a partial signature forgery (M^*, y_i^*, i) such that

1. $\mathsf{TS.PartSignVerify}(\mathsf{TSig.pp}, M^*, y_i^*) = 1$

2. $y_i^* = (\sigma_i^*, \tilde{\pi}_i^*) \neq \mathsf{TS.PartSign}(\mathsf{TSig.pp}, \mathsf{TSig.sk}_i, M^*)$.

$\mathsf{TS.PartSignVerify}(\mathsf{TSig.pp}, M^*, y_i^*)$ first computes $\mathsf{CT}_\sigma \leftarrow \mathsf{HE.Eval}(\mathsf{HE.PK}, \mathcal{C}_{M^*}, \mathsf{CT}_{\mathsf{Sig.sk}})$ and outputs 1 iff
$\mathsf{HS.HVerify}(\mathsf{HS.pk}, \alpha, \sigma_i^*, \tau_i, (\pi_{\tau_i}, \tilde{\pi}_i^*)) = 1$, where $\alpha = \mathsf{HS.Process}(\mathsf{HS.pp}, \mathcal{C}_{\mathsf{PS}})$. Thus, $\mathcal{A}$ wins the experiment iff both the following two conditions are true.

1. For $\alpha = \mathsf{HS.Process}(\mathsf{HS.pp}, \mathcal{C}_{\mathsf{PS}})$

$$\mathsf{HS.HVerify}(\mathsf{HS.pk}, \alpha, \sigma_i^*, \tau_i, (\pi_{\tau_i}, \tilde{\pi}_i^*)) = 1,$$

2. $y_i^* = (\sigma_i^*, \tilde{\pi}_i^*) \neq \mathsf{TS.PartSign}(\mathsf{TSig.pp}, \mathsf{TSig.sk}_i, M^*)$, which implies $\sigma_i^* \neq \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_\sigma) + e'_i$, which in turn is same as

$$\sigma_i^* \neq \mathcal{C}_{\mathsf{PS}}(\mathsf{sk}_i, \mathsf{sprf}_i).$$

But this is a case for valid forgery of type 2 (Definition A.3) against HS scheme, which can happen only with negligible probability. Note that since (τ_i, π_{τ_i}) are part of HS.pp, case of type 2 in HS security definition is inherently applied.

□

4.2 On the Optimality of Our Flooding

We show that the flooding amount that we achieved is optimal for our threshold signature scheme. To argue this, we show how to attack it if the flooding amount is below $\Omega(\sqrt{Q})$. For simplicity, we restrict to the case of $N = 2$. Recall that in our construction, $\mathsf{TS.PartSign}(\mathsf{TSig.pp}, \mathsf{TSig.sk}_i, M)$ outputs $\sigma_{i,M} = \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + e'_{i,M}$, where $\mathsf{TSig.sk}_i = (\mathsf{sk}_i, \mathsf{sprf}_i)$.[4] W.l.o.g, assume that the adversary gets the partial signing key $\mathsf{TSig.sk}_2$ and the response for any signing query is a partial signature corresponding to party P_1. For any message M of its choice, the adversary receives $\sigma_{1,M} = \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_{\sigma_M}) + e'_{1,M}$. From this the adversary can compute:

$$\begin{aligned} \sigma_{1,M} + \mathsf{HE.decode}_0(\mathsf{sk}_2, \mathsf{CT}_{\sigma_M}) &= \mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_{\sigma_M}) + e'_{1,M} \\ &= \sigma_M + \mathsf{err}_M + e'_{1,M}, \end{aligned}$$

where err_M is the error in CT_{σ_M}. Note that if the adversary succeeds in computing err_M for polynomially many M's, then it can compute HE.SK.

As a warm-up, we show that if the error $e'_{1,M}$ is randomized, small and of center 0, then the adversary can indeed compute err_M. Later, we will show that even for deterministic flooding $e'_{1,M}$, there exist secure signature schemes for which the attack can be extended. Since the adversary knows the key share sk_2, it can compute $\sigma_{2,M}$ on its own and hence can compute $\sigma_M = \mathsf{TS.Combine}(\mathsf{TSig.pp}, \sigma_{1,M}, \sigma_{2,M})$. Hence, from $\sigma_M + \mathsf{err}_M + e'_{1,M}$, the adversary can compute $\mathsf{err}_M + e'_{1,M}$. Since, the signature scheme is deterministic, err_M depends only on M. Thus, if the same message is queried for signature multiple times, then each time the term err_M remains the same, but since flooding is randomized, the term $e'_{1,M}$ is different.

To compute err_M, the adversary issues all Q signing queries for the same message M and receives $\sigma_{1,M}^{(1)}, \ldots, \sigma_{1,M}^{(Q)}$, where $\sigma_{1,M}^{(i)}$ denotes the partial signature returned for message M in the ith query. From these responses the adversary gets Q different values of the form

$$w^i = \mathsf{err}_M + e'^i_{1,M} \tag{4.2}$$

Since err_M is same, taking average on both sides of Equation (4.2) over all the Q samples, we get $\frac{\sum_{i\in[Q]} w^i}{Q} = \mathsf{err}_M + \frac{\sum_{i\in[Q]} e'^i_{1,M}}{Q}$. If $|\frac{1}{Q}\sum_{i\in[Q]} e'^i_{1,M}| < 1/2$, then the adversary can recover err_M as

[4] We focus only on the $\sigma_{i,M}$ component of PartSign's output since the second component, the HS signature of $\sigma_{i,M}$, is not relevant here.

$\mathsf{err}_M = \left\lfloor \frac{1}{Q} \sum_{i\in[Q]} w^i \right\rceil$. As $e'^1_{1,M}, \ldots, e'^Q_{1,M}$ are independently and identically distributed with mean 0, by Hoeffding's inequality, we have

$$\Pr\left[\left|\frac{\sum_{i\in[Q]} e'^i_{1,M}}{Q}\right| < 1/2\right] \geq 1 - 2\mathsf{exp}(-\frac{Q}{2s^2}).$$

If $Q \geq \Omega(s^2 \log \lambda)$, then $1-2\mathsf{exp}(-Q/(2s^2)) \geq 1-\lambda^{-\Omega(1)}$, in which case the adversary can recover err_M with probability sufficiently close to 1 to recover sufficiently many err_M's to compute $\mathsf{HE.SK}$. To prevent this, we do need s to grow at least proportionally to $\sqrt{Q}$. Next, we provide the argument for deterministic flooding error.

Attack for Deterministic Error. In the argument for randomized error, the fact that $e'^i_{1,M}$ is randomized is crucial - the same attack strategy would not work if $e'^i_{1,M}$ is deterministic, i.e., the same across different signature queries for the same message M. Note that it is possible to modify the scheme to make $e'^i_{1,M}$ deterministic, by deriving it from a PRF evaluated in M. However, as promised above, we show that in case of deterministic flooding too, there exist secure signature schemes for which a variant of the attack can (heuristically) apply.

We consider a special (contrived) signature scheme $\mathsf{Sig}' = (\mathsf{Sig}'.\mathsf{KeyGen}, \mathsf{Sig}'.\mathsf{Sign}, \mathsf{Sig}'.\mathsf{Verify})$ derived from a secure signature scheme $\mathsf{Sig} = (\mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$ as follows:

1. $\mathsf{Sig}'.\mathsf{KeyGen}$ is identical to $\mathsf{Sig.KeyGen}$.
   Let $(\mathsf{Sig.sk}, \mathsf{Sig.vk})$ be the signing and verification keys, respectively, and $\mathsf{Sig.sk}_i$ denote the ith bit of $\mathsf{Sig.sk}$ for $i \in [\ell]$, where ℓ is the bit-length of $\mathsf{Sig.sk}$.
2. $\mathsf{Sig}'.\mathsf{Sign}(\mathsf{Sig.sk}, M)$ is modified as follows:
   - Compute $\sigma_M = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, M)$. Set $\sigma'_M := \sigma_M$.
   - For i from 1 to ℓ: if $\mathsf{Sig.sk}_i = 0$, then set $\sigma'_M := \sigma'_M \| \mathsf{Sig.sk}_i$.
   - Output σ'_M.
3. $\mathsf{Sig}'.\mathsf{Verify}(\mathsf{Sig.vk}, M, \sigma'_M)$ is defined as $\mathsf{Sig.Verify}(\mathsf{Sig.vk}, M, \sigma_M)$, where σ_M is obtained from σ'_M by removing all bits after the kth bit, where k is the bit-length of signatures in Sig.

We assume that the signing key of Sig is a uniform bit-string among those with the same number of 0's and 1's. This is without loss of generality, as we may use an extractor to get a uniform bit-string and a PRG to expand randomness, if necessary.

Security of Sig': Since $\mathsf{Sig.sk}$ has always $\ell/2$ bits equal to 0, the number of zeroes appended to the signature will be $\ell/2$ and hence does not leak any extra information to the adversary. Hence, it follows easily that if Sig is a secure signature scheme, then so is Sig'.

Now, consider using Sig' to instantiate our threshold signature scheme. Then, for any message M, the HE ciphertext CT_{σ_M} now additionally includes homomorphically evaluated encryptions of $\{\mathsf{Sig.sk}_i\}_{i\in[\ell]:\mathsf{Sig.sk}_i=0}$. Let $\mathsf{CT}_{\sigma_M}, \mathsf{err}_M, e'_M$ respectively to denote the encryption of σ_M, the error in CT_{σ_M} and the flooding noise added to partial decryption of CT_{σ_M}. Let $\mathsf{CT}^*, \mathsf{err}^*$ and e^*_M denote the components corresponding to $\{\mathsf{Sig.sk}_i\}_{i\in[\ell]:\mathsf{Sig.sk}_i=0}$.

For any message M, the adversary can compute $\mathsf{err}_M + e'_M$ as described previously, from which the adversary gets $\mathsf{err}^* + e^*_M$. If the adversary manages to compute err^* (for sufficiently many instances), then it can also recover $\mathsf{HE.SK}$.

Note that err^* is independent of any message and hence is constant across different messages, while e^*_M does depend on M and is different for different messages. This gives an attack strategy. To compute err^*, the adversary issues Q signing queries on different messages $\{M_j\}_{j\in[Q]}$, and from the received partial signatures, derives the values for

$$w^*_j = \mathsf{err}^* + e^*_{M_j} \quad \text{for } j \in [Q],$$

This equation is of the same form as Equation (4.2).

Heuristically, one would expect the $e^*_{M_j}$ to behave as independent and identically distributed random variables with centre 0. Hence, we can argue in similar way that if $Q \geq \Omega(s^2 \log \lambda)$ then the adversary can recover err^* with probability $1 - 1/\operatorname{poly}(\lambda)$. This implies that for hiding err^*, the standard deviation parameter s must grow at least proportionally to $\sqrt{Q}$.

5 Adaptive Security for Threshold Signatures

As discussed in Section 1, we provide two constructions to improve the selective security achieved by [12]. Below, we describe our construction in the ROM, which satisfies partially adaptive unforgeability where the adversary is allowed to make signing queries before revealing the set of corrupted parties, but must reveal the set of all the corrupted parties all at once. We provide our construction in the standard model with *pre-processing* that satisfies fully adaptive unforgeability in Appendix C.

5.1 Partially Adaptive Unforgeability

A TS scheme satisfies partially adaptive unforgeability if for any adversary $\mathcal{A}$ with run-time $2^{o(\lambda)}$, the output of the following experiment is 1 with probability $2^{-\Omega(\lambda)}$:
$\mathsf{Expt}_{\mathcal{A},\mathsf{TS},PA-uf}(1^\lambda)$:

1. On input the security parameter λ, $\mathcal{A}$ outputs an access structure $\mathsf{A} \in \mathbb{S}$.
2. The challenger runs the $\mathsf{TS.KeyGen}(1^\lambda)$ algorithm and generates public parameters pp, verification key vk and a set of N key shares $\{\mathsf{sk}_i\}_{i=1}^N$. It sends pp and vk to $\mathcal{A}$.
3. Adversary $\mathcal{A}$ issues polynomial number of adaptive signing queries where in each query, $\mathcal{A}$ outputs a message m and the challenger computes $\sigma_i \leftarrow \mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m)$ for $i \in [N]$ and provides $\{\sigma_i\}_{i=1}^N$ to $\mathcal{A}$.
4. Adversary $\mathcal{A}$ then outputs a maximal invalid party set $S \subseteq [N]$ to get key shares sk_i for $i \in S$.
5. Challenger provides the set of keys $\{\mathsf{sk}_i\}_{i\in S}$ to $\mathcal{A}$.
6. Adversary $\mathcal{A}$ continues to issue polynomial number of adaptive signing queries of the form (m, i), where $i \in [N] \setminus S$, to get a partial signature σ_i for m. For each query the challenger computes σ_i as $\mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m)$ and provides it to $\mathcal{A}$.
7. At the end of the experiment, adversary $\mathcal{A}$ outputs a forgery (m^*, σ^*). The experiment outputs 1 if $\mathsf{TS.Verify}(\mathsf{vk}, m^*, \sigma^*) = \mathit{accept}$ and m^* was not queried previously as a signing query.

5.1.1 Construction.

We use the same building blocks for construction as those used for the non-adaptive construction. We also use two keyed hash function modelled as random oracles: $H : \{0,1\}^\lambda \times \{0,1\}^* \to \mathbb{Z}_q^N$ and $H_1 : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^r$. The construction is provided in Figure 2.

Correctness. From the correctness of the $\mathsf{HE.Eval}$ algorithm, $\mathsf{CT}_{\sigma_M} = \mathsf{HE.Eval}(\mathsf{HE.PK}, \mathcal{C}_M, \mathsf{CT}_{\mathsf{Sig.sk}})$ is an encryption of $\mathcal{C}_M(\mathsf{Sig.sk}) = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, M) = \sigma_M$, which decrypts with the HE secret key $\mathsf{HE.SK}$. So, $\mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_{\sigma_M}) = \sigma_M \lfloor q/2 \rceil + e$. The signature computed by the $\mathsf{TS.Combine}$ algorithm is

$$
\begin{aligned}
&\mathsf{HE.decode}_1\Big(\sum_{i=1}^{N} \sigma_{i,M}\Big) \\
&= \mathsf{HE.decode}_1\Big(\sum_{i=1}^{N} \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + \sum_{i=1}^{N} r_{i,M} + \sum_{i=1}^{N} e'_i\Big) \\
&= \mathsf{HE.decode}_1\Big(\sum_{i=1}^{N} \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + \sum_{i=1}^{N} H(K,M)^T R_i + \sum_{i=1}^{N} e'_i\Big) \\
&= \mathsf{HE.decode}_1\Big(\mathsf{HE.decode}_0\Big(\sum_{i=1}^{N} \mathsf{sk}_i, \mathsf{CT}_{\sigma_M}\Big) + H(K,M)^T \sum_{i=1}^{N} R_i + \sum_{i=1}^{N} e'_i\Big) \\
&= \mathsf{HE.decode}_1\Big(\mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_{\sigma_M}) + \mathbf{0} + \sum_{i=1}^{N} e'_i\Big) \qquad \Big(\text{because } \sum_{i=1}^{N} R_i = \mathbf{0}\Big) \\
&= \mathsf{HE.decode}_1\Big(\sigma_M \lfloor q/2 \rceil + e + \sum_{i=1}^{N} e'_i\Big) = \sigma_M
\end{aligned}
$$

Proof of Security.

Theorem 5.1. *Assume the signature scheme* Sig *satisfies unforgeability,* HE *is a semantically secure homomorphic encryption scheme (Definition 2.18),* HS *is context hiding homomorphic signature scheme (Definition A.4) and* Share *satisfies privacy (Definition 2.26). Then the* TS *construction in Section 5 satisfies partially adaptive unforgeability in ROM if the flooding error is of size* $\mathrm{poly}(\lambda)\sqrt{Q}$*, where* Q *is the upper bound on the number of signing queries.*

Proof. The security of the construction can be argued using the following hybrids:

Hybrid_0: The real world.

Hybrid_1: Same as Hybrid_0 except that now instead of using $\mathsf{HS.SigEval}$ algorithm to compute the homomorphic signature $\tilde{\pi}_{i,M}$ on $\sigma_{i,M}$ in $y_{i,M} = (\sigma_{i,M}, \tilde{\pi}_{i,M})$, the challenger simulates $\tilde{\pi}_{i,M}$ as

$$\tilde{\pi}_{i,M} = \mathsf{Sim}(\mathsf{HS.sk}, \alpha, \sigma_{i,M}, \tau_i, \pi_{\tau_i}),$$

where $\alpha = \mathsf{HS.Process}(\mathsf{HS.pp}, \mathcal{C}_{\mathsf{PS}})$.

$\mathsf{TS.KeyGen}(1^\lambda)$: Upon input the security parameter λ, do the following:

1. Randomly choose $K \leftarrow \{0,1\}^\lambda$ and N vectors $R_1, R_2, \ldots, R_N \in \mathbb{Z}_q^N$ such that $\sum_{i=1}^N R_i = \mathbf{0}$.
2. Generate the keys for the signature scheme $(\mathsf{Sig.vk}, \mathsf{Sig.sk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$.
3. Generate the keys for the HE scheme $(\mathsf{HE.PK}, \mathsf{HE.SK}) \leftarrow \mathsf{HE.KeyGen}(1^\lambda)$ and share $\mathsf{HE.SK}$ into N shares as $(\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_N) = \mathsf{Share}(\mathsf{HE.SK})$ such that $\sum_{i=1}^N \mathsf{sk}_i = \mathsf{HE.SK}$.
4. Compute an HE encryption of the signing key as $\mathsf{CT}_{\mathsf{Sig.sk}} = \mathsf{HE.Enc}(\mathsf{HE.PK}, \mathsf{Sig.sk})$.
5. For each party P_i, randomly choose a tag $\tau_i \in \{0,1\}^*$, a hash key $\mathsf{hkey}_i \leftarrow \{0,1\}^\lambda$ and generate HS public parameters $\mathsf{HS.pp} \leftarrow \mathsf{HS.PrmsGen}(1^\lambda, 1^n)$ and HS public and signing keys as $(\mathsf{HS.pk}, \mathsf{HS.sk}) \leftarrow \mathsf{HS.KeyGen}(1^\lambda, \mathsf{HS.pp})$. Here, n is the length of input to PartSign circuit which depends on $(\mathsf{HE.SK}, K, R_i, \mathsf{hkey}_i)$.
6. Compute $(\pi_{\tau_i}, \pi_i) = \mathsf{HS.Sign}(\mathsf{HS.sk}, (\mathsf{sk}_i, K, R_i, \mathsf{hkey}_i), \tau_i)$.
7. Output $\mathsf{TSig.pp} = (\mathsf{HE.PK}, \mathsf{HS.pp}, \mathsf{HS.pk}, \mathsf{CT}_{\mathsf{Sig.sk}}, \{\tau_i, \pi_{\tau_i}\}_{i=1}^N)$, $\mathsf{TSig.vk} = \mathsf{Sig.vk}$, $\mathsf{TSig.sk} = \{\mathsf{TSig.sk}_i = (\mathsf{sk}_i, K, R_i, \mathsf{hkey}_i, \pi_i)\}_{i=1}^N$.

$\mathsf{TS.PartSign}(\mathsf{TSig.pp}, \mathsf{TSig.sk}_i, M)$: Upon input the public parameters $\mathsf{TSig.pp}$, the key share $\mathsf{TSig.sk}_i = (\mathsf{sk}_i, K, R_i, \mathsf{hkey}_i, \pi_i)$ and a message M, do the following:

1. Compute $u_i = H_1(\mathsf{hkey}_i, M)$ and sample $e_i' \leftarrow \mathcal{D}_s(u_i)$.
2. Let $\mathcal{C}_M$ be the signing circuit, with message M being hardcoded. Compute an HE encryption of signature σ_M as $\mathsf{CT}_{\sigma_M} = \mathsf{HE.Eval}(\mathsf{HE.PK}, \mathcal{C}_M, \mathsf{CT}_{\mathsf{Sig.sk}})$.
3. Compute $r_{i,M} = H(K, M)^T R_i$.
4. Compute $\sigma_{i,M} = \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{i,M} + e_i'$.
5. Let $\mathcal{C}_{\mathsf{PS}}$ be the circuit to compute $\mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + H(K, M)^T R_i + e_i'$ in which CT_{σ_M} is hardcoded and the key share $\mathsf{TSig.sk}_i$ is given as the input.
   (a) Compute $\pi_{i,M}^* = \mathsf{HS.SignEval}(\mathsf{HS.pp}, \mathcal{C}_{\mathsf{PS}}, \pi_{\tau_i}, (\mathsf{sk}_i, K, R_i, \mathsf{hkey}_i), \pi_i)$.
   (b) Compute $\tilde{\pi}_{i,M} = \mathsf{HS.Hide}(\mathsf{HS.pk}, \sigma_{i,M}, \pi_{i,M}^*)$.
6. Output $y_{i,M} = (\sigma_{i,M}, \tilde{\pi}_{i,M})$.

$\mathsf{TS.PartSignVerify}(\mathsf{TSig.pp}, M, y_{i,M})$: Upon input the public parameters $\mathsf{TSig.pp}$, message M, and a partial signature $y_{i,M}$, the partial signature verifier does the following.

1. Computes $\mathsf{CT}_{\sigma_M} = \mathsf{HE.Eval}(\mathsf{HE.PK}, \mathcal{C}_M, \mathsf{CT}_{\mathsf{Sig.sk}})$.
2. Defines circuit $\mathcal{C}_{\mathsf{PS}}$ as described before and computes $\alpha = \mathsf{HS.Process}(\mathsf{HS.pp}, \mathcal{C}_{\mathsf{PS}})$.
3. Parses $y_{i,M}$ as $(\sigma_{i,M}, \tilde{\pi}_{i,M})$ and outputs

$$\mathsf{HS.HVerify}(\mathsf{HS.pp}, \alpha, \sigma_{i,M}, \tau_i, (\pi_{\tau_i}, \tilde{\pi}_{i,M})).$$

$\mathsf{TS.Combine}(\mathsf{TSig.pp}, \{y_{i,M}\}_{i\in[N]})$: Upon input the public parameters $\mathsf{TSig.pp}$ and a set of partial signatures $\{y_{i,M}\}_{i\in[N]}$, parses $y_{i,M}$ as $(\sigma_{i,M}, \tilde{\pi}_{i,M})$ and outputs $\sigma_M = \mathsf{HE.decode}_1(\sum_{i=1}^N \sigma_{i,M})$.

$\mathsf{TS.Verify}(\mathsf{TSig.vk}, M, \sigma_M)$: Upon input the verification key $\mathsf{TSig.vk}$, a message M and a signature σ_M, output $\mathsf{Sig.Verify}(\mathsf{TSig.vk}, M, \sigma_M)$.

Figure 2 Partially Adaptive Threshold Signature.

Hybrid_2: Same as Hybrid_1, except that the randomness u_i used in sampling the flooding noise in $\mathsf{PartSign}$ algorithm is chosen uniformly randomly from $\{0,1\}^r$ and then H_1 is programmed as $H_1(\mathsf{hkey}_i, M) = u_i$. For random oracle queries for hash H_1 by the adversary on an input x, the challenger first checks if $H_1(x)$ is already set. If so, then returns that value else chooses a value uniformly randomly from $\{0,1\}^r$ and saves and returns it.

Hybrid_3: Same as Hybrid_2 except that the value of $H(K, M)$ for all M in pre corruption signing queries is set in reverse order, i.e., firstly partial signatures are computed and then $H(K, M)$ is set accordingly as follows:

1. The challenger computes $\mathsf{CT}_{\sigma_M} = \mathsf{HE.Eval}(\mathsf{HE.PK}, \mathcal{C}_M, \mathsf{CT}_{\mathsf{Sig.sk}})$.
2. It then computes $\mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_{\sigma_M})$ and divides it into N shares $\{s_{i,M}\}_{i=1}^N$ such that $\sum_{i=1}^N s_{i,M} = \mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_{\sigma_M})$.
3. Returns partial signatures as $\{\sigma_{i,M} = s_{i,M} + e'_i\}_{i=1}^N$. Also, if a message M is repeated for signing query, then the challenger uses same $\{s_{i,M}\}_{i=1}^N$ shares of $\mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_{\sigma_M})$ again.
4. When the adversary outputs the set S of corrupted parties, the challenger first sets the value of $H(K, M)$ for each M in pre corruption signing queries as described next, and then provides key shares for $i \in S$ to the adversary.
   - $\forall i \in [N]$, compute $r_{i,M} = s_{i,M} - \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M})$.
   - Solve for vector $\mathbf{b} \in \mathbb{Z}_q^N$ such that $\forall\ i \in [N], \mathbf{b}^T R_i = r_{i,M}$. Set $H(K, M) = \mathbf{b}$. Note that since there are $N-1$ independent equations in N unknowns, such a $\mathbf{b}$ exists and can be computed.
5. To answer a random oracle query for hash function H on input x, the challenger first checks if the value is already set, if so then returns that value, else randomly samples a fresh vector $\mathbf{r}_x$ and sets and returns $H(x) = \mathbf{r}_x$.

Hybrid_4: Same as Hybrid_3, except that now the signing queries are answered differently. For each pre-corruption signing query for a message M, the challenger computes $\sigma_{i,M}$ as follows:

1. Computes $\sigma_M = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, M)$ and generates random shares of $\sigma_M \lfloor q/2 \rceil$ as $\{s_{i,M}\}_{i=1}^N \leftarrow \mathsf{Share}(\sigma_M \lfloor q/2 \rceil)$ such that $\sum_{i=1}^N s_{i,M} = \sigma_M \lfloor q/2 \rceil$.
2. Returns $\sigma_{i,M} = s_{i,M} + e'_i$, where $e'_i \leftarrow \mathcal{D}_s$

When the adversary outputs the set S of corrupted parties, the challenger does the following:

1. Let $PreQ$ be the set of messages for which signing queries were made before. Then for each $M \in PreQ$ it does the following. For each $i \in S$, computes $r_{i,M} = s_{i,M} - \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M})$. Computes $\mathbf{b}$ such that $\forall\ i \in S, \mathbf{b}^T R_i = r_{i,M}$. Sets $H(K, M) = \mathbf{b}$. Such a $\mathbf{b}$ exists and can be computed since there are only $N-1$ equations to solve in N unknowns.
2. Returns the secret key shares $\{\mathsf{TSig.sk}_i\}_{i \in S}$.

For each post corruption signing query on message M, the challenger does the following. Let the uncorrupted party be P_a.

1. Computes $\mathsf{CT}_{\sigma_M} = \mathsf{HE.Eval}(\mathsf{HE.PK}, \mathcal{C}_M, \mathsf{CT}_{\mathsf{Sig.sk}})$.
2. For each $i \in S$, computes $\sigma'_{i,M} = \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + H(K, M)^T R_i$ and $\sigma_{i,M} = \sigma'_{i,M} + e'_i$, where $e'_i \leftarrow \mathcal{D}_s$.
3. Computes $\sigma_M = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, M)$.
4. Sets $\sigma_{a,M} = \sigma_M \lfloor q/2 \rceil - \sum_{i \in S} \sigma'_{i,M} + e'_a$, where $e'_a \leftarrow \mathcal{D}_s$.
5. Sends $\sigma_{a,M}$ to the adversary.

Hybrid_5: Same as Hybrid_4 except that now instead of secret sharing $\mathsf{HE.SK}$ to generate sk_i in $\mathsf{TSig.sk}_i$, sk_i is generated as secret share of zero vector as $\{\mathsf{sk}_i\}_{i=1}^N \leftarrow \mathsf{Share}(\mathbf{0})$.

Hybrid_6: Same as Hybrid_5 except that now $\mathsf{CT}_{\mathsf{Sig.sk}}$ in public parameters $\mathsf{TSig.pp}$ is replaced by $\mathsf{CT}_{\mathbf{0}}$, i.e., an HE encryption of zero vector.

Indistinguishability of Hybrids. Next, we show that consecutive hybrids are indistinguishable.

Claim 5.2. *If the underlying homomorphic signature scheme* HS *is context hiding then* Hybrid_0 *and* Hybrid_1 *are indistinguishable.*

Proof. The two hybrids differ only in the way $\tilde{\pi}_{i,M}$ is computed. In Hybrid_0 it is computed using $\mathsf{HS.SignEval}$ while in Hybrid_1 it is generated by HS simulator. Hence, from the context hiding property of HS, the two hybrids are indistinguishable. □

Claim 5.3. *If* H_1 *is modeled as random oracle then* Hybrid_1 *and* Hybrid_2 *are indistinguishable.*

Proof. The two hybrids differ only in the way u_is are computed while computing partial signatures. In Hybrid_1, $u_i = H_1(\mathsf{hkey}_i, M)$, while in Hybrid_2, it is chosen uniformly randomly and then H_1 is programmed accordingly. Since H_1 is modeled as random oracle the two hybrids are indistinguishable in adversary's view. □

Claim 5.4. Hybrid_2 *and* Hybrid_3 *are indistinguishable in random oracle model.*

Proof. The two hybrids differ only in the order in which $H(K, M)$ and $r_{i,M} = H(K, M)^T R_i$ are computed in pre-corruption queries. In Hybrid_2, $H(K, M)$ is set first and then $r_{i,M}$ is computed accordingly. In Hybrid_3, $r_{i,M}$ is fixed first and then $H(K, M)$ is programmed after the adversary reveals the set S of corrupted parties such that $H(K, M)^T R_i = r_{i,M}$ is satisfied for each $i \in S$.

Since the adversary gets to know the secret K only after revealing the set S of corrupted parties, there is only negligible probability that the adversary makes a random oracle query for input (K, M) before revealing the set S (which could lead to inconsistent values for $H(K, M)$). Hence, setting $H(K, M)$ for pre-corruption queries in the two hybrids in the above described ways, does not change adversary's view. □

Claim 5.5. *Assume that the flooding error is of the order* $\mathrm{poly}(\lambda) \cdot \sqrt{Q}$. *If there is an adversary who can win the* $\mathsf{Expt}_{\mathcal{A},\mathsf{TS},PA-uf}(1^\lambda)$ *game in* Hybrid_3 *with probability* ϵ, *then then its probability of winning in* Hybrid_4 *is at least* $\epsilon^2/2$.

Proof. Let the adversary makes Q signing queries. Wlog let the corrupted parties be $\{P_2, \ldots, P_N\}$. Then in the adversary's view the two hybrids differ only in the error term in $\sigma_{1,M}$, as shown below. Let $e'_1 \leftarrow \mathcal{D}_s$. In Hybrid_3, for pre-corruption queries, the partial signature $\sigma_{1,M}$ is computed as

$$
\begin{aligned}
\sigma_{1,M} &= s_{1,M} + e'_1 \\
&= \sum_{i=1}^{N} s_{i,M} - \sum_{i=2}^{N} s_{i,M} + e'_1 \\
&= \mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_{\sigma_M}) - \sum_{i=2}^{N} s_{i,M} + e'_1 \\
&= \mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_{\sigma_M}) - \sum_{i=2}^{N} (\mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{i,M}) + e'_1 \\
&= \mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_{\sigma_M}) - \sum_{i=1}^{N} \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) \\
&\quad + \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_{\sigma_M}) - \sum_{i=2}^{N} H(K,M)^T R_i + e'_1 \\
&= \mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_{\sigma_M}) - \mathsf{HE.decode}_0(\sum_{i=1}^{N} \mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) \\
&\quad + \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_{\sigma_M}) - \sum_{i=1}^{N} H(K,M)^T R_i + H(K,M)^T R_1 + e'_1 \\
&= \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_{\sigma_M}) + H(K,M)^T R_1 + e'_1 \quad (\because \sum_{i=1}^{N} R_i = 0;\ \sum_{i=1}^{N} \mathsf{sk}_i = \mathsf{HE.SK})
\end{aligned}
$$

In Hybrid_3, for any post-corruption signing query on message M, the partial signature $\sigma_{1,M}$ is computed as:

$$\mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_{\sigma_M}) + H(K,M)^T R_1 + e'_1$$

In Hybrid_4, for pre-corruption queries, we have

$$\begin{aligned}
\sigma_{1,M} &= s_{1,M} + e'_1 \\
&= \sigma_M \lfloor q/2 \rceil - \sum_{i=2}^{N} s_{i,M} + e'_1 \\
&= \sigma_M \lfloor q/2 \rceil - \sum_{i=2}^{N} (\mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + H(K,M)^T R_i) + e'_1 \\
&= \sigma_M \lfloor q/2 \rceil - \sum_{i=1}^{N} \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) - \sum_{i=1}^{N} H(K,M)^T R_i \\
&+ \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_{\sigma_M}) + H(K,M)^T R_1 + e'_1 \\
&= \sigma_M \lfloor q/2 \rceil - \sum_{i=1}^{N} \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) - H(K,M)^T \sum_{i=1}^{N} R_i \\
&+ \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_{\sigma_M}) + H(K,M)^T R_1 + e'_1 \\
&= \sigma_M \lfloor q/2 \rceil - \mathsf{HE.decode}_0(\sum_{i=1}^{N} \mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) \\
&+ \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_{\sigma_M}) + H(K,M)^T R_1 + e'_1 \qquad (\because \sum_{i=1}^{N} R_i = 0) \\
&= \sigma_M \lfloor q/2 \rceil - \mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_{\sigma_M}) + \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_{\sigma_M}) \\
&\quad + H(K,M)^T R_1 + e'_1 \\
&= \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_{\sigma}) + H(K,M)^T R_1 + e + e'_1
\end{aligned}$$

In Hybrid_4, for any post-corruption query for a message M, we have

$$\begin{aligned}
\sigma_{1,M} &= \sigma_M . \lfloor q/2 \rceil - \sum_{i=2}^{N} (\mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + H(K,M)^T R_i) + e'_1 \\
&= \sigma_M . \lfloor q/2 \rceil - \sum_{i=1}^{N} (\mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + H(K,M)^T R_i) \\
&+ \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_{\sigma_M}) + H(K,M)^T R_1 + e'_1 \\
&= \sigma_M . \lfloor q/2 \rceil - \mathsf{HE.decode}_0(\sum_{i=1}^{N} \mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + H(K,M)^T \sum_{i=1}^{N} R_i \\
&+ \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_{\sigma_M}) + H(K,M)^T R_1 + e'_1 \\
&= \sigma_M . \lfloor q/2 \rceil - \mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_{\sigma_M}) + \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_{\sigma_M}) \\
&\quad + H(K,M)^T R_1 + e'_1 \\
&= \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_{\sigma_M}) + H(K,M)^T R_1 + (e'_1 + e).
\end{aligned}$$

Thus, the difference in the two hybrids is in the error terms in σ_1. In Hybrid_3, the error is e'_1, while in Hybrid_4, it is $e'_1 + e$. This is the same case as in Section 4. Hence we can use exactly the

same analysis using Rényi Divergence as in Claim 4.4 to show that the claim is true when s is of the order $\text{poly}(\lambda) \cdot \sqrt{Q}$. □

Claim 5.6. *Assuming the privacy property of secret sharing scheme* Share, Hybrid_4 *and* Hybrid_5 *are indistinguishable.*

Proof. The only difference between Hybrid_4 and Hybrid_5 is in the way the key shares $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$ are generated. In Hybrid_4, $\{\mathsf{sk}_i\}_{i\in[N]} \leftarrow \mathsf{Share}(\mathsf{HE.SK})$, while in Hybrid_5, $\{\mathsf{sk}_i\}_{i\in[N]} \leftarrow \mathsf{Share}(\mathbf{0})$. Since the adversary is given the key shares only for an invalid set of parties, the two distributions are identical due to the privacy property of secret sharing scheme Share. □

Claim 5.7. *Assume that* HE *is semantically secure. Then* Hybrid_5 *and* Hybrid_6 *are computationally indistinguishable.*

Proof. Let $\mathcal{A}$ be an adversary who can distinguish Hybrid_5 and Hybrid_6. Then we construct an adversary $\mathcal{B}$ against the HE scheme as follows.

- Upon receiving $\mathsf{HE.PK}$ from the HE challenger, $\mathcal{B}$ runs $(\mathsf{Sig.sk}, \mathsf{Sig.vk}) \leftarrow \mathsf{Sig.KeyGen}$ and sends the challenge messages $m_0 = \mathsf{Sig.sk}$ and $m_1 = \mathbf{0}$ to the HE challenger.
- The challenger replies with ciphertext CT_b.
- $\mathcal{B}$ generates secret shares of zero vector as $\{\mathsf{sk}_i\}_{i=1}^N \leftarrow \mathsf{Share}(\mathbf{0})$. It also generates all other components of $\mathsf{TSig.pp}, \mathsf{TSig.vk}$ and $\mathsf{TSig.sk}_i$ for $i \in [N]$ and invokes $\mathcal{A}$.
- Adversary $\mathcal{B}$ answers all the queries (both signing and secret key shares, as well as random oracle queries) of $\mathcal{A}$, as defined. Note that to respond to these queries, $\mathsf{HE.SK}$ is not needed and hence $\mathcal{B}$ can respond to them on its own.
- Finally, if $\mathcal{A}$ guesses to be in Hybrid_5, then $\mathcal{B}$ replies with $b' = 0$, else $b' = 1$ to the HE challenger.

□

Claim 5.8. *If the underlying signature scheme* Sig *is unforgeable, then the unforgeability experiment* $\mathsf{Expt}_{\mathcal{A},\mathsf{TS},PA-uf}$ *in* Hybrid_6 *cannot be won.*

Proof. Let $\mathcal{A}$ be an adversary that wins the unforgeability game in Hybrid_6. Then we can construct an adversary $\mathcal{B}$ against the underlying signature scheme as follows:

- On receiving verification key $\mathsf{Sig.vk}$ from Sig challenger, adversary $\mathcal{B}$ generates all the other secret and public values defined for Hybrid_6 on its own.
- To answer a pre-corruption as well as a post-corruption signing query from $\mathcal{A}$ for any message M, adversary $\mathcal{B}$ issues a signing query for same message M to the Sig challenger and receives σ_M. For pre-corruption queries, it simulates partial signatures using σ_M, for each $i \in [N]$, as defined for the hybrid. For post-corruption queries, for each $j \in S$, adversary $\mathcal{B}$ computes partial signatures using the key share $\mathsf{TSig.sk}_i$ and simulates the signature for $i \notin S$ using σ_M as defined for the hybrid.

- Adversary $\mathcal{B}$ answers the key share queries for set S of corrupted parties issued by $\mathcal{A}$ as defined for the hybrid. Note that the key shares does not depend on $\mathsf{Sig.sk}$ and hence, adversary $\mathcal{B}$ can send a valid response to $\mathcal{A}$.
- In the end, let (M^*, σ^*) be the forgery returned by $\mathcal{A}$. Then $\mathcal{B}$ returns (M^*, σ^*) to the Sig challenger.

Since $\mathcal{B}$ issues signing queries for only those messages for which $\mathcal{A}$ issues partial signature queries, if (M^*, σ^*) is a valid forgery for $\mathcal{A}$, then it is also a valid forgery for $\mathcal{B}$. □

This proves the partially adaptive unforgeability of our construction. □

Robustness. It can be seen that if HS is a multi data secure homomorphic signature, then the construction of TS satisfies robustness. The proof is the same as in Claim 4.8.

6 Blind Signatures

In this section, we describe our two-round construction of blind signatures in the random oracle model. Our construction uses the following building blocks:

- A *circuit private* homomorphic encryption scheme $\mathsf{HE} = (\mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Dec}, \mathsf{HE.Eval})$ instantiated using [15] (denoted by BdPMW).
- An "HE-friendly" signature scheme $\mathsf{Sig} = (\mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$ as constructed in Section 3.
- Zero-knowledge proofs of knowledge (ZKPoK) for linear relations with small coefficients, i.e., for the existence of a vector $\mathbf{t}$ such that $\mathbf{t}$ has low Euclidean norm and $\mathbf{Bt} = \mathbf{u} \bmod q$ for some public $(\mathbf{B}, \mathbf{u}, q)$. We require the extraction algorithm to exactly recover $\mathbf{t}$. Efficient realizations of such proof systems have been recently proposed [73, 13, 31].
- A hash function H modelled as a random oracle.

Maliciously Secure Circuit Private HE. Our construction makes use of a circuit private HE which is secure in the malicious setting. We leverage the random oracle model to upgrade a semi-honest circuit private HE (Definition 2.20) to malicious security using NIZK proofs (see [58] for a discussion). To instantiate the underlying semi honest circuit private HE we use the construction from [15], which modifies the GSW HE scheme [41].

Theorem 6.1 ([15])**.** *There exists a homomorphic encryption scheme* BdPMW *for branching programs that is semi honest circuit private and whose security is based on the LWE assumption with polynomial noise-to-modulus ratio.*

In BdPMW, the secret key SK and public key PK are, respectively:

$$\mathbf{s} = (-\bar{\mathbf{s}}, 1) \quad \text{and} \quad \hat{\mathbf{A}} = \begin{pmatrix} \mathbf{A} \\ \bar{\mathbf{s}}^\top \mathbf{A} + \mathbf{e}^\top \end{pmatrix}$$

where $\bar{\mathbf{s}} \leftarrow \mathbb{Z}_q^{n-1}$, $\mathbf{A} \leftarrow \mathbb{Z}_q^{(n-1)\times m}$ and $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\alpha}$. Here $m = nk$ with $k = \lceil \log_2 q \rceil$.

We modify the BdPMW scheme to make it maliciously circuit private (denoted BdPMW′) as follows:

1. We sample the secret key $\bar{\mathbf{s}}$ from the LWE error distribution, so that it has small coefficients. This does not impact functionality of BdPMW, nor its semantic security under the LWE assumption [5].
2. The matrix $\mathbf{A}$ is generated using ROM.
3. The public key PK contains a ZKPoK that the last row of $\hat{\mathbf{A}}$ is of the form $\mathbf{x}^\top\mathbf{A} + \mathbf{y}^\top$ for some low-norm vector $(\mathbf{x}, \mathbf{y})$.
4. The ciphertext contains a ZKPoK that it is of the form $\mathsf{CT} = \hat{\mathbf{A}}\mathbf{R} + \mu\mathbf{G}$ for some low-norm $\mathbf{R}$ and binary μ, where $\mathbf{G}$ is obtained by appending the matrix $(1, 2, \ldots, 2^{k-1}) \otimes \mathbf{I}_{n-1}$ on top of the zero vector of dimension $m = nk$. Such a ZKPoK is for example given in [73, Section 4.2].

Claim 6.2. *The* BdPMW′ *construction satisfies malicious circuit privacy.*

Proof. We argue malicious circuit privacy for BdPMW′ by observing that the proof of semi-honest circuit privacy in [15] only makes use of the following properties: i) the matrix $\mathbf{A}$ must be generated uniformly to satisfy the condition in [15, Lemma 3.1], ii) the ciphertext and public key must be structured as in the honest scheme, so that [15, Lemma 3.1] may be applied during the analysis of circuit privacy. We observe in particular, that [15, Lemma 3.1] makes no assumption about the distribution of the error terms in the public key or the ciphertext, nor does it rely on any distributional requirement from the secret key. In particular, if we choose $\bar{\mathbf{s}}$ from the error distribution, this does not impact circuit privacy. Hence, by ensuring that $\mathbf{A}$ is uniform and that PK and CT are well-formed by providing zero knowledge proofs, we satisfy all the conditions required for the proof of circuit privacy in [15] to follow. □

6.1 Construction

Setup. $\mathsf{Gen}(1^\lambda)$: Upon input the security parameter 1^λ, invoke $(\mathsf{Sig.sk}, \mathsf{Sig.vk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$ using the signature scheme provided in Section 3, and output $(\mathsf{Sig.sk}, \mathsf{Sig.vk})$.

Signing. $\langle \mathcal{S}(\mathsf{Sig.sk}), \mathcal{U}(\mathsf{Sig.vk}, m) \rangle$:

1. **User:**
   - User $\mathcal{U}$ samples the HE secret key SK as in BdPMW′. It computes $\mathbf{A} = H(\mathsf{Sig.vk}, \mathsf{id})$ where id is a user identifier. It samples both $\bar{\mathbf{s}}$ and $\mathbf{e}$ from the LWE error distribution and computes the PK as $\hat{\mathbf{A}} = \begin{pmatrix} \mathbf{A} \\ \bar{\mathbf{s}}^\top\mathbf{A} + \mathbf{e}^\top \end{pmatrix}$
   - User $\mathcal{U}$ generates a ZKPoK π_{SK} to prove that it knows a low-norm vector $(\mathbf{x}, \mathbf{y})$ such that the last row of $\hat{\mathbf{A}}$ is of the form $\mathbf{x}^\top\mathbf{A} + \mathbf{y}^\top$.
   - It computes $\mathsf{CT}_m = \mathsf{HE.Enc}(\mathsf{PK}, m)$ and a proof π_{CT} that CT_m is well-formed, i.e., $\mathsf{CT}_m = \hat{\mathbf{A}}\mathbf{R} + \mu\mathbf{G}$, for a low-norm $\mathbf{R}$ and a binary μ.
   - It sends $(\mathsf{PK}, \pi_{\mathsf{SK}}, \mathsf{CT}_m, \pi_{\mathsf{CT}})$ to signer $\mathcal{S}$.

2. **Signer:**
   - Upon receiving $(\mathsf{PK}, \pi_{\mathsf{SK}}, \mathsf{CT}_m, \pi_{\mathsf{CT}})$, signer $\mathcal{S}$ verifies the proofs π_{SK} and π_{CT} and outputs $\perp$ if any of these fails.
   - It computes $\mathsf{CT}_\sigma = \mathsf{HE.Eval}(\mathsf{PK}, \mathsf{Sig.Sign}_{\mathsf{Sig.sk}}, \mathsf{CT}_m)$. It sends CT_σ to user $\mathcal{U}$.
3. **User:** User $\mathcal{U}$ runs $\mathsf{HE.Dec}(\mathsf{SK}, \mathsf{CT}_\sigma)$ to recover σ and outputs it.

Verifying. $\mathsf{Vrfy}(\mathsf{Sig.vk}, m, \sigma)$ is identical to $\mathsf{Sig.Verify}$.

Correctness. From the correctness of the ZKPoK schemes, the signer accepts π_{SK} and π_{CT} (and does not abort) if the user computed the HE public key and ciphertexts correctly. From the correctness of $\mathsf{HE.Eval}(\mathsf{Sig.Sign}_{\mathsf{Sig.sk}}, \mathsf{CT}_m)$, CT_σ is an HE encryption of $\mathsf{Sig.Sign}_{\mathsf{Sig.sk}}(m) = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, m) = \sigma$. Hence, from the correctness of the HE decryption algorithm, $\mathsf{HE.Dec}(\mathsf{Sig.sk}, \mathsf{CT}_\sigma)$ outputs a signature $\sigma = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, m)$. It passes verification, by correctness of the $\mathsf{Sig.Verify}$ algorithm.

Security. We show honest signer blindness of the construction in the following theorem.

Theorem 6.3. *Assume that the underlying proof systems are ZKPoK. Then the BS construction satisfies (honest signer) blindness.*

Proof. The argument proceeds via a sequence of hybrids.

Hybrid_0: This is the real world.

1. The challenger generates $(\mathsf{Sig.sk}, \mathsf{Sig.vk})$ and returns $(\mathsf{Sig.sk}, \mathsf{Sig.vk})$ to the adversary $\mathcal{S}^*$.
2. The signer $\mathcal{S}^*$ outputs two messages m_0 and m_1. The challenger picks a random bit b.
3. The signer $\mathcal{S}^*$ interacts concurrently with $\mathcal{U}_1(\mathsf{Sig.vk}, m_b)$ and $\mathcal{U}_2(\mathsf{Sig.vk}, m_{\bar{b}})$, played by the challenger as follows:
   (a) User $\mathcal{U}_1$ (resp. $\mathcal{U}_2$) generates $\mathbf{A}_1 = H(\mathsf{Sig.vk}, \mathsf{id}_1)$ (resp. $\mathbf{A}_2 = H(\mathsf{Sig.vk}, \mathsf{id}_2)$) using the random oracle. Users $\mathcal{U}_1$ and $\mathcal{U}_2$ generate the HE secret keys SK_1, SK_2 and public keys $\mathsf{PK}_1 = \hat{\mathbf{A}}_1$ and $\mathsf{PK}_2 = \hat{\mathbf{A}}_2$ honestly, along with proofs π^1_{SK} and π^2_{SK} of the well-formedness of the public keys.
   (b) Additionally, users $\mathcal{U}_1$ and $\mathcal{U}_2$ provide their respective ciphertexts $\mathsf{CT}_b = \mathsf{HE.Enc}(\mathsf{PK}_1, m_b)$ and $\mathsf{CT}_{\bar{b}} = \mathsf{HE.Enc}(\mathsf{PK}_2, m_{\bar{b}})$ along with their proofs π^1_{CT} and π^2_{CT} to $\mathcal{S}^*$.
   (c) The signer $\mathcal{S}^*$ evaluates the signing algorithm homomorphically on ciphertexts CT_b and $\mathsf{CT}_{\bar{b}}$ to obtain CT_{σ_b} and $\mathsf{CT}_{\sigma_{\bar{b}}}$, respectively.
   (d) After receiving the evaluated HE ciphertexts from the signer $\mathcal{S}^*$, users $\mathcal{U}_1$ and $\mathcal{U}_2$ decrypt them using the HE secret keys SK_1 and SK_2 to obtain signatures σ_b and $\sigma_{\bar{b}}$ respectively.
   (e) The signer $\mathcal{S}^*$ is given σ_0, σ_1.
   (f) The signer $\mathcal{S}^*$ outputs its guess for bit b.

Hybrid_1: In this world, the proofs π^1_{SK} and π^2_{SK} are replaced with simulated proofs.

Hybrid_2: In this world, the proofs π^1_{CT} and π^2_{CT} are replaced with simulated proofs.

Hybrid_3: In this world, at Step 3(d), the signatures σ_b and $\sigma_{\bar{b}}$ are generated using $\mathsf{Sig.Sign}$ directly.

Hybrid_4: In this world, we replace $\mathsf{HE.Enc}(\mathsf{PK}_1, m_b)$ by $\mathsf{HE.Enc}(\mathsf{PK}_1, 0)$ and $\mathsf{HE.Enc}(\mathsf{PK}_2, m_{\bar{b}})$ by $\mathsf{HE.Enc}(\mathsf{PK}_2, 0)$.

One can see that the advantage of the adversary in Hybrid_4 is 0 since the bit b is information theoretically hidden. We proceed to argue that consecutive hybrids are indistinguishable. Indistinguishability of Hybrid_0 and Hybrid_1 as well as Hybrid_1 and Hybrid_2 follows from the zero-knowledge property of the underlying ZKPoKs. Indistinguishability of Hybrid_2 and Hybrid_3 follows from the correctness of BdPMW' and Sig.

Indistinguishability of Hybrid_3 and Hybrid_4 follows from the semantic security of BdPMW'. The proof is a standard reduction to the semantic security. Note that due to the previous hybrids, the blindness challenger does not need the HE secret key for any operations and can obtain the HE public keys and ciphertexts from the HE challenger. The proofs π_{SK} and π_{CT} are simulated, and the signatures σ_b and $\sigma_{\bar{b}}$ are computed directly using the signing key $\mathsf{Sig.sk}$. □

To upgrade BS from honest signer blindness to full-fledged blindness, we modify the scheme as follows:

- We use a ZKPoK to construct a proof π_{vk} that the verification key $\mathsf{Sig.vk}$ is well-formed (as we use Lyubashevsky's signature scheme, this is again a linear statement with a low-norm secret vector), and have the signer append π_{vk} to its verification key $\mathsf{Sig.vk}$.
- We use a homomorphic signature scheme enjoying context hiding security to authenticate the correct homomorphic evaluation of the $\mathsf{Sig.Sign}$ signing algorithm.

We defer the formal description and analysis of this variant of BS to the full version of this article.

Next, we show that our construction achieves one-more unforgeability.

Theorem 6.4. *Assume that the underlying signature scheme satisfies UF-CMA security. Then the BS construction satisfies one-more unforgeability.*

Proof. Assume there exists an adversary $\mathcal{U}^*$ who wins the one-more unforgeability game against the BS blind signature. Then we build an adversary $\mathcal{B}$ against the UF-CMA security of the underlying signature scheme Sig. $\mathcal{B}$ does the following:

1. It obtains $\mathsf{Sig.vk}$ from the signature challenger and forwards it to $\mathcal{U}^*$.
2. It runs the signing protocol with adversary $\mathcal{U}^*$ simulating the BS signer as follows:
   (a) Adversary $\mathcal{U}^*$ outputs the HE public key PK along with proof of knowledge π_{PK} of a corresponding SK. Adversary $\mathcal{B}$ rewinds $\mathcal{U}^*$ to extract SK.
   (b) Adversary $\mathcal{U}^*$ also outputs ℓ ciphertexts $\mathsf{CT}_1, \ldots, \mathsf{CT}_\ell$ along with their proofs $\pi_{\mathsf{CT}_1}, \ldots, \pi_{\mathsf{CT}_\ell}$. Adversary $\mathcal{B}$ uses SK to decrypt $\mathsf{CT}_1, \ldots, \mathsf{CT}_\ell$ to obtain $m_1, \ldots, m_\ell$.
   (c) Adversary $\mathcal{B}$ sends $m_1, \ldots, m_\ell$ to the UF-CMA signature challenger and obtains signatures $\sigma_1, \ldots, \sigma_\ell$.

(d) It encrypts $\sigma_1, \ldots, \sigma_\ell$ using HE to obtain $\mathsf{CT}'_{\sigma_1}, \ldots, \mathsf{CT}'_{\sigma_\ell}$. It runs $\mathsf{HE.Eval}(\mathsf{PK}, \mathcal{C}^0, (\mathsf{CT}'_{\sigma_i}, \star))$ for all $i \in [\ell]$ to obtain $\mathsf{CT}_{\sigma_1} \ldots, \mathsf{CT}_{\sigma_\ell}$. Here $\mathcal{C}^0$ is a dummy circuit that has the same depth and number of inputs as the $\mathsf{Sig.Sign}$ signing circuit and outputs its first input (see Definition 2.20), and $\star$ represents as many independent encryptions of 0 as required. Finally, it returns $\mathsf{CT}_{\sigma_1} \ldots, \mathsf{CT}_{\sigma_\ell}$ to $\mathcal{U}^*$.

(e) When $\mathcal{U}^*$ outputs $\ell + 1$ distinct message-signature pairs (m_i, σ_i) for $i \in [\ell + 1]$ that pass verification, it outputs any of these for which m_i had not been queried to the UF-CMA signature challenger.

As BdPMW' satisfies malicious circuit privacy, it follows that $\mathcal{U}^*$'s view is indistinguishable from what it expects. As a result, it follows that the success probability of $\mathcal{B}$ is a polynomial function of the success probability of $\mathcal{U}^*$. □

6.2 Comparison with the Hauck *et al* scheme

Below, we describe the main sources of inefficiency in the scheme of [43] and how we overcome these:

1. *Noise Flooding.* [43] relies on a variant of noise flooding, used with the Short Integer Solution problem (SIS). For λ-bit security, it leads to setting the SIS parameters such that $n \log q = \widetilde{\Omega}(\lambda^3)$, which is a lower bound on the signature bit-length. More precisely, the noise flooding derives from the requirement that the lattice-based linear hash function must be (ϵ, Q)-regular (see [43, Section 3]) with an ϵ that is $2^{-\Omega(\lambda)}$ for [43, Theorem 1] to be applicable to adversaries succeeding with probability $2^{-o(\lambda)}$. This is achieved in [43, Section 6] by considering two balls of the same radius that is $2^{\Omega(\lambda)}$ larger than the offset between the ball centers. In contrast, by a more careful proof using Rényi divergence, we only require flooding of $\widetilde{\Omega}(\sqrt{Q_S})$, where Q_S is the bound on the number of generated signatures.

2. *Loss in Security Proof.* The scheme by Hauck *et al* is an adaptation of the Okamoto-Schnorr blind signature [57] from the discrete logarithm setting to the lattice setting. The best known security proof for the Okamoto-Schnorr blind signature under standard assumptions [64], suffers from a loss in advantage larger than $2^{Q_S}/|\mathcal{C}|$. Here Q_S denotes an upper bound on the number of generated signatures and $\mathcal{C}$ denotes the challenge space. Note that $\log |\mathcal{C}|$ is a lower bound on the signature bit-length. This advantage loss stems from the ability of a malicious user to perform concurrent protocol executions with the signer and was recently shown to be intrinsic [10]. In the lattice setting, the proof from [43] suffers from a similar loss, which drastically limits Q_S or forces to have signature bit-lengths that grow at least linearly in Q_S (note that no devastating attack as the one from [10] is known for the scheme [43], so a better proof could possibly exist). In contrast, our protocol has only two rounds and hence concurrency issues do not come up.

These differences lead to the following comparison in terms of concrete parameters. As discussed in Section 3.3, by using a variant of Dilithium-G with ≈ 128 classical bit security, the size of our signature is below 3 KB, where the adversary is limited to getting 256 signatures. In contrast, using the parameters provided by [43], the authors obtain signatures of size ≈ 7.73MB, for adversaries limited to getting 7 signatures. In terms of verification cost, our scheme also significantly outperforms the one of Hauck *et al*, as verification boils down to Dilithium-G signature verification.

At the outset, the user has to generate HE keys and ciphertexts and appropriate NIZK proofs. A standard trick used in the context of HE to reduce communication complexity is to use HE in conjunction with an efficient, plain symmetric key encryption scheme SKE. Let K be the secret key of SKE. Then, the user computes an HE encryption of K and an SKE encryption of the message m that it wishes to sign. To sign the message, the signer first computes an HE evaluation of a "translation" circuit that takes the HE ciphertext of K and the SKE ciphertext of message m and outputs an HE ciphertext of m by running the SKE decryption circuit homomorphically on the SKE ciphertext. As the SKE expansion factor is much less than the HE expansion factor, this reduces the communication cost of the first round very significantly. Another advantage of this approach is that we now need to only provide proof of well-formedness of the single HE ciphertext of K, since well-formedness of the SKE ciphertexts when translated to HE is automatic by the correctness of the HE decryption algorithm. The downside is that it increases the signer cost, but probably not very significantly, because it already has to homomorphically evaluate a hash function (see below).

The transcript of the second round of communication is an HE ciphertext that decrypts to the desired signature. Its bit-size greatly depends on how much work we push to the signer side. Given that the last steps of the signing algorithm are linear algebra operations, it is likely that the chosen HE solution would be BGV [16] or FV [32] with multiple plaintext slots [70]. In that case, one could use a modulus-switch before sending the ciphertext, to decrease its size. A more costly strategy would be to bootstrap to a format that has smaller ciphertext expansion factor (note that no more HE computations have to be performed, at that stage). Depending on the chosen solution, one may probably expect a blow-up between a factor of 20 and 200. Starting with a 3 KB signature, this remains below the communication cost of the Hauck *et al* scheme, which is ≈ 33.2 MB.[5]

The comparison is more complex, and likely less favourable, for the signer cost. It is difficult to assess it precisely, as this depends on specific choices of building blocks, as discussed below.

6.3 Towards an Instantiation

The core task in instantiating the blind signature scheme based on the rejection-free Lyubashevsky signature from Section 3 is to select components that can be efficiently evaluated homomorphically. A central difficulty is the choice of the hash function that is modelled as a random oracle: choosing an "HE-friendly" hash function is a highly nontrivial question. While there has been some effort in the community to design block ciphers that minimize multiplicative size and depth of description, so as to be more HE-friendly [2, 1], this question has not been studied sufficiently for the case of hash functions which must model random oracles, to the best of our knowledge. A starting point idea, discussed in [24], is to use a block cipher with small multiplicative complexity such as LowMC [2] as a permutation in a sponge construction [11]. This would result in a hash function which behaves like a random oracle, and with low multiplicative size and depth. An alternative path would be to start from the FLIP stream cipher [56], which is HE-friendly for the GSW homomorphic encryption scheme [41].

Another difficulty in concrete instantiation is to handle the diverse formats of the quantities involved during the homomorphic signing. If the message to be signed is encrypted using an SKE, as suggested above, then the first step of the signer is to convert it to an HE-ciphertext. Next, the

[5] Using the notations from [43, Figure 9], the first transcript has size $\eta n \log_2(q)$, which is ≈ 26 MB, the second transcript has size $n \log_2(2d_c)$, which is much smaller, and the third transcript has size $mn \log_2(2d_s)$, which is ≈ 7.2 MB.

message is fed to the hash function (along with a vector that can be computed in the clear). Hence, the first HE-format should be compatible with the one needed for evaluating the hash function. In Dilithium-G, the output of the hash function is a sparse polynomial with ternary coefficients with exactly 60 non-zero entries (which are uniform and independent). Mapping the output of the hash function to such a format may prove complex to do homormorphically, and it may be better to choose a more amenable format for the signature challenge.

Other decisions include how to choose the HE scheme [16, 32] with appropriately chosen plaintext slots [70] to perform linear algebra operations, how to compress the signature elements to decrease bitlength in lieu of homomorphism – for instance, Dilithium-G uses a Huffman encoding but this may be expensive to perform under an HE layer.

Acknowledgments. The authors thank Léo Ducas and Adeline Roux-Langlois for insightful discussions. This work was partly supported by the DST "Swarnajayanti" fellowship, an Indo-French CEFIPRA project, National Blockchain Project, the CCD Centre of Excellence, European Union Horizon 2020 Research and Innovation Program Grant 780701, and BPI-France in the context of the national project RISQ (P141580). Part of the research corresponding to this work was conducted while the first two authors were visiting the Simons Institute for the Theory of Computing.

References

[1] M. R. Albrecht, L. Grassi, C. Rechberger, A. Roy, and T. Tiessen. MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In *ASIACRYPT*, 2016.

[2] M. R. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, and M. Zohner. Ciphers for MPC and FHE. In *EUROCRYPT*, 2015.

[3] N. A. Alkadri, R. E. Bansarkhani, and J. Buchmann. BLAZE: practical lattice-based blind signatures for privacy-preserving applications. In *Financial Cryptography*, 2020.

[4] N. A. Alkadri, R. E. Bansarkhani, and J. Buchmann. On lattice-based interactive protocols: An approach with less or no aborts. In *ACISP*, 2020.

[5] B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*, 2009.

[6] S. Bai and S. D. Galbraith. An improved compression technique for signatures based on learning with errors. In *CT-RSA*, 2014.

[7] S. Bai, T. Lepoint, A. Roux-Langlois, A. Sakzad, D. Stehlé, and R. Steinfeld. Improved security proofs in lattice-based cryptography: using the Rényi divergence rather than the statistical distance. *J. Cryptol.*, 2018.

[8] M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *CCS*, 2006.

[9] R. Bendlin and I. Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In *TCC*, 2010.

[10] F. Benhamouda, T. Lepoint, M. Orrù, and M. Raykova. On the (in)security of ROS. *IACR Cryptol. ePrint Arch.*, 2020.

[11] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. On the indifferentiability of the sponge construction. In *EUROCRYPT*, 2008.

[12] D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. R. Rasmussen, and A. Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In *CRYPTO*, 2018.

[13] J. Bootle, V. Lyubashevsky, and G. Seiler. Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In *CRYPTO*, 2019.

[14] S. Bouaziz-Ermann, S. Canard, G. Eberhart, G. Kaim, A. Roux-Langlois, and J. Traoré. Lattice-based (partially) blind signature without restart. *IACR Cryptol. ePrint Arch.*, 2020.

[15] F. Bourse, R. del Pino, M. Minelli, and H. Wee. FHE circuit privacy almost for free. In *CRYPTO*, 2016.

[16] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theory*, 2014.

[17] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé. Classical hardness of learning with errors. In *STOC*, 2013.

[18] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, 2011.

[19] G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker. Two-party ECDSA from hash proof systems and efficient instantiations. In *CRYPTO*, 2019.

[20] G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker. Bandwidth-efficient threshold EC-DSA. In *PKC*, 2020.

[21] D. Chaum. Blind signatures for untraceable payments. In *CRYPTO*, 1982.

[22] D. Cozzo and N. P. Smart. Sharing the LUOV: threshold post-quantum signatures. In *IMACC*, 2019.

[23] I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, 2012.

[24] D. Derler, S. Ramacher, and D. Slamanig. Post-quantum zero-knowledge proofs for accumulators with applications to ring signatures from symmetric-key primitives. In *PQCrypto*, 2018.

[25] Y. Desmedt. Threshold cryptography. *European Transactions on Telecommunications*, 1994.

[26] J. Doerner, Y. Kondi, E. Lee, and A. Shelat. Threshold ECDSA from ECDSA assumptions: The multiparty case. In *S&P*, 2019.

[27] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018.

[28] L. Ducas, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé. CRYSTALS – Dilithium: Digital signatures from module lattices. Cryptology ePrint Archive, 2017. Version 1, dated 27/06/2017.

[29] L. Ducas and T. Prest. Fast Fourier orthogonalization. In *ISSAC*, 2016.

[30] L. Ducas and D. Stehlé. Sanitization of FHE ciphertexts. In *EUROCRYPT*, 2016.

[31] M. F. Esgin, N. K. Nguyen, and G. Seiler. Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. 2020.

[32] J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, 2012.

[33] M. Fischlin and D. Schröder. On the impossibility of three-move blind signature schemes. In *EUROCRYPT*, 2010.

[34] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang. Falcon: Fast-Fourier lattice-based compact signatures over NTRU. Technical report. Specification v1.0, available at https://falcon-sign.info/.

[35] T. K. Frederiksen, M. Keller, E. Orsini, and P. Scholl. A unified approach to MPC with preprocessing using OT. In *ASIACRYPT*, 2015.

[36] S. Garg and D. Gupta. Efficient round optimal blind signatures. In *EUROCRYPT*, 2014.

[37] S. Garg, V. Rao, A. Sahai, D. Schröder, and D. Unruh. Round optimal blind signatures. In *CRYPTO*, 2011.

[38] R. Gennaro and S. Goldfeder. Fast multiparty threshold ECDSA with fast trustless setup. In *CCS*, 2018.

[39] C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.

[40] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, 2008.

[41] C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, 2013.

[42] S. Goldwasser, Y. T. Kalai, C. Peikert, and V. Vaikuntanathan. Robustness of the learning with errors assumption. In *ICS*, 2010.

[43] E. Hauck, E. Kiltz, J. Loss, and N. K. Nguyen. Lattice-based blind signatures, revisited. In *CRYPTO*, 2020.

[44] G. Herold, E. Kirshanova, and A. May. On the asymptotic complexity of solving LWE. *Des. Codes Cryptogr.*, 2018.

[45] J. Howe, T. Prest, T. Ricosset, and M. Rossi. Isochronous gaussian sampling: From inception to implementation. In *PQCrypto*, 2020.

[46] S. Ibrahim, M. Kamat, M. Salleh, and S. A. Aziz. Secure E-voting with blind signature. In *NCTT*, 2003.

[47] A. Juels, M. Luby, and R. Ostrovsky. Security of blind digital signatures (extended abstract). In *CRYPTO*, 1997.

[48] A. Langlois, D. Stehlé, and R. Steinfeld. GGHLite: More efficient multilinear maps from ideal lattices. In *EUROCRYPT*, 2014.

[49] H. Q. Le, W. Susilo, T. X. Khuc, M. K. Bui, and D. H. Duong. A blind signature from module lattices. In *DSC*, 2019.

[50] Y. Lindell and A. Nof. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In *CCS*, 2018.

[51] S. Ling, K. Nguyen, D. Stehlé, and H. Wang. Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In *PKC*, 2013.

[52] V. Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT*, 2009.

[53] V. Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, 2012.

[54] V. Lyubashevsky and D. Micciancio. Generalized compact knapsacks are collision resistant. In *ICALP*, 2006.

[55] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, 2010.

[56] P. Méaux, A. Journault, F. Standaert, and C. Carlet. Towards stream ciphers for efficient FHE with low-noise ciphertexts. In *EUROCRYPT*, 2016.

[57] T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *CRYPTO*, 1992.

[58] R. Ostrovsky, A. Paskin-Cherniavsky, and B. Paskin-Cherniavsky. Maliciously circuit-private FHE. In *CRYPTO*, 2014.

[59] D. Papachristoudis, D. Hristu-Varsakelis, F. Baldimtsi, and G. Stephanides. Leakage-resilient lattice-based partially blind signatures. *IET Information Security*, 2019.

[60] C. Peikert and A. Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *TCC*, 2006.

[61] D. Pointcheval and J. Stern. Provably secure blind signature schemes. In *ASIACRYPT*, 1996.

[62] D. Pointcheval and J. Stern. Security proofs for signature schemes. In *EUROCRYPT*, 1996.

[63] D. Pointcheval and J. Stern. New blind signatures equivalent to factorization (extended abstract). In *CCS*, 1997.

[64] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *J. Cryptol.*, 2000.

[65] T. Pornin and T. Prest. More efficient algorithms for the NTRU key generation using the field norm. In *PKC*, 2019.

[66] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 2009.

[67] M. Rückert. Lattice-based blind signatures. In *ASIACRYPT*, 2010.

[68] C.-P. Schnorr. Efficient signature generation by smart cards. *J. Cryptol.*, 1991.

[69] P. Schwabe, R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, G. Seiler, and D. Stehlé. CRYSTALS-Kyber: Algorithm specifications and supporting documentation (version 3.0). https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions.

[70] N. P. Smart and F. Vercauteren. Fully homomorphic SIMD operations. *Des. Codes Cryptogr.*, 2014.

[71] D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa. Efficient public key encryption based on ideal lattices. In *ASIACRYPT*, 2009.

[72] K. Takashima and A. Takayasu. Tighter security for efficient lattice cryptography via the rényi divergence of optimized orders. In *ProvSec*, 2015.

[73] R. Yang, M. H. Au, Z. Zhang, Q. Xu, Z. Yu, and W. Whyte. Efficient lattice-based zero-knowledge arguments with standard soundness: Construction and applications. In *CRYPTO*, 2019.

[74] X. Yi and K.-Y. Lam. A new blind ECDSA scheme for bitcoin transaction anonymity. In *Asia-CCS*, 2019.

[75] C. Yin, S. Huang, P. Su, and C. Gao. Secure routing for large-scale wireless sensor networks. In *ICCT*, 2003.

[76] R. K. Zhao, R. Steinfeld, and A. Sakzad. COSAC: compact and scalable arbitrary-centered discrete Gaussian sampling over integers. In *PQCrypto*, 2020.

Supplementary Material

A Additional Preliminaries

A.1 Multi-data Homomorphic Signature.

A homomorphic signature scheme is a signature scheme that allows computations on authenticated data. In a multi-data homomorphic signature scheme, the signer can sign many different datasets of arbitrary size. Each dataset is tied to some label τ (e.g., the name of the dataset) and the verifier is assumed to know the label of the dataset over which it wishes to verify computations.

Definition A.1 (Multi-data Homomorphic Signature)**.** A *multi-data homomorphic signature* for messages over a set $\mathcal{X}$ is a tuple of PPT algorithms $(\mathsf{PrmsGen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{SignEval}, \mathsf{Process}, \mathsf{Verify})$ with the following syntax.

- $\mathsf{prms} \leftarrow \mathsf{PrmsGen}(1^\lambda, 1^N)$: Gets the security parameter λ and a data-size bound N and generates public parameters prms.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{prms})$: Produces a public verification key pk and a secret signing key sk.
- $(\sigma_\tau, \sigma_1, \ldots, \sigma_N) \leftarrow \mathsf{Sign}(\mathsf{sk}, (x_1, \ldots, x_N), \tau)$: Signs some data $(x_1, \ldots, x_N) \in \mathcal{X}^*$ under a label $\tau \in \{0,1\}^*$.
- $\sigma^* = \mathsf{SignEval}(\mathsf{prms}, g, \sigma_\tau, (x_1, \sigma_1), (x_\ell, \sigma_\ell))$: Homomorphically computes a signature σ^* for $g(x_1, \ldots, x_N)$.
- $\alpha_g \leftarrow \mathsf{Process}(\mathsf{prms}, g)$: Produces a "public-key" α_g for the function g.
- $\mathsf{Verify}(\mathsf{pk}, \alpha_g, y, \tau, (\sigma_\tau, \sigma^*))$: Verifies that $y \in \mathcal{X}$ is indeed the output of the function g over the data signed with label τ. We can define the "combined verification procedure" $\mathsf{Verify}^*(\mathsf{pk}, g, y, \tau, \sigma_\tau, \sigma^*)$ as: Compute $\alpha_g \leftarrow \mathsf{Process}(\mathsf{prms}, g)$ and output $\mathsf{Verify}(\mathsf{pk}, \alpha_g, y, \tau, (\sigma_\tau, \sigma^*))$.

A homomorphic signature should satisfy the correctness and security properties defined below.

Definition A.2 (Correctness)**.** *Correctness of Signing.* Let $\mathsf{id}_i : \mathcal{X}^N \to \mathcal{X}$ be a canonical description of the function $\mathsf{id}_i(x_1, ..., x_N) = x_i$ (i.e., a circuit consisting of a single wire taking the i'th input to the output). We require that for any $\mathsf{prms} \leftarrow \mathsf{PrmsGen}(1^\lambda, 1^N)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{prms})$, $(x_1, \ldots, x_N) \in \mathcal{X}^N$, any $\tau \in \{0,1\}^*$, and any $(\sigma_\tau, \sigma_1, \ldots, \sigma_N) \leftarrow \mathsf{Sign}(\mathsf{sk}, (x_1, \ldots, x_N), \tau)$ following must satisfy: $\mathsf{Verify}^*(\mathsf{pk}, \mathsf{id}_i, x_i, \tau, (\sigma_\tau, \sigma_i)) = \mathsf{accept}$. In other words, the pair (σ_τ, σ_i) certifies x_i as the ith data item of the data with label τ.
Correctness of Evaluation. For any functions $h_1, \ldots, h_\ell$ with $h_i : \mathcal{X}^N \to \mathcal{X}$ for $i \in [\ell]$, any function $g : \mathcal{X}^\ell \to \mathcal{X}$, any $(x_1, \ldots, x_\ell) \in \mathcal{X}^\ell$, any $\tau \in \{0,1\}^*$ and any $(\sigma_\tau, \sigma_1, \ldots, \sigma_\ell)$:

$$\begin{aligned} &\{\{\mathsf{Verify}(\mathsf{pk}, h_i, x_i, \tau, (\sigma_\tau, \sigma_i)) = \mathsf{accept}\}_{i \in [\ell]}, \\ &\sigma^* \leftarrow \mathsf{SignEval}(\mathsf{prms}, g, \sigma_\tau, (x_1, \sigma_1), (x_\ell, \sigma_\ell))\} \\ \Rightarrow\ &\mathsf{Verify}^*(\mathsf{pk}, (g \circ \bar{h}), g(x_1, \ldots, x_\ell), \tau, (\sigma_\tau, \sigma^*)) = \mathsf{accept}. \end{aligned}$$

In other words, if the signatures (σ_τ, σ_i) certify x_i as the outputs of function h_i over the data labeled with τ for all $i \in [\ell]$, then (σ_τ, σ^*) certifies $g(x_1, \ldots, x_\ell)$ as the output of $g \circ \bar{h}$ over the data labeled with τ.

Definition A.3 (Security)**.** The security is defined via the following game between an attacker $\mathcal{A}$ and a challenger:

- The challenger runs $\mathsf{prms} \leftarrow \mathsf{PrmsGen}(1^\lambda, 1^N)$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{prms}, 1^\lambda)$, and gives $\mathsf{prms}, \mathsf{pk}$ to the attacker $\mathcal{A}$.
- Signing queries: The attacker $\mathcal{A}$ can ask an arbitrary number of signing queries. In each query j, the attacker chooses a fresh tag τ_j which was never queried previously and a message $(x_{j1}, \ldots, x_{jN_j}) \in \mathcal{X}^*$. The challenger responds with
$$(\sigma_{\tau_j}, \sigma_{j1}, \ldots, \sigma_{jN_j}) \leftarrow \mathsf{Sign}(\mathsf{sk}, (x_{j1}, \ldots, x_{jN_j}), \tau_j).$$
- The attacker $\mathcal{A}$ outputs a function $g : \mathcal{X}^{N'} \to \mathcal{X}$ and values $\tau, y', (\sigma'_\tau, \sigma')$. The attacker wins if $\mathsf{Verify}^*(\mathsf{pk}, g, y', \tau, (\sigma'_\tau, \sigma')) = \mathsf{accept}$ and either:
  - *Type 1 forgery*: $\tau \notin \{\tau_j\}_j$ or $\tau = \tau_j$ for some j but $N' \neq N_j$, i.e., the signing query with label τ was never made or there is a mismatch between the size of the data signed under label τ and the arity of the function g.
  - *Type 2 forgery*: $\tau = \tau_j$ for some j with corresponding message $x_{j,1}, \ldots, x_{j,N'}$ such that (a) g is admissible on $x_{j,1}, \ldots, x_{j,N'}$ and (b) $y' \neq g(x_{j,1}, \ldots, x_{j,N'})$.

We require that for all $\mathcal{A}$ with run-time $2^{o(\lambda)}$, we have $\Pr[\mathcal{A} \text{ wins}] \leq 2^{-\Omega(\lambda)}$ in the above game.

We now give a simulation-based notion of context-hiding security, requiring that a context hiding signature $\tilde{\sigma}$ can be simulated given the knowledge of only the computation g and output y, but without any other knowledge of underlying data. The simulation remains indistinguishable even given the underlying data, the underlying signatures, and even the public/secret key of the scheme. In other words, the derived signature does not reveal anything beyond the output of the computation even to an attacker that may have some partial information on the underlying values.

Definition A.4 (Context Hiding)**.** A multi-data homomorphic signature supports context hiding if there exist additional PPT procedures $\tilde{\sigma} \leftarrow \mathsf{Hide}(\mathsf{pk}, y, \sigma)$, $\mathsf{HVerify}(\mathsf{pk}, g, \mathsf{Process}(g), y, \tau, (\sigma_\tau, \tilde{\sigma}))$ such that:

- *Correctness:* For any $\mathsf{prms} \leftarrow \mathsf{PrmsGen}(1^\lambda, 1^N)$, any $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{prms})$ and any $\alpha, y, \tau, \sigma_\tau, \sigma$ such that $\mathsf{Verify}(\mathsf{pk}, \alpha, y, \tau, (\sigma_\tau, \sigma)) = \mathsf{accept}$, for any $\tilde{\sigma} \leftarrow \mathsf{Hide}(\mathsf{pk}, y, \sigma)$ we have
$$\mathsf{HVerify}(\mathsf{pk}, \alpha, y, \tau, (\sigma_\tau, \tilde{\sigma}) = \mathsf{accept}.$$
- *Unforgeability:* Multi-data signature security holds when we replace the Verify procedure by $\mathsf{HVerify}$ in the security game.
- *Context hiding security:* Firstly, in the procedure $(\sigma_\tau, \{\sigma_i\}_{i\in[N]}) \leftarrow \mathsf{Sign}(\mathsf{sk}, \{x_i\}_{i\in[N]}, \tau)$, we require that σ_τ can only depend on (sk, N, τ) but not on the data $\{x_i\}$. Secondly, we require that there is a simulator Sim such that for any fixed (worst-case) choice of prms, $(\mathsf{pk}, \mathsf{sk})$ and any $\alpha, y, \tau, \sigma_\tau, \sigma$ such that $\mathsf{Verify}(\mathsf{pk}, \alpha, y, \tau, (\sigma_\tau, \sigma)) = \mathsf{accept}$, we have that the distributions $\mathsf{Hide}(\mathsf{pk}, y, \sigma)$ and $\mathsf{Sim}(\mathsf{sk}, \alpha, y, \tau, \sigma_\tau)$ are indistinguishable, where the randomness is only over the random coins of the simulator and the Hide procedure. We say that such schemes are statistically context hiding if the above indistinguishability holds statistically.

B Missing Details in Section 3

B.1 Proof of Lemma 3.5

Let $D_H = \{\mathbf{c} : \mathbf{c} \in \{-1, 0, 1\}^k, \|\mathbf{c}\|_1 \leq \alpha\}$. Using $\mathcal{F}$ we design $\mathcal{B}$ for the SIS problem. Given $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $\mathcal{B}$ does the following:

1. Randomly selects $\mathbf{S} \leftarrow \{-d, \ldots, 0, \ldots, d\}^{m \times k}$. Sets $\mathsf{Sign.vk} = (\mathbf{A}, \mathbf{T} = \mathbf{AS})$ and $\mathsf{Sign.sk} = \mathbf{S}$.
2. Sets random coins ϕ and ψ to be used by the forger $\mathcal{F}$ and the signing challenger, respectively.
3. Randomly selects $\mathbf{r}_1, \ldots, \mathbf{r}_t \leftarrow D_H$, where $t = Q_S + Q_H$. These values are used to answer random oracle queries by the forger or to program the random oracle in the process of answering signing queries.
4. Calls a sub-routine $\mathcal{A}(\mathbf{A}, \mathbf{T}, \phi, \psi, \{\mathbf{r}_1, \ldots, \mathbf{r}_t\})$ which initializes the forging algorithm $\mathcal{F}$ with the verification key $\mathsf{Sign.vk}$. Then runs $\mathcal{F}$ using the random coins ϕ.
   (a) To answer any signing query issued by $\mathcal{F}$, the random oracle is programmed with the first unused value $\mathbf{r}_i$. Similarly, any random oracle query made by $\mathcal{F}$ is answered with the first unused value $\mathbf{r}_i$. If a random oracle query is repeated on the same input, then the same value is returned (for this, $\mathcal{A}$ maintains a list).
   (b) With probability δ, $\mathcal{F}$ outputs a valid forgery $(\mu, (\mathbf{z}, \mathbf{c}))$. $\mathcal{B}$ uses this to find a small vector $\mathbf{v}$ such that $\mathbf{Av} = \mathbf{0}$.

Let us refer to the list $\{\mathbf{r}_1, \ldots, \mathbf{r}_t\}$ as R. Then we can assume that $\mathbf{c} \in R$ and was programmed on some input $\mathbf{w} = \mathbf{Az} - \mathbf{Tc}$ for some RO or signing query. Because otherwise it means $\mathcal{F}$ has produced $\mathbf{c}$ such that $H(\mathbf{w}, \mu) = \mathbf{c}$ which is possible with probability at most $\frac{1}{|D_H|}$. Hence with probability at least $(\delta - \frac{1}{|D_H|})$, $\mathcal{F}$ outputs a forgery where $\mathbf{c} \in R$. Now, two cases are possible:

Case(i): $\mathbf{c} = \mathbf{r}_i$, where $\mathbf{r}_i$ is programmed while answering a signing query on message μ'. Let the signature returned by the challenger be $(\mathbf{z}', \mathbf{r}_i)$. Thus, $H(\mathbf{Az}' - \mathbf{Tr}_i, \mu') = \mathbf{r}_i = H(\mathbf{Az} - \mathbf{Tr}_i, \mu)$. This implies that $(\mathbf{Az}' - \mathbf{Tr}_i, \mu') = (\mathbf{Az} - \mathbf{Tr}_i, \mu)$ (because otherwise it means that the forger has found a different preimage for $\mathbf{r}_i$). Thus, $\mu = \mu'$ and $\mathbf{Az}' - \mathbf{Tr}_i = \mathbf{Az} - \mathbf{Tr}_i \implies \mathbf{A}(\mathbf{z} - \mathbf{z}') = \mathbf{0}$. But $(\mathbf{z} - \mathbf{z}') \neq \mathbf{0}$ because in that case the forgery is not a valid one since both message and signature are same as the one already queried for. Thus $\mathcal{B}$ can output $\mathbf{v} = \mathbf{z} - \mathbf{z}'$. Also, since $\|\mathbf{z}\| \leq \gamma$, $\|\mathbf{z} - \mathbf{z}'\| \leq 2\gamma$.

Case(ii): $\mathbf{c} = \mathbf{r}_i$, where $\mathbf{r}_i$ is response to a random oracle query by $\mathcal{F}$. Adversary $\mathcal{B}$ notes the signature $(\mathbf{z}, \mathbf{r}_i)$, randomly selects $\mathbf{r}'_i, \ldots, \mathbf{r}'_k$ from D_H and again runs the subroutine $\mathcal{A}(\mathbf{A}, \mathbf{T}, \phi, \psi, \{\mathbf{r}_1, \ldots, \mathbf{r}_{i-1}, \mathbf{r}'_i, \ldots, \mathbf{r}'_t\})$. By using general forking lemma ([8, Lemma 1]), probability that $\mathcal{F}$ will use $\mathbf{r}'_i$ in generating the forgery (i.e., outputs $(\mathbf{z}', \mathbf{r}'_i)$ as the new forgery) is at least

$$\left(\delta - \frac{1}{|D_H|}\right)\left(\frac{\delta - 1/|D_H|}{Q_S + Q_H} - \frac{1}{|D_H|}\right).$$

Hence, with the above probability we get $\mathbf{Az} - \mathbf{Tr}_i = \mathbf{Az}' - \mathbf{Tr}'_i$ which implies $\mathbf{A}(\mathbf{z} - \mathbf{z}') + \mathbf{A}(\mathbf{Sr}'_i - \mathbf{S}r_i) = \mathbf{0}$. Thus $\mathcal{B}$ can set $\mathbf{v} = \mathbf{z} - \mathbf{z}' + \mathbf{Sr}'_i - \mathbf{Sr}_i$ which has size at most $2\gamma + 2\sqrt{m}d\alpha$.

All that is now left is to show that $\mathbf{z} - \mathbf{z}' + \mathbf{Sr}'_i - \mathbf{Sr}_i$ is not zero. Here, we use the result from [53, Lemma 5.2] that with probability at least $(1 - \epsilon)$, where $\epsilon = \frac{q^n}{(2d+1)^m}$, there exists another

$\mathbf{S}'$ which differs from $\mathbf{S}$ only at column i such that $\mathbf{AS} = \mathbf{AS}'$. Since the secret $\mathbf{S}$ is not given to the subroutine $\mathcal{A}$, none of the responses depend on $\mathbf{S}$. Also, if $\mathbf{z} - \mathbf{z}' + \mathbf{S}\mathbf{r}'_i - \mathbf{S}\mathbf{r}_i = \mathbf{0}$, then $\mathbf{z} - \mathbf{z}' + \mathbf{S}'\mathbf{r}'_i - \mathbf{S}'\mathbf{r}_i$ cannot be zero. Thus with probability at least $1/2$ the secret $\mathbf{S}$ chosen will satisfy the condition that $\mathbf{z} - \mathbf{z}' + \mathbf{S}\mathbf{r}'_i - \mathbf{S}\mathbf{r}_i \neq \mathbf{0}$. Combining all the cases, we get that $\mathcal{B}$ succeeds in outputting a vector $\mathbf{v}$ such that $\|\mathbf{v}\| \leq (2\gamma + 2\sqrt{m}d\alpha)$ and $\mathbf{Av} = 0$ with probability at least

$$\left(\frac{1}{2} - \frac{\epsilon}{2}\right)\left(\delta - \frac{1}{|D_H|}\right)\left(\frac{\delta - 1/|D_H|}{Q_S + Q_H} - \frac{1}{|D_H|}\right).$$

B.2 Optimality of Flooding in Section 3

In this section, we show that the flooding amount used in the construction in Section 3 is essentially optimal, and in particular that the dependence on $\sqrt{Q}$ is necessary. In more detail, we show that if the flooding noise is smaller than this, then an adversary can recover the signing key. Note that this attack is folklore, we recall it for the sake of completeness.

Statistical Attack. Recall that the signature for message M_i is of the form $(\mathbf{z}_i, \mathbf{c}_i)$, where $\mathbf{z}_i = \mathbf{S}\mathbf{c}_i + \mathbf{y}_i$, $\mathbf{c}_i \in \{-1, 0, 1\}^k$, $\|\mathbf{c}_i\|_1 \leq \alpha$, and $\mathbf{S}$ is the signing key. The adversary can obtain many such pairs corresponding to different messages. Let Q be the maximum number of signing queries that the adversary can make. Let $\mathbf{S}_i$ represents the ith row of matrix $\mathbf{S}$. Let $\mathbf{c}_{ij}$, $\mathbf{y}_{ij}$ and $\mathbf{S}_{ij}$ represent the jth entry in vectors $\mathbf{c}_i$, $\mathbf{y}_i$ and $\mathbf{S}_i$ respectively. Consider such tuples $(\mathbf{z}_i, \mathbf{c}_i)$ where $\mathbf{c}_{i1} = 1$. Let $B \subseteq [Q]$ be the set of such indices. The adversary gets approximately $Q/3$ such tuples corresponding to $i \in B$. For each i, using the first row of $\mathbf{S}$, we may write:

$$\mathbf{S}_{11} + \sum_{j=2}^{k} \mathbf{S}_{1j}\mathbf{c}_{ij} + \mathbf{y}_{i1} = \mathbf{z}_{i1} \tag{B.1}$$

We denote the average of $\sum_{j=2}^{k} \mathbf{S}_{1j}\mathbf{c}_{ij} + \mathbf{y}_{i1}$ over $i \in B$ as avg. We show that unless $\mathbf{y}_{i1}$ is $O(\sqrt{Q})$, we can recover $\mathbf{S}_{11}$. To conduct the attack, we bound each summand of avg separately.

Claim B.1. *Let $t_1 < 1/2$ be a positive constant and Q, k, d, α be as above. Then,*

$$\Pr\left[\left|\frac{\sum_{i\in B}\sum_{j=2}^{k} \mathbf{S}_{1j}\mathbf{c}_{ij}}{|B|}\right| < t_1\right] \geq 1 - 2\exp\left(\frac{-Qt_1^2}{6(\alpha-1)^2d^2}\right)$$

Proof. Note that $\sum_{j=2}^{k} \mathbf{S}_{1j}\mathbf{c}_{ij}$ takes values in the range $[-(\alpha-1)d, (\alpha-1)d]$, with mean at 0. In more detail, let X be a random variable, with mean 0 and bound $[-(\alpha-1)d, (\alpha-1)d]$, then for some positive constant $t_1 < 1/2$, we have from Hoeffding's bound

$$\begin{aligned}
&\Pr[|\bar{X} - E[X]| \geq t_1] && \leq 2\exp\left(\frac{-(Q/3)t_1^2}{2(\alpha-1)^2d^2}\right) \\
\implies \quad &\Pr[|\bar{X}| \geq t_1] && \leq 2\exp\left(\frac{-Qt_1^2}{6(\alpha-1)^2d^2}\right) \\
\implies \quad &\Pr[|\bar{X}| < t_1] && \geq 1 - 2\exp\left(\frac{-Qt_1^2}{6(\alpha-1)^2d^2}\right)
\end{aligned}$$

Since d is small, in particular if $(6(\alpha-1)^2d^2 < Qt_1^2)$, then $1 - 2\exp(\frac{-Qt_1^2}{6(\alpha-1)^2d^2})$ is non-negligible. □

Let us assume that the average of $\mathbf{y}_{i1}$ is also smaller than $1/2 - t_1$ with non negligible probability. Then, $\mathsf{avg} < 1/2$ with non negligible probability. Summing both sides of Equation B.1 over the set B, we get

$$\mathbf{S}_{11} + \mathsf{avg} = \frac{\sum_{i \in B} \mathbf{z}_{i1}}{|B|}$$

In this case the adversary can successfully recover $\mathbf{S}_{11}$ as

$$\mathbf{S}_{11} = \left\lfloor \frac{\sum_{i \in B} \mathbf{z}_{i1}}{|B|} \right\rceil$$

We now examine how large $\mathbf{y}_{i1}$ must be to avoid this attack. Let $Y \leftarrow \mathcal{D}_\sigma$ be the random variable representing the distribution of $\mathbf{y}_{i1}$ values. Then from Hoeffding's bound, for some constant c' and $t_2 < (1/2 - t_1)$,

$$\begin{aligned} & \Pr[|\bar{Y} - E[Y]| \ \ \geq t_2] \leq 2\exp(-c'Qt_2^2/3\sigma^2) \\ \implies & \Pr[|\bar{Y}| \ \ \geq t_2] \leq 2\exp(-c'Qt_2^2/3\sigma^2) \\ \implies & \Pr[\ |\bar{Y}| \ \ < t_2] \geq 1 - 2\exp(-c'Qt_2^2/3\sigma^2) \end{aligned}$$

Thus, if $3\sigma^2 < c'Qt_2^2$, then $1 - 2\exp(-c'Qt_2^2/3\sigma^2)$ is non-negligible. Hence, for the average of $\mathbf{y}_{i1}$ to be greater than t_2, we need that $3\sigma^2 \geq c'Qt_2^2$, i.e. σ must grow proportional to $\sqrt{Q}$.

C Fully Adaptive Unforgeability in the Preprocessing Model

A TS scheme satisfies fully adaptive unforgeability if for any adversary $\mathcal{A}$ with run-time $2^{o(\lambda)}$, the output of the following experiment is 1 with probability $2^{-\Omega(\lambda)}$:
$\mathsf{Expt}_{\mathcal{A},\mathsf{TS},FA-uf}(1^\lambda)$:

1. On input the security parameter λ, the adversary $\mathcal{A}$ outputs an access structure $\mathsf{A} \in \mathbb{S}$.
2. The challenger runs the $\mathsf{TS.KeyGen}(1^\lambda)$ algorithm and generates public parameters pp, verification key vk and the set of N key shares $\{\mathsf{sk}_i\}_{i=1}^N$. It sends pp and vk to $\mathcal{A}$.
3. Initialize $S = \phi$. Repeat:
   - Adversary $\mathcal{A}$ issues a polynomial number of adaptive signing queries where in each query, $\mathcal{A}$ outputs a message m and the challenger computes $\sigma_i \leftarrow \mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m)$ for all $i \in [N] \setminus S$, and provides $\{\sigma_i\}_{i \in [N] \setminus S}$ to $\mathcal{A}$.
   - Adversary $\mathcal{A}$ issues a query for key shares for a set $S' \subseteq [N]$ of parties, such that $S \cup S'$ is an invalid set of parties.
   - Challenger provides the set of key shares $\{\mathsf{sk}_i\}_{i \in S'}$ to $\mathcal{A}$ and sets $S = S \cup S'$
4. Adversary $\mathcal{A}$ continues to issue a polynomial number of adaptive queries of the form (m, i), where $i \in [N] \setminus S$ to get partial signature σ_i on m. For each query, the challenger computes σ_i as $\mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m)$ and provides it to $\mathcal{A}$.
5. At the end of the experiment, adversary $\mathcal{A}$ outputs a forgery (m^*, σ^*). The experiment outputs 1 if $\mathsf{TS.Verify}(\mathsf{vk}, m^*, \sigma^*) = \mathit{accept}$ and m^* was not queried previously as a signing query.

Construction. In this section we provide our construction for fully adaptive threshold signatures in the standard model but with pre-processing, where signers must be provided correlated randomness of length proportional to the number of signing queries. We emphasize that this correlated randomness is independent of messages, and that this processing can be done in an offline phase before any messages are made available. The informed reader may notice similarities with the "MPC with Preprocessing" model (please see [35] and references therein).

The construction in standard model differs from the one in ROM in the way the random values $r_{i,j}$ are chosen. In this construction, $r_{i,j}$ is sampled directly for all possible signing query j in such a way that for each $j, \sum_{i=1}^{N} r_{i,j} = \mathbf{0}$. This helps to achieve full adaptivity because when key shares of one or more parties in $S' \subseteq [N]$ are revealed to the adversary, it does not fix $r_{i,j}$ values for $i \in [N] \setminus S'$. This gives the challenger the flexibility to simulate partial signature for uncorrupted parties and adjust their randomness $r_{i,j}$ later. Let Q be the maximum number of signing queries, then the signature scheme is defined as follows. For a stateless scheme, we use a collision resistant hash function H which maps a message to an index in $[Q]$.

Let $H : \{0,1\}^* \to [Q]$ be a collision resistant hash function and $H_1 : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^r$ be a hash function modelled as random oracle.

$\mathsf{TS.KeyGen}(1^\lambda)$: Upon input the security parameter λ, do the following:

1. For $j = 1$ to Q, generate random values $\{r_{ij}\}_{i=1}^{N}$ such that $\sum_{i=1}^{N} r_{ij} = 0$.
2. Generate the verification and signing keys for the signature scheme $(\mathsf{Sig.vk}, \mathsf{Sig.sk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$.
3. Generate the public and the secret keys for the HE scheme $(\mathsf{HE.PK}, \mathsf{HE.SK}) \leftarrow \mathsf{HE.KeyGen}(1^\lambda)$ and generate N shares of $\mathsf{HE.SK}$ as $(\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_N) = \mathsf{Share}(\mathsf{HE.SK})$ such that $\sum_{i=1}^{N} \mathsf{sk}_i = \mathsf{HE.SK}$.
4. Compute an HE encryption of $\mathsf{Sig.sk}$ as $\mathsf{CT}_{\mathsf{Sig.sk}} \leftarrow \mathsf{HE.Enc}(\mathsf{HE.PK}, \mathsf{Sig.sk})$.
5. For each P_i, randomly choose a tag $\tau_i \in \{0,1\}^*$ and a hash key $\mathsf{hkey}_i \leftarrow \{0,1\}^\lambda$. Generate HS public parameters $\mathsf{HS.pp} \leftarrow \mathsf{HS.PrmsGen}(1^\lambda, 1^n)$, and the public and signing keys as $(\mathsf{HS.pk}, \mathsf{HS.sk}) \leftarrow \mathsf{HS.KeyGen}(1^\lambda, \mathsf{HS.pp})$. Here, n is the bit length of input to PartSign circuit which depends on $(\mathsf{HE.SK}, r_{ij}, \mathsf{hkey}_i)$.
6. Compute $(\pi_{\tau_i}, \pi_i) = \mathsf{HS.Sign}(\mathsf{HS.sk}, (\mathsf{sk}_i, \{r_{ij}\}_{j\in[Q]}, \mathsf{hkey}_i), \tau_i)$ for each $i \in [N]$.
7. Output $\mathsf{TSig.pp} = (\mathsf{HE.PK}, \mathsf{HS.pp}, \mathsf{HS.pk}, \mathsf{CT}_{\mathsf{Sig.sk}}, \{\tau_i, \pi_{\tau_i}\}_{i=1}^{N})$, $\mathsf{TSig.vk} = \mathsf{Sig.vk}$, $\mathsf{TSig.sk} = \{\mathsf{TSig.sk}_i = (\mathsf{sk}_i, \{r_{ij}\}_{j\in[Q]}, \mathsf{hkey}_i, \pi_i)\}_{i=1}^{N}$.

$\mathsf{TS.PartSign}(\mathsf{TSig.pp}, \mathsf{TSig.sk}_i, M)$: Upon input the public parameters TSig.pp, a partial signing key $\mathsf{TSig.sk}_i = (\mathsf{sk}_i, \{r_{ij}\}_{j\in[Q]}, \mathsf{hkey}_i, \pi_i)$ and a message M, do the following:

1. Compute $j = H(M)$, $u_i = H_1(\mathsf{hkey}_i, M)$ and sample $e_i' \leftarrow \mathcal{D}_s(u_i)$.
2. Let $\mathcal{C}_M$ be the Sig.Sign circuit with message M being hardcoded. Compute HE encryption of signature σ_M as $\mathsf{CT}_{\sigma_M} \leftarrow \mathsf{HE.Eval}(\mathsf{HE.PK}, \mathcal{C}_M, \mathsf{CT}_{\mathsf{Sig.sk}})$.
3. Compute $\sigma_{i,M} = \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{ij} + e_i'$.
4. Let $\mathcal{C}_{\mathsf{PS}}$ be a circuit with CT_{σ_M} being hardcoded and which takes as input the key share $\mathsf{TSig.sk}_i$ to compute $\mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{ij} + e_i'$.
   - Compute $\pi_{i,M}^* = \mathsf{HS.SignEval}(\mathsf{HS.pp}, \mathcal{C}_{\mathsf{PS}}, \pi_{\tau_i}, (\mathsf{sk}_i, \{r_{ij}\}_{j\in[Q]}, \mathsf{hkey}_i), \pi_i)$.

- Compute $\tilde{\pi}_{i,M} = \mathsf{HS.Hide}(\mathsf{HS.pk}, \sigma_{i,M}, \pi^*_{i,M})$.

5. Output $y_{i,M} = (\sigma_{i,M}, \tilde{\pi}_{i,M})$.

$\mathsf{TS.PartSignVerify}(\mathsf{TSig.pp}, M, y_{i,M})$: Upon input the public parameters $\mathsf{TSig.pp}$, message M, and a partial signature $y_{i,M}$, the verifier does the following.

1. Computes $\mathsf{CT}_{\sigma_M} = \mathsf{HE.Eval}(\mathsf{HE.PK}, \mathcal{C}_M, \mathsf{CT}_{\mathsf{Sig.sk}})$.
2. Defines circuit $\mathcal{C}_{\mathsf{PS}}$ as described before and computes $\alpha = \mathsf{HS.Process}(\mathsf{HS.pp}, \mathcal{C}_{\mathsf{PS}})$.
3. Parses $y_{i,M}$ as $(\sigma_{i,M}, \tilde{\pi}_{i,M})$ and outputs $\mathsf{HS.HVerify}(\mathsf{HS.pp}, \alpha, \sigma_{i,M}, \tau_i, (\pi_{\tau_i}, \tilde{\pi}_{i,M}))$.

$\mathsf{TS.Combine}(\mathsf{TSig.pp}, \{y_{i,M}\}_{i\in[N]})$: Upon input the public parameters $\mathsf{TSig.pp}$ and a set of partial signatures $\{y_{i,M}\}_{i\in[N]}$, parse $y_{i,M}$ as $(\sigma_{i,M}, \tilde{\pi}_{i,M})$ and output $\sigma_M = \mathsf{HE.decode}_1(\sum_{i=1}^N \sigma_{i,M})$.

$\mathsf{TS.Verify}(\mathsf{TSig.vk}, M, \sigma_M)$: Upon input a verification key $\mathsf{TSig.vk}$, a message M and signature σ_M, output $\mathsf{Sig.Verify}(\mathsf{TSig.vk}, M, \sigma_M)$.

Correctness. From the correctness of $\mathsf{HE.Eval}$, $\mathsf{CT}_{\sigma_M} = \mathsf{HE.Eval}(\mathsf{HE.PK}, \mathcal{C}_M, \mathsf{CT}_{\mathsf{Sig.sk}})$ is the encryption of $\mathcal{C}_M(\mathsf{Sig.sk}) = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, M) = \sigma_M$, which decrypts with the HE secret key $\mathsf{HE.SK}$. So, $\mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_{\sigma_M}) = \sigma_M \lfloor q/2 \rceil + e$. The signature computed by the $\mathsf{TS.Combine}$ algorithm is

$$\begin{aligned}
\mathsf{HE.decode}_1(\sum_{i=1}^N \sigma_{i,M}) &= \mathsf{HE.decode}_1(\sum_{i=1}^N \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + \sum_{i=1}^N r_{ij} + \sum_{i=1}^N e'_i) \\
&= \mathsf{HE.decode}_1(\mathsf{HE.decode}_0(\sum_{i=1}^N \mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + 0 + \sum_{i=1}^N e'_i) \\
&= \mathsf{HE.decode}_1(\mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_{\sigma_M}) + \sum_{i=1}^N e'_i) \\
&= \mathsf{HE.decode}_1(\sigma_M \lfloor q/2 \rceil + e + \sum_{i=1}^N e'_i) = \sigma_M.
\end{aligned}$$

Theorem C.1. *Assume the signature scheme* Sig *satisfies unforgeability,* HE *is a CPA secure homomorphic encryption scheme (Definition 2.18),* HS *is context hiding homomorphic signature scheme (Definition A.4) and* Share *satisfies privacy (Definition 2.26). Then the above construction satisfies adaptive unforgeability if the flooding error is of the size* $\mathrm{poly}(\lambda)\sqrt{Q}$*, where* Q *is the number of signing queries.*

Proof. The security of the construction can be argued using the following hybrids:

Hybrid_0: The real world.

Hybrid_1 : Same as Hybrid_0, except that now instead of using $\mathsf{HS.SigEval}$ algorithm to compute the homomorphic signature $\tilde{\pi}_{i,M}$ on $\sigma_{i,M}$ in $y_{i,M} = (\sigma_{i,M}, \tilde{\pi}_{i,M})$, the challenger simulates $\tilde{\pi}_{i,M}$ as $\tilde{\pi}_{i,M} = \mathsf{Sim}(\mathsf{HS.sk}, \alpha, \sigma_{i,M}, \tau_i, \pi_{\tau_i})$, where $\alpha = \mathsf{HS.Process}(\mathsf{HS.pp}, \mathcal{C}_{\mathsf{PS}})$.

Hybrid_2: Same as Hybrid_1, except that now the randomness u_i used in sampling flooding noise in $\mathsf{PartSign}$ algorithm is chosen uniformly randomly from $\{0,1\}^r$ and then H_1 is programmed as $H_1(\mathsf{hkey}_i, M) = u_i$. For random oracle queries by the adversary on an input x, the challenger first checks if $H_1(x)$ is already set. If so, then returns it else chooses a value uniformly randomly from $\{0,1\}^r$ and saves and returns it.

Hybrid_3: Same as Hybrid_2 except that now the r_{ij} values are set in a different order. In particular, for each $i \in [N]$, let $PreQ_i$ be the set of messages for which partial signatures are computed before corrupting P_i. Then, for each $j \in \{H(M) : M \in PreQ_i\}$, r_{ij} is set in reverse order, i.e the challenger first computes the partial signature $\sigma_{i,M}$ and then sets the value for $r_{i,H(M)}$ accordingly as follows. For any signing query on message M, let $S_M \subseteq [N]$ be the set of parties corrupted by the adversary so far. Then to compute $\sigma_{i,M}$, the challenger does the following.

1. If M was queried before then returns $\sigma'_{iM} + e'_i$ for each $i \in [N] \setminus S$, where $\sigma'_{i,M}$ is the same value as in the earlier response, but e'_i is sampled afresh. Else,
2. Computes $\mathsf{CT}_{\sigma_M} = \mathsf{HE.Eval}(\mathsf{HE.PK}, \mathcal{C}_M, \mathsf{Sig.sk})$ and
3. For each $i \in S_M$, computes $\sigma'_{i,M} = \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{i,H(M)}$ and sets $\sigma_{i,M} = \sigma'_{i,M} + e'_i$.
4. For $i \in [N] \setminus S_M$,
   - Divide $\mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_{\sigma_M}) - \sum_{k \in S_M} \sigma'_{k,M}$ into $N - |S_M|$ shares $\{s_{i,M}\}_{i \in [N] \setminus S_M}$.
   - Set $\sigma'_{i,M} = s_{i,M}$ and $\sigma_{i,M} = \sigma'_{i,M} + e'_i$
5. Return $\{\sigma_{i,M}\}_{i \in [N] \setminus S}$ to the adversary.

When the adversary asks for key share for some $i \in [N]$, the challenger does the following.

1. For each $M \in PreQ_i$, computes $r_{i,H(M)} = s_{i,M} - \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M})$.
2. For $j \in [Q] \setminus \{H(M) : M \in PreQ_i\}$ chooses $r_{i,j}$ randomly.

Hybrid_4: Same as Hybrid_3, except the following changes:

To answer signing query on any message M, for each $i \in S_M$, the challenger computes the response as in Hybrid_3, but for $i \in [N] \setminus S_M$ the challenger simulates the response differently from Hybrid_3. In particular, instead of sharing $\mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_{\sigma_M}) - \sum_{i \in S_M} \sigma'_{i,M}$ among $N - |S_M|$ uncorrupted parties, the challenger now shares $\sigma_M \lfloor q/2 \rceil - \sum_{i \in S_M} \sigma'_{i,M}$ among the uncorrupted parties as described below.

1. Computes $\mathsf{CT}_{\sigma_M} = \mathsf{HE.Eval}(\mathsf{HE.PK}, \mathcal{C}_M, \mathsf{CT}_{\mathsf{Sig.sk}})$ and
   $\sigma_M = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, M)$.
2. For each $i \in S_M$, compute $j = H(M)$, $\sigma'_{i,M} = \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{ij}$, and $\sigma_{i,M} = \sigma'_{i,M} + e'_i$, where $e'_i \leftarrow \mathcal{D}_s$.

3. For each $i \in [N] \setminus S_M$, do the following: divide $\sigma_M \lfloor q/2 \rceil - \sum_{k \in S_M} (\sigma'_{k,M})$ into $N - |S_M|$ shares to get $\{s_{i,M}\}_{i \in [N] \setminus S_M}$. Set $\sigma_{i,M} = s_{i,M} + e'_i$.
4. Return $\sigma_{i,M}$ for $i \in [N] \setminus S_M$.

Response to key share queries is given in the same way as in the previous hybrid.

Hybrid_5: Same as Hybrid_4, except that now the challenger shares zero vector $\{\mathsf{sk}_i\}_{i=1}^N \leftarrow \mathsf{Share}(\mathbf{0})$ instead of $\mathsf{HE.SK}$ to generate key shares sk_i in $\mathsf{TSig.sk}_i$.

Hybrid_6: Same as Hybrid_5, except that now $\mathsf{CT}_{\mathsf{Sig.sk}}$ in public parameters, $\mathsf{TSig.pp}$ is replaced by $\mathsf{CT}_{\mathbf{0}}$, i.e., an HE encryption of zero vector.

Indistinguishability of Hybrids. Next, we show that consecutive hybrids are indistinguishable.

Proof for indistinguishability between Hybrid_0, Hybrid_1 and Hybrid_2 is the same as for partially adaptive construction (Section 5.1.1).

Claim C.2. Hybrid_2 *and* Hybrid_3 *are statistically indistinguishable*

Proof. Let Query be the set of all messages for which the adversary issued signing queries during the experiment. Let $T = \{H(M)\}_{M \in \mathsf{Query}}$. Then Hybrid_2 and Hybrid_3 differ only in the way $\{r_{i,j}\}_{i \in [N], j \in T}$ are set. Consider any $j \in T$. Let $j = H(M)$ for some $M \in \mathsf{Query}$. Then in Hybrid_3, $\{r_{i,j}\}_{i \in S_M}$ are randomly chosen. For $i \in [N] \setminus S_M$, $r_{i,j} = s_{i,j} - \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M})$, where $\{s_{i,j}\}_{i \in [N] \setminus S_M}$ are obtained by randomly sharing $\mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_{\sigma_M}) - \sum_{k \in S_M} (\mathsf{HE.decode}_0(\mathsf{sk}_k, \mathsf{CT}_{\sigma_M}) + r_{k,j})$ into $N - |S_M|$ shares. Thus,

$$
\begin{aligned}
\mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_{\sigma_M}) &= \sum_{k \in S_M} \mathsf{HE.decode}_0(\mathsf{sk}_k, \mathsf{CT}_{\sigma_M}) + \sum_{k \in S_M} r_{k,j} + \sum_{i \in [N] \setminus S_M} s_{i,j} \\
&= \sum_{k \in S_M} \mathsf{HE.decode}_0(\mathsf{sk}_k, \mathsf{CT}_{\sigma_M}) + \sum_{k \in S_M} r_{k,j} \\
&+ \sum_{i \in [N] \setminus S_M} \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + \sum_{i \in [N] \setminus S_M} r_{i,j} \\
&= \sum_{i \in [N]} \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + \sum_{i \in [N]} r_{i,j} \\
&= \mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_{\sigma_M}) + \sum_{i \in [N]} r_{i,j}
\end{aligned}
$$

This implies $\sum_{i \in [N]} r_{i,j} = 0$, and since $\{s_{i,j}\}_{i \in [N] \setminus S_M}$ are random shares, we can conclude that $\{r_{i,j}\}_{i \in [N]}$ are random shares of 0, which is same as Hybrid_2. □

Claim C.3. *Assume that the flooding error is of the order* $\mathrm{poly}(\lambda) \cdot \sqrt{Q}$. *If there is an adversary who can win the* $\mathsf{Expt}_{\mathcal{A},\mathsf{TS},FA-uf}(1^\lambda)$ *game in* Hybrid_3 *with probability* ϵ, *then its probability of winning in* Hybrid_4 *is at least* $\epsilon^2/2$.

Proof. Let the adversary issues Q signing queries. Wlog let the corrupted parties be $\{P_2, \ldots, P_N\}$. Then, the two hybrids differ only in the error term in $\sigma_{1,M}$, as shown below.

Consider any signing query for a message M. Let S_M be the set of corrupted parties so far and let $H(M) = j$ and $e'_1 \leftarrow \mathcal{D}_s$. Then,

In Hybrid_4, we have:

$$\begin{aligned}
\sigma_1 &= s_{1,M} + e'_1 \\
&= \sigma_M \lfloor q/2 \rceil - \sum_{i \in S_M} \sigma'_{i,M} - \sum_{\substack{i \in [N] \setminus S_M \\ i \neq 1}} s_{i,M} + e'_1 \\
&= \sigma_M \lfloor q/2 \rceil - \sum_{i \in S_M} (\mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{ij}) \\
&\quad - \sum_{\substack{i \in [N] \setminus S_M \\ i \neq 1}} (\mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{ij}) + e'_1 \\
&= \sigma_M \lfloor q/2 \rceil - \sum_{i \in [N]} \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) - \sum_{i \in [N] \setminus \{1\}} r_{ij} + \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_{\sigma_M}) + e'_1 \\
&= \sigma_M \lfloor q/2 \rceil - \mathsf{HE.decode}_0(\sum_{i \in [N]} \mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) - \sum_{i \in [N] \setminus \{1\}} r_{ij} + \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_{\sigma_M}) + e'_1 \\
&= \sigma_M \lfloor q/2 \rceil - \mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_{\sigma_M}) - \sum_{i \in [N] \setminus \{1\}} r_{ij} + \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_{\sigma_M}) + e'_1 \\
&= \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_{\sigma_M}) + r_{1j} + e + e'_1
\end{aligned}$$

In Hybrid_3,

$$\begin{aligned}
\sigma_{1,M} &= s_{1,M} + e'_1 \\
&= \mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_{\sigma_M}) - \sum_{i \in S_M} \sigma'_{i,M} - \sum_{\substack{i \in [N] \setminus S_M \\ i \neq 1}} s_{i,M} + e'_1 \\
&= \mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_{\sigma_M}) - \sum_{i \in S_M} (\mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{ij}) \\
&\quad - \sum_{\substack{i \in [N] \setminus S_M \\ i \neq 1}} (\mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{ij}) + e'_1 \\
&= \mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_{\sigma_M}) - \sum_{i \in [N]} \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) - \sum_{i \in [N] \setminus \{1\}} r_{ij} \\
&\quad + \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_{\sigma_M}) + e'_1 \\
&= \mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_{\sigma_M}) - \mathsf{HE.decode}_0(\sum_{i \in [N]} \mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) - \sum_{i \in [N] \setminus \{1\}} r_{ij} \\
&\quad + \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_{\sigma_M}) + e'_1 \\
&= \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_{\sigma_M}) + e'_1 - \sum_{i \in [N] \setminus \{1\}} r_{ij} \\
&= \mathsf{HE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_{\sigma_M}) + r_{1j} + e'_1
\end{aligned}$$

In the third step, $\sigma'_{i,M}$ is computed as $\mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{ij}$, while $s_{i,M} = \mathsf{HE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{ij}$ because of the setting of r_{ij} such that the equality holds. In the last step we replace

$-\sum_{i\in[N]\setminus\{1\}} r_{ij}$ by r_{1j} because $\sum_{i\in[N]} r_{ij} = 0$. However note that r_{1j} is never actually set since $\mathsf{TSig.sk}_1$ is never queried for.

Thus, the difference in the two hybrids is in the error terms in σ_1. In Hybrid_3, the error is e'_1, while in Hybrid_4, it is $e'_1 + e$. This is the same case as in Section 4. Hence, the claim can be proved in the same way as Claim 4.4.

□

Claim C.4. *Assuming the privacy property of secret sharing scheme,* Share, Hybrid_4 *and* Hybrid_5 *are indistinguishable.*

Proof. The only difference between Hybrid_4 and Hybrid_5 is in the way the key shares $\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_N$ are generated. In Hybrid_4 $(\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_N \leftarrow \mathsf{Share}(\mathsf{HE.SK})$, while in Hybrid_5, $(\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{Share}(\mathbf{0})$. Hence as long as the adversary is given the key shares for an invalid set of parties, the two distributions are indistinguishable due to the privacy property of secret sharing scheme Share. □

Claim C.5. *Assume the* HE *scheme is semantically secure. Then* Hybrid_5 *and* Hybrid_6 *are indistinguishable.*

Proof. Let $\mathcal{A}$ be an adversary who can distinguish Hybrid_5 and Hybrid_6 with non-negligible probability ϵ. Then we construct an adversary $\mathcal{B}$ against the HE scheme. The proof is similar to the one in Claim 5.7. □

Claim C.6. *If the underlying signature scheme* Sig *is unforgeable, then the unforgeability game in* Hybrid_6 *cannot be won.*

Proof. Let $\mathcal{A}$ be an adversary that wins the experiment $\mathsf{Expt}_{\mathcal{A},\mathsf{TS},FA-uf}(1^\lambda)$ in Hybrid_6. We can construct an adversary $\mathcal{B}$ against the signature scheme in the same way as described in the proof of Claim 5.8

Above arguments prove the fully adaptive unforgeability of our threshold signature scheme. □

□

Robustness.

Claim C.7. *If* HS *is multi data secure homomorphic signature, then the construction of* TS *satisfies robustness.*

Proof. The proof is same as in Claim 4.8. □

D Threshold Signatures for t-out-of-N access structures

In this section we give a general construction for t-*out-of*-N access structure using $\{0,1\}$-LSSS.

The construction uses following building blocks:

1. A special homomorphic encryption scheme $\mathsf{HE} = (\mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Dec}, \mathsf{HE.Eval})$. Let B be the error bound of the HE scheme.
2. A UF-CMA signature scheme $\mathsf{Sig} = (\mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$.
3. A t out of N $\{0,1\}$-LSSS, Share.

Construction

$\mathsf{TS.KeyGen}(1^\lambda, t)$: Upon input the security parameter λ and the threshold t do the following:

1. Generate the verification and signing keys for the signature scheme $(\mathsf{Sig.vk}, \mathsf{Sig.sk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$.
2. Generate the keys for the HE scheme $(\mathsf{HE.PK}, \mathsf{HE.SK}) \leftarrow \mathsf{HE.KeyGen}(1^\lambda)$ and compute an HE encryption of the signing key as $\mathsf{CT}_{\mathsf{Sig.sk}} = \mathsf{HE.Enc}(\mathsf{HE.PK}, \mathsf{Sig.sk})$.
3. Share the HE secret key as: $\{\mathsf{TSig.sk}_i\}_{i=1}^N \leftarrow \mathsf{Share}(\mathsf{HE.SK})$. Note that for $\{0,1\}$-LSSS, each $\mathsf{TSig.sk}_i$ can be a set of more than one secret shares.
   Notation: Let $\mathbf{M}$ be the share matrix of dimension $\ell \times N$. Then for $i \in [N]$, T_i refers to the partition of $[\ell]$ corresponding to party P_i and $\mathsf{TSig.sk}_i = \{\mathsf{sk}_j\}_{j \in T_i}$, where sk_j, is the jth (out of ℓ shares) share of $\mathsf{HE.SK}$.
4. Output $\mathsf{TSig.pp} = \{\mathsf{HE.PK}, \mathsf{CT}_{\mathsf{Sig.sk}}\}$, $\mathsf{TSig.vk} = \mathsf{Sig.vk}$, $\mathsf{TSig.sk} = \{\mathsf{TSig.sk}_i\}_{i=1}^N$.

$\mathsf{TS.PartSign}(\mathsf{TSig.pp}, \mathsf{TSig.sk}_i, m)$: Upon input the public parameters $\mathsf{TSig.pp}$, the partial signing key $\mathsf{TSig.sk}_i$ and a message m, do the following:

1. Let $\mathcal{C}_m$ be the signing circuit, with message m being hardcoded. Homomorphically compute an HE encryption of the signature as $\mathsf{CT}_\sigma = \mathsf{HE.Eval}(\mathsf{HE.PK}, \mathcal{C}_m, \mathsf{CT}_{\mathsf{Sig.sk}})$.
2. Output $\sigma_i = \{\hat{\sigma}_j\}_{j \in T_i}$, where $\hat{\sigma}_j = \mathsf{HE.decode}_0(\mathsf{sk}_j, \mathsf{CT}_\sigma) + e'_j$, where, $e'_j \leftarrow \mathcal{D}_s$.

$\mathsf{TS.Combine}(\mathsf{TSig.pp}, \{\sigma_i\}_{i \in S})$: Upon input the public parameters $\mathsf{TSig.pp}$ and a set of partial signatures $\{\sigma_i\}_{i \in S}$, where $S \subseteq [N]$, the $\mathsf{Combine}$ algorithm first checks if $|S| \geq t$ - if not then output $\perp$, else computes a minimum valid share set $T \subseteq \bigcup_{i \in S} T_i$ and outputs

$$\sigma_m = \mathsf{HE.decode}_1(\sum_{j \in T} \hat{\sigma}_j).$$

$\mathsf{TS.Verify}(\mathsf{TSig.vk}, m, \sigma_m)$: Upon input the verification key $\mathsf{TSig.vk}$, a message m and signature σ_m, output $\mathsf{Sig.Verify}(\mathsf{TSig.vk}, m, \sigma_m)$.

Correctness. From the correctness of $\mathsf{HE.Eval}$ algorithm, $\mathsf{CT}_\sigma = \mathsf{HE.Eval}(\mathsf{HE.PK}, \mathcal{C}_m, \mathsf{CT}_{\mathsf{Sig.sk}})$ is the encryption of $\mathcal{C}_m(\mathsf{Sig.sk}) = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, m) = \sigma_m$, which decrypts with the HE secret key $\mathsf{HE.SK}$. So, $\mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_\sigma) = \sigma_m \lfloor q/2 \rceil + e$, where e is the error in CT_σ. The signature

computed by the TS.Combine algorithm is

$$\begin{aligned}
\mathsf{HE.decode}_1(\sum_{j\in T}\hat{\sigma}_j) &= \mathsf{HE.decode}_1(\sum_{j\in T}\mathsf{HE.decode}_0(\mathsf{sk}_j,\mathsf{CT}_\sigma)+\sum_{j\in T}e'_j)\\
&= \mathsf{HE.decode}_1(\mathsf{HE.decode}_0(\sum_{j\in T}\mathsf{sk}_j,\mathsf{CT}_\sigma)+\sum_{j\in T}e'_j)\\
&\quad \text{(from linearity of } \mathsf{HE.decode}_0)\\
&= \mathsf{HE.decode}_1(\mathsf{HE.decode}_0(\mathsf{HE.SK},\mathsf{CT}_\sigma)+\sum_{j\in T}e'_j)\\
&\quad \text{(from correctness of Share algorithm.)}\\
&= \mathsf{HE.decode}_1(\sigma_m\lfloor q/2\rceil+e+\sum_{j\in T}e'_j)=\sigma_m.
\end{aligned}$$

Unforgeability

Theorem D.1. *Assume* HE *is a homomorphic encryption that satisfies security (Definition 2.18),* Share *is a* $\{0,1\}$*-*LSSS *t-out-of-N secret sharing scheme that satisfies privacy(Definition 2.26) and* Sig *is a signature scheme that satisfies unforgeability then the above construction of threshold signature satisfies unforgeability.*

Proof. We prove the theorem using following hybrids.

Hybrid_0 : Is the real world; i.e. the challenger generates the signing key shares, $\{\mathsf{TSig.sk}_i\}_{i\in[N]}$, the verification key TSig.vk, and the public parameters TSig.pp and sends TSig.vk and TSig.pp to the adversary, $\mathcal{A}$. $\mathcal{A}$ outputs a maximal invalid party set S^* (i.e. $|S^*|=t-1$) tofor which the challenger returns the corresponding key shares $\{\mathsf{TSig.sk}_i\}_{i\in S^*}$. In response to (partial) signing query (m,i) for any $i\in[N]\setminus S^*$, the challenger computes it as per the scheme.

Hybrid_1: Same as Hybrid_0 except that now the signing queries are answered differently.

1. On receiving the (invalid) party set S^* from $\mathcal{A}$, the challenger commits to a maximal invalid share set T^* which contains $\bigcup_{i\in S^*}T_i$.
2. To respond to any partial signing query (m,i) on message m for party P_i ($i\in[N]\setminus S^*$), the challenger computes $\hat{\sigma}_j$ for $j\in T_i$ as follows:
   - If $j\in T_i\cap T^*$, then computes $\hat{\sigma}_j=\mathsf{HE.decode}_0(\mathsf{sk}_j,\mathsf{CT}_\sigma)+e'_j$, i.e. as in the real world.
   - If $j\notin T_i\cap T^*$, then the challenger does the following: computes a minimal valid share set $T\subseteq T^*\cup\{j\}$. (Note that such a set always exists and contains j because T^* is a maximal invalid share set and $j\notin T^*$, hence $T^*\cup\{j\}$ is a valid share set.) Computes $\sigma_m=\mathsf{Sig.Sign}(\mathsf{Sig.sk},m)$ and $\{\hat{\sigma}_{j'}=\mathsf{HE.decode}_0(\mathsf{sk}_{j'},\mathsf{CT}_\sigma)+e'_{j'})\}_{j'\in T\setminus\{j\}}$ Then it computes $\hat{\sigma}_j$ as

$$\hat{\sigma}_j=\lfloor q/2\rceil\cdot\sigma_m-\sum_{j'\in T\setminus\{j\}}\hat{\sigma}_{j'}+e'_j.$$

Hybrid_2: Same as Hybrid_1, except that instead of sharing $\mathsf{HE.SK}$ the challenger now computes key shares as $(\mathsf{TSig.sk}_1, \ldots, \mathsf{TSig.sk}_N) \leftarrow \mathsf{Share}(\mathbf{0}, t)$.

Hybrid_3: Same as Hybrid_2, except that $\mathsf{CT}_{\mathsf{Sig.sk}}$ in $\mathsf{TSig.pp}$ is replaced by $\mathsf{CT}_{\mathbf{0}}$, i.e. ciphertext of $\mathbf{0}$ using same encryption key $\mathsf{HE.PK}$.

Indistinguishability of Hybrids. Next, we show that consecutive hybrids are indistinguishable.

Claim D.2. *If the flooding error is of the size* $\mathrm{poly}(\lambda)\sqrt{Q}$*, then if there is an adversary who can win the game in* Hybrid_0 *with probability* ϵ*, then its probability of winning in* Hybrid_1 *is at least* $\epsilon^2/2$.

Proof. Let the number of signing queries that an adversary can make be bounded by Q.

The two hybrids differ only in the error term in partial signatures returned by the challenger. Let the adversary issues partial signing query for (m, i). Let T_i, S^* and T^* be as defined in Hybrid_1. Then for $j \in T_i \cap T^*$, $\hat{\sigma}_j$ is computed in the same way in both the hybrids. The difference is in the error term in $\hat{\sigma}_j$ for $j \in T_i \setminus T^*$.

Let us focus on one such j. Let $e'_j \leftarrow \mathcal{D}_s$ and T be a minimal valid share set contained in $T^* \cup \{j\}$.

In Hybrid_0, we have:

$$\hat{\sigma}_j = \mathsf{HE.decode}_0(\mathsf{sk}_j, \mathsf{CT}_\sigma) + e'_j.$$

In Hybrid_1, we have:

$$\begin{aligned}
\hat{\sigma}_j &= \sigma_m.\lfloor q/2 \rceil - \sum_{j' \in T \setminus \{j\}} \mathsf{HE.decode}_0(\mathsf{sk}_{j'}, \mathsf{CT}_\sigma) + e'_j \\
&= \sigma_m.\lfloor q/2 \rceil - \sum_{j' \in T} \mathsf{HE.decode}_0(\mathsf{sk}_{j'}, \mathsf{CT}_\sigma) + \mathsf{HE.decode}_0(\mathsf{sk}_j, \mathsf{CT}_\sigma) + e'_j \\
&= \sigma_m.\lfloor q/2 \rceil - \mathsf{HE.decode}_0(\sum_{j' \in T} \mathsf{sk}_{j'}, \mathsf{CT}_\sigma) + \mathsf{HE.decode}_0(\mathsf{sk}_j, \mathsf{CT}_\sigma) + e'_j \\
&= \sigma_m.\lfloor q/2 \rceil - \mathsf{HE.decode}_0(\mathsf{HE.SK}, \mathsf{CT}_\sigma) + \mathsf{HE.decode}_0(\mathsf{sk}_j, \mathsf{CT}_\sigma) + e'_j \\
&= \sigma_m.\lfloor q/2 \rceil - \sigma_m.\lfloor q/2 \rceil + e + \mathsf{HE.decode}_0(\mathsf{sk}_j, \mathsf{CT}_\sigma) + e'_j \\
&= \mathsf{HE.decode}_0(\mathsf{sk}_j, \mathsf{CT}_\sigma) + (e'_j + e)
\end{aligned}$$

Thus, the difference in the two hybrids is in the error terms in $\hat{\sigma}_j$ for $j \in T_i \setminus T^*$. In Hybrid_0, the error is e'_j, while in Hybrid_1, it is $e'_j + e$. The proof is similar to the Rényi Divergence based proof given for claim 4.4, with few modifications made for t-out-of-N access structure. Since e'_j is sampled from a discrete Gaussian with std. deviation s, we consider the distributions $\mathcal{D}_{s,0}$ and $\mathcal{D}_{s,|e|}$ for each $j \in T_i \setminus T^*$. For simplicity, we refer to the error distribution in Hybrid_0 as D_0 and in Hybrid_1 as D_1. Then $D_0 = \mathcal{D}_{\mathbb{Z}^{n_i}, s, \mathbf{0}}$ and $D_1 = \mathcal{D}_{\mathbb{Z}^{n_i}, s, \mathbf{e}}$, where $\mathbf{e} = (e, \ldots, e$ ($|T_i \setminus T^*|$ times)) and $n_i = |T_i \setminus T^*|$. To begin with, consider an adversary that makes a single signing query. In this case, let E represent the event that the adversary wins the game. Then, we assumed that

$$D_0(E) = \epsilon.$$

From the probability preservation property (Lemma 2.13), we have

$$D_1(E) \geq \frac{D_0(E)^{\frac{a}{a-1}}}{R_a(D_0\|D_1)}, \text{ for } a \in (1,\infty)$$

From Lemma 2.14,

$$\begin{aligned} R_a(D_0\|D_1) &= \exp(a\pi\frac{\|\mathbf{e}\|^2}{s^2}) \\ &= \exp(a\pi\frac{e^2\cdot|T_i\setminus T^*|}{s^2}) \end{aligned}$$

In general, let the adversary adaptively issues Q signing queries as $(m_1,i_1),\ldots,(m_Q,i_Q)$. Then, $D_0=\mathcal{D}_{\mathbb{Z}^d,s,\mathbf{0}}$ and $D_1=\mathcal{D}_{\mathbb{Z}^d,s,\mathbf{e}}$, where

$$\mathbf{e} = \underbrace{(e_{i_1},\ldots,e_{i_1})}_{|T_{i_1}\setminus T^*|\text{ times}},\ldots,\underbrace{(e_{i_Q},\ldots,e_{i_Q})}_{|T_{i_Q}\setminus T^*|\text{ times}},$$

where $d=\sum_{j\in Q} n_{i_j}$ and e_{i_j} is the error in $\mathsf{CT}_{\sigma_{M_j}}$. Errors $e_{i_1},\ldots,e_{i_Q}$ can be mutually dependent and each $e_{i_j}\leq B_{eval}$ for $1\leq j\leq Q$. Thus, for Q queries,

$$\begin{aligned} R_a(D_0\|D_1) &= \exp(a\pi\frac{\|\mathbf{e}\|^2}{s^2}) \\ &= \exp(a\pi\frac{\sum_{j\in[Q]}|T_{i_j}\setminus T^*|\cdot e_{i_j}^2}{s^2}) \\ &\leq \exp(a\pi\frac{\sum_{j\in[Q]}\ell\cdot B_{eval}^2}{s^2}) \\ &= \exp(a\pi\frac{Q\cdot\ell\cdot B_{eval}^2}{s^2}) \end{aligned}$$

Setting $s=B_{eval}\cdot\sqrt{\ell\cdot Q\lambda}$, where ℓ is bounded by $\mathrm{poly}(N)$, we get

$$R_a(D_0\|D_1)\leq\exp(\frac{a\pi}{\lambda})$$

Therefore,

$$\begin{aligned} D_1(E) &\geq \frac{D_0(E)^{\frac{a}{a-1}}}{R_a(D_0\|D_1)} \\ &\geq D_0(E)^{\frac{a}{a-1}}\exp(-\frac{a\pi}{\lambda}) \end{aligned}$$

The claim is proved by taking $a=2$. Thus, if the probability of success in Hybrid_0 is non-negligible then it is non-negligible in Hybrid_1 as well. □

Claim D.3. *Assume that* Share *is secure sharing scheme, then* Hybrid_1 *and* Hybrid_2 *are indistinguishable.*

Proof. The two hybrids differ only in the generation of key shares. In Hybrid_1, $\{\mathsf{TSig.sk}_i\}_{i\in[N]}=\mathsf{Share}(\mathsf{HE.SK},t)$, while in Hybrid_2 $\{\mathsf{TSig.sk}_i\}_{i\in[N]}$ is computed as $\mathsf{Share}(\mathbf{0},t)$. Hence by the privacy property (Definition 2.26) of Share the two hybrids are identical since the key shares given to the adversary is only for an invalid set of participants. □

Claim D.4. *Assume the* HE *scheme is secure (Definition 2.18). Then* Hybrid_2 *and* Hybrid_3 *are indistinguishable.*

Proof. Let $\mathcal{A}$ be an adversary who can distinguish Hybrid_2 and Hybrid_3 with non-negligible probability ϵ. Then we construct an adversary $\mathcal{B}$ against the HE scheme as follows.

1. Adversary $\mathcal{B}$ receives HE.PK from the HE challenger.
2. It generates $(\mathsf{Sig.sk}, \mathsf{Sig.vk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$ and computes the key shares as $\{\mathsf{TSig.sk}_1, \ldots, \mathsf{TSig.sk}_N\} \leftarrow \mathsf{Share}($
3. It sends the challenge messages: $m_0 = \mathsf{Sig.sk}$ and $m_1 = \mathbf{0}$ to the HE challenger.
4. The HE challenger responds with a ciphertext CT_b.
5. Adversary $\mathcal{B}$ sets $\mathsf{TSig.pp} = \{\mathsf{HE.PK}, \mathsf{CT}_b\}$ and sends $\mathsf{Sig.vk}, \mathsf{TSig.pp}$ to $\mathcal{A}$.
6. Adversary $\mathcal{A}$ outputs a maximal invalid party set S^* for which $\mathcal{B}$ returns $\{\mathsf{TSig.sk}_i\}_{i \in S^*}$.
7. To answer PartSign query, (m, i), for any message m and participant P_i by $\mathcal{A}$, $\mathcal{B}$ computes $\mathsf{CT}_{\sigma_m} = \mathsf{HE.Eval}(\mathsf{HE.PK}, \mathcal{C}_m, \mathsf{CT}_b)$ and then computes σ_j in the way described in the hybrids.
8. Finally, if $\mathcal{A}$ outputs its guess as Hybrid_2, then $\mathcal{B}$ sends $b' = 0$, else $b' = 1$ to the HE challenger.

□

Claim D.5. *If the underlying signature scheme* Sig *is unforgeable, then the unforgeability game in* Hybrid_3 *cannot be won.*

Proof. Let $\mathcal{A}$ be an adversary that wins the unforgeability game in Hybrid_3. We can construct an adversary $\mathcal{B}$ against the signature scheme as follows:

1. On receiving verification key Sig.vk from Sig-challenger, $\mathcal{B}$ runs HE.KeyGen to get HE.PK, HE.SK; computes key shares $\{\mathsf{TSig.sk}_i\}_{i=1}^N \leftarrow \mathsf{Share}(\mathbf{0}, t)$ and $\mathsf{CT}_\mathbf{0}$ and sends $\{\mathsf{TSig.vk} = \mathsf{Sig.vk}, \mathsf{TSig.pp} = (\mathsf{HE.PK}, \mathsf{CT}_0)\}$ to $\mathcal{A}$.
2. $\mathcal{A}$ outputs a maximal invalid party set S^* to which $\mathcal{B}$ responds with $\{\mathsf{TSig.sk}_i\}_{i \in S^*}\}$.
3. To simulate PartSign query on message m for any participant P_i, $\mathcal{B}$ requires σ_m which it gets by issuing signing query on message m to Sig challenger and computes σ_i as described in the hybrid.
4. At the end of the experiment, let (m^*, σ^*) be a message-signature pair returned by $\mathcal{A}$, then $\mathcal{B}$ also returns the same pair to the Sig challenger.

Since $\mathcal{B}$ issues signing queries to Sig challenger on only those messages m, for which $\mathcal{A}$ issues signing queries, if (m^*, σ^*) is a valid forgery for $\mathcal{A}$, then it is a valid forgery for $\mathcal{B}$ as well. □

The arguments above prove the unforgeability of our threshold signature scheme if the underlying HE scheme, secret sharing scheme and signature scheme are secure. □

Robustness

To add robustness in the above scheme, we can use context hiding secure homomorphic signature in the same way as in Section 4. In particular, the KeyGen algorithm also includes in $\mathsf{TSig.sk}_i$ a HS signature of party P_i's key shares. To prove the honest evaluation of $\mathsf{PartSign}$ algorithm, the signer homomorphically computes a signature on $\sigma_{i,m}$ and gives it to the verifier. The unforgeability property of HS provides robustness and its context hiding property ensures that unforgeability of TS is maintained.