Round-Optimal Lattice-Based Threshold Signatures, Revisited

Shweta Agrawal*, Damien Stehlé**, and Anshu Yadav***

Abstract. Threshold signature schemes enable distribution of the signature issuing capability to multiple users, to mitigate the threat of signing key compromise. Though a classic primitive, these signatures have witnessed a surge of interest in recent times due to relevance to modern applications like blockchains and cryptocurrencies. In this work, we study round-optimal threshold signatures in the post-quantum regime and improve the only known lattice-based construction by Boneh *et al* [CRYPTO'18] as follows:

- *Efficiency.* We reduce the amount of noise flooding used in the construction from $2^{\Omega(\lambda)}$ down to $\sqrt{Q}$, where Q 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 the signature bit-lengths from $\widetilde{O}(\lambda^3)$ to $\widetilde{O}(\lambda)$, bringing them significantly closer to practice. Our improvement relies on a careful analysis using Rényi divergence rather than statistical distance in the security proof.
- *Instantiation.* The construction of Boneh *et al* requires a standard signature scheme to be evaluated homomorphically. To instantiate this, we provide a homomorphism-friendly variant of Lyubashevsky's signature [EUROCRYPT '12] which achieves low circuit depth by being "rejection-free" and uses an optimal, moderate noise flooding of $\sqrt{Q}$, matching the above.
- *Towards Adaptive Security.* The construction of Boneh *et al* satisfies only selective security, where all the corrupted parties must be announced before any signing query is made. We improve this in two ways: in the Random Oracle Model, 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 preprocessing phase.

1 Introduction

A threshold signature [23] distributes the signature issuing capacity among several users, so that a signature can be generated only if a sufficient number of users collaborate to sign a message. In more detail, each of N parties holds a partial signing key, and any set of parties at least as large as a given threshold $t \leq N$ can participate in a protocol to generate a signature. Security requires that a valid signature cannot be generated if fewer than t parties cooperate.

A central motivation for constructing threshold signatures is to decentralize the trust placed in the signing authority, thus reducing the risk of the signing key being compromised. While threshold signatures have been studied for a long time [43, 24, 14, 32, 30, 44, 25, 20, 15, 13, 37, 21, 31, 6], they have received renewed attention in recent years due to numerous applications in modern topics such as cryptocurrencies and blockchains. Most prior work has focused on creating distributed versions of ECDSA or Schnorr signatures [44, 30, 25, 14, 15] which are not quantum secure. From conjectured post-quantum assumptions such as those related to Euclidean lattices, much less is known, especially with optimal round complexity.

1.1 Prior Work

The thresholdisation of lattice-based signatures from the NIST post-quantum cryptography project has been investigated in [19] but the resulting candidates incur several rounds of communication. A threshold signature

* IIT Madras, shweta@cse.iitm.ac.in
** ENS de Lyon and Institut Universitaire de France, damien.stehle@ens-lyon.fr
*** IIT Madras, anshu.yadav06@gmail.com

restricted to $t = N$ was proposed in [22] but it also involves possibly many rounds, because of aborts. To the best of our knowledge, the only lattice-based, round-optimal threshold signature construction is by Boneh *et al* [8] (henceforth BGGJKRS), relying on the Learning With Errors problem (LWE). However, while this construction provided the first feasibility result for a long-standing open problem, it suffers from the following drawbacks:

1. *Noise Flooding and Impact on Parameters.* It makes use of the so-called "noise flooding" technique [33, 5, 36], 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 success probability $2^{-o(\lambda)}$ where λ is the security parameter, 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 LWE parametrisation. 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., [38]) 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 leads to $\widetilde{\Omega}(\lambda^3)$-bit signature sizes – prohibitively expensive in practice.
2. *Instantiating Underlying Signature.* It requires a standard signature scheme to be evaluated homomorphically. BGGJKRS do not suggest a candidate and existing lattice based signatures are not suitable – the GPV signature scheme [34] and its practical versions [27, 51, 28] seem ill-suited, as the signing algorithm is very sequential, and the required 1-dimensional Gaussian samples are obtained via algorithms based on rejection sampling (see, e.g., [39, 57]) that are costly to transform into circuits. The other candidate is Lyubashevsky's signature scheme [46, 47]. 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.
3. *Selective Security.* It only achieves a very restricted notion of *selective* security, where all the corrupted parties must be announced before any partial signing query is made. To obtain security in the more realistic *adaptive* setting, one option is to invoke complexity leveraging, which consists in guessing at the outset which parties will be corrupted. This is not only dissatisfying as a solution but also leads to a further degradation of the parameters.

1.2 Our Contributions

In this work, we improve the construction from [8] as follows:

- *Efficiency.* We decrease the noise flooding ratio from $2^{\Omega(\lambda)}$ down to $\sqrt{Q}$, where Q is the bound on the number of generated signatures. This gives a one-round threshold signature of bit-length growing as $\widetilde{O}(\lambda \log^2 Q)$, which is $\widetilde{O}(\lambda)$ for any polynomially bounded Q,[1] in contrast with $\widetilde{O}(\lambda^3)$ for the construction from [8]. These bit-lengths are obtained when relying on the ring variants of SIS and LWE [48, 50, 55, 49]. Additionally, we show that the amount of noise flooding used in this construction is *optimal*, by exhibiting an attack when a smaller noise flooding ratio is used.
- *Instantiation.* To instantiate the signature underlying BGGJKRS, we provide a homomorphism friendly variant of Lyubashevsky's signature [EUROCRYPT '12] which achieves low circuit depth. We remove the rejection sampling at the expense of adding moderate noise of size $\sqrt{Q}$, matching the above. Again, we show that this amount of flooding is optimal by demonstrating an attack when smaller flooding is used.

[1] For many applications, the bound Q is quite limited and can be considered to be a small polynomial in λ. For example, for applications pertaining to cryptocurrencies, the bound Q may capture 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 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 limited. `https://www.blockchain.com/charts/n-transactions`, `https://www.statista.com/statistics/647374/worldwide-blockchain-wallet-users/`

- *Selective versus Adaptive.* As discussed above, the construction BGGJKRS satisfies only selective security. We improve this in two ways: in the Random Oracle Model (ROM), in which a hash function is being modeled as a uniformly sampled function with the same domain and range, 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 where signers must be provided correlated randomness of length proportional to the number of signing queries. The informed reader may notice similarities with the "MPC with Preprocessing" model, please see [29] and references therein [2].

1.3 Technical Overview

Recap of BGGJKRS Threshold Signatures. The round-optimal threshold signatures provided by [8] are designed using a "universal thresholdizer" which enables the thresholdizing of a number of primitives. This thresholdizer is itself instantiated using a threshold version of "special" fully homomorphic encryption (FHE), which in turn can be constructed using the LWE assumption. In threshold fully homomorphic encryption (TFHE), the setup algorithm takes as input a threshold t and produces a set of decryption key shares $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$ for the parties such that every party can perform a partial decryption using its own decryption key and any t out of N partial decryptions can be combined into a complete decryption of the ciphertext in a single round.

In more detail, the TFHE construction of BGGJKRS leverages the fact that the decryption in LWE based FHE schemes [12, 10, 35] requires to compute an inner product of the ciphertext ct with the secret key sk, followed by a rounding operation. Since inner product is a linear operation, a natural approach to thresholdize FHE decryption is by applying a Shamir t-out-of-N secret sharing to sk. This will yield N keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$, which can be distributed to the N users. Now, to decrypt a ciphertext ct, each user can compute the inner product with its individual secret key sk_i as its partial decryption m_i. To combine any t partial decryptions into the final decryption, the combiner chooses Lagrange coefficients $\gamma_1, \ldots, \gamma_t$ so that $\sum_i \gamma_i \mathsf{sk}_i = \mathsf{sk}$. Then, she computes

$$\sum_i \gamma_i m_i = \sum_i \gamma_i \langle \mathsf{ct}, \mathsf{sk}_i \rangle = \langle \mathsf{ct}, \sum_i \gamma_i \mathsf{sk}_i \rangle = \langle \mathsf{ct}, \mathsf{sk} \rangle,$$

followed by rounding, as desired. However, this appealingly simple construction turns out to be insecure. This is because each time a party computes a partial decryption, it leaks information about its secret share sk_i via the inner product with (the public) value ct.

To get around this insecurity, a natural approach is to add noise to the partial decryption which quickly transforms a simple computation to intractable. However, care must be taken to ensure that this added noise does not affect correctness, since it is later multiplied by the Lagrange coefficients during reconstruction: the previous $\sum_i \gamma_i m_i$ will now become $\sum_i \gamma_i (m_i + e_i)$ for some noise terms e_i. BGGJKRS propose two solutions – one to use a secret sharing scheme whose reconstruction coefficients are binary, and another, to "clear the denominators" by observing that since the Lagrange coefficients are rational numbers, it is possible to scale them to be integers. The exact details are irrelevant for the current discussion and hence omitted (please refer to [8] for more details).

To use this technique to construct threshold signatures, the authors propose the following. Choose a signature scheme Sig, compute an FHE encryption $\mathsf{ct}_{\mathsf{sk}}$ of its signing key Sig.sk and let each signer homomorphically evaluate the signing algorithm for a message μ on this ciphertext. In more detail, given $\mathsf{ct}_{\mathsf{sk}} = \mathsf{FHE.Enc}(\mathsf{Sig.sk})$, each party first computes $\mathsf{FHE.Eval}(C, \mathsf{ct}_{\mathsf{sk}})$ where C is the circuit $\mathsf{Sig.Sign}(\mu, \cdot)$. By correctness of FHE, this yields an FHE encryption of the signature $\sigma = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, \mu)$. To this ciphertext, the thresholdization trick described above may now be applied.

[2] Note that we can trade the offline sharing of correlated randomness with an additional communication round in the signing protocol – however, this would destroy round optimality.

Modeling the Adversary and Effect on Parameters. In their analysis, BGGJKRS consider the complexity-theory security requirement of "no polynomial time attacks", corresponding to assuming attacks with advantage $\epsilon = \lambda^{-O(1)}$ and run-time $\lambda^{O(1)}$. However, for practically motivated primitives like threshold signatures, it is more meaningful to consider attackers with advantage $2^{-o(\lambda)}$ and run-time $2^{o(\lambda)}$. We choose our adversarial model so that all attacks should be exponential while all honest algorithms run in polynomial time. Compared to the complexity-theory definition of security, this provides a much more significant (and practically meaningful) hardness gap between honest and malicious parties.

For subexponentially strong attackers as described above, the noise flooding used in BGGJKRS is exponential, severely damaging the practicality of the scheme, despite the exciting developments in practical FHE [18, 58, 41, 17, 16, 26]. In more detail, the proof requires to make the statistical distance between some noise terms e' and $e + e'$ small, so that knowing $e + e'$ is essentially the same as knowing e', which does not carry sensitive information. To get security against attackers with advantage $2^{-o(\lambda)}$, the statistical distance must be set to $2^{-\Omega(\lambda)}$ and, as a result, the standard deviation of e' must be a factor $2^{\Omega(\lambda)}$ larger than the upper bound on $|e|$.

Tightening Analysis via Rényi divergence. In this work, we examine whether this flooding noise can be improved so that the impact of flooding e by e' on efficiency is minimised. To this end, we explore using *Rényi divergence* rather than statistical distance to bound the distance between distributions in the security proof. Rényi divergence has been used in prior work as a replacement to the statistical distance in lattice based cryptography [42, 45, 7, 52, 40, 3, 9, 2, 4]. To understand why this may be beneficial, let us first see how statistical distance is used in typical security proofs of cryptography. Let $\mathcal{P}$ and $\mathcal{Q}$ be two non-vanishing probability distributions over a common measurable support X. Typical security proofs consider a hard problem relying on some ideal distribution $\mathcal{Q}$, and then replace this ideal distribution by a real world distribution $\mathcal{P}$. When the statistical distance $\Delta(\mathcal{Q}, \mathcal{P})$ between the two distributions is small, the problem remains hard, implying security. This is made rigorous by the so-called "probability preservation" property which says that for any measurable event $E \subseteq X$, we have $\mathcal{Q}(E) \geq \mathcal{P}(E) - \Delta(\mathcal{Q}, \mathcal{P})$.

Let us now define Rényi Divergence (RD). For $a \in (1, \infty)$, the RD of order a is defined by $R_a(\mathcal{P}||\mathcal{Q}) = \left(\sum_{x \in X} \frac{\mathcal{P}(x)^a}{\mathcal{Q}(x)^{a-1}}\right)^{\frac{1}{a-1}}$. It enjoys an analogous probability preservation property, though multiplicative as against additive. For $E \subseteq X$, we have $\mathcal{Q}(E) \geq \mathcal{P}(E)^{\frac{a}{a-1}} / R_a(\mathcal{P}||\mathcal{Q})$. Thus, if an event E occurs with significant probability under $\mathcal{P}$, and if the SD or the RD is small, then the event E also occurs with significant probability under $\mathcal{Q}$. As discussed in [4], probability preservation in SD is meaningful when the distance is smaller than any $\mathcal{P}(E)$ that the security proof is required to deal with – if $\mathcal{P}(E) \geq \epsilon$ for some ϵ, then we require that $\Delta(\mathcal{Q}, \mathcal{P}) < \epsilon$. The analogous requirement for RD is $R_a(\mathcal{P}||\mathcal{Q}) \leq \mathsf{poly}(1/\epsilon)$. Bai et al. [4] observed that RD is often less demanding than SD in proofs. This is because RD between distributions may be small enough to suffice for RD probability preservation while SD may be too large for the SD probability preservation to be applicable. Thus, RD can often serve as a better tool for security analysis, especially in applications with search-type security definitions, like signatures.

In this work, we study the applicability of RD analysis in the construction of threshold signatures. Building upon the above approach, we show that a limited flooding growing as $\sqrt{Q}$ suffices in BGGJKRS, where Q is the number of signing queries made by the attacker. We note that this is a substantial improvement in practice, since the number of sign queries is typically very different, and much smaller, than the run time of the adversary. Note that signature queries require active participation by an honest user and there is no reason for an honest user to keep replying after an overly high number of queries that clearly shows adversarial behavior. As a concrete example, in the NIST post quantum project [1], adversarial runtimes can go up to 2^{256} in some security levels, but the number of signature queries is always bounded by 2^{64} (which is itself an overly conservative bound in many scenarios). Thus, dependence on the number of queries is significantly better than exponential dependence on the security parameter, and this leads to a significant improvement in the signature bit size.

Optimality of our Moderate Flooding. We also 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.

This attack may be avoided by making the flooding noise a deterministic function of the message, e.g., by using a pseudo-random function evaluated on the message to generate the noise. We show that this modification is not sufficient to make the threshold signature construction secure. For this purpose, we design a signature scheme which includes "useless" information in the signature: this information does not affect correctness nor security of the signature itself, but allows us to recreate the attack described above on the resulting threshold signature. We start with a secure signature scheme $\mathsf{Sig} = (\mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$ whose signing key is a uniform bit-string among those with the same number of 0's and 1's. Now, let us consider a special signature scheme $\mathsf{Sig}' = (\mathsf{Sig}'.\mathsf{KeyGen}, \mathsf{Sig}'.\mathsf{Sign}, \mathsf{Sig}'.\mathsf{Verify})$ derived from Sig by modifying the signing algorithm as follows: for $i \in [|\mathsf{Sig.sk}|]$, if $\mathsf{Sig.sk}_i = 0$, then append a 0 to the signature. Since our signing key has exactly half as many 0's as 1's, this leads to a string of $|\mathsf{Sig.sk}|/2$ zeroes being appended to every signature: this does not leak any information and does not affect correctness (it is simply ignored during verification). Now, consider using Sig' to instantiate our threshold signature scheme. Then, for any message M, the FHE ciphertext CT_{σ_M} now additionally includes homomorphically evaluated encryptions of $\{\mathsf{Sig.sk}_i\}_{i \in [|\mathsf{Sig.sk}|]:\mathsf{Sig.sk}_i=0}$. Note that these extra encryptions are designed to be a deterministic function of the secret key so that across multiple messages, the corresponding error term (obtained via homomorphic evaluation) will not change. On the other hand, the message-dependent error terms can be assumed to change across messages. Due to this, the error term recovered by the adversary will be a sum of a fixed term (dependent only on the secret key) plus a fresh term per signature, which allows it to recreate the first attack. Please see Section 3 for more details.

Homomorphism-Friendly Signature. Next, we provide a variant of Lyubashevsky's signature scheme [47] which enjoys low circuit depth and is homomorphism friendly. As discussed above, Lyubashevsky's signature contains a rejection sampling step, whose purpose is to make the distribution of the resultant signature canonical, but this step is cumbersome to implement homomorphically. We show that by using RD analysis in place of statistical distance, analogously to the case of threshold signatures discussed above, the rejection sampling step can be replaced by noise flooding of moderate magnitude $\sqrt{Q}$. Additionally, we show that this flooding is optimal – please see Section 4 for details.

Towards Adaptive Security. Another limitation of the construction of $\mathsf{BGGJKRS}$ 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. At a high level, 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.

For more details, let us consider the case of an N-out-of-N threshold signature. In the simulation, the challenger knows which party is honest at the start of the game, e.g., player N. Now, the challenger can sample FHE secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_{N-1}$ randomly, implicitly setting the last share as $\mathsf{sk} - \sum_{i \in [N-1]} \mathsf{sk}_i$. To invoke the unforgeability of the underlying signature scheme Sig, the challenger must "forget" the signing key $\mathsf{Sig.sk}$ at some point in the proof, and rely on the Sig challenger to return signatures, which it then encrypts using the (public key) FHE scheme. By correctness of FHE, this is the same as computing the signing circuit for a given message on the ciphertext containing the secret key, which is what happens in the real world. However, the FHE encryption of signing key $\mathsf{Sig.sk}$ is provided as part of the public parameters in the real world, which in turn means that the FHE secret key must be "forgotten" so that the FHE ciphertext of $\mathsf{Sig.sk}$ is indistinguishable from a dummy value. Yet the challenger must return legitimate partial signatures of queried

messages m_j in the security game, which in turn are (noisy) partial decryptions of the FHE ciphertexts $\widehat{\sigma_j}$ of signatures σ_j. Knowing all the corrupt secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_{N-1}$ from the outset enables the challenger to walk this tightrope successfully – it obtains σ_j from the Sig challenger, computes an FHE encryption $\widehat{\sigma_j}$ of this, computes partial decryptions using $\mathsf{sk}_1, \ldots, \mathsf{sk}_{N-1}$, floods these with appropriate noise and returns these to the adversary.

In the adaptive game, the honest player is not known at the beginning of the game so the challenger is unable to sample FHE secret key shares as described above. When requested for a partial signatures for message m_j, it can obtain the corresponding signature σ_j and can FHE encrypt it, but cannot decrypt it using secret key shares which are unavailable. To preserve correctness and indistinguishability from the real world, it is forced to return (noisy) random secret shares $\{\sigma_{i,j}\}_{i\in[N],j\in\mathsf{poly}}$ of σ_j as partial signatures, for unbounded j. Later if user 1 is corrupted (say), the adversary receives the secret key share sk_1. Now, to preserve indistinguishability, the challenger must explain the partial signatures $\{\sigma_{1,j}\}_{j\in\mathsf{poly}}$ corresponding to user 1 as $\langle\widehat{\sigma}_j, \mathsf{sk}_1\rangle \approx \sigma_{1,j}$, which seems impossible for unbounded j.

We overcome this hurdle in the ROM by having the challenger simulate all partial keys as though corresponding to a corrupt user and when 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. In more detail, we modify the signing key to additionally contain a random secret share of $\mathbf{0}$, i.e., each party is provided a vector $\mathbf{v}_i$ of length N, such that $\sum_{i\in[N]} \mathbf{v}_i = \mathbf{0}$. In the scheme, to compute a partial signature for a message m_j, the partial signing algorithm first computes $r_{i,j} = H(j,K)^\mathsf{T}\ \mathbf{v}_i$ where $H(j,K)$ is a random vector of length N, and K is a secret value required for a technical reason that we will not discuss here. It then returns $\langle\widehat{\sigma}_j, \mathsf{sk}_i\rangle + \mathsf{noise}_{i,j} + r_{i,j}$. By linearity, it holds that $\sum_{i\in[N]} H(j,K)^\mathsf{T}\ \mathbf{v}_i = \mathbf{0}$, so correctness is not affected. But the unbounded programmability of the ROM helps us overcome the aforementioned impasse in the proof. Now, the challenger answers partial signature queries by returning (noisy) random secret shares $\{\sigma_{i,j}\}_{i\in[N],j\in\mathsf{poly}}$ of σ_j. When later, user 1 is corrupted, it can correctly explain the returned signatures as follows: it samples sk_1, computes $d_{1,j} = \langle\widehat{\sigma}_j, \mathsf{sk}_1\rangle + \mathsf{noise}$ and sets $r_{1,j} = \sigma_{1,j} - d_{1,j}$. Now we may program $H(j,K)$ so that $r_{i,j} = H(j,K)^\mathsf{T}\ \mathbf{v}_i$ for all j – it can be checked that there are enough degrees of freedom to satisfy these equations. However, since all secrets of a user are revealed when it is corrupted, the value $H(j,K)$ is fixed when even a single user is corrupted. This is why we require that all corruptions be made simultaneously and only achieve the restricted notion of "partial" adaptivity.

We also provide a construction in the standard model which achieves full adaptivity where users can be corrupted at arbitrary points in the security game. But this construction is only secure 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 [29] and references therein). We refer the reader to Section 5 and 6 for more details.

2 Preliminaries

In this section, we define the notations and other preliminaries used in our work.

Notation. We write vectors with bold small letters and matrices with bold capital letters. For any vector $\mathbf{v}$, $\mathbf{v}[i]$ or $\mathbf{v}_i$ denotes its ith element unless otherwise indicated. 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 (see Def. 2.29) 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.*
- $\mathsf{TS.PartSignVerify}(\mathsf{pp}, m, \sigma_i) \to \mathsf{accept}/\mathsf{reject}$*: On input the public parameters* pp*, a message $m \in \{0,1\}^*$ and a partial signature σ_i, the partial signature verification algorithm outputs accept or reject.*
- $\mathsf{TS.Combine}(\mathsf{pp}, \{\sigma_i\}_{i\in S}) \to \sigma_m$*: On input the public parameters* pp *and the partial signatures $\{\sigma_i\}_{i\in S}$ for $S \in \mathsf{A}$, the combining algorithm outputs a full signature σ_m.*
- $\mathsf{TS.Verify}(\mathsf{vk}, m, \sigma_m) \to \mathsf{accept}/\mathsf{reject}$*: On input a verification key* vk*, a message m and a signature σ_m, the verification algorithm outputs accept or reject.*

A TS scheme should satisfy the following requirements.

Definition 2.2 (Compactness). *A* TS *scheme for $\mathbb{S}$ satisfies compactness if there exist polynomials $\mathrm{poly}_1(\cdot), \mathrm{poly}_2(\cdot)$ 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 $\sigma_m \leftarrow \mathsf{TS.Combine}(\mathsf{pp}, \{\sigma_i\}_{i\in S})$, we have that $|\sigma_m| \le \mathrm{poly}_1(\lambda)$ and $|\mathsf{vk}| \le \mathrm{poly}_2(\lambda)$.*

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

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 issues polynomial number of following two types of queries in any order*
   - *Corruption queries: $\mathcal{A}$ outputs a party $i \in [N]$ which it wants to corrupt. In response, the challenger returns the key share sk_i.*
   - *Signature queries: $\mathcal{A}$ outputs a query of the form (m, i), where m is a message and $i \in [N]$, to get partial signature σ_i for m. The challenger computes σ_i as $\mathsf{TS.PartSign}(\mathsf{pp}, \mathsf{sk}_i, m)$ and provides it to $\mathcal{A}$.*
4. *At the end of the experiment, adversary $\mathcal{A}$ outputs a message-signature pair (m^*, σ^*).*
5. *The experiment outputs 1 if both of the following conditions are met: (i) Let $S \subseteq [N]$ be the set of corrupted parties, then S is a an invalid party set, i.e. $S \notin \mathsf{A}$ (ii) m^* was not queried previously as a signing query and $\mathsf{TS.Verify}(\mathsf{vk}, m^*, \sigma^*) = accept$.*

We also consider following weaker notions of unforgeability.

Definition 2.6 (Partially Adaptive Unforgeability). *Here, all the corruptions are done all at once. That is, Step 3, is now changed as follows:*

– *$\mathcal{A}$ issues polynomial number of signing queries of the form (m, i) adaptively and gets corresponding σ_i's.*
– *$\mathcal{A}$ outputs a set $S \subseteq [N]$ such that $S \notin \mathsf{A}$. The challenger returns $\{\mathsf{sk}_i\}_{i \in S}$.*
– *$\mathcal{A}$ continues to issue polynomial number of more signing queries of the form (m, i) adaptively, and gets corresponding σ_i.*

Rest of the steps remain the same.

Definition 2.7 (Selective Unforgeability). *In this case, all the corruptions happen before any signing query. That is, Step 3, is now further changed as follows:*

– *$\mathcal{A}$ outputs a set $S \subseteq [N]$ such that $S \notin \mathsf{A}$. The challenger returns $\{\mathsf{sk}_i\}_{i \in S}$.*
– *$\mathcal{A}$ issues polynomial number of signing queries of the form (m, i) adaptively, and gets corresponding σ_i.*

Rest of the steps remain the same.

Definition 2.8 (Robustness). *A TS scheme for $\mathbb{S}$ satisfies robustness if for all λ, the following holds. For any 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 $2^{-\Omega(\lambda)}$:*

– *On input the security parameter 1^λ, the adversary outputs an access structure $\mathsf{A} \in \mathbb{S}$.*
– *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 $\mathcal{A}$.*
– *Adversary $\mathcal{A}$ outputs a partial signature forgery (m^*, σ_i^*, i).*
– *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 Fully Homomorphic Encryption (FHE)

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

Definition 2.9 (Fully Homomorphic Encryption). *A fully homomorphic encryption scheme FHE is a tuple of PPT algorithms $\mathsf{FHE} = (\mathsf{FHE.KeyGen}, \mathsf{FHE.Enc}, \mathsf{FHE.Eval}, \mathsf{FHE.Dec})$ defined as follows:*

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

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

Definition 2.10 (Compactness). *We say that an FHE scheme is compact if there exists a polynomial function $f(\cdot, \cdot)$ such that for all λ, depth bound d, circuit $\mathcal{C} : \{0,1\}^k \rightarrow \{0,1\}$ of depth at most d, and $\mu_i \in \{0,1\}$ for $i \in [k]$, the following holds: for $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda, 1^d)$, $\mathsf{ct}_i \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, \mu_i)$ for $i \in [k]$, $\hat{\mathsf{ct}} \leftarrow \mathsf{FHE.Eval}(\mathsf{pk}, \mathcal{C}, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$, the bit-length of $\hat{ct}$ is at most $f(\lambda, d)$.*

Definition 2.11 (Correctness). *We say that an FHE scheme is correct if for all λ, depth bound d, circuit $\mathcal{C} : \{0,1\}^k \rightarrow \{0,1\}$ of depth at most d, and $\mu_i \in \{0,1\}$ for $i \in [k]$, the following holds: for $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda, 1^d)$, $\mathsf{ct}_i \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, \mu_i)$ for $i \in [k]$, $\hat{\mathsf{ct}} \leftarrow \mathsf{FHE.Eval}(\mathsf{pk}, \mathcal{C}, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$, we have*

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

Definition 2.12 (Security). *We say that an* FHE *scheme is secure if for all λ and depth bound d, the following holds: for any adversary $\mathcal{A}$ with run-time $2^{o(\lambda)}$, the following experiment outputs 1 with probability $2^{-\Omega(\lambda)}$:*

1. *On input the security parameter λ and a depth bound d, the challenger runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda, 1^d)$ and $\mathsf{ct} \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, b)$ 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* FHE having some additional properties as described in [8]. These properties are satisfied by direct adaptations of typical FHE schemes such as [12, 35] (see, e.g., [8, Appendix B]).

Definition 2.13 (Special FHE). *An* FHE *scheme is a special* FHE *scheme if it satisfies the following properties:*

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

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

2.3 Threshold Fully Homomorphic Encryption

Definition 2.14 (Threshold Fully Homomorphic Encryption). *A threshold fully 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* $\mathsf{TFHE} = (\mathsf{TFHE.KeyGen}, \mathsf{TFHE.Enc}, \mathsf{TFHE.Eval}, \mathsf{TFHE.PartDec}, \mathsf{TFHE.FinDec})$ *with the following specifications:*

- $\mathsf{TFHE.KeyGen}(1^\lambda, 1^d, \mathsf{A}) \to (\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N)$*: 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 $\{\mathsf{sk}_i\}_{i=1}^N$.*
- $\mathsf{TFHE.Enc}(\mathsf{pk}, \mu) \to \mathsf{ct}$*: On input a public key* pk *and a single bit message $\mu \in \{0,1\}$, the encryption algorithm outputs a ciphertext* ct*.*
- $\mathsf{TFHE.Eval}(\mathsf{pk}, \mathcal{C}, \mathsf{ct}_1, \mathsf{ct}_2, \ldots, \mathsf{ct}_k) \to \hat{\mathsf{ct}}$*: 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* $\hat{\mathsf{ct}}$*.*
- $\mathsf{TFHE.PartDec}(\mathsf{pk}, \mathsf{sk}_i, \mathsf{ct}) \to p_i$*: On input a public key* pk*, a secret key share* sk_i *and a ciphertext* ct*, the partial decryption algorithm outputs a partial decryption p_i corresponding to the party P_i.*
- $\mathsf{TFHE.FinDec}(\mathsf{pk}, \{p_i\}_{i \in S}) \to \hat{\mu}$ *: 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* TFHE *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 $(\mathsf{pp}, sk_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{TFHE.KeyGen}(1^\lambda, 1^d, \mathsf{A})$, $\mathsf{ct}_i \leftarrow \mathsf{TFHE.Enc}(\mathsf{pk}, \mu_i)$ for $i \in [k]$, $\hat{\mathsf{ct}} \leftarrow \mathsf{TFHE.Eval}(\mathsf{pk}, \mathcal{C}, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$:*

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

Semantic security. *A* TFHE *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 $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{TFHE.KeyGen}(1^\lambda, 1^d, \mathsf{A})$ 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 $\mathsf{TFHE.Enc}(\mathsf{pk}, b)$, 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 TFHE 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 $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{TFHE.KeyGen}(1^\lambda, 1^d, \mathsf{A})$ 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 $\{\mathsf{ct}_i = \mathsf{TFHE.Enc}(\mathsf{pk}, \mu_i)\}_{i \in [k]}$ 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 $\hat{\mathsf{ct}} \leftarrow \mathsf{TFHE.Eval}(\mathsf{pk}, \mathcal{C}, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ and sends $\{\mathsf{TFHE.PartDec}(\mathsf{pk}, \mathsf{sk}_i, \hat{\mathsf{ct}})\}_{i \in S}$ 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 $\{\mathsf{ct}_i = \mathsf{TFHE.Enc}(\mathsf{pk}, \mu_i)\}_{i \in [k]}$ 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.*

2.4 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 2.15 (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.*

- $\mathsf{PrmsGen}(1^\lambda, 1^N) \to \mathsf{prms}$*: Gets the security parameter λ and a data-size bound N and generates public parameters* prms.
- $\mathsf{KeyGen}(1^\lambda, \mathsf{prms}) \to (\mathsf{pk}, \mathsf{sk})$*: Produces a public verification key* pk *and a secret signing key* sk.
- $\mathsf{Sign}(\mathsf{sk}, (x_1, \ldots, x_N), \tau) \to (\sigma_\tau, \sigma_1, \ldots, \sigma_N)$*: Signs some data $(x_1, \ldots, x_N) \in \mathcal{X}^*$ under a label $\tau \in \{0,1\}^*$.*
- $\mathsf{SignEval}(\mathsf{prms}, g, \sigma_\tau, (x_1, \sigma_1), (x_\ell, \sigma_\ell)) \to \sigma^*$*: Homomorphically computes a signature σ^* for $g(x_1, \ldots, x_N)$.*

- $\mathsf{Process}(\mathsf{prms}, g) \to \alpha_g$*: Produces a "public-key"* α_g *for the function* g*.*
- $\mathsf{Verify}(\mathsf{pk}, \alpha_g, y, \tau, (\sigma_\tau, \sigma^*)) \to \mathsf{accept/reject}$*: 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 2.16 (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* i*th 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 2.17 (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 2.18 (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* 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.*

2.5 Lattices and Discrete Gaussians

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

Definition 2.19 (Lattice). *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.*

Definition 2.20 (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:*

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

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* $\mathbf{c}$.

Definition 2.21 (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* $\rho_{1/s}(\Lambda^* \setminus \{0\}) \leq \epsilon$.

2.6 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 [53, 11].

Definition 2.22 (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* $\mathsf{LWE}_{q,\alpha,m}$ *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 2.23 (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 $\mathbf{t} = \mathbf{As}$. Then given $\mathbf{A}, \mathbf{t}$, the task is to find $\mathbf{s}' \in \{-d, \ldots, 0, \ldots, d\}^m$ such that $\mathbf{As}' = \mathbf{t}$.*

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

Definition 2.24 (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 $\mathbf{Av} = 0$ 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 $\mathbf{v}$.

2.6.1 Other Useful Lemmas

Lemma 2.25 (Adapted from [47, Lemma 4.4]).

1. *For any $k > 0$, $\Pr[|z| > k\sigma; z \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}] \leq 2\exp(-\pi k^2)$.*
2. *For any $\sigma \geq 3$, $H_\infty(\mathcal{D}_{\mathbb{Z}^m,\sigma}) \geq m$.*
3. *For any $k > 1/\sqrt{2\pi}$, $\Pr[\|\mathbf{z}\| > k\sigma\sqrt{2\pi m};\ \mathbf{z} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}] < (k\sqrt{2\pi})^m \exp(\frac{m}{2}(1 - 2\pi k^2))$.*

2.7 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 [4], which may help obtain security proofs for smaller scheme parameters and may sometimes lead to simpler proofs.

Definition 2.26 (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

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

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.

The following lemma is borrowed from [4, Lemma 2.9], with the exception of the multiplicativity property for non-independent variables, which is borrowed from [54, Proposition 2].

Lemma 2.27. *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 $a \in [1, \infty]$.*
  - *$R_a(P||Q) \leq R_a(P_1||Q_1) \cdot \max_{y_1 \in Y_1} R_a(P_{2|1}(\cdot|y_1)||Q_{2|1}(\cdot|y_1))$.*
- ***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 [56].

Lemma 2.28 ([4]). *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 $a \in (1, +\infty)$*

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

2.8 Secret Sharing

We now recall some standard definitions related to secret sharing.

Definition 2.29 (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 2.30 (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 $|S| \geq t$. 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 2.31 (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 2.32 (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:*

- $\mathsf{SS.Share}(\mathsf{k}, \mathsf{A}) \rightarrow (s_1, \ldots, s_N)$*: 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.*
- $\mathsf{SS.Combine}(B) \rightarrow \mathsf{k}$*: 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 2.33 (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 2.34 (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 2.35 (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* j*th 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 2.36. *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 [8] 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.

3 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* [8]. We also show that the limited flooding that we use is in fact optimal, and 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 and restricted to selective security. We extend it to to adaptive security in Section 5 and Section 6 and t out of N threshold in Section 7. In Section 4, we show how to instantiate the underlying signature scheme using a variant of Lyubashevsky's signature [47] with matching moderate flooding.

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

Our scheme is similar to the one in [8]. 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 fully homomorphic encryption scheme $\mathsf{FHE} = (\mathsf{FHE.KeyGen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval})$. As in [8], we also assume that the $\mathsf{FHE.Dec}$ can be divided into two sub-algorithms: $\mathsf{FHE.decode}_0$ and $\mathsf{FHE.decode}_1$ as defined in Section 2.2.
- 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.

The construction is provided in Figure 1.

3.1.1 Correctness From the correctness of $\mathsf{FHE.Eval}$ 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 FHE secret key $\mathsf{FHE.sk}$.
So, $\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_\sigma) = \sigma_M \lfloor q/2 \rceil + e$. The signature computed by the $\mathsf{TS.Combine}$ algorithm is

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

3.1.2 Unforgeability For security, we prove the following theorem.

Theorem 3.1. *Assume F is a secure PRF,* Sig *is a UF-CMA secure signature scheme,* FHE *is a secure fully homomorphic encryption scheme (Definition 2.12),* Share *is a secret sharing scheme that satisfies privacy (Definition 2.34) and* HS *is a context hiding secure homomorphic signature scheme (Definitions 2.18). Then the construction of threshold signatures in Figure 1 satisfies selective unforgeablity (Definition 2.7) 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 $\mathcal{A}$ 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 $\tilde{\pi}_1$ in $\mathsf{PartSign}$ is now generated using HS simulator as $\tilde{\pi}_1 = \mathsf{HS.Sim}(\mathsf{HS.sk}, \alpha, \sigma_1, \tau_1, \pi_{\tau_1})$, 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. For each party P_i, sample a PRF key $\mathsf{sprf}_i \leftarrow \mathcal{K}$.
2. Generate the signature scheme's keys $(\mathsf{Sig.vk}, \mathsf{Sig.sk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$.
3. Generate the FHE keys $(\mathsf{FHE.pk}, \mathsf{FHE.sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda)$ and compute an FHE encryption of $\mathsf{Sig.sk}$ as $\mathsf{CT}_{\mathsf{Sig.sk}} \leftarrow \mathsf{FHE.Enc}(\mathsf{FHE.pk}, \mathsf{Sig.sk})$.
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 $(\mathsf{FHE.sk}, \mathsf{sprf}_i)$.
5. Share $\mathsf{FHE.sk}$ as $\{\mathsf{sk}_i\}_{i=1}^N \leftarrow \mathsf{Share}(\mathsf{FHE.sk})$ such that $\sum_{i=1}^N \mathsf{sk}_i = \mathsf{FHE.sk}$.
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 $\mathsf{TSig.pp} = \{\mathsf{FHE.pk}, \mathsf{CT}_{\mathsf{Sig.sk}}, \mathsf{HS.pp}, \mathsf{HS.pk}, \{\tau_i, \pi_{\tau_i}\}_{i=1}^N\}$, $\mathsf{TSig.vk} = \mathsf{Sig.vk}$, $\mathsf{TSig.sk} = \{\mathsf{TSig.sk}_i = (\mathsf{sk}_i, \mathsf{sprf}_i, \pi_i)\}_{i=1}^N$.

$\mathsf{TS.PartSign}(\mathsf{TSig.pp}, \mathsf{TSig.sk}_i, M)$: 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 $(\mathsf{FHE.pk}, \mathsf{CT}_{\mathsf{Sig.sk}}, \mathsf{HS.pp}, \mathsf{HS.pk}, \{\tau_i, \pi_{\tau_i}\}_{i=1}^N\})$ and $\mathsf{TSig.sk}_i$ 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 $\mathsf{CT}_\sigma = \mathsf{FHE.Eval}(\mathsf{FHE.pk}, \mathcal{C}_M, \mathsf{CT}_{\mathsf{Sig.sk}})$.
3. Compute $\sigma_i = \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_\sigma) + e_i'$.
4. This step computes a homomorphic signature $\tilde{\pi}_i$ on partial signature σ_i to provide robustness (Definition 2.8).
   Let $\mathcal{C}_{\mathsf{PS}}$ be the circuit to compute $\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_\sigma) + e_i'$ in which CT_σ is hardcoded and the FHE 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 $\sigma_M = \mathsf{FHE.decode}_1(\sum_{i=1}^N \sigma_i)$.

$\mathsf{TS.PartSignVerify}(\mathsf{TSig.pp}, M, y_i)$: Upon 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 $\mathsf{CT}_\sigma = \mathsf{FHE.Eval}(\mathsf{FHE.pk}, \mathcal{C}_M, \mathsf{CT}_{\mathsf{Sig.sk}})$.
2. Compute $\alpha = \mathsf{HS.Process}(\mathsf{HS.pp}, \mathcal{C}_{\mathsf{PS}})$, where $\mathcal{C}_{\mathsf{PS}}$ is as described above.
3. Parse y_i as $(\sigma_i, \tilde{\pi}_i)$ and 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 $B_{eval} \leq \mathrm{poly}(\lambda)$ is a bound on the FHE decryption noise after homomorphic evaluation of the signing circuit $\mathcal{C}_M$, and Q is the bound on the number of signatures.

Fig. 1. Optimization of Boneh et al Threshold Signature Scheme.

Hybrid_2: Same as Hybrid_1 except that to compute $\sigma_1 = \mathsf{FHE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_\sigma) + e_1'$, 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. Computes $\mathsf{CT}_\sigma = \mathsf{FHE.Eval}(\mathsf{FHE.pk}, \mathcal{C}_M, \mathsf{CT}_{\mathsf{Sig.sk}})$ and $\{\sigma_i' = \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_\sigma)\}_{i\in[2,N]}$.
2. Computes $\sigma_M = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, M)$ and set $\sigma_1 = \sigma_M \left\lfloor \frac{q}{2} \right\rceil - \sum_{i=2}^N \sigma_i' + e_1'$, where $e_1' \leftarrow \mathcal{D}_s$.

Hybrid_4: Same as Hybrid_3 except that instead of sharing $\mathsf{FHE.sk}$, now the challenger generates the FHE 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 $\mathsf{CT}_{\mathbf{0}} = \mathsf{FHE.Enc}(\mathsf{FHE.pk}, \mathbf{0})$.

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

Claim 3.2 *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, $\tilde{\pi}_1 = \mathsf{HS.Sim}(\mathsf{HS.sk}, \alpha, \sigma_1, \tau_1, \pi_{\tau_1})$, where $\alpha = \mathsf{HS.Process}(\mathsf{HS.pp}, \mathcal{C}_{\mathsf{PS}})$. Hence, the two hybrids are indistinguishable because of the context hiding property of HS which ensures that $\mathsf{HS.Hide}(\mathsf{HS.pk}, \sigma_1, \pi_1^*) \approx \mathsf{HS.Sim}(\mathsf{HS.sk}, \alpha, \sigma_1, \tau_1, \pi_{\tau_1})$.

Claim 3.3 *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 3.4 *If there is an adversary that can win the unforgeability game in* Hybrid_2 *with probability* ϵ*, then its probability of winning the game in* Hybrid_3 *is at least* $\epsilon^2/2$.

Proof. Let the number of signing queries that the adversary makes be Q. The two hybrids differ only in the error term in σ_1, as shown below. In Hybrid_2, we have $\sigma_1 = \mathsf{FHE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_\sigma) + e_1'$, for $e_1' \leftarrow \mathcal{D}_s$. In Hybrid_3, we have:

$$\begin{aligned}
\sigma_1 &= \sigma_M.\lfloor q/2 \rceil - \sum\nolimits_{i=2}^{N} \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_\sigma) + e_1' \\
&= \sigma_M.\lfloor q/2 \rceil - \sum\nolimits_{i=1}^{N} \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_\sigma) + \mathsf{FHE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_\sigma) + e_1' \\
&= \sigma_M.\lfloor q/2 \rceil - \mathsf{FHE.decode}_0(\sum\nolimits_{i=1}^{N} \mathsf{sk}_i, \mathsf{CT}_\sigma) + \mathsf{FHE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_\sigma) + e_1' \\
&= \sigma_M.\lfloor q/2 \rceil - \mathsf{FHE.decode}_0(\mathsf{sk}, \mathsf{CT}_\sigma) + \mathsf{FHE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_\sigma) + e_1' \\
&= \sigma_M.\lfloor q/2 \rceil - \sigma_M.\lfloor q/2 \rceil + e + \mathsf{FHE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_\sigma) + e_1' \\
&= \mathsf{FHE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_\sigma) + (e_1' + e),
\end{aligned}$$

for some e satisfying $|e| \leq B_{eval}$. Thus, in Hybrid_2, the error term in σ_1 is e_1', while in Hybrid_3, it is $e_1' + e$, where, $e_1' \leftarrow \mathcal{D}_s$, and e is the error in FHE ciphertext CT_σ.

Recall the distribution seen by the adversary – the public parameters $\mathsf{TSig.pp}$, the verification key $\mathsf{TSig.vk}$, the corrupted secret key shares $\mathsf{TSig.sk}_i$, the messages M_j and corresponding partial signatures $(\sigma_j, \tilde{\pi}_j)$. Note that since messages are chosen adaptively, their distribution depends on previous signature queries and responses, and in particular on the differently generated error terms in both hybrids. On the other hand $\mathsf{TSig.pp}$, $\mathsf{TSig.vk}$, $\{\mathsf{TSig.sk}_i\}$, $\{\tilde{\pi}_j\}$ are constructed identically in both hybrids and independently from the rest (in particular these error terms): we implicitly assume that they are fixed and known, and exclude them from the analysis. We refer to the distribution to be considered in Hybrid_2 as D_2 and in Hybrid_3 as D_3.

Let E_j be the random variables corresponding to the error term in CT_{σ_j} in the j-th response and $\mathcal{E}_j^{(2)}$ and $\mathcal{E}_j^{(3)}$ be their distributions in Hybrids 2 and 3, respectively. Similarly, let M_j be the random variable corresponding to the queried message in j-th query and $\mathcal{M}_j^{(2)}$ and $\mathcal{M}_j^{(3)}$ be their distributions in Hybrids 2 and 3, respectively. Then, from the discussion above, we have $\mathcal{E}_j^{(2)} = \mathcal{D}_s$ and $\mathcal{E}_j^{(3)} = \mathcal{D}_{s,e_j}$ for all $j \in [Q]$, where e_j is the error in CT_{σ_j} and can depend upon previous queries and responses.

Overall, we have $D_k = (\mathcal{E}_Q^{(k)}, \mathcal{M}_Q^{(k)}, \mathcal{E}_{Q-1}^{(k)}, \mathcal{M}_{Q-1}^{(k)}, \ldots, \mathcal{E}_1^{(k)}, \mathcal{M}_1^{(k)})$ for $k \in \{2, 3\}$ and

$$R_a(D_2 \| D_3) = R_a(\mathcal{E}_Q^{(2)}, \mathcal{M}_Q^{(2)}, \ldots, \mathcal{E}_1^{(2)}, \mathcal{M}_1^{(2)} \parallel \mathcal{E}_Q^{(3)}, \mathcal{M}_Q^{(3)}, \ldots, \mathcal{E}_1^{(3)}, \mathcal{M}_1^{(3)}). \tag{3.1}$$

Applying the multiplicativity property of the Rényi divergence (Lemma 2.27), we obtain that $R_a(D_2 \| D_3)$ is bounded from above by

$$\begin{aligned}
&\max_{x \in X} R_a(\mathcal{E}_Q^{(2)} | X = x \parallel \mathcal{E}_Q^{(3)} | X = x) \cdot R_a(\mathcal{M}_Q^{(2)}, \ldots, \mathcal{E}_1^{(2)}, \mathcal{M}_1^{(2)} \parallel \mathcal{M}_Q^{(3)}, \ldots, \mathcal{E}_1^{(3)}, \mathcal{M}_1^{(3)}) \\
= &\max_{x \in X} R_a(\mathcal{D}_s | X = x \parallel \mathcal{D}_{s,e_Q} | X = x) \cdot R_a(\mathcal{M}_Q^{(2)}, \ldots, \mathcal{E}_1^{(2)}, \mathcal{M}_1^{(2)} \parallel \mathcal{M}_Q^{(3)}, \ldots, \mathcal{E}_1^{(3)}, \mathcal{M}_1^{(3)}),
\end{aligned} \tag{3.2}$$

where $X = (M_Q, E_{Q-1}, \ldots, E_1)$ and e_Q is the error term in CT_{σ_Q}; note that e_Q may depend on the sample from X (which differs in Hybrids 2 and 3) and is bounded by B_{eval}. Then applying Lemma 2.28 in Equation (3.2), we get

$$\begin{aligned} R_a(D_2\|D_3) &\leq \exp(a\pi\|e_Q\|^2/s^2) \cdot R_a(\mathcal{M}_Q^{(2)}, \ldots, \mathcal{E}_1^{(2)}, \mathcal{M}_1^{(2)} \parallel \mathcal{M}_Q^{(3)}, \ldots, \mathcal{E}_1^{(3)}, \mathcal{M}_1^{(3)}) \\ &\leq \exp(a\pi B_{eval}^2/s^2) \cdot R_a(\mathcal{M}_Q^{(2)}, \ldots, \mathcal{E}_1^{(2)}, \mathcal{M}_1^{(2)} \parallel \mathcal{M}_Q^{(3)}, \ldots, \mathcal{E}_1^{(3)}, \mathcal{M}_1^{(3)}). \end{aligned}$$

Further, since M_Q is a function of $E_{Q-1}, M_{Q-1}, \ldots, E_1, M_1$, the data processing inequality (Lemma 2.27) gives

$$\begin{aligned} R_a(\mathcal{M}_Q^{(2)}, \mathcal{E}_{Q-1}^{(2)}, \ldots, \mathcal{E}_1^{(2)}, \mathcal{M}_1^{(2)} \parallel \mathcal{M}_Q^{(3)}, \mathcal{E}_{Q-1}^{(3)}, \ldots, \mathcal{E}_1^{(3)}, \mathcal{M}_1^{(3)}) \\ \leq R_a(\mathcal{E}_{Q-1}^{(2)}, \ldots, \mathcal{E}_1^{(2)}, \mathcal{M}_1^{(2)} \parallel \mathcal{E}_{Q-1}^{(3)}, \ldots, \mathcal{E}_1^{(3)}, \mathcal{M}_1^{(3)}), \end{aligned}$$

Hence, we get

$$\begin{aligned} R_a(D_2\|D_3) &\leq \exp(a\pi B_{eval}^2/s^2) \cdot R_a(\mathcal{E}_{Q-1}^{(2)}, \ldots, \mathcal{E}_1^{(2)}, \mathcal{M}_1^{(2)} \parallel \mathcal{E}_{Q-1}^{(3)}, \ldots, \mathcal{E}_1^{(3)}, \mathcal{M}_1^{(3)}) \\ &\leq \exp(a\pi B_{eval}^2 Q/s^2), \end{aligned}$$

where the last inequality follows from induction.

As $s = B_{eval} \cdot \sqrt{Q\lambda}$, we get $R_a(D_2\|D_3) \leq \exp(a\pi/\lambda)$. Therefore, from the probability preservation property of the Rényi divergence (Lemma 2.27), we have $D_3(\mathsf{E}) \geq \frac{D_2(\mathsf{E})^{\frac{a}{a-1}}}{R_a(D_2\|D_3)} \geq D_2(\mathsf{E})^{\frac{a}{a-1}} \exp(-\frac{a\pi}{\lambda})$. The result is obtained by setting $a = 2$.

Claim 3.5 *Assume that* Share *is a secret sharing scheme that satisfies privacy (Definition 2.34). 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 $(\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{Share}(\mathsf{FHE.sk})$, while in Hybrid_4, $(\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{Share}(\mathbf{0})$. Since, the adversary is given secret shares for an invalid set of parties, distribution in the two hybrids are identical.

Claim 3.6 *Assume* FHE *is a fully homomorphic encryption that satisfies security (Definition 2.12). 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 FHE scheme as follows.

1. After receiving $\mathsf{FHE.pk}$ from the FHE challenger, $\mathcal{B}$ generates $(\mathsf{Sig.sk}, \mathsf{Sig.vk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$ and the 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 $m_1 = \mathbf{0}$ to the FHE challenger.
4. After receiving the challenge ciphertext CT_b from the FHE challenger, $\mathcal{B}$ constructs $\mathsf{TSig.pp}$ using CT_b. It also generates $\mathsf{TSig.vk}$ and $\mathsf{TSig.sk}$ as defined for the hybrid. In particular, note that in both the hybrids, the key shares $\{\mathsf{sk}_i\}_{i=1}^N$ are generated as random secret shares of $\mathbf{0}$ in place of $\mathsf{FHE.sk}$ and hence $\mathcal{B}$ does not need $\mathsf{FHE.sk}$ to answer key queries.
5. To answer a $\mathsf{PartSign}$ query for a message M issued by adversary $\mathcal{A}$, $\mathcal{B}$ computes σ_1 as follows. It computes $\sigma = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, M)$, samples $e_1' \leftarrow \mathcal{D}_s$ and returns $\sigma_1 = \sigma - \sum_{i=2}^N \mathsf{TS.decode}_0(\mathsf{TSig.sk}_i, \mathsf{CT}_b) + e_1'$.
6. Finally, $\mathcal{A}$ outputs a guess bit b'. $\mathcal{B}$ returns the same to the FHE challenger.

Clearly, if $b = 0$, then $\mathcal{B}$ simulates Hybrid_4, else Hybrid_5 with $\mathcal{A}$. Hence if $\mathcal{A}$ wins with non-negligible probability in distinguishing the two hybrids then so does $\mathcal{B}$ against the FHE challenger.

Finally the proof of Theorem 3.1 completes with the following claim.

Claim 3.7 *If the underlying signature scheme* Sig *is unforgeable, then the advantage of the adversary in the unforgeability game of Definition 2.7 is negligible 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 a verification key $\mathsf{Sig.vk}$ from Sig challenger, $\mathcal{B}$ generates $(\mathsf{FHE.sk}, \mathsf{FHE.pk})$, $\mathsf{HS.pp}$, $(\mathsf{HS.pk}, \mathsf{HS.sk})$ and all the other values required to define $\mathsf{TSig.pp}, \mathsf{TSig.vk}$ and $\mathsf{TSig.sk}$ on its own. In particular, since in Hybrid_5, $\mathsf{TSig.pp}$ contains $\mathsf{CT_0}$ instead of $\mathsf{CT}_{\mathsf{Sig.sk}}$, $\mathcal{B}$ does not require $\mathsf{Sig.sk}$ to generate a valid $\mathsf{TSig.pp}$.
2. $\mathcal{B}$ then sends $\mathsf{TSig.vk} = \mathsf{Sig.vk}, \mathsf{TSig.pp}, \{\mathsf{TSig.sk}_i\}_{i=2}^{N}$ to $\mathcal{A}$.
3. To simulate $\mathsf{PartSign}$ query σ_1 for any message M, $\mathcal{B}$ needs σ_M. For this, it issues a signing query on message M to the Sig challenger and receives σ_M.
4. In the end, let (M^*, σ^*) be the forgery returned by $\mathcal{A}$. Then $\mathcal{B}$ returns the same to the Sig challenger.

Since $\mathcal{B}$ issues signing queries on only those messages 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.

3.2 Robustness

Claim 3.8 *If* HS *is multi data secure (Definition 2.17) homomorphic signature, then the construction of* TS *in Figure 1 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 $\mathsf{CT}_\sigma \leftarrow \mathsf{FHE.Eval}(\mathsf{FHE.pk}, \mathcal{C}_{M^*}, \mathsf{CT}_{\mathsf{Sig.sk}})$
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
$\sigma_i^* \neq \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_\sigma) + e_i'$, 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 2.17) against HS scheme, which can happen only with negligible probability. Note that since (τ_i, π_{τ_i}) are part of $\mathsf{HS.pp}$, case of type 2 in HS security definition is inherently applied.

3.3 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 $\sigma_{i,M} =$

$\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + e'_{i,M}$, where $\mathsf{TSig.sk}_i = (\mathsf{sk}_i, \mathsf{sprf}_i)$.[3] 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 $\sigma_{1,M} = \mathsf{FHE.decode}_0(\mathsf{sk}_1, \mathsf{CT}_{\sigma_M}) + e'_{1,M}$. From this the adversary can compute:

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

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 $\mathsf{FHE.sk}$.

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 $\sigma^{(1)}_{1,M}, \ldots, \sigma^{(Q)}_{1,M}$, where $\sigma^{(i)}_{1,M}$ denotes the partial signature returned for message M in the ith query. From these responses the adversary gets Q different values of the form

$$w^i = \mathsf{err}_M + e'^i_{1,M} \tag{3.3}$$

Since err_M is same, taking average on both sides of Equation (3.3) 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 $\mathsf{err}_M = \left\lfloor \frac{1}{Q}\sum_{i\in[Q]} w^i \right\rceil$. As $e'^1_{1,M}, \ldots, e'^Q_{1,M}$ are independently and identically distributed with mean 0, by Hoeffding's inequality, we have

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

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 $\mathsf{FHE.SK}$. To prevent this, we do need s to grow at least proportionally to $\sqrt{Q}$.

3.3.1 Attack for Deterministic Error In the argument for randomized error, the fact that $e'^i_{1,M}$ is randomized is crucial. However, as discussed in Section 1, we can extend the attack for the case of deterministic flooding as well, by exhibiting a secure signature scheme (with deterministic flooding) for which a variant of the attack can apply.

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 the bits after the kth bit, where k is the bit-length of signatures in Sig.

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

Above, 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. 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'. However, as discussed in Section 1, our attack can be generalized to work for this setting.

The Attack Now, consider using Sig' to instantiate our threshold signature scheme. Then, for any message M, the FHE 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 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 $\mathsf{FHE.sk}$.

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

Observe that the above equation is of the same form as Equation (3.3). 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 $\sqrt{Q}$.

4 Instantiating Threshold Signatures: Rejection-Free Lyubashevsky

Here, we provide an FHE friendly variant of Lyubashevsky's signature scheme from [47]. Our construction uses a hash function $H : \{0,1\}^* \to D_H := \{\mathbf{v} : \mathbf{v} \in \{-1, 0, 1\}^k; \|\mathbf{v}\|_1 \leq \alpha\}$, modeled as a random oracle. Here α is a parameter, typically much smaller than k. The signature scheme is described in Figure 2.

$\mathsf{KeyGen}(1^\lambda)$: Upon input the security parameter λ, set $q, n, m, \beta, k, d, \sigma$ such that $n = \Omega(\lambda)$ and the scheme is secure (see Theorem 4.1); then do the following:

1. Sample $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$ and $\mathbf{S} \leftarrow \{-d, \ldots, 0, \ldots, d\}^{m\times k}$.
2. Set $\mathbf{T} = \mathbf{AS}$.
3. Output $\mathsf{vk} = (\mathbf{A}, \mathbf{T})$, $\mathsf{sk} = \mathbf{S}$.

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

1. Sample $\mathbf{y} \leftarrow \mathcal{D}_{\mathbf{Z}^m,\sigma}$.
2. Set $\mathbf{c} = H(\mathbf{Ay}, \mu)$.
3. Set $\mathbf{z} = \mathbf{y} + \mathbf{Sc}$.
4. 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 $\gamma = (2\sigma + \alpha d)\sqrt{m}$.
2. Check if $H(\mathbf{Az} - \mathbf{Tc}, \mu) = \mathbf{c}$.
3. If both checks pass, then accept, else reject.

Fig. 2. Lyubashevsky's Signature Without Aborts

Correctness. Since $\mathbf{z} = \mathbf{y} + \mathbf{Sc}$, where $\mathbf{y} \leftarrow \mathcal{D}_{\mathbf{Z}^m,\sigma}$, we have $\|\mathbf{z}\| \leq 2\sigma\sqrt{m} + \|\mathbf{Sc}\|$ with probability $1 - 2^{-\Omega(\lambda)}$, using standard Gaussian tail bounds (see, e.g., Lemma 2.25). Since $\|\mathbf{S}\|_\infty \leq d$ and $\|\mathbf{c}\|_1 \leq \alpha$, we have

$\|\mathbf{Sc}\| \leq d\alpha\sqrt{m}$. This gives us $\|\mathbf{z}\| \leq (2\sigma + d\alpha)\sqrt{m}$ 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. We establish security via the following theorem.

Theorem 4.1. *Assume that $m > \lambda + (n \log q)/\log(2d+1)$, $\sigma \geq \alpha d\sqrt{mQ}$ where Q is the maximum number of signing queries an attacker can make and $|D_H| \geq 2^\lambda$. Assume further that $\mathsf{SIS}_{q,n,m,\beta}$ is hard for $\beta = 2\gamma + 2d\alpha\sqrt{m}$. Then the construction in Figure 2 satisfies* UF-CMA *in the random oracle model.*

Proof. We prove the security via the following hybrids:

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

Hybrid_1: In this hybrid the challenger responds to the signing query for any message μ as follows.

1. Sample $\mathbf{y} \leftarrow \mathcal{D}_{\mathbf{Z}^m,\sigma}$ as in the previous hybrid.
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_2: 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 $\mathbf{z} \leftarrow \mathcal{D}_{\mathbf{Z}^m,\sigma}$.
3. Set $H(\mathbf{Az} - \mathbf{Tc}, \mu) = \mathbf{c}$.
4. Output $(\mathbf{z}, \mathbf{c})$.

The only difference between Hybrid_0 and Hybrid_1 is that in Hybrid_1, the output value for H is chosen at random, and then programmed as the answer to $H(\mathbf{Ay}, \mu)$, without checking whether the value for $(\mathbf{Ay}, \mu)$ is already set, when a signing query for μ is made. By the definition of the random oracle, the two hybrids are identical if the same input $(\mathbf{Ay}, \mu)$ is not programmed twice throughout hash and sign queries, and forgery. The distinguishing advantage is therefore bounded as $Q(Q + Q_H + 1) \cdot 2^{-h}$, where h is the min-entropy of $\mathbf{Ay}$ for $\mathbf{y} \leftarrow \mathcal{D}_{\mathbf{Z}^m,\sigma}$. Standard arguments show that this is negligible.

The result now follows from the two claims below.

Claim 4.2 *If there is an adversary that makes at most Q signing queries and can win the game in Hybrid_1 with probability δ, then its probability of winning in Hybrid_2 is polynomial in δ, if $\sigma \geq \alpha d\sqrt{mQ}$.*

Proof. Wlog, we assume that the adversary makes exactly Q queries. The only difference between the two hybrids is in the value of $\mathbf{z}$. Let us refer to the joint distribution of all $(\mathbf{z}, \mathbf{c})$'s in Hybrid_1 as D_1 and that in Hybrid_2 as D_2. Note that the $\mathbf{c}_i$'s are sampled identically in both hybrids, and independently from all the rest. For $i \in [Q]$, the vector $\mathbf{z}_i$ is from distribution $Z_{1i} := \mathcal{D}_{\mathbf{Z}^m,\sigma,\mathbf{Sc}_i}$ in Hybrid_1 and from distribution $Z_{2i} = \mathcal{D}_{\mathbf{Z}^m,\sigma}$ in Hybrid_2. By Lemma 2.28, we have

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

Recall from the correctness proof that we have $\|\mathbf{Sc}_i\| \leq d\alpha\sqrt{m}$.

Let D_{1i} (resp. D_{2i}) be the distribution of $(\mathbf{z}_i, \mathbf{c}_i)$'s in Hybrid_1 (resp. Hybrid_2). As $\mathbf{c}_i$ is identically distributed in both games, we have, by using the multiplicativity property of Rényi divergence (Lemma 2.27):

$$R_a[D_{1i} \| D_{2i}] \leq 1 \cdot \max_{\mathbf{c}_i} R_a[Z_{1i} \| Z_{2i}] \leq \exp\left(a\pi \frac{(d\alpha\sqrt{m})^2}{\sigma^2}\right).$$

As $D_1 = (D_{1i})_i$ and $D_2 = (D_{2i})_i$, by using the multiplicativity property of the Rényi divergence once more, we get:

$$R_a(D_1 \| D_2) \leq \exp\left(a\pi \frac{Q(d\alpha\sqrt{m})^2}{\sigma^2}\right) \leq \exp(a\pi), \quad \text{for any } a \in (1, \infty). \tag{4.1}$$

Now, the view of the adversary in both hybrids includes the verification key vk, the queried messages M_i and the signature replies $(\mathbf{z}_i, \mathbf{c}_i)$ for $i \in [Q]$. As the distribution of vk is identical in both games and vk is sampled independently from all the rest, we may implicitly assume that it is fixed. As they are chosen adaptively, the μ_i's may depend on the previous queries and replies. But the dependence of the responses on the messages is broken by the random oracle (unlike in the proof of Claim 3.4). Hence, the $(\mathbf{c}_i, \mathbf{z}_i)$'s are independent of the μ_i's in both the hybrids. As the μ_i's are functions of the $(\mathbf{c}_i, \mathbf{z}_i)$'s, by the data processing inequality of the Rényi divergence (Lemma 2.27), we have

$$R_a(V_1 \| V_2) \leq R_a(D_1 \| D_2), \tag{4.2}$$

where V_1 (resp. V_2) is the adversary's view in Hybrid_1 (resp. Hybrid_2).

Let E denote the event that the adversary wins the game. Then by our assumption, we have $D_1(\mathsf{E}) = \delta$. From the probability preservation property (Lemma 2.27) of the Rényi divergence, we get:

$$V_2(\mathsf{E}) \geq \frac{\delta^{\frac{a}{a-1}}}{R_a(V_1 \| V_2)}, \text{ for any } a \in (1, \infty). \tag{4.3}$$

Using Equations (4.1), (4.2) and (4.3), we obtain that $V_2(\mathsf{E}) \geq \delta^{\frac{a}{a-1}} \exp(-a\pi)$. Taking any value of $a > 1$ provides the result.

Claim 4.3 *Let D_H be the range of the random oracle H. If there is a forger $\mathcal{F}$ that makes at most Q signing queries and Q_H random oracle queries, and succeeds in forging a valid signature with probability δ in Hybrid_2, 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 $\mathbf{A}\mathbf{v} = \mathbf{0}$, with probability at least*

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

This claim and its proof are identical to [47, Lemma 5.4]. Note that under the conditions of Theorem 4.1, the latter probability lower bound is $\geq \delta^2/(2(Q_H + Q)) - 2^{-\Omega(\lambda)}$.

Note that the condition $\sigma \geq \alpha d\sqrt{mQ}$ from Theorem 4.1 forces to set a modulus q and a SIS bound β that grow linearly with $\sqrt{Q}$. To ensure λ bits of security, one may choose n growing linearly with $\sqrt{Q}$. Overall, if using a Ring-SIS or Module-SIS instantiation, then the bit-length of the signature grows linearly with $n \log q$ and hence with $\log^2 Q$.

Next, we show that the flooding noise used in the above construction is essentially optimal by exhibiting an attack when the flooding noise is smaller.

4.1 Optimality of Flooding

In this section, we show that the flooding amount used in the construction in Figure 2 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.

4.1.1 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:

$$\mathbf{S}_{11} + \sum_{j=2}^{k} \mathbf{S}_{1j}\mathbf{c}_{ij} + \mathbf{y}_{i1} = \mathbf{z}_{i1} \tag{4.4}$$

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

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

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 support $[-(\alpha-1)d, (\alpha-1)d]$, then for some positive constant $t_1 < 1/2$, we have from Hoeffding's bound

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

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 4.4 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

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

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 $t_2 < (1/2 - t_1)$,

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

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 σ must grow proportional 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 [8]. Below, we describe our construction in the ROM, which satisfies partially adaptive unforgeability (Definition 2.6). We provide our construction in the standard model with *pre-processing* that satisfies fully adaptive unforgeability (Definition 2.5) in Section 6.

5.1 Partially Adaptive Unforgeability

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 3.

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

1. Randomly choose $K \leftarrow \{0,1\}^\lambda$ and N vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N \in \mathbb{Z}_q^N$ such that $\sum_{i=1}^N \mathbf{v}_i = \mathbf{0}$.
2. Generate $(\mathsf{Sig.vk}, \mathsf{Sig.sk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$ and $(\mathsf{FHE.pk}, \mathsf{FHE.sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda)$ and share $\mathsf{FHE.sk}$ into N shares as $(\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{Share}(\mathsf{FHE.sk})$ such that $\sum_{i=1}^N \mathsf{sk}_i = \mathsf{FHE.sk}$.
3. Compute an FHE encryption of the signing key as $\mathsf{CT}_{\mathsf{Sig.sk}} = \mathsf{FHE.Enc}(\mathsf{FHE.pk}, \mathsf{Sig.sk})$.
4. 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 $\mathsf{PartSign}$ circuit which depends on $(\mathsf{FHE.sk}, K, \mathbf{v}_i, \mathsf{hkey}_i)$.
5. Compute $(\pi_{\tau_i}, \pi_i) = \mathsf{HS.Sign}(\mathsf{HS.sk}, (\mathsf{sk}_i, K, \mathbf{v}_i, \mathsf{hkey}_i), \tau_i)$.
6. Output $\mathsf{TSig.pp} = (\mathsf{FHE.pk}, \mathsf{HS.pp}, \mathsf{HS.pk}, \mathsf{CT}_{\mathsf{Sig.sk}}, \{\tau_i, \pi_{\tau_i}\}_{i=1}^N)$, $\mathsf{TSig.vk} = \mathsf{Sig.vk}$, $\mathsf{TSig.sk} = \{\mathsf{TSig.sk}_i = (\mathsf{sk}_i, K, \mathbf{v}_i, \mathsf{hkey}_i, \pi_i)\}_{i=1}^N$.

$\mathsf{TS.PartSign}(\mathsf{TSig.pp}, \mathsf{TSig.sk}_i, M)$: Upon input the public parameters $\mathsf{TSig.pp}$, the key share $\mathsf{TSig.sk}_i = (\mathsf{sk}_i, K, \mathbf{v}_i, \mathsf{hkey}_i, \pi_i)$ 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 FHE encryption of signature σ_M as $\mathsf{CT}_{\sigma_M} = \mathsf{FHE.Eval}(\mathsf{FHE.pk}, \mathcal{C}_M, \mathsf{CT}_{\mathsf{Sig.sk}})$.
3. Compute $r_{i,M} = H(K, M)^T \mathbf{v}_i$ and $\sigma_{i,M} = \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{i,M} + e_i'$.
4. This step computes a homomorphic signature $\tilde{\pi}_{i,M}$ on partial signature $\sigma_{i,M}$ to provide robustness (Definition 2.8).
   Let $\mathcal{C}_{\mathsf{PS}}$ be the circuit to compute $\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + H(K, M)^T \mathbf{v}_i + e_i'$ in which CT_{σ_M} is hardcoded and the key share $\mathsf{TSig.sk}_i$ is given as the input. Compute $\pi_{i,M}^* = \mathsf{HS.SignEval}(\mathsf{HS.pp}, \mathcal{C}_{\mathsf{PS}}, \pi_{\tau_i}, (\mathsf{sk}_i, K, \mathbf{v}_i, \mathsf{hkey}_i), \pi_i)$ and $\tilde{\pi}_{i,M} = \mathsf{HS.Hide}(\mathsf{HS.pk}, \sigma_{i,M}, \pi_{i,M}^*)$.
5. Output $y_{i,M} = (\sigma_{i,M}, \tilde{\pi}_{i,M})$.

The algorithms $\mathsf{TS.PartSignVerify}$, $\mathsf{TS.Combine}$ and $\mathsf{TS.Verify}$ are identical to those in section 3.1.

Fig. 3. Partially Adaptive Threshold Signature Scheme.

5.2 Proof of Correctness

The correctness can be argued in the same way as that in Section 3.1. The $\mathsf{TS.Combine}$ algorithm outputs $\mathsf{FHE.decode}_1(\sum_{i=1}^N \sigma_{i,M})$, where $\sigma_{i,M} = \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + e_i' + r_{i,M}$. First observe that $\sum_{i=1}^N r_{i,M} = \sum_{i=1}^N H(K, M)^T \mathbf{v}_i = H(K, M)^T \sum_{i=1}^N \mathbf{v}_i = 0$, because $\sum_{i=1}^N \mathbf{v}_i = \mathbf{0}$. Hence, $\mathsf{FHE.decode}_1(\sum_{i=1}^N \sigma_{i,M}) = \mathsf{FHE.decode}_1((\sum_{i=1}^N \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + e_i') + \sum_{i=1}^N r_{i,M}) = \mathsf{FHE.decode}_1(\sum_{i=1}^N (\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + e_i') + \mathbf{0}) = \sigma_M$, where the last equality can be derived in the same way as in Section 3.1.

5.3 Unforgeability

We prove that the construction in Figure 3 satisfy partially adaptive unforgeability via the following theorem.

Theorem 5.1. *Assume the signature scheme* Sig *satisfies unforgeability,* FHE *is a semantically secure fully homomorphic encryption scheme (Definition 2.12),* HS *is context hiding homomorphic signature scheme (Definition 2.18) and* Share *satisfies privacy (Definition 2.34). Then the* TS *construction in Figure 3 satisfies*

partially adaptive unforgeability (Definition 2.6) 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 theorem can be proved using the following hybrids:

Hybrid_0 and Hybrid_1 are the same as that in the proof of Theorem 3.1.

Hybrid_2: Same as Hybrid_1, except that the randomness u_i used in sampling e'_i in $\sigma_{i,M}$ 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 $\mathcal{A}$ 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 each M in pre corruption signing query is set in the reverse order, i.e., firstly partial signatures are computed and then $H(K, M)$ is set accordingly as follows:

1. The challenger computes $\mathsf{CT}_{\sigma_M} = \mathsf{FHE.Eval}(\mathsf{FHE.pk}, \mathcal{C}_M, \mathsf{CT}_{\mathsf{Sig.sk}})$.
2. It then computes $\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M})$ and generates N shares as $\{s_{i,M}\}_{i=1}^N \leftarrow \mathsf{Share}(\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}))$.
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 $\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M})$ again.
4. When the adversary $\mathcal{A}$ outputs the set S of corrupted parties, the challenger first programs 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 $\mathcal{A}$.
   - Programming $H(K, M)$: $\forall i \in [N]$, compute $r_{i,M} = s_{i,M} - \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M})$ and solve for vector $\mathbf{b}_M \in \mathbb{Z}_q^N$ such that $\forall\ i \in [N], \mathbf{b}_M^T \mathbf{v}_i = r_{i,M}$. Set $H(K, M) = \mathbf{b}_M$. Note that since there are $N-1$ independent equations in N unknowns, such a $\mathbf{b}_M$ 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 $\mathcal{A}$ 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 $r_{i,M} = s_{i,M} - \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M})$. Computes $\mathbf{b}_M$ such that $\forall\ i \in S,\ \mathbf{b}_M^T \mathbf{v}_i = r_{i,M}$. Sets $H(K, M) = \mathbf{b}_M$. Such a $\mathbf{b}_M$ exists and can be computed since there are only $N-1$ equations to satisfy 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 honest party be P_a, i.e. $S = [N] \setminus \{a\}$.

1. Computes $\mathsf{CT}_{\sigma_M} = \mathsf{FHE.Eval}(\mathsf{FHE.pk}, \mathcal{C}_M, \mathsf{CT}_{\mathsf{Sig.sk}})$ and $\sigma_M = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, M)$.
2. For each $i \in S$, computes $\sigma'_{i,M} = \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + H(K, M)^T \mathbf{v}_i$ and $\sigma_{i,M} = \sigma'_{i,M} + e'_i$, where $e'_i \leftarrow \mathcal{D}_s$.
3. Returns $\sigma_{a,M} = \sigma_M \lfloor q/2 \rceil - \sum_{i \in S} \sigma'_{i,M} + e'_a$, where $e'_a \leftarrow \mathcal{D}_s$.

Hybrid_5 and Hybrid_6: are the same as Hybrid_4 and Hybrid_5, respectively, defined in the proof of Theorem 3.1.

Indistinguishability of Hybrids Now we show that the 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 a random oracle the two hybrids are indistinguishable in adversary's view.

Claim 5.4 Hybrid_2 *and* Hybrid_3 *are statistically indistinguishable.*

Proof. The two hybrids differ only in the order in which $H(K, M)$ and $r_{i,M} = H(K, M)^T \mathbf{v}_i$ 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 $H(K, M)^T \mathbf{v}_i = r_{i,M}$ is satisfied for each $i \in S$. Next we show that this change in order of computation does not change adversary's view.

Let P_a be the honest party. Then, observe that the adversary receives the following values: $H(K, M)$, $\{\mathsf{sk}_i\}_{i\in[N]\setminus\{a\}}, \{\mathbf{v}_i\}_{i=1}^N$ and $\{\sigma_{i,M}\}_{i=1}^N$ in the two hybrids. We show that their joint distribution in the two hybrids is indistinguishable.

Firstly, consider $\sigma_{i,M}^{(2)}$ and $\sigma_{i,M}^{(3)}$, where superscripts indicate the respective hybrids. Recall that $\sigma_{i,M}^{(2)} = s_{i,M}^{(2)} + e'_{i,M}$ and $\sigma_{i,M}^{(3)} = s_{i,M}^{(3)} + e'_{i,M}$ where the added noise is sampled from the same distribution in both the hybrids. Hence we focus on $s_{i,M}^{(2)}$ and $s_{i,M}^{(3)}$. $s_{i,M}^{(2)} = \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + H(K, M)^{\mathsf{T}} \mathbf{v}_i$. Recall that $\{\mathsf{sk}_i\}_{i\in[N]}$ are random secret shares of $\mathsf{FHE.sk}$ and $\{\mathbf{v}_i\}_{i\in[N]}$ are random secret shares of $\mathbf{0}$. Hence, by linearity property of Share, $\{\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M})\}_{i\in[N]}$ are secret shares of $\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M})$, $\{H(K, M)^{\mathsf{T}} \mathbf{v}_i\}_{i\in[N]}$ are secret shares of 0 and hence $\{\mathsf{s}_{i,M}^{(2)}\}_{i\in[N]}$ are secret shares of $\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M})$. Moreover, since H is modeled as a random oracle, we have that $\{H(K, M)^{\mathsf{T}} \mathbf{v}_i\}_{i\in[N]}$ is a random secret sharing of 0, due to which $s_{i,M}^{(2)}$ are a random secret sharing of $\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M})$.

On the other hand, $\{s_{i,M}^{(3)}\}_{i\in[N]}$ are also random secret shares of $\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M})$, by design. Hence, they have the same distribution.

Next, observe that the adversary has sk_i for $i \in [N] \setminus \{a\}$ and hence it can compute $r_{i,M} = s_{i,M} - \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M})$ which is supposed to be equal to $H(K, M)^{\mathsf{T}} \mathbf{v}_i$ for $i \in [N] \setminus \{a\}$. Thus, given $\mathbf{v}_i$, sk_i, $s_{i,M}$ for $i \in [N] \setminus \{a\}$, $H(K, M)$ is a random vector from the set $\{\mathbf{b} : \mathbf{b}^{\mathsf{T}} \mathbf{v}_i = s_{i,M} - \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) \; \forall \; i \in [N] \setminus \{a\}\}$. Again, the same is true for $H(K, M)$ in Hybrid_3 by design. Hence, the joint distribution of adversary's view in the two hybrids is indistinguishable.

Finally, 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* $\mathrm{poly}(\lambda) \cdot \sqrt{Q}$. *Then if there is an adversary that can win the unforgeability game in* Hybrid_3 *with probability* ϵ, *then its probability of winning the game in* Hybrid_4 *is at least* $\epsilon^2/2$.

Proof. Let the adversary makes Q signing queries and let theP_a be the honest party. Then in the adversary's view the two hybrids differ only in the error term in $\sigma_{a,M}$, as shown below.

Let $e'_a \leftarrow \mathcal{D}_s$. In Hybrid_3, for pre-corruption queries, the partial signature $\sigma_{a,M}$ is computed as follows:

$$\begin{aligned}
\sigma_{a,M} &= s_{a,M} + e'_a \\
&= \sum_{i=1}^{N} s_{i,M} - \sum_{i\in[N]\setminus\{a\}} s_{i,M} + e'_a \\
&= \mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) - \sum_{i\in[N]\setminus\{a\}} s_{i,M} + e'_a \\
&= \mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) - \sum_{i\in[N]\setminus\{a\}} (\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + H(K,M)^{\mathsf{T}}\mathbf{v}_i) + e'_a \\
&= \mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) - \sum_{i=1}^{N} \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) \\
&\quad + \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) - \sum_{i=1}^{N} H(K,M)^{T}\mathbf{v}_i + H(K,M)^{\mathsf{T}}\mathbf{v}_a + e'_a \\
&= \mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) - \mathsf{FHE.decode}_0(\sum_{i=1}^{N} \mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) \\
&\quad + \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) - \sum_{i=1}^{N} H(K,M)^{T}\mathbf{v}_i + H(K,M)^{T}\mathbf{v}_a + e'_a \\
&= \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) + H(K,M)^{T}\mathbf{v}_a + e'_a \quad (\because \sum_{i=1}^{N} \mathbf{v}_i = 0;\ \sum_{i=1}^{N} \mathsf{sk}_i = \mathsf{FHE.sk})
\end{aligned}$$

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

$$\mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) + H(K,M)^{\mathsf{T}}\mathbf{v}_a + e'_a$$

In Hybrid_4, for pre-corruption queries, we have

$$\begin{aligned}
\sigma_{a,M} &= s_{a,M} + e'_a \\
&= \sigma_M \lfloor q/2 \rceil - \sum_{i\in[N]\setminus\{a\}} s_{i,M} + e'_a \\
&= \sigma_M \lfloor q/2 \rceil - \sum_{i\in[N]\setminus\{a\}} (\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + H(K,M)^{T}\mathbf{v}_i) + e'_a \\
&= \sigma_M \lfloor q/2 \rceil - \sum_{i=1}^{N} \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) - \sum_{i=1}^{N} H(K,M)^{T}\mathbf{v}_i \\
&\quad + \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) + H(K,M)^{T}\mathbf{v}_a + e'_a \\
&= \sigma_M \lfloor q/2 \rceil - \mathsf{FHE.decode}_0(\sum_{i=1}^{N} \mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) - H(K,M)^{T} \sum_{i=1}^{N} \mathbf{v}_i \\
&\quad + \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) + H(K,M)^{T}\mathbf{v}_a + e'_a \\
&= \sigma_M \lfloor q/2 \rceil - \mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) + \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) \\
&\quad + H(K,M)^{T}\mathbf{v}_a + e'_a \\
&= \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma}) + H(K,M)^{T}\mathbf{v}_a + e + e'_a
\end{aligned}$$

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

$$\begin{aligned}
\sigma_{a,M} &= s_{a,M} + e'_a \\
&= \sigma_M.\lfloor q/2 \rceil - \sum_{i\in[N]\setminus\{a\}} (\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + H(K,M)^T\mathbf{v}_i) + e'_a \\
&= \sigma_M.\lfloor q/2 \rceil - \sum_{i=1}^{N} (\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + H(K,M)^T\mathbf{v}_i) \\
&\quad + \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) + H(K,M)^T\mathbf{v}_a + e'_a \\
&= \sigma_M.\lfloor q/2 \rceil - \mathsf{FHE.decode}_0(\sum_{i=1}^{N} \mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + H(K,M)^T \sum_{i=1}^{N} \mathbf{v}_i \\
&\quad + \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) + H(K,M)^T\mathbf{v}_a + e'_a \\
&= \sigma_M.\lfloor q/2 \rceil - \mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) + \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) \\
&\quad + H(K,M)^T\mathbf{v}_a + e'_a \\
&= \mathsf{FHE.decode}_0(\mathsf{sk}_a, \mathsf{CT}_{\sigma_M}) + H(K,M)^T\mathbf{v}_a + (e'_a + e).
\end{aligned}$$

Thus, the difference in the two hybrids is in the error terms in $\sigma_{a,M}$. In Hybrid_3, the error is e'_a, while in Hybrid_4, it is $e'_a + e$. This is the same case as in Claim 3.4 in Section 3. Hence we can use exactly the same analysis using Rényi Divergence as in the proof of Claim 3.4, to complete the proof.

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, $\{\mathsf{sk}_i\}_{i\in[N]} \leftarrow \mathsf{Share}(\mathsf{FHE.sk})$, 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* FHE *is semantically secure. Then* Hybrid_5 *and* Hybrid_6 *are computationally indistinguishable.*

Proof. The proof is via standard reduction to FHE security and is similar to the proof of Claim 3.6.

Finally the proof for Theorem 5.1 completes with the following claim.

Claim 5.8 *If the underlying signature scheme* Sig *is unforgeable, then the advantage of the adversary in the unforgeability game of Definition 2.6 is negligible in* Hybrid_6.

Proof. The proof is via standard reduction to Sig security and is similar to the proof of Claim 3.7.

5.4 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 that for Claim 3.8.

6 Fully Adaptive Unforgeability in the Preprocessing Model

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 [29] and references therein).

6.1 Construction

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 $j, \sum_{i=1}^{N} r_{i,j} = 0$. 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. For a stateless scheme, we use a collision resistant hash function $H\{0,1\}^* \to [Q]$, which maps a message to an index in $[Q]$. We also use $H_1 : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^r$ modelled as random oracle as also used in the partially adaptive construction. The construction is provided in Figure 4.

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

1. For $j = 1$ to Q, generate random values $\{r_{ij}\}_{i=1}^{N} \leftarrow \mathsf{Share}(0)$, such that $\sum_{i=1}^{N} r_{ij} = 0$.
2. Generate $(\mathsf{Sig.vk}, \mathsf{Sig.sk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$, $(\mathsf{FHE.pk}, \mathsf{FHE.sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda)$ and N shares of $\mathsf{FHE.sk}$ as $(\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{Share}(\mathsf{FHE.sk})$ such that $\sum_{i=1}^{N} \mathsf{sk}_i = \mathsf{FHE.sk}$.
3. Compute an FHE encryption of $\mathsf{Sig.sk}$ as $\mathsf{CT}_{\mathsf{Sig.sk}} \leftarrow \mathsf{FHE.Enc}(\mathsf{FHE.pk}, \mathsf{Sig.sk})$.
4. For each P_i, sample a tag $\tau_i \leftarrow \{0,1\}^*$ 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 the 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 $\mathsf{PartSign}$ circuit which depends on $(\mathsf{FHE.sk}, r_{ij}, \mathsf{hkey}_i)$.
5. 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 $i \in [N]$.
6. Output $\mathsf{TSig.pp} = (\mathsf{FHE.pk}, \mathsf{HS.pp}, \mathsf{HS.pk}, \mathsf{CT}_{\mathsf{Sig.sk}}, \{\tau_i, \pi_{\tau_i}\}_{i=1}^{N})$, $\mathsf{TSig.vk} = \mathsf{Sig.vk}$, $\mathsf{TSig.sk} = \{\mathsf{TSig.sk}_i = (\mathsf{sk}_i, \{r_{ij}\}_{j\in[Q]}, \mathsf{hkey}_i, \pi_i)\}_{i=1}^{N}$.

$\mathsf{TS.PartSign}(\mathsf{TSig.pp}, \mathsf{TSig.sk}_i, M)$: Upon input the public parameters $\mathsf{TSig.pp}$, 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 $\mathsf{Sig.Sign}$ circuit with message M being hardcoded. Compute FHE encryption of signature σ_M as $\mathsf{CT}_{\sigma_M} \leftarrow \mathsf{FHE.Eval}(\mathsf{FHE.pk}, \mathcal{C}_M, \mathsf{CT}_{\mathsf{Sig.sk}})$.
3. Compute $\sigma_{i,M} = \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{ij} + e_i'$.
4. This step computes a homomorphic signature $\tilde{\pi}_{i,M}$ on partial signature $\sigma_{i,M}$ to provide robustness (Definition 2.8).
   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 $\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{ij} + e_i'$.
   - Compute $\pi^*_{i,M} = \mathsf{HS.SignEval}(\mathsf{HS.pp}, \mathcal{C}_{\mathsf{PS}}, \pi_{\tau_i}, (\mathsf{sk}_i, r_{ij}, \mathsf{hkey}_i), \pi_i)$.
   - 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 $\mathsf{CT}_{\sigma_M} = \mathsf{FHE.Eval}(\mathsf{FHE.pk}, \mathcal{C}_M, \mathsf{CT}_{\mathsf{Sig.sk}})$.
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 $\sigma_M = \mathsf{FHE.decode}_1(\sum_{i=1}^{N} \sigma_{i,M})$.

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

Fig. 4. Fully Adaptive Threshold Signature Scheme.

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

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

6.2 Unforgeability

Theorem 6.1. *Assume the signature scheme* Sig *satisfies unforgeability,* FHE *is a CPA secure fully homomorphic encryption scheme (Definition 2.12),* HS *is context hiding homomorphic signature scheme (Definition 2.18) and* Share *satisfies privacy (Definition 2.34). Then the construction in Figure 4 satisfies adaptive unforgeability (Definition 2.5) 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 $\sigma_{i,M}$, the challenger simulates $\tilde{\pi}_{i,M}$ as $\tilde{\pi}_{i,M} = \mathsf{HS.Sim}(\mathsf{HS.sk}, \alpha, \sigma_{i,M}, \tau_i, \pi_{\tau_i})$, 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 $r_{i,H(M)}$ as defined below. 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 $i \in [N] \setminus S_M$, where $\sigma'_{i,M}$ is the same value as in the earlier response, but e'_i is sampled afresh.
2. Else, computes $\mathsf{CT}_{\sigma_M} = \mathsf{FHE.Eval}(\mathsf{FHE.pk}, \mathcal{C}_M, \mathsf{Sig.sk})$ and does the following:
   - For each $i \in S_M$, computes $\sigma'_{i,M} = \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{i,H(M)}$ and sets $\sigma_{i,M} = \sigma'_{i,M} + e'_i$.
   - For $i \in [N] \setminus S_M$,
     - Divide $\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) - \sum_{k \in S_M} \sigma'_{k,M}$ into $N - |S_M|$ random shares, $\{s_{i,M}\}_{i \in [N] \setminus S_M}$.
     - Set $\sigma'_{i,M} = s_{i,M}$ and $\sigma_{i,M} = \sigma'_{i,M} + e'_i$.
3. Return $\{\sigma_{i,M}\}_{i \in [N] \setminus S_M}$ to the adversary.

When the adversary makes a key query for some $i \in [N]$, the challenger does the following.

1. For each $M \in PreQ_i$, computes $r_{i,H(M)} = s_{i,M} - \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M})$.

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 signatures as in Hybrid_3, but for $i \in [N] \setminus S_M$ the challenger simulates the signatures as follows: instead of sharing $\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) - \sum_{i \in S_M} \sigma'_{i,M}$, the challenger now shares $\sigma_M \lfloor q/2 \rceil - \sum_{i \in S_M} \sigma'_{i,M}$ between the uncorrupted parties as described below.

1. Compute $\mathsf{CT}_{\sigma_M} = \mathsf{FHE.Eval}(\mathsf{FHE.pk}, \mathcal{C}_M, \mathsf{CT}_{\mathsf{Sig.sk}})$ and $\sigma_M = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, M)$.
2. For each $i \in S_M$, compute $\sigma'_{i,M} = \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{i,H(M)}$ and $\sigma_{i,M} = \sigma'_{i,M} + e'_i$, where $e'_i \leftarrow \mathcal{D}_s$.
3. For uncorrupted parties do the following: divide $\sigma_M \lfloor q/2 \rceil - \sum_{k \in S_M} \sigma'_{k,M}$ into $N - |S_M|$ random shares, $\{s_{i,M}\}_{i \in [N] \setminus S_M}$. Set $\sigma_{i,M} = s_{i,M} + e'_i$ for $i \in [N] \setminus S_M$.
4. Return $\sigma_{i,M}$ for $i \in [N] \setminus S_M$.

Key queries are answered in the same way as in the previous hybrid.

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

Hybrid_6: Same as Hybrid_5, except that now $\mathsf{CT}_{\mathsf{Sig.sk}}$ in $\mathsf{TSig.pp}$ is replaced by $\mathsf{CT}_{\mathbf{0}} = \mathsf{FHE.Enc}(\mathsf{FHE.pk}, \mathbf{0})$.

6.2.1 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 that in the proof of Theorem 5.1 in Section 5.

Claim 6.2 Hybrid_2 *and* Hybrid_3 *are statistically indistinguishable*

Proof. Observe that the two hybrids differ only in the way the $\{r_{i,j}\}$ shares are set. For any message M, in Hybrid_2, $\{r_{i,j}\}_{i \in [N]} \leftarrow \mathsf{Share}(0)$, where $j = H(M)$. In Hybrid_3, $\{r_{i,j}\}_{i \in S_M}$ are randomly chosen and for $i \in [N] \setminus S_M$, $r_{i,j} = s_{i,j} - \mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M})$, where $\{s_{i,j}\}_{i \in [N] \setminus S_M}$ are obtained by randomly sharing $\mathsf{FHE.decode}_0(\mathsf{FHE.sk}, \mathsf{CT}_{\sigma_M}) - \sum_{k \in S_M} (\mathsf{FHE.decode}_0(\mathsf{sk}_k, \mathsf{CT}_{\sigma_M}) + r_{k,j})$ into $N - |S_M|$ shares. Thus,

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

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 6.3 *Assume that the flooding error is of the order* $\mathrm{poly}(\lambda) \cdot \sqrt{Q}$. *If there is an adversary who can win the unforgeability game as per Definition 2.5 in* Hybrid_3 *with probability* ϵ, *then its probability of winning the game in* Hybrid_4 *is at least* $\epsilon^2/2$.

Proof. Let the adversary issues Q signing queries. Let P_a be the honest party for some $a \in [N]$. Then the two hybrids differ only in the error term in $\sigma_{a,M}$, 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 $e'_a \leftarrow \mathcal{D}_s$. Then,

In Hybrid_4, we have:

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

On the other hand in Hybrid_3,

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

In the third step, $\sigma'_{i,M}$ is computed as $\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{ij}$, while $s_{i,M}$ is replaced with $\mathsf{FHE.decode}_0(\mathsf{sk}_i, \mathsf{CT}_{\sigma_M}) + r_{ij}$ because of the way $r_{i,j}$ is set. In the last step we replace $-\sum_{i \in [N] \setminus \{a\}} r_{ij}$ by r_{aj} because $\sum_{i \in [N]} r_{ij} = 0$. However note that r_{aj} is never actually set since $\mathsf{TSig.sk}_a$ is never queried for.

Thus, the difference in the two hybrids is in the error terms in σ_a. In Hybrid_3, the error is e'_a, while in Hybrid_4, it is $e'_a + e$. This is the same case as in Claim 3.4 in Section 3. Hence we can use exactly the same analysis using Rényi Divergence as in the proof of Claim 3.4, to complete the proof.

Indistinguishability between $\mathsf{Hybrid}_4, \mathsf{Hybrid}_5$ and Hybrid_6 has the same argument as that for indistinguishability between $\mathsf{Hybrid}_4, \mathsf{Hybrid}_5$ and Hybrid_6 in the proof of Theorem 5.1.

Finally the proof of Theorem 6.1 completes with the following claim.

Claim 6.4 *If the underlying signature scheme* Sig *is unforgeable, then the advantage of the adversary in the unforgeability game of Definition 2.5 is negligible in* Hybrid_6.

Proof. The proof is via standard reduction to Sig security and is similar to the proof of Claim 3.7.

6.3 Robustness

Claim 6.5 *If* HS *is multi data secure homomorphic signature, then the above construction of* TS *satisfies robustness.*

Proof. The proof is same as the proof for Claim 3.8.

7 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.

7.1 Construction

The construction uses following building blocks:

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

To keep the construction simple, we have omitted the steps required for robustness. The robustness can be achieved using homomorphic signatures in the same way as in the previous constructions. The construction is provided in Figure 5.

7.2 Correctness

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

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

$\mathsf{TS.KeyGen}(1^\lambda, t)$: 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 FHE scheme $(\mathsf{FHE.pk}, \mathsf{FHE.sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda)$ and compute $\mathsf{CT}_{\mathsf{Sig.sk}} = \mathsf{FHE.Enc}(\mathsf{FHE.pk}, \mathsf{Sig.sk})$.
3. Share the FHE secret key as: $\{\mathsf{TSig.sk}_i\}_{i=1}^N \leftarrow \mathsf{Share}(\mathsf{FHE.sk}, t)$. 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 (Def. 2.35) 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 $\mathsf{FHE.sk}$.
4. Output $\mathsf{TSig.pp} = \{\mathsf{FHE.pk}, \mathsf{CT}_{\mathsf{Sig.sk}}\}$, $\mathsf{TSig.vk} = \mathsf{Sig.vk}$, $\mathsf{TSig.sk} = \{\mathsf{TSig.sk}_i\}_{i=1}^N$.

$\mathsf{TS.PartSign}(\mathsf{TSig.pp}, \mathsf{TSig.sk}_i, M)$: 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. Compute $\mathsf{CT}_\sigma = \mathsf{FHE.Eval}(\mathsf{FHE.pk}, \mathcal{C}_M, \mathsf{CT}_{\mathsf{Sig.sk}})$.
2. Output $\sigma_i = \{\hat{\sigma}_j\}_{j \in T_i}$, where $\hat{\sigma}_j = \mathsf{FHE.decode}_0(\mathsf{sk}_j, \mathsf{CT}_\sigma) + e'_j$, 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 $\sigma_i = \{\hat{\sigma}_j\}_{j \in T_i}$ and $S \subseteq [N]$, the Combine algorithm first checks if $|S| \geq t$. If not, then outputs $\perp$, else computes a minimum valid share set (Def. 2.36) $T \subseteq \bigcup_{i \in S} T_i$ and outputs

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

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

Fig. 5. t-out-of$N$ Threshold Signature Scheme.

7.3 Unforgeability

Theorem 7.1. *Assume* FHE *is a secure fully homomorphic encryption as per Definition 2.12,* Share *is a* $\{0,1\}$-LSSS *t-out-of-N secret sharing scheme that satisfies Definition 2.34 and* Sig *is a signature scheme that satisfies (UF-CMA) unforgeability. Then the construction of threshold signature in Figure 5 satisfies selective unforgeability (Definition 2.7).*

Proof. We prove the theorem using following hybrids.

Hybrid_0 : Same as the real world.

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

1. Upon 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 answer any partial signing query (M, i) for message M and 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 $\hat{\sigma}_j = \mathsf{FHE.decode}_0(\mathsf{sk}_j, \mathsf{CT}_\sigma) + e'_j$, 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.) and $\sigma_M = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, M)$. Then it computes $\hat{\sigma}_j$ as
   $$\hat{\sigma}_j = \lfloor q/2 \rceil \cdot \sigma_M - \sum_{j' \in T \setminus \{j\}} \mathsf{FHE.decode}_0(\mathsf{sk}_{j'}, \mathsf{CT}_\sigma) + e'_j.$$

Hybrid_2: Same as Hybrid_1, except that instead of sharing $\mathsf{FHE.sk}$ the challenger now shares $\mathbf{0}$ to compute 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 $\mathsf{CT}_{\mathbf{0}} = \mathsf{FHE.Enc}(\mathsf{FHE.pk}, \mathbf{0})$.

7.3.1 Indistinguishability of Hybrids

Next, we show that consecutive hybrids are indistinguishable.

Claim 7.2 *If the flooding error is of the size* $\mathrm{poly}(\lambda)\sqrt{Q}$*, then if there is an adversary who can win the unforgeability game in* Hybrid_0 *with probability* ϵ*, then its probability of winning the game 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:

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

In Hybrid_1, we have:

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

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 uses Rényi Divergence and is similar to the proof given for Claim 3.4. We refer to the distribution to be considered in Hybrid_1 and Hybrid_2 by D_1 and D_2, respectively. Let E_k be the random variable for error vector corresponding to the error terms in ciphertexts returned in response to the k-th query. Let Y_k be the random variable for k-th query which is of the form (M_k, i_k). Then as we discussed in proof of Claim 3.4, $D_t = (\mathcal{E}_Q^{(t)}, \mathcal{M}_Q^{(t)}, \mathcal{E}_{Q-1}^{(t)}, \mathcal{M}_{Q-1}^{(t)}, \ldots, \mathcal{E}_1^{(t)}, \mathcal{M}_1^{(t)})$ for $t \in \{1, 2\}$, where $\mathcal{E}_k^{(t)}$ is the distribution of E_k in Hybrid_t and $\mathcal{M}_k^{(t)}$ is the distribution of Y_k in Hybrid_t. From the above analysis, we have that for any given query (M_k, i_k), $\mathcal{E}_k^{(1)} = \mathcal{D}_{\mathbb{Z}^{n_k}, s, \mathbf{0}}$ and $\mathcal{E}_k^{(2)} = \mathcal{D}_{\mathbb{Z}^{n_k}, s, \mathbf{e}_k}$, where $\mathbf{e}_k = (e_k, \ldots, e_k)$ is a vector of length $n_k = |T_{i_k} \setminus T^*|$ and $|e_k| \leq B_{eval}$ and $n_k \leq \ell$, where ℓ is the number of rows in the share matrix (see Definition 2.35) and is bounded by $\mathrm{poly}(N)$. Then,

$$R_a(D_1 \| D_2) = R_a(\mathcal{E}_Q^{(1)}, \mathcal{M}_Q^{(1)}, \ldots, \mathcal{E}_1^{(1)}, \mathcal{M}_1^{(1)} \parallel \mathcal{E}_Q^{(2)}, \mathcal{M}_Q^{(2)}, \ldots, \mathcal{E}_1^{(2)}, \mathcal{M}_1^{(2)}). \tag{7.1}$$

Applying the multiplicativity property of the Rényi divergence (Lemma 2.27), we obtain that $R_a(D_1 \| D_1)$ is bounded from above by

$$\begin{aligned}
&\max_{x \in X} R_a(\mathcal{E}_Q^{(1)} | X = x \parallel \mathcal{E}_Q^{(2)} | X = x) \cdot R_a(\mathcal{M}_Q^{(1)}, \ldots, \mathcal{E}_1^{(1)}, \mathcal{M}_1^{(1)} \parallel \mathcal{M}_Q^{(2)}, \ldots, \mathcal{E}_1^{(2)}, \mathcal{M}_1^{(2)}) \\
= &\max_{x \in X} R_a(\mathcal{D}_{\mathbb{Z}^{n_Q}, s, \mathbf{0}} | X = x \parallel \mathcal{D}_{\mathbb{Z}^{n_Q}, s, \mathbf{e}_Q} | X = x) \cdot R_a(\mathcal{M}_Q^{(1)}, \ldots, \mathcal{E}_1^{(1)}, \mathcal{M}_1^{(1)} \parallel \mathcal{M}_Q^{(2)}, \ldots, \mathcal{E}_1^{(2)}, \mathcal{M}_1^{(2)}),
\end{aligned} \tag{7.2}$$

where $X = (Y_Q, E_{Q-1}, \ldots, E_1)$.

Then applying Lemma 2.28 in Equation (7.2), we get

$$\begin{aligned} R_a(D_1\|D_2) &\leq \max_{x\in X} \ \exp(a\pi\|\mathbf{e}_Q\|^2/s^2)\cdot R_a(\mathcal{M}_Q^{(1)},\ldots,\mathcal{E}_1^{(1)},\mathcal{M}_1^{(1)} \parallel \mathcal{M}_Q^{(2)},\ldots,\mathcal{E}_1^{(2)},\mathcal{M}_1^{(2)}) \\ &= \max_{x\in X} \ \exp(a\pi n_Q e_Q^2/s^2)\cdot R_a(\mathcal{M}_Q^{(1)},\ldots,\mathcal{E}_1^{(1)},\mathcal{M}_1^{(1)} \parallel \mathcal{M}_Q^{(2)},\ldots,\mathcal{E}_1^{(2)},\mathcal{M}_1^{(2)}) \\ &\leq \exp(a\pi\ell B_{eval}^2/s^2)\cdot R_a(\mathcal{M}_Q^{(1)},\ldots,\mathcal{E}_1^{(1)},\mathcal{M}_1^{(1)} \parallel \mathcal{M}_Q^{(2)},\ldots,\mathcal{E}_1^{(2)},\mathcal{M}_1^{(2)}). \end{aligned}$$

Further, from the data processing inequality (Lemma 2.27),

$$\begin{aligned} R_a(\mathcal{M}_Q^{(1)},\mathcal{E}_{Q-1}^{(1)},\ldots,\mathcal{E}_1^{(1)},\mathcal{M}_1^{(1)} \parallel \mathcal{M}_Q^{(2)},\mathcal{E}_{Q-1}^{(2)},\ldots,\mathcal{E}_1^{(2)},\mathcal{M}_1^{(2)}) \\ \leq R_a(\mathcal{E}_{Q-1}^{(1)},\ldots,\mathcal{E}_1^{(1)},\mathcal{M}_1^{(1)} \parallel \mathcal{E}_{Q-1}^{(2)},\ldots,\mathcal{E}_1^{(2)},\mathcal{M}_1^{(2)}), \end{aligned}$$

Hence, we get

$$\begin{aligned} R_a(D_1\|D_2) &\leq \exp(a\pi\ell B_{eval}^2/s^2)\cdot R_a(\mathcal{E}_{Q-1}^{(2)},\ldots,\mathcal{E}_1^{(2)},\mathcal{M}_1^{(2)} \parallel \mathcal{E}_{Q-1}^{(3)},\ldots,\mathcal{E}_1^{(3)},\mathcal{M}_1^{(3)}) \\ &\leq \exp(\frac{a\pi Q\ell B_{eval}^2}{s^2}), \end{aligned}$$

where the second inequality follows from induction.

Setting $s = B_{eval}\cdot\sqrt{\ell\cdot Q\lambda}$, we get

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

Therefore,

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

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 7.3 *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, $\{\mathsf{TSig.sk}_i\}_{i\in[N]} = \mathsf{Share}(\mathsf{FHE.sk}, t)$, 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 2.34) of Share the two hybrids are identical in the adversary's view since it receives the key shares only for an invalid set of participants.

Claim 7.4 *Assume the* FHE *scheme is secure (Definition 2.12). Then* Hybrid_2 *and* Hybrid_3 *are indistinguishable.*

Proof. The proof is similar to the proof of Claim 3.6.

Finally the proof of Theorem 7.1 completes with the following claim. The proof of the claim is similar to the proof of Claim 3.7 and is omitted.

Claim 7.5 *If the underlying signature scheme* Sig *is unforgeable, then the the adversary cannot win the unforgeability game in* Hybrid_3 *with non-negligible advantage.*

7.4 Robustness

To add robustness in the above scheme, we can use context hiding secure homomorphic signature in the same way as in Section 3. 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.

7.5 Construction for adaptive unforgeability

The above construction of t-out-of-N threshold signatures can be made partially adaptive unforgeable in the same way as in Section 5. In particular, for t-out-of-N access structure, the random shares of $\mathbf{0}$, i.e. vector $\mathbf{v}_i$ in $\mathsf{TSig.sk}_i$ for $i \in [N]$ are now computed as $\{\mathbf{v}_i\}_{i\in[N]} \leftarrow \mathsf{Share}(\mathbf{0}, t)$. Similarly for fully adaptive unforgeablity the construction can be modified in the same way as in Section 6. In particular, for t-out-of-N adaptation, the random shares of 0 in $\mathsf{TSig.sk}_i$ are now generated as: $\forall\ j \in [Q], \{r_{i,j}\}_{i\in[N]} \leftarrow \mathsf{Share}(0, t)$.

Acknowledgements This work was partly supported by the DST "Swarnajayanti" fellowship, National Blockchain Project, European Union Horizon 2020 Research and Innovation Program Grant 780701, BPI-France in the context of the national project RISQ (P141580), and the ANR AMIRAL project (ANR-21-ASTR-0016). Part of the research corresponding to this work was conducted while the authors were visiting the Simons Institute for the Theory of Computing.

References

1. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf, 2017.
2. M. R. Albrecht and A. Deo. Large modulus ring-LWE $\geq$ module-LWE. In *ASIACRYPT*, 2017.
3. E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe. Post-quantum key exchange—a new hope. In *USENIX Security*, 2016.
4. 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.
5. R. Bendlin and I. Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In *TCC*, 2010.
6. R. Bendlin, S. Krehbiel, and C. Peikert. How to share a lattice trapdoor: threshold protocols for signatures and (H) IBE. In *ACNS*, 2013.
7. A. Bogdanov, S. Guo, D. Masny, S. Richelson, and A. Rosen. On the hardness of learning with rounding over small modulus. In *TCC*, 2016.
8. 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.
9. J. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from LWE. In *ACM CCS*, 2016.
10. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, 2012.
11. Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé. Classical hardness of learning with errors. In *STOC*, 2013.
12. Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, 2011.
13. R. Canetti, R. Gennaro, S. Goldfeder, N. Makriyannis, and U. Peled. UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In *CCS*, 2020.
14. G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker. Two-party ECDSA from hash proof systems and efficient instantiations. In *CRYPTO*, 2019.

15. G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker. Bandwidth-efficient threshold EC-DSA. In *PKC*, 2020.
16. J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song. Bootstrapping for approximate homomorphic encryption. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT*, 2018.
17. J. H. Cheon, A. Kim, M. Kim, and Y. Song. Homomorphic encryption for arithmetic of approximate numbers. In *ASIACRYPT*, 2017.
18. I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. TFHE: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 2020.
19. D. Cozzo and N. P. Smart. Sharing the LUOV: threshold post-quantum signatures. In *IMACC*, 2019.
20. A. Dalskov, C. Orlandi, M. Keller, K. Shrishak, and H. Shulman. Securing DNSSEC keys via threshold ECDSA from generic MPC. In *ESORICS*, 2020.
21. I. Damgård, T. P. Jakobsen, J. B. Nielsen, J. I. Pagter, and M. B. Østergård. Fast threshold ECDSA with honest majority. In *SCN*, 2020.
22. I. Damgård, C. Orlandi, A. Takahashi, and M. Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. In *PKC*, 2021.
23. Y. Desmedt. Threshold cryptography. *European Transactions on Telecommunications*, 1994.
24. J. Doerner, Y. Kondi, E. Lee, and A. Shelat. Secure two-party threshold ECDSA from ECDSA assumptions. In *S& P*, 2018.
25. J. Doerner, Y. Kondi, E. Lee, and A. Shelat. Threshold ECDSA from ECDSA assumptions: The multiparty case. In *S&P*, 2019.
26. L. Ducas and D. Micciancio. FHEW: bootstrapping homomorphic encryption in less than a second. In *EUROCRYPT*, 2015.
27. L. Ducas and T. Prest. Fast Fourier orthogonalization. In *ISSAC*, 2016.
28. 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. Specification v1.0, available at `https://falcon-sign.info/`.
29. T. K. Frederiksen, M. Keller, E. Orsini, and P. Scholl. A unified approach to MPC with preprocessing using OT. In *ASIACRYPT*, 2015.
30. R. Gennaro and S. Goldfeder. Fast multiparty threshold ECDSA with fast trustless setup. In *CCS*, 2018.
31. R. Gennaro and S. Goldfeder. One round threshold ECDSA with identifiable abort. *IACR Cryptol. ePrint Arch.*, 2020.
32. R. Gennaro, S. Goldfeder, and A. Narayanan. Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In *ACNS*, 2016.
33. C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.
34. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, 2008.
35. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, 2013.
36. S. Goldwasser, Y. T. Kalai, C. Peikert, and V. Vaikuntanathan. Robustness of the learning with errors assumption. In *ICS*, 2010.
37. A. Gągol, J. Kula, D. Straszak, and M. Świętek. Threshold ECDSA for decentralized asset custody. *Cryptology ePrint Archive*, 2020.
38. G. Herold, E. Kirshanova, and A. May. On the asymptotic complexity of solving LWE. *Des. Codes Cryptogr.*, 2018.
39. J. Howe, T. Prest, T. Ricosset, and M. Rossi. Isochronous gaussian sampling: From inception to implementation. In *PQCrypto*, 2020.
40. A. Hülsing, T. Lange, and K. Smeets. Rounded gaussians – fast and secure constant-time sampling for lattice-based crypto. In *PKC*, 2018.
41. K. Kluczniak and L. Schild. Fdfb: Full domain functional bootstrapping towards practical fully homomorphic encryption. *arXiv preprint arXiv:2109.02731*, 2021.
42. A. Langlois, D. Stehlé, and R. Steinfeld. GGHLite: More efficient multilinear maps from ideal lattices. In *EUROCRYPT*, 2014.
43. Y. Lindell. Fast secure two-party ECDSA signing. In *CRYPTO*, 2017.
44. Y. Lindell and A. Nof. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In *CCS*, 2018.

45. S. Ling, D. H. Phan, D. Stehlé, and R. Steinfeld. Hardness of k-LWE and applications in traitor tracing. In *CRYPTO*, 2014.
46. V. Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT*, 2009.
47. V. Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, 2012.
48. V. Lyubashevsky and D. Micciancio. Generalized compact knapsacks are collision resistant. In *ICALP*, 2006.
49. V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, 2010.
50. C. Peikert and A. Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *TCC*, 2006.
51. T. Pornin and T. Prest. More efficient algorithms for the NTRU key generation using the field norm. In *PKC*, 2019.
52. T. Prest. Sharper bounds in lattice-based cryptography using the Rényi divergence. In *ASIACRYPT*, 2017.
53. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 2009.
54. M. Rossi. *Extended Security of Lattice-Based Cryptography*. PhD thesis, Université PSL, 2020. `https://www.di.ens.fr/~mrossi/docs/thesis.pdf`.
55. D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa. Efficient public key encryption based on ideal lattices. In *ASIACRYPT*, 2009.
56. K. Takashima and A. Takayasu. Tighter security for efficient lattice cryptography via the Rényi divergence of optimized orders. In *ProvSec*, 2015.
57. R. K. Zhao, R. Steinfeld, and A. Sakzad. COSAC: compact and scalable arbitrary-centered discrete Gaussian sampling over integers. In *PQCrypto*, 2020.
58. R. Zhu, C. Ding, and Y. Huang. Practical MPC+ FHE with applications in secure multi-partyneural network evaluation. *IACR Cryptol. ePrint Arch.*, 2020.

Table of Contents