


Round-Optimal Lattice-Based Threshold Signatures, Revisited

Shweta Agrawal 

Indian Institute of Technology, Madras, India

Damien Stehlé 

ENS de Lyon, France

Institut Universitaire de France, Paris, France

Anshu Yadav 

Indian Institute of Technology, Madras, India

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 $\tilde{O}(\lambda^3)$ to $\tilde{O}(\lambda)$, 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.

2012 ACM Subject Classification Security and privacy → Cryptography

Keywords and phrases Post-Quantum Cryptography, Lattices, Threshold Signatures

Digital Object Identifier 10.4230/LIPIcs.ICALP.2022.8

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version:* <https://eprint.iacr.org/2022/634.pdf>

Funding 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). The work is also partially supported by Microsoft Research Travel Grants to travel and present the paper at the conference.

Acknowledgements Part of the research corresponding to this work was conducted while the authors were visiting the Simons Institute for the Theory of Computing.



© Shweta Agrawal, Damien Stehlé, and Anshu Yadav;
licensed under Creative Commons License CC-BY 4.0

49th International Colloquium on Automata, Languages, and Programming (ICALP 2022).

Editors: Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff;

Article No. 8; pp. 8:1–8:20



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



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, 33, 31, 44, 25, 20, 15, 13, 30, 21, 32, 7], 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, 31, 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 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. [9] (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 [34, 6, 37], 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 $\exp(\tilde{O}(n \log q / \log^2 \alpha))$ (see, e.g., [38]) this leads to setting $n \log q = \tilde{\Omega}(\lambda^3)$. As the signature shares have bit-sizes that grow as $\Omega(n \log q)$, this leads to $\tilde{\Omega}(\lambda^3)$ -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 [35] 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, 56]) 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 [9] 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 $\tilde{O}(\lambda \log^2 Q)$, which is $\tilde{O}(\lambda)$ for any polynomially bounded Q ,¹ in contrast with $\tilde{O}(\lambda^3)$ for the construction from [9]. These bit-lengths are obtained when relying on the ring variants of SIS and LWE [48, 50, 54, 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.
- *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 therein².

1.3 Technical Overview

Recap of BGGJKRS Threshold Signatures. The round-optimal threshold signatures provided by [9] 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

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/>

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.

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 sk_1, \dots, 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, 11, 36] 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 sk_1, \dots, 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 $\gamma_1, \dots, \gamma_t$ so that $\sum_i \gamma_i sk_i = sk$. Then, she computes

$$\sum_i \gamma_i m_i = \sum_i \gamma_i \langle ct, sk_i \rangle = \langle ct, \sum_i \gamma_i sk_i \rangle = \langle ct, sk \rangle,$$

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 [9] 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 ct_{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 $ct_{sk} = \text{FHE.Enc}(\text{Sig.sk})$, each party first computes $\text{FHE.Eval}(C, ct_{sk})$ where C is the circuit $\text{Sig.Sign}(\mu, \cdot)$. By correctness of FHE, this yields an FHE encryption of the signature $\sigma = \text{Sig.Sign}(\text{Sig.sk}, \mu)$. To this ciphertext, the thresholdization trick described above may now be applied.

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, 57, 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, 8, 52, 40, 4, 10, 3, 5]. 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 [5], 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 $R_a(\mathcal{P}||\mathcal{Q}) \leq \text{poly}(1/\epsilon)$. Bai et al. [5] 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 $\text{err}_M + e_{1,M}$ 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 $\text{Sig} = (\text{Sig.KeyGen}, \text{Sig.Sign}, \text{Sig.Verify})$ 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 $\text{Sig}' = (\text{Sig}'.\text{KeyGen}, \text{Sig}'.\text{Sign}, \text{Sig}'.\text{Verify})$ derived from Sig by modifying the signing algorithm as follows: for $i \in [|\text{Sig.sk}|]$, if $\text{Sig.sk}_i = 0$, 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 $|\text{Sig.sk}|/2$ 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 $\{\text{Sig.sk}_i\}_{i \in [|\text{Sig.sk}|]: \text{Sig.sk}_i = 0}$. 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 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 $\mathbf{sk}_1, \dots, \mathbf{sk}_{N-1}$ randomly, implicitly setting the last share as $\mathbf{sk} - \sum_{i \in [N-1]} \mathbf{sk}_i$. To invoke the unforgeability of the underlying signature scheme Sig , the challenger must “forget” the signing key 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 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 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 $\hat{\sigma}_j$ of signatures σ_j . Knowing all the corrupt secret keys $\mathbf{sk}_1, \dots, \mathbf{sk}_{N-1}$ from the outset enables the challenger to walk this tightrope successfully – it obtains σ_j from the Sig challenger, computes an FHE encryption $\hat{\sigma}_j$ of this, computes partial decryptions using $\mathbf{sk}_1, \dots, \mathbf{sk}_{N-1}$, 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 $\{\sigma_{i,j}\}_{i \in [N], j \in \text{poly}}$ of σ_j as partial signatures, for unbounded j . Later if user 1 is corrupted (say), the adversary receives the secret key share \mathbf{sk}_1 . Now, to preserve indistinguishability, the challenger must explain the partial signatures $\{\sigma_{1,j}\}_{j \in \text{poly}}$ corresponding to user 1 as $\langle \hat{\sigma}_j, \mathbf{sk}_1 \rangle \approx \sigma_{1,j}$, 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 $r_{i,j} = H(j, K)^\top \mathbf{v}_i$ 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 $\langle \hat{\sigma}_j, \mathbf{sk}_i \rangle + \text{noise}_{i,j} + r_{i,j}$. By linearity, it holds that $\sum_{i \in [N]} H(j, K)^\top \mathbf{v}_i = \mathbf{0}$, 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 $\{\sigma_{i,j}\}_{i \in [N], j \in \text{poly}}$ of σ_j . When later, user 1 is corrupted, it can correctly explain the returned signatures as follows: it samples \mathbf{sk}_1 , computes $d_{1,j} = \langle \hat{\sigma}_j, \mathbf{sk}_1 \rangle + \text{noise}$ and sets $r_{1,j} = \sigma_{1,j} - d_{1,j}$. Now we may program $H(j, K)$ so that $r_{i,j} = H(j, K)^\top \mathbf{v}_i$ 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 for more details.

2 Preliminaries

In this section, we define some preliminaries used in our work. Please refer to the full version of the paper [2] for additional preliminaries.

► **Definition 1** (Threshold Signatures). *Let $P = \{P_1, \dots, P_N\}$ be a set of N parties. A threshold signature scheme for a class of efficient access structures \mathbb{S} on P is a tuple of PPT algorithms denoted by $\text{TS} = (\text{TS.KeyGen}, \text{TS.PartSign}, \text{TS.PartSignVerify}, \text{TS.Combine}, \text{TS.Verify})$ defined as follows:*

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

A TS scheme should satisfy the following requirements.

► **Definition 2** (Compactness). *A TS scheme for \mathbb{S} satisfies compactness if there exist polynomials $\text{poly}_1(\cdot), \text{poly}_2(\cdot)$ such that for all $\lambda, A \in \mathbb{S}$ and $S \in A$, the following holds. For $(\text{pp}, \text{vk}, \{\text{sk}_i\}_{i=1}^N) \leftarrow \text{TS.KeyGen}(1^\lambda, A)$, $\sigma_i \leftarrow \text{TS.PartSign}(\text{pp}, \text{sk}_i, m)$ for $i \in S$, and $\sigma_m \leftarrow \text{TS.Combine}(\text{pp}, \{\sigma_i\}_{i \in S})$, we have that $|\sigma_m| \leq \text{poly}_1(\lambda)$ and $|\text{vk}| \leq \text{poly}_2(\lambda)$.*

► **Definition 3** (Evaluation Correctness). *A signature scheme TS for \mathbb{S} satisfies evaluation correctness if for all $\lambda, A \in \mathbb{S}$ and $S \in A$, the following holds. For $(\text{pp}, \text{vk}, \{\text{sk}_i\}_{i=1}^N) \leftarrow \text{TS.KeyGen}(1^\lambda, A)$, $\sigma_i \leftarrow \text{TS.PartSign}(\text{pp}, \text{sk}_i, m)$ for $i \in [N]$ and $\sigma_m \leftarrow \text{TS.Combine}(\text{pp}, \{\sigma_i\}_{i \in S})$, we have that $\Pr[\text{TS.Verify}(\text{vk}, m, \sigma_m) = \text{accept}] \geq 1 - \lambda^{-\omega(1)}$.*

► **Definition 4** (Partial Verification Correctness). *A signature scheme TS for \mathbb{S} satisfies partial verification correctness if for all λ and $A \in \mathbb{S}$, the following holds. For $(\text{pp}, \text{vk}, \{\text{sk}_i\}_{i=1}^N) \leftarrow \text{TS.KeyGen}(1^\lambda, A)$, $\Pr[\text{TS.PartSignVerify}(\text{pp}, m, \text{TS.PartSign}(\text{pp}, \text{sk}_i, m)) = 1] = 1 - \lambda^{-\omega(1)}$.*

► **Definition 5** (Unforgeability). *A TS scheme is unforgeable if for any adversary \mathcal{A} with runtime $2^{o(\lambda)}$, the output of the following experiment $\text{Expt}_{\mathcal{A}, \text{TS}, \text{uf}}(1^\lambda)$ is 1 with probability $2^{-\Omega(\lambda)}$:*

1. On input the security parameter λ , the adversary outputs an access structure $A \in \mathbb{S}$.
2. Challenger runs the $\text{TS.KeyGen}(1^\lambda)$ algorithm and generates public parameters pp , verification key vk and set of N key shares $\{\text{sk}_i\}_{i=1}^N$. 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 $\text{TS.PartSign}(\text{pp}, \text{sk}_i, m)$ 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 an invalid party set, i.e. $S \not\subseteq \mathcal{A}$ (ii) m^* was not queried previously as a signing query and $\text{TS.Verify}(\text{vk}, m^*, \sigma^*) = \text{accept}$.
- We also consider following weaker notions of unforgeability.

► **Definition 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 $S \not\subseteq \mathcal{A}$. The challenger returns $\{\text{sk}_i\}_{i \in S}$.
 - \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 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 $S \not\subseteq \mathcal{A}$. The challenger returns $\{\text{sk}_i\}_{i \in S}$.
 - \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 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 $\text{Expt}_{\mathcal{A}, \text{TS}, \text{rb}}(1^\lambda)$ outputs 1 with probability $2^{-\Omega(\lambda)}$:

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

2.1 Rényi Divergence

► **Definition 9 (Rényi Divergence).** Let P and Q be any two discrete probability distributions such that $\text{Supp}(P) \subseteq \text{Supp}(Q)$. Then for $a \in (1, \infty)$, the Rényi Divergence of order a is defined by: $R_a(P||Q) = \left(\sum_{x \in \text{Supp}(P)} \frac{P(x)^a}{Q(x)^{a-1}} \right)^{\frac{1}{a-1}}$.

The following lemma is borrowed from [5, Lemma 2.9], with the exception of the multiplicativity property for non-independent variables, which is borrowed from [53, Proposition 2].

► **Lemma 10.** Let $a \in [1, \infty]$. Let P and Q denote distributions with $\text{Supp}(P) \subseteq \text{Supp}(Q)$. 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 $E \subseteq \text{Supp}(Q)$ be an arbitrary event. If $a \in (1, \infty)$, then $P(E) \geq P(E)^{\frac{a}{a-1}} / R_a(P||Q)$.
- **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 $a \in [1, \infty]$.
 - $R_a(P||Q) \leq R_a(P_1||Q_1) \cdot \max_{y_1 \in Y_1} R_a(P_{2|1}(\cdot|y_1)||Q_{2|1}(\cdot|y_1))$.

We will use the following RD bounds. Note that proof tightness can often be improved by optimizing over a , as suggested in [55].

► **Lemma 11** ([5]). 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 $a \in (1, +\infty)$

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

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. [9]. 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 t out of N threshold in the full version of the paper [2] and to adaptive security in Section 5. In Section 4, we show how to instantiate the underlying signature scheme using a variant of Lyubashevsky's signature [47] with matching moderate flooding.

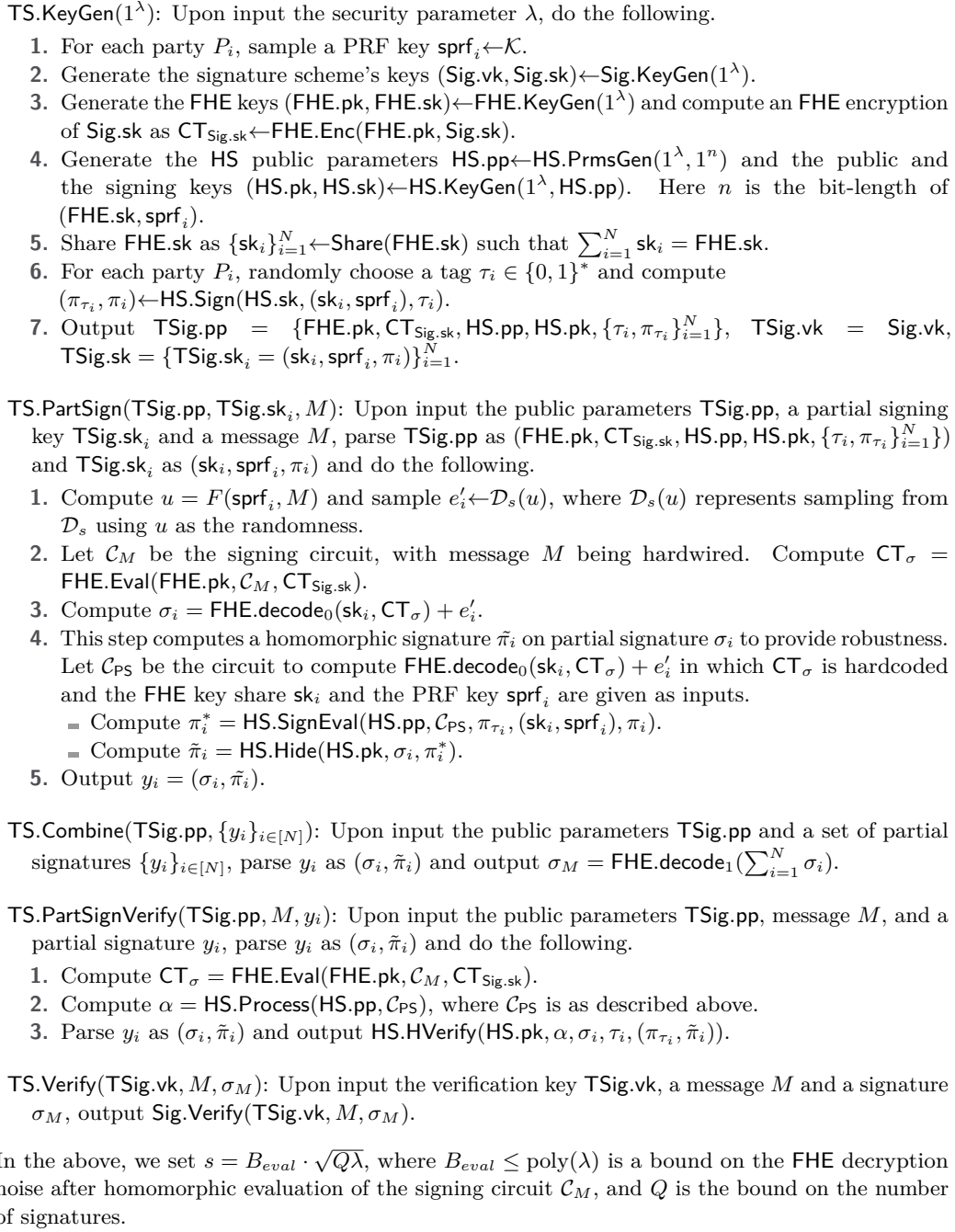
3.1 Optimizing the Boneh et al. scheme using the Rényi Divergence

Our scheme is similar to the one in [9] and is provided in Figure 1. The construction uses the following building blocks:

- A PRF $F : \mathcal{K} \times \{0, 1\}^* \rightarrow \{0, 1\}^r$, 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 $\text{FHE} = (\text{FHE.KeyGen}, \text{FHE.Enc}, \text{FHE.Dec}, \text{FHE.Eval})$. As in [9], we also assume that the FHE.Dec can be divided into two sub-algorithms: FHE.decode_0 and FHE.decode_1 .
- A UF-CMA signature scheme $\text{Sig} = (\text{Sig.KeyGen}, \text{Sig.Sign}, \text{Sig.Verify})$.
- A context hiding homomorphic signature scheme $\text{HS} = (\text{HS.PrmsGen}, \text{HS.KeyGen}, \text{HS.Sign}, \text{HS.SignEval}, \text{HS.Process}, \text{HS.Verify}, \text{HS.Hide}, \text{HS.HVerify})$ to provide robustness.
- An N out of N secret sharing scheme Share .

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

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



■ **Figure 1** Optimization of Boneh et al Threshold Signature Scheme.

3.1.1 Unforgeability

For security, we prove the following theorem.

► **Theorem 12.** *Assume F is a secure PRF, Sig is UF-CMA secure, FHE satisfies semantic security, Share satisfies privacy and HS is context hiding. Then the construction of threshold signatures in Figure 1 satisfies selective unforgeability (Definition 7) if the flooding noise is of the size $\text{poly}(\lambda) \cdot \sqrt{Q}$, where Q is the number of the signing queries.*

8:12 Round-Optimal Lattice-Based Threshold Signatures, Revisited

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., $S = [N] \setminus \{1\}$.

Hybrid₀: This is the real world.

Hybrid₁: Same as **Hybrid₀**, except that $\tilde{\pi}_1$ in **PartSign** is now generated using HS simulator as $\tilde{\pi}_1 = \text{HS.Sim}(\text{HS.sk}, \alpha, \sigma_1, \tau_1, \pi_{\tau_1})$, where $\alpha = \text{HS.Process}(\text{HS.pp}, \mathcal{C}_{\text{PS}})$.

Hybrid₂: Same as **Hybrid₁** except that to compute $\sigma_1 = \text{FHE.decode}_0(\text{sk}_1, \text{CT}_\sigma) + e'_1$, the randomness u used to sample $e'_1 \leftarrow \mathcal{D}_s(u)$ is chosen uniformly randomly instead of computing it using the PRF.

Hybrid₃: Same as **Hybrid₂**, except that now, for signing query for $(M, 1)$, the challenger simulates σ_1 as follows:

1. Computes $\text{CT}_\sigma = \text{FHE.Eval}(\text{FHE.pk}, \mathcal{C}_M, \text{CT}_{\text{Sig.sk}})$ and $\{\sigma'_i = \text{FHE.decode}_0(\text{sk}_i, \text{CT}_\sigma)\}_{i \in [2, N]}$.
2. Computes $\sigma_M = \text{Sig.Sign}(\text{Sig.sk}, M)$ and set $\sigma_1 = \sigma_M \lfloor \frac{q}{2} \rfloor - \sum_{i=2}^N \sigma'_i + e'_1$, where $e'_1 \leftarrow \mathcal{D}_s$.

Hybrid₄: Same as **Hybrid₃** except that instead of sharing FHE.sk , now the challenger generates the FHE key shares as $\{\text{sk}_i\}_{i=1}^N \leftarrow \text{Share}(\mathbf{0})$.

Hybrid₅: Same as **Hybrid₄**, except that $\text{CT}_{\text{Sig.sk}}$ in **TSig.pp** is replaced by $\text{CT}_0 = \text{FHE.Enc}(\text{FHE.pk}, \mathbf{0})$.

Detailed proofs of indistinguishability are provided in the full version [2]. Below, we provide the proof for the main new claim in our work.

► **Claim 13.** *If there is an adversary that can win the unforgeability game in **Hybrid₂** with probability ϵ , then its probability of winning the game in **Hybrid₃** 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 **Hybrid₂**, we have $\sigma_1 = \text{FHE.decode}_0(\text{sk}_1, \text{CT}_\sigma) + e'_1$, for $e'_1 \leftarrow \mathcal{D}_s$. In **Hybrid₃**, we have:

$$\begin{aligned}
 \sigma_1 &= \sigma_M \cdot \lfloor q/2 \rfloor - \sum_{i=2}^N \text{FHE.decode}_0(\text{sk}_i, \text{CT}_\sigma) + e'_1 \\
 &= \sigma_M \cdot \lfloor q/2 \rfloor - \sum_{i=1}^N \text{FHE.decode}_0(\text{sk}_i, \text{CT}_\sigma) + \text{FHE.decode}_0(\text{sk}_1, \text{CT}_\sigma) + e'_1 \\
 &= \sigma_M \cdot \lfloor q/2 \rfloor - \text{FHE.decode}_0\left(\sum_{i=1}^N \text{sk}_i, \text{CT}_\sigma\right) + \text{FHE.decode}_0(\text{sk}_1, \text{CT}_\sigma) + e'_1 \\
 &= \sigma_M \cdot \lfloor q/2 \rfloor - \text{FHE.decode}_0(\text{sk}, \text{CT}_\sigma) + \text{FHE.decode}_0(\text{sk}_1, \text{CT}_\sigma) + e'_1 \\
 &= \sigma_M \cdot \lfloor q/2 \rfloor - \sigma_M \cdot \lfloor q/2 \rfloor + e + \text{FHE.decode}_0(\text{sk}_1, \text{CT}_\sigma) + e'_1 \\
 &= \text{FHE.decode}_0(\text{sk}_1, \text{CT}_\sigma) + (e'_1 + e),
 \end{aligned}$$

for some e satisfying $|e| \leq B_{\text{eval}}$. Thus, in **Hybrid₂**, the error term in σ_1 is e'_1 , while in **Hybrid₃**, 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 **TSig.pp**, the verification key **TSig.vk**, the corrupted secret key shares **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 **TSig.pp**, **TSig.vk**, $\{\text{TSig.sk}_i\}$, $\{\tilde{\pi}_j\}$ 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 **Hybrid₂** as D_2 and in **Hybrid₃** 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 $D_k = (\mathcal{E}_Q^{(k)}, \mathcal{M}_Q^{(k)}, \mathcal{E}_{Q-1}^{(k)}, \mathcal{M}_{Q-1}^{(k)}, \dots, \mathcal{E}_1^{(k)}, \mathcal{M}_1^{(k)})$ for $k \in \{2, 3\}$ and

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

Applying the multiplicativity property of the Rényi divergence (Lemma 10), we obtain that $R_a(D_2 \| D_3)$ is bounded from above by

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

where $X = (M_Q, E_{Q-1}, \dots, E_1)$ 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 11 in Equation (3.2), we get

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

Further, since M_Q is a function of $E_{Q-1}, M_{Q-1}, \dots, E_1, M_1$, the data processing inequality (Lemma 10) gives

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

Hence, we get

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

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 10), we have $D_3(\mathbf{E}) \geq \frac{D_2(\mathbf{E})^{\frac{a}{a-1}}}{R_a(D_2 \| D_3)} \geq D_2(\mathbf{E})^{\frac{a}{a-1}} \exp(-\frac{a\pi}{\lambda})$. The result is obtained by setting $a = 2$. \blacktriangleleft

3.2 On the Optimality of Our Flooding

We show that the flooding amount that we achieved is optimal for our threshold signature scheme. To argue this, we show how to attack it if the flooding amount is below $\Omega(\sqrt{Q})$. For simplicity, we restrict to the case of $N = 2$. Recall that in our construction, $\text{TS.PartSign}(\text{TSig.pp}, \text{TSig.sk}_i, M)$ outputs $\sigma_{i,M} = \text{FHE.decode}_0(\text{sk}_i, \text{CT}_{\sigma_M}) + e'_{i,M}$, where $\text{TSig.sk}_i = (\text{sk}_i, \text{sprf}_i)$.³ W.l.o.g, assume that the adversary gets the partial signing key 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 $\sigma_{1,M} = \text{FHE.decode}_0(\text{sk}_1, \text{CT}_{\sigma_M}) + e'_{1,M}$. From this the adversary can compute:

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

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

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 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 $\sigma_M = \text{TS.Combine}(\text{TSig.pp}, \sigma_{1,M}, \sigma_{2,M})$. Hence, from $\sigma_M + \text{err}_M + e'_{1,M}$, the adversary can compute $\text{err}_M + e'_{1,M}$. Since, the signature scheme is deterministic, err_M depends only on M . Thus, if the same message is queried for signature multiple times, then each time the term err_M remains the same, but since flooding is randomized, the term $e'_{1,M}$ is different.

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

$$w^i = \text{err}_M + e'_{1,M}{}^i \quad (3.3)$$

Since err_M is same, taking average on both sides of Equation (3.3) over all the Q samples, we get $\frac{\sum_{i \in [Q]} w^i}{Q} = \text{err}_M + \frac{\sum_{i \in [Q]} e'_{1,M}{}^i}{Q}$. If $|\frac{1}{Q} \sum_{i \in [Q]} e'_{1,M}{}^i| < 1/2$, then the adversary can recover err_M as $\text{err}_M = \lfloor \frac{1}{Q} \sum_{i \in [Q]} w^i \rfloor$. As $e'_{1,M}{}^1, \dots, e'_{1,M}{}^Q$ are independently and identically distributed with mean 0, by Hoeffding's inequality, we have

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

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

3.2.1 Attack for Deterministic Error

In the argument for randomized error, the fact that $e'_{1,M}{}^i$ 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 $\text{Sig}' = (\text{Sig}'.\text{KeyGen}, \text{Sig}'.\text{Sign}, \text{Sig}'.\text{Verify})$ derived from a secure signature scheme $\text{Sig} = (\text{Sig}.\text{KeyGen}, \text{Sig}.\text{Sign}, \text{Sig}.\text{Verify})$ as follows:

1. $\text{Sig}'.\text{KeyGen}$ is identical to $\text{Sig}.\text{KeyGen}$. Let $(\text{Sig}.\text{sk}, \text{Sig}.\text{vk})$ be the signing and verification keys, respectively, and $\text{Sig}.\text{sk}_i$ denote the i th bit of $\text{Sig}.\text{sk}$ for $i \in [\ell]$, where ℓ is the bit-length of $\text{Sig}.\text{sk}$.
2. $\text{Sig}'.\text{Sign}(\text{Sig}.\text{sk}, M)$ is modified as follows:
 - Compute $\sigma_M = \text{Sig}.\text{Sign}(\text{Sig}.\text{sk}, M)$. Set $\sigma'_M := \sigma_M$.
 - For i from 1 to ℓ : if $\text{Sig}.\text{sk}_i = 0$, then set $\sigma'_M := \sigma'_M \parallel \text{Sig}.\text{sk}_i$.
 - Output σ'_M .

3. $\text{Sig}'.\text{Verify}(\text{Sig}.\text{vk}, M, \sigma'_M)$ is defined as $\text{Sig}.\text{Verify}(\text{Sig}.\text{vk}, M, \sigma_M)$, where σ_M is obtained from σ'_M by removing all the bits after the k th bit, where k is the bit-length of signatures in Sig .

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 $\text{Sig}.\text{sk}$ 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.

3.2.1.1 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 $\{\text{Sig}.\text{sk}_i\}_{i \in [\ell]: \text{Sig}.\text{sk}_i=0}$. Let $\text{CT}_{\sigma_M}, \text{err}_M, e'_M$ 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 $\text{CT}^*, \text{err}^*$ and e_M^* denote the components corresponding to $\{\text{Sig}.\text{sk}_i\}_{i \in [\ell]: \text{Sig}.\text{sk}_i=0}$.

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

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 $w_j^* = \text{err}^* + e_{M_j}^*$ 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 $1 - 1/\text{poly}(\lambda)$. 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 $H : \{0, 1\}^* \rightarrow D_H := \{\mathbf{v} : \mathbf{v} \in \{-1, 0, 1\}^k; \|\mathbf{v}\|_1 \leq \alpha\}$, modeled as a random oracle. Here α is a parameter, typically much smaller than k . The signature scheme is described in Figure 2.

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

Security. We establish security via the following theorem. Because of space constraints, proof of the theorem is given in the full version of the paper [2].

► **Theorem 14.** *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 $\text{SIS}_{q,n,m,\beta}$ is hard for $\beta = 2\gamma + 2d\alpha\sqrt{m}$. Then the construction in Figure 2 satisfies UF-CMA in the random oracle model.*

KeyGen(1^λ): 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 14); then do the following:

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

Sign(sk, μ): 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 $\mathbf{c} = H(\mathbf{A}\mathbf{y}, \mu)$.
3. Set $\mathbf{z} = \mathbf{y} + \mathbf{S}\mathbf{c}$.
4. Output (\mathbf{z}, \mathbf{c}) .

Verify($\text{vk}, \mu, (\mathbf{z}, \mathbf{c})$): 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 $H(\mathbf{A}\mathbf{z} - \mathbf{T}\mathbf{c}, \mu) = \mathbf{c}$.
3. If both checks pass, then accept, else reject.

■ **Figure 2** Lyubashevsky’s Signature Without Aborts.

5 Adaptive Security for Threshold Signatures

As discussed in Section 1, we provide two constructions to improve the selective security achieved by [9]. Below, we describe our construction in the ROM, which satisfies partially adaptive unforgeability (Definition 6). Due to space constraints, we provide our construction in the standard model with *pre-processing* that satisfies fully adaptive unforgeability (Definition 5) in the full version.

5.1 Partially Adaptive Unforgeability

We use the same building blocks as in Section 3.1 for construction. We also use two keyed hash function modelled as random oracles: $H : \{0, 1\}^\lambda \times \{0, 1\}^* \rightarrow \mathbb{Z}_q^N$ and $H_1 : \{0, 1\}^\lambda \times \{0, 1\}^* \rightarrow \{0, 1\}^r$. The construction is provided in Figure 3.

Correctness and Unforgeability. The proof of correctness is similar to that in Section 3.1 and is provided in the full version of the paper [2]. We prove partially adaptive unforgeability via the following theorem.

► **Theorem 15.** *Assume Sig satisfies unforgeability, FHE is semantically secure, HS is context hiding and Share satisfies privacy. Then the TS construction in Figure 3 satisfies partially adaptive unforgeability (Definition 6) in ROM if the flooding error is of size $\text{poly}(\lambda)\sqrt{Q}$, where Q is the upper bound on the number of signing queries.*

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

Hybrid₀ and Hybrid₁ are the same as that in the proof of Theorem 12.

Hybrid₂: Same as Hybrid₁, 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 $H_1(\text{hkey}_i, M) = u_i$. 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.

TS.KeyGen(1^λ): Upon input the security parameter λ , do the following:

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

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

1. Compute $u_i = H_1(\text{hkey}_i, M)$ 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 $\text{CT}_{\sigma_M} = \text{FHE.Eval}(\text{FHE.pk}, \mathcal{C}_M, \text{CT}_{\text{Sig.sk}})$.
3. Compute $r_{i,M} = H(K, M)^T \mathbf{v}_i$ and $\sigma_{i,M} = \text{FHE.decode}_0(\text{sk}_i, \text{CT}_{\sigma_M}) + r_{i,M} + e'_i$.
4. This step computes a homomorphic signature $\tilde{\pi}_{i,M}$ on $\sigma_{i,M}$ to provide robustness. Let \mathcal{C}_{PS} be the circuit to compute $\text{FHE.decode}_0(\text{sk}_i, \text{CT}_{\sigma_M}) + H(K, M)^T \mathbf{v}_i + e'_i$ in which CT_{σ_M} is hardcoded and the key share TSig.sk_i is given as the input. Compute $\pi_{i,M}^* = \text{HS.SignEval}(\text{HS.pp}, \mathcal{C}_{\text{PS}}, \pi_{\tau_i}, (\text{sk}_i, K, \mathbf{v}_i, \text{hkey}_i), \pi_i)$ and $\tilde{\pi}_{i,M} = \text{HS.Hide}(\text{HS.pk}, \sigma_{i,M}, \pi_{i,M}^*)$.
5. Output $y_{i,M} = (\sigma_{i,M}, \tilde{\pi}_{i,M})$.

Algorithms TS.PartSignVerify, TS.Combine and TS.Verify are identical to those in Section 3.1.

■ **Figure 3** Partially Adaptive Threshold Signature Scheme.

Hybrid₃: Same as Hybrid₂ 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 $\text{CT}_{\sigma_M} = \text{FHE.Eval}(\text{FHE.pk}, \mathcal{C}_M, \text{CT}_{\text{Sig.sk}})$.
2. It then computes $\text{FHE.decode}_0(\text{FHE.sk}, \text{CT}_{\sigma_M})$ and generates N shares as $\{s_{i,M}\}_{i=1}^N \leftarrow \text{Share}(\text{FHE.decode}_0(\text{FHE.sk}, \text{CT}_{\sigma_M}))$.
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 $\text{FHE.decode}_0(\text{FHE.sk}, \text{CT}_{\sigma_M})$ 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 $r_{i,M} = s_{i,M} - \text{FHE.decode}_0(\text{sk}_i, \text{CT}_{\sigma_M})$ 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 $H(x) = \mathbf{r}_x$.

Hybrid₄: Same as Hybrid₃, except that now the signing queries are answered differently. For each pre-corruption signing query for a message M , $\sigma_{i,M}$ is computed as follows:

1. The challenger computes $\sigma_M = \text{Sig.Sign}(\text{Sig.sk}, M)$ and generates random shares of $\sigma_M \lfloor q/2 \rfloor$ as $\{s_{i,M}\}_{i=1}^N \leftarrow \text{Share}(\sigma_M \lfloor q/2 \rfloor)$ such that $\sum_{i=1}^N s_{i,M} = \sigma_M \lfloor q/2 \rfloor$.
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 $r_{i,M} = s_{i,M} - \text{FHE.decode}_0(\text{sk}_i, \text{CT}_{\sigma_M})$. 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 $\{\text{TSig.sk}_i\}_{i \in S}$.

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

Hybrid₅ and Hybrid₆: are the same as Hybrid₄ and Hybrid₅, respectively, defined in the proof of Theorem 12.

In the full version of the paper we show that consecutive hybrids are indistinguishable and that the probability of the adversary winning the unforgeability game (Definition 6) is negligible in Hybrid₆. ◀

References

- 1 Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. URL: <https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf>, 2017.
- 2 Shweta Agrawal, Damien Stehlé, and Anshu Yadav. Round-optimal lattice-based threshold signatures, revisited. Cryptology eprint Archive, 2022.
- 3 Martin R Albrecht and Amit Deo. Large modulus ring-LWE \geq module-LWE. In *ASIACRYPT*, 2017.
- 4 Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange—a new hope. In *USENIX Security*, 2016.
- 5 Shi Bai, Tancrede Lepoint, Adeline Roux-Langlois, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: using the Rényi divergence rather than the statistical distance. *J. Cryptol.*, 2018.
- 6 Rikke Bendlin and Ivan Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In *TCC*, 2010.
- 7 Rikke Bendlin, Sara Krehbiel, and Chris Peikert. How to share a lattice trapdoor: threshold protocols for signatures and (H) IBE. In *ACNS*, 2013.
- 8 Andrej Bogdanov, Siyao Guo, Daniel Masny, Silas Richelson, and Alon Rosen. On the hardness of learning with rounding over small modulus. In *TCC*, 2016.
- 9 Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In *CRYPTO*, 2018.
- 10 Joppe Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from LWE. In *ACM CCS*, 2016.
- 11 Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, 2012.

- 12 Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, 2011.
- 13 Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In *CCS*, 2020.
- 14 Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Two-party ECDSA from hash proof systems and efficient instantiations. In *CRYPTO*, 2019.
- 15 Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Bandwidth-efficient threshold EC-DNA. In *PKC*, 2020.
- 16 Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for approximate homomorphic encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT*, 2018.
- 17 Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *ASIACRYPT*, 2017.
- 18 Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 2020.
- 19 Daniele Cozzo and Nigel P. Smart. Sharing the LUOV: threshold post-quantum signatures. In *IMACC*, 2019.
- 20 Anders Dalskov, Claudio Orlandi, Marcel Keller, Kris Shrishak, and Haya Shulman. Securing DNSSEC keys via threshold ECDSA from generic MPC. In *ESORICS*, 2020.
- 21 Ivan Damgård, Thomas Pelle Jakobsen, Jesper Buus Nielsen, Jakob Illeborg Pagter, and Michael Bækvang Østergård. Fast threshold ECDSA with honest majority. In *SCN*, 2020.
- 22 Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n -out-of- n and multi-signatures and trapdoor commitment from lattices. In *PKC*, 2021.
- 23 Yvo Desmedt. Threshold cryptography. *European Transactions on Telecommunications*, 1994.
- 24 Jack Doerner, Yashvanth Kondi, Eysa Lee, and Abhi Shelat. Secure two-party threshold ECDSA from ECDSA assumptions. In *S&P*, 2018.
- 25 Jack Doerner, Yashvanth Kondi, Eysa Lee, and Abhi Shelat. Threshold ECDSA from ECDSA assumptions: The multiparty case. In *S&P*, 2019.
- 26 Léo Ducas and Daniele Micciancio. FHEW: bootstrapping homomorphic encryption in less than a second. In *EUROCRYPT*, 2015.
- 27 Léo Ducas and Thomas Prest. Fast Fourier orthogonalization. In *ISSAC*, 2016.
- 28 Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-Fourier lattice-based compact signatures over NTRU. Specification v1.0, available at <https://falcon-sign.info/>.
- 29 Tore Kasper Frederiksen, Marcel Keller, Emmanuela Orsini, and Peter Scholl. A unified approach to MPC with preprocessing using OT. In *ASIACRYPT*, 2015.
- 30 Adam Gągól, Jędrzej Kula, Damian Straszak, and Michał Świątek. Threshold ECDSA for decentralized asset custody. *Cryptology ePrint Archive*, 2020.
- 31 Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ECDSA with fast trustless setup. In *CCS*, 2018.
- 32 Rosario Gennaro and Steven Goldfeder. One round threshold ECDSA with identifiable abort. *IACR Cryptol. ePrint Arch.*, 2020.
- 33 Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In *ACNS*, 2016.
- 34 Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. URL: crypto.stanford.edu/craig.
- 35 Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, 2008.
- 36 Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, 2013.

- 37 Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In *ICS*, 2010.
- 38 Gottfried Herold, Elena Kirshanova, and Alexander May. On the asymptotic complexity of solving LWE. *Des. Codes Cryptogr.*, 2018.
- 39 James Howe, Thomas Prest, Thomas Ricosset, and Mélissa Rossi. Isochronous gaussian sampling: From inception to implementation. In *PQCrypto*, 2020.
- 40 Andreas Hülsing, Tanja Lange, and Kit Smeets. Rounded gaussians – fast and secure constant-time sampling for lattice-based crypto. In *PKC*, 2018.
- 41 Kamil Kluczniak and Leonard Schild. Fdfb: Full domain functional bootstrapping towards practical fully homomorphic encryption. *arXiv preprint*, 2021. [arXiv:2109.02731](https://arxiv.org/abs/2109.02731).
- 42 Adeline Langlois, Damien Stehlé, and Ron Steinfeld. GGHLite: More efficient multilinear maps from ideal lattices. In *EUROCRYPT*, 2014.
- 43 Yehuda Lindell. Fast secure two-party ECDSA signing. In *CRYPTO*, 2017.
- 44 Yehuda Lindell and Ariel Nof. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In *CCS*, 2018.
- 45 San Ling, Duong Hieu Phan, Damien Stehlé, and Ron Steinfeld. Hardness of k -LWE and applications in traitor tracing. In *CRYPTO*, 2014.
- 46 Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT*, 2009.
- 47 Vadim Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, 2012.
- 48 Vadim Lyubashevsky and Daniele Micciancio. Generalized compact knapsacks are collision resistant. In *ICALP*, 2006.
- 49 Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, 2010.
- 50 Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *TCC*, 2006.
- 51 Thomas Pornin and Thomas Prest. More efficient algorithms for the NTRU key generation using the field norm. In *PKC*, 2019.
- 52 Thomas Prest. Sharper bounds in lattice-based cryptography using the Rényi divergence. In *ASIACRYPT*, 2017.
- 53 Mélissa Rossi. *Extended Security of Lattice-Based Cryptography*. PhD thesis, Université PSL, 2020. URL: <https://www.di.ens.fr/~mrossi/docs/thesis.pdf>.
- 54 Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In *ASIACRYPT*, 2009.
- 55 Katsuyuki Takashima and Atsushi Takayasu. Tighter security for efficient lattice cryptography via the Rényi divergence of optimized orders. In *ProvSec*, 2015.
- 56 Raymond K. Zhao, Ron Steinfeld, and Amin Sakzad. COSAC: compact and scalable arbitrary-centered discrete Gaussian sampling over integers. In *PQCrypto*, 2020.
- 57 Ruiyu Zhu, Changchang Ding, and Yan Huang. Practical MPC+ FHE with applications in secure multi-party neural network evaluation. *IACR Cryptol. ePrint Arch.*, 2020.