Simple and Efficient Lattice Threshold Signatures with Identifiable Aborts

Rafael del Pino[1], Thomas Espitau[1], Guilhem Niot[1,2], and Thomas Prest[1]

[1] PQShield
rafael.del.pino@pqshield.com, thomas@espitau.com, guilhem@gniot.fr, thomas.prest@pqshield.com
[2] Univ Rennes, CNRS, IRISA

Abstract. We introduce simple yet efficient lattice-based threshold signatures with identifiable aborts, secure under the MLWE assumption. Central to our construction are novel Distributed Key Generation with Short Shares (sDKG) protocols over lattices, ensuring short shares, small reconstruction coefficients, and linear evaluation of honest shares. This uniquely realizes the "threshold designer's dream": *signature shares double as valid signatures under the corresponding secret key shares*. With two concrete instantiations (ramp and replicated secret sharings), our schemes match Threshold Raccoon (del Pino et al. EUROCRYPT 2024)'s compact ∼10kB size. Further, we unveil 'Death Star Detection', a new algorithm that enhances identifiable aborts by efficiently spotting short vector adversarial correlations, of interest beyond threshold signatures.

1 Introduction

A threshold signature scheme (TSS) is a specific type of multiparty computation scheme that distributes a signing key sk across N devices, so that any subset of T devices may interact to sign a message jointly, but strictly less than T cannot. The recent announcement of the NIST First Call for Multiparty Threshold Schemes [BP23], including proposals for post-quantum secure schemes highlighted the need to standardize practical TSS. While there are several efficient, secure TSS constructions based on RSA, ECDSA and Schnorr signatures, the upcoming post-quantum standards Dilithium, SPHINCS+, and Falcon are highly impractical to thresholdize and build efficient TSS based on post-quantum assumptions has remained a largely open problem until recently. In the past two years, several lattice-based TSS have been proposed, which can sometimes achieve a high number of signers T (for example, up to 1024 in [DKM+24]).

Identifiable aborts. While the baseline security notion for TSS is unforgeability, it is often insufficient as distributed protocols must resist a broader class of attacks than non-distributed primitives. For example, malicious parties may sabotage the signing process so that it does not output valid signatures. One solution is to design protocols that can reliably identify such malicious parties, a property commonly referred to as *identifiable aborts (IA)*.

Unfortunately, adding this property to existing lattice TSS seems difficult. For example, the signing procedures of [DKM+24, EKT24, KRT24, BKL+24] rely on the use of one-time masks that are generated by passing a secret value into a PRF, a step in which misbehavior seems nontrivial to detect without relying on heavy tools such as non-interactive zero-knowledge proofs (NIZKs). Such an heavy approach has been recently used in [dPKN+25], but remains very cumbersome. Similarly, rejection sampling, present in [DOTT22, BTT22, ADP24], provides a means for malicious parties to abort without raising suspicion.

Our contributions. In this work, we address the challenge of constructing *efficient* threshold signatures with identifiable aborts, notably without resorting to Non-Interactive Zero-Knowledge proofs (NIZKs) or rejection sampling. We propose an elegantly simple solution, consisting in this design rationale:

"Signature shares are valid signatures for their corresponding key shares".

This inherent property allows the honesty verification *for free* by simply *validating individual shares against the respective public key share*. This organic approach further allows a generic optimization leveraging geometric properties of short random vectors. Specifically, we exploit the concentration of Gaussian mass to detect anomalous correlations between shares, thereby enhancing the detection of collusions among malicious users. To formalize this, our framework requires a secret sharing/key generation with three critical properties:

1. The output must consist of *short shares*.
2. The reconstruction coefficients must be *small*.
3. It must enable *functional evaluation* over the honest shares.

In particular, this provides a general framework for generating secret sharing schemes with short shares, a critical feature for applications such as lattice trapdoor sampling and advanced properties like Identifiable Aborts (IA), where the shortness property is paramount. Our scheme notably provides more comprehensive properties than $\{0,1\}$-Linear Secret Sharing Schemes (LSSS) [BGG+18], which only guarantee small reconstruction coefficients. As such, we showcase this construction in the context of *lattice-based threshold signature schemes (TSS) with identifiable aborts (IA)*. Crucially, our methodology circumvents the need for cumbersome cryptographic machinery such as general multiparty computation (MPC), NIZKs, or fully homomorphic encryption (FHE). To our knowledge, this paper introduces *the most efficient lattice-based TSS with IA*. We instantiate this framework in two complementary ways:

1. The first construction uses *replicated* secret sharing and demonstrates efficiency for small values of N and T (typically, $N \leqslant 16$).
2. The second construction, based on *ramp sharing*, is tailored for efficiency in large user sets (e.g., $T, N > 1000$). In this variant, the privacy threshold T_1 may be strictly smaller than the correctness threshold T_2, whereas in standard TSS, one typically finds $T_1 = T_2 - 1$.

Additionally, we note that our framework can be applied to TSS schemes based on rejection sampling [DOTT22, BTT22, ADP24] relying on short partial secrets to extend them to the T-out-of-N setting, although with increased concrete parameters. This was done by a follow-up work [dPN25], which introduced a compact threshold signature with rejection sampling leveraging our framework.

Related Works

Classical v. Post-quantum. There exists a myriad of TSS based on RSA [GHKR08], Schnorr signatures [KG20, CKM23] and ECDSA [CGG+20, CCL+23] with many of these satisfying IA [KG20, CGG+20, CCL+23]. Lattice TSS based on plain Fiat-Shamir (without rejection sampling) have been proposed recently. They either rely on threshold FHE [ASY22, GKS24] or on more standard tools [DKM+24, EKT24, KRT24, CATZ24]. The underlying standard signature in these works and ours are similar to [dEK+23]. Some lattice-based TSS have also been based on different paradigms such as trapdoor sampling [BKP13], hash-then-sign with noise flooding [EEN+24, ENP24], and Fiat-Shamir with aborts leveraging linear homomorphism [DOTT22, BTT22, Che23, ADP24] or MPC [TPCZ23, DKLS25].

Withstanding malicious users. Achieving resistance against misbehaving parties in lattice-based TSS schemes has been particularly challenging and several approaches have been proposed: [BKP13] laid a theoretical foundation for lattice-based TSS with *robustness*; i.e. ensuring that a valid signature is output even in the presence of malicious users. [ENP24] improved the practicality of robust lattice TSS by relying on approximate shortness proof in the honest majority setting. The notion of *identifiable aborts* (IA) was introduced as a relaxation of robustness to improve the practicality of TSS, and only aims at *detecting* misbehaving users rather than correcting. [BGG+18, ASY22, GKS24] introduced TSS with IA based on threshold FHE. Unfortunately, these constructions come at a high practical cost or with a short-proof slack constraining parameterization.

Most recently, del Pino et al. [dPKN+25] proposed a mechanism to identify aborts in Threshold Raccoon. They take a radically different approach from that of this paper with a deferred abort identification relying on cryptographic proofs rather than simple partial signature verification. Although their construction may scale to larger thresholds, it comes at the prohibitive cost of increased communication (about $60+6T$ additional kB per party) and complex implementation. Additionally, it seems hard to distribute its key generation due to the need for commitment on secret shares, while this is builtin in our solution.

2 Technical Overview

2.1 The big picture of thresholdizing lattice-based Fiat-Shamir

Several lines of work were explored to build lattice-based threshold signatures. We focus here on solutions that do not rely on heavy tools such as general MPC,

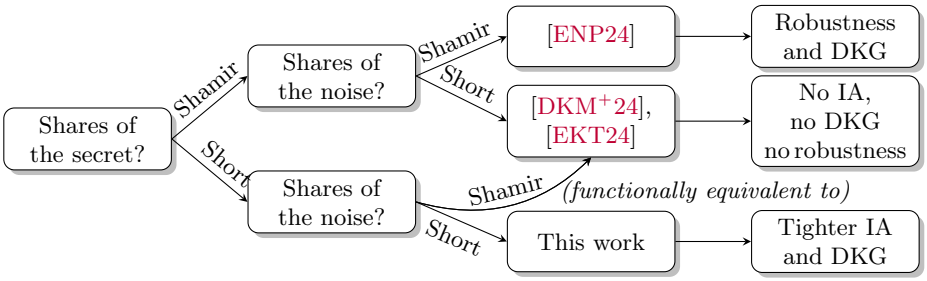


Fig. 1: Concurrent T-out-of-N threshold Fiat-Shamir signatures on lattices classified by whether their secret and noise are shared using either short secret sharing or Shamir secret sharing. The choice of secret sharing for the secret key and the noise has a massive impact on the resulting properties.

NIZK, or FHE. Inside this scope, two approaches stand out for their practical efficiency. The first builds on *Fiat-Shamir with aborts* [DOTT21, BTT22], and have each party perform a rejection-based partial signing. These schemes are, however, limited to the N-out-of-N setting, and produce somewhat large signatures.

The second approach has built upon Shamir's secret sharing combined with noise flooding. They produce signatures of the form $c \cdot \mathbf{s} + \sum_i \mathbf{r}_i$ where $\mathbf{s}$ is the secret and the $\mathbf{r}_i$ are short vectors produced locally by each party. From this, the natural idea would be to perform partial signing by simply computing $\mathbf{z}_i \coloneqq c \cdot \lambda_i [\![\mathbf{s}]\!]_i + \mathbf{r}_i$, however this would leak the share $[\![\mathbf{s}]\!]_i$. Two ways appeared to perform this computation safely: (i) add a T-out-of-T share of 0 [DKM+24] to $\mathbf{z}_i$ so that it appears uniform and hides the shares of $\mathbf{s}$, or (ii) each party samples and dispatches a T-out-of-N sharing of $\mathbf{r}_i$ [ENP24] and rather computes $\mathbf{z}_i \coloneqq c \cdot [\![\mathbf{s}]\!]_i + [\![\mathbf{r}_i]\!]$.

In this paper, we introduce an alternative approach relying on short shares (a) to prevent the caveats above in computing $\mathbf{z}_i$ safely, and (b) that can be applied to lift schemes based on Fiat-Shamir with aborts to the T-out-of-N setting. Adapting these techniques to the Fiat-Shamir without aborts gives a very natural way to provide IA in lattice-based TSS. In the following paragraphs, we illustrate our techniques for noise-flooded schemes, starting from a N-out-of-N threshold signature and lifting it to a T-out-of-N one.

2.2 From basic noise-flooded Fiat-Shamir...

We first recall a simple construction of a lattice-based Fiat-Shamir signature scheme that avoids rejection sampling. It builds over a ring $\mathcal{R}_q$ of polynomials reduced modulo an integer q. $\mathcal{R}_q$ is equipped with usual norms computed over the coefficients embedding lifted to $\mathbb{Z} \cap [-\lfloor q/2 \rfloor, q - \lfloor q/2 \rfloor)$. Aside from the challenge c computation, the scheme consists solely of linear operations modulo q, making it highly suitable for thresholdization [ASY22, DKM+24, GKS24, EKT24].

Key generation. Sample a short secret $(\mathbf{s}, \mathbf{e}) \in \mathcal{R}_q^\ell \times \mathcal{R}_q^k$, and reveal the public key $\mathsf{vk} := \mathbf{b} := \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$, where $\mathbf{A} \in \mathcal{R}_q^{k \times \ell}$ is a random matrix. The public key is indistinguishable from uniform under the MLWE assumption.

Signature generation. The signer samples a short mask $(\mathbf{r}, \mathbf{e}')$, and computes a commitment $\mathbf{w} = \mathbf{A} \cdot \mathbf{r} + \mathbf{e}'$, then a challenge $c := H(\mathsf{msg}, \mathbf{w})$ and a response $(\mathbf{z} = c \cdot \mathbf{s} + \mathbf{r}, \mathbf{y} = c \cdot \mathbf{e} + \mathbf{e}')$. The signature is the tuple $\mathsf{sig} := (c, \mathbf{z}, \mathbf{y})$.

Verification. Given a signature $\mathsf{sig} = (c, \mathbf{z}, \mathbf{y})$, the verifier reconstructs $\mathbf{w} = \mathbf{A} \cdot \mathbf{z} + \mathbf{y} - c \cdot \mathbf{b}$. It then verifies that $(\mathbf{z}, \mathbf{y})$ are small and that $c = H(\mathsf{msg}, \mathbf{w})$. The hardness of forgery is directly related to the MSIS problem, provided that the challenge space $\mathcal{C}$ from which c is sampled has sufficient entropy, for instance, using $\mathcal{C} = \{c \in \mathcal{R}_q \mid \|c\|_\infty = 1 \wedge \|c\|_1 = \omega\}$ as in [DKM+24].

2.3 ...to N-out-of-N TSS with IA

This elementary scheme easily converts into an N-out-of-N threshold signature scheme (TSS) as follows. First, individual key pairs $(\mathbf{s}_i, \mathbf{b}_i)$ are generated for each party, which are then linearly aggregated to compute the signing and verification keys: $(\mathbf{s}, \mathbf{b}) = \left(\sum_{i \in [N]} \mathbf{s}_i, \sum_{i \in [N]} \mathbf{b}_i\right)$. To sign, parties:

1. Generate individual commitments $\mathbf{w}_i := \mathbf{A} \cdot \mathbf{r}_i + \mathbf{e}'_i$ in parallel;
2. Aggregate individual commitments into a single commitment $\mathbf{w} = \sum_{i \in [N]} \mathbf{w}_i$ and generate a common challenge;
3. Locally generate individual responses $(\mathbf{z}_i, \mathbf{y}_i) := c \cdot (\mathbf{s}_i, \mathbf{e}_i) + (\mathbf{r}_i, \mathbf{e}'_i)$.

We can identify misbehaviour from party i by seeing its partial signature as a signature for the keypair $(\mathbf{s}_i, \mathbf{b}_i)$ and running the signature verification algorithm on it. Finally, by linearity, all partial signatures can be aggregated into a single signature $(c, \mathbf{z} = \sum_i \mathbf{z}_i, \mathbf{y} = \sum_i \mathbf{y}_i)$.

While partial signatures leak the individual shares in part, security is guaranteed as long as there remains sufficient entropy in the secret, even after such a leak, notably when the adversary learns a biased center for a Gaussian-distributed secret or gains hints about it. Fortunately, the MLWE assumption is resilient to certain biases:

- It is possible to reveal public hints about the secret vector $\mathbf{s}$ of the form $c \cdot \mathbf{s} + \mathbf{r}$ (when $\mathbf{s}$ and $\mathbf{r}$ are Gaussian with appropriate parameters) without compromising the uniformity of $\mathbf{b}$ —this is referred to as the Hint-MLWE assumption [KLSS23], which tightly reduces to MLWE.
- An adversary may introduce an offset in $\mathbf{b}$, but $\mathbf{b}$ will still appear uniformly random as long as the offset is independent of $\mathbf{s}$.

Thus, the security of our basic scheme boils down to the MSIS and Hint-MLWE problems. Next, we aim to transform this basic N-out-of-N threshold scheme into a generic T-out-of-N TSS while retaining efficient IA. The natural approach for this conversion is to replace additive secret sharing with Shamir's secret sharing. However, this no longer guarantees that the partial secret keys $\mathbf{s}_i$ remain small, and it now requires additional steps to achieve security (e.g., [DKM+24] requires pairwise ephemeral one-time pads to utilize this technique). It also breaks the IA property, as partial signatures are no longer valid as they are not short.

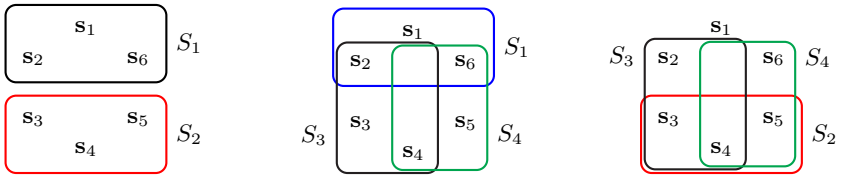
2.4 Two T-out-of-N TSS constructions from Short Secret-Sharing

With Replicated Secret Sharing. A first way to circumvent this issue is through (additive) T-out-of-N *replicated* secret sharing. Introduced in [ISN87], multiple threshold schemes have used replicated secret sharing due to its purely additive structure. To our knowledge, the threshold decryption scheme of [BD10] is the only previous scheme that used replicated secret sharing with small shares; it is, however, worth noting that additional difficulties arise when considering signatures rather than decryption since adding small shares can reveal secret information. Each user i knows a tuple of short secrets $(\mathbf{s}_1^i, \ldots, \mathbf{s}_n^i)$ such that: (i) any set S of T users can reconstruct the secret $\mathbf{s}$ as a subset-sum of the secrets $(\mathbf{s}_j^i)_{j\in S, i\in[n]}$, and (ii) no set of less than T users can reconstruct $\mathbf{s}$. Given such a secret sharing and a function $\mathcal{F}$ that maps sets of T users to the appropriate subset of partial secrets (i.e., $\mathcal{F}(S,i) = \{i_1, \ldots, i_k\}$ corresponds to the indices that party $i \in S$ adds to the subset sum to recover $\mathbf{s}$), one can bootstrap our previous construction into a T-out-of-N TSS by replacing $\mathbf{s}_i$ with $\mathbf{s}_i = \sum_{j\in\mathcal{F}(S,i)} \mathbf{s}_j^i$.

Such a suitable secret sharing can be constructed with $\binom{N}{T-1}$ secrets in total, and this number is optimal—we prove this fact in all generality in Section D.1. While it may seem prohibitive, our construction is very efficient for small parameters, such as $N = 16$. Indeed, while the *computational* complexity of the signature is linear in the number of secrets, the public key is unchanged, and the secret keys can be compressed using seeds and trees. Perhaps counterintuitively, adding up to $\binom{N}{T-1}$ secrets to constructing the partial verification key $\mathbf{s}_i$ has *virtually no impact* on the signature size. This is thanks to our use of the Hint-MLWE assumption [DKM+24, ENP24, BKL+24] allowing us to largely decorrelate the signature size from the secret key size when setting parameters.

With (T_1, T_2)-out-of-N TSS from Ramp Secret Sharing. Since there is no construction for purely additive secret sharing with a size lower than $\binom{N}{T-1}$, we can also consider a more general setting for threshold secret sharing, where up to T_1 parties can be corrupted, and $T_2 > T_1$ are needed to sign (in T-out-of-N TSSs, we would simply use $T_2 = T_1 + 1$, but we consider larger gaps in all generality). Our construction relies on the *phase transition phenomenon* of the so-called *coupon collector problem*—i.e., that a quasi-linear number of samples are required to collect a full collection of random coupons.

The secret key $\mathbf{s}$ is now shared as $\mathbf{s} = \mathbf{s}_1 + \ldots + \mathbf{s}_n$, and each user $i \in [N]$ is given the secret $\mathbf{s}_{\boldsymbol{\pi}(i)}$, where $\boldsymbol{\pi}$ is a PRF from $[N]$ to $[n]$. Under standard arguments, a random set of $T_1 \ll n \log n$ users will, with overwhelming probability (w.o.p.), not be able to reconstruct the secret, while a set of $T_2 \gg n \log n$ users will w.o.p be able to reconstruct it. While this idea provides a ramp secret sharing, its parameters are far from satisfactory. We can show that to have a probability $2^{-\kappa}$ that T_1 users cannot recover $\mathbf{s}$ and that T_2 users can, asymptotically we obtain $\frac{T_2}{T_1} \underset{n\to\infty}{\sim} 1 + \frac{\kappa \ln 2 + \ln(\kappa \ln 2)}{\ln n}$. This ratio tends towards 1 but with a slow decay.



(a) Not 2-private. (b) 3-correct. (c) Not 3-correct

Fig. 2: Example of a 3-out-of-4 secret sharing that is neither perfectly 2-private nor 3-correct. Here, the users 1, 2, 3 and 4 are (randomly) given the sets $S_1 = \{\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_6\}$, $S_2 = \{\mathbf{s}_3, \mathbf{s}_4, \mathbf{s}_4\}$, $S_3 = \{\mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_4\}$ and $S_4 = \{\mathbf{s}_4, \mathbf{s}_5, \mathbf{s}_6\}$ respectively. Since T_1-privacy and T_2-correctness are only achieved with constant probability, we amplify both using the techniques described above.

Amplifying Correctness. To avoid this issue, we can use a constant bound for the probability that T_2 parties can reconstruct $\mathbf{s}$, and use amplification to make it overwhelming. We secret share *the same secret* $\mathbf{s}$, m times in parallel:

$$\mathbf{s} = (\mathbf{s}_1^1 + \ldots + \mathbf{s}_n^1) = \ldots = (\mathbf{s}_1^m + \ldots + \mathbf{s}_n^m)$$

For each $j \in [m]$, this is efficiently done by sampling small $\mathbf{s}_1^j, \ldots, \mathbf{s}_{n-1}^j$, then setting $\mathbf{s}_n^j = \mathbf{s} - \sum_{i=1}^{n-1} \mathbf{s}_i^j$. For instance, suppose that the probability that T_2 parties possess all the $\mathbf{s}_i$ needed to reconstruct $\mathbf{s}$ is $1/2$ instead of $1 - 2^{-\kappa}$, then the probability that a set of T_2 users knows all the secrets in at least one row is $\geqslant 1 - 2^{-m}$. To sign, a set of T_2 users has to agree (efficiently) on a row where all the secrets are known and use the previous signing algorithm. The refined asymptotic ramp ratio now roughly satisfies $\frac{T_2}{T_1} \underset{n\to\infty}{\sim} 1 + \frac{\ln(\kappa \ln 2)}{\ln n} + \frac{\kappa \ln 2}{m \ln n}$.

Amplifying Privacy. We can also apply the same amplification idea to T_1. We will consider p distinct secrets $\mathbf{s}_1, ..., \mathbf{s}_p$, each being shared m times and set the signing key as $\mathbf{s} = \mathbf{s}_1 + \ldots + \mathbf{s}_p$. In other words, we have a two-dimensional secret sharing of $\mathbf{s}$. To reconstruct $\mathbf{s}$, a set of parties needs to be able to reconstruct all $\mathbf{s}_i$. For instance, assuming that the probability that T_1 parties can reconstruct one secret is $1/2$ instead of $2^{-\kappa}$, T_1 parties can reconstruct the whole secret with probability 2^{-p}. The asymptotic ramp ratio becomes $\frac{T_2}{T_1} \underset{n\to\infty}{\sim} 1 + \frac{\ln(\kappa/p \ln 2)}{\ln n} + \frac{(\kappa + \log p) \ln 2}{m \ln n}$. This extra amplification slightly reduces the gap between T_1 and T_2. As many parameters are now involved, however, it requires a careful analysis to find a good tradeoff between the probability that a set of T_1 users can reconstruct the secret (which must stay negligible) and the probability that a set of T_2 users can reconstruct the secret (which shall remain overwhelming). All in all, we get a ramp secret sharing where the gap factor is at most 8, as shown in Fig. 8.

Comparison of our two sharing methods. Our two methods offer a tradeoff between scalability and tightness. On the one hand, we build on Replicated Secret Sharing to build a standard sharing, where up to $T - 1$ parties can be corrupted. However, with a number of shares growing exponentially with T, this

approach only supports small thresholds (up to 16 parties). On the other hand, we propose a Ramp Secret Sharing, which scales even to large thresholds T, but has a small gap between the reconstruction and security thresholds.

2.5 Abstracting the Requirements: Short DKG with Leaks

Both solutions require a secret sharing scheme with specific properties, modelable as an enhanced *Distributed Key Generation* (DKG) protocol. To ensure the IA property and security, shares must remain *short* and reconstruction coefficients *small*. *Linear evaluation* over honest shares is crucial for functional correctness, and the secret must be protected despite statistical leaks from corrupted shares. Standard DKG *zero-knowledge* properties are too stringent due to the leaks, thus requiring a weaker simulatability notion. For usability, functional evaluation over honest shares should not leak them. However, common simulation strategies, relying only on public and corrupted shares, fail for functions that preserve shortness. To address this, we introduce an extension for the ramp setting and functional evaluation with partial leakage, leveraging the *fragmentary knowledge* property introduced in [ENP24] for *Verifiable Short Secret Sharing* schemes. This ensures public key security despite small leakages. Our simulatability notion guarantees that the adversary's view (corrupted users' views and function evaluations of honest shares) can be simulated from public material sampled from a target distribution. We coin this key generation protocol *Distributed Key Generation with Short Shares*, or sDKG (see Section 4).

2.6 Correlation Detection via Inner Product Testing

We can also reflect on the minimal requirements and the geometric interpretation of the IA property. From an abstract viewpoint, we can model the partial signatures as T normally distributed vectors $X_0, \ldots, X_{T-1} \in \mathbb{R}^d$ and consider their aggregation $X = \sum_i X_i$. We expect vectors produced by honest parties to have norms concentrated around a public constant B. Therefore, partial signatures that exceed B should be rejected. However, this crude bound only guarantees that the final signature will verify $\|X\| \leqslant T\,B$.

Dishonest parties, however, could collude to produce partial signatures X_i that are highly correlated (see Fig. 3a), thus increasing the norm of the sum X while individually staying within bounds close to B. This would allow them to approach the worst-case bound $T\,B$ while still appearing to behave honestly. In contrast, if all parties are honest, the sum X should grow as $\sqrt{T}\,B$. Hence, if X grows significantly bigger than $\sqrt{T}\,B$, we can infer that a subset of signers has colluded and introduced correlation. We formalize this intuition. By the linearity of expectation and trace and bounding the covariance, we get:

$$T \leqslant \mathbb{E}\left[\|X\|^2\right] = \sum_i \mathbb{E}\left[\|X_i\|^2\right] + \sum_{i \neq j} \mathrm{Tr}(\mathrm{Cov}(X_i, X_j)) \leqslant T^2,$$

where both bounds are tight: the lower bound holds for independent variables, and the upper bound applies to maximally correlated variables sharing the same direction, see Fig. 3 for an illustration.



(a) Worst-case (colinear) (b) Average-case (random)

Fig. 3: Representing the sum X of 3 unitary vectors: worst-case vs average-case.

A classical analysis based on the concentration of Gaussian mass would assert that the sum is close to its expected value. Thus, the gap (of size $\sqrt{T}$) provides enough information to detect correlations: the more correlated the vectors are, the further the result deviates from the lower bound. Therefore, a vector X_i is deemed valid only if it lies on the gray spherical cap of Fig. 4. Algebraically, this translates into a simple inner product test of the form: $\frac{\langle X_i, X - X_i \rangle}{\|X - X_i\|^2} < \alpha$ for a properly chosen α. We call this the *Death Star* test, due to its resemblance to a certain space station from a galaxy far, far away. Fig. 5 illustrates the efficiency of our technique (—✳—) compared to the (naive) Minkowski bound (—■—). In practice, for the parameters selected in our TSS, this translates in gaining *for free* between **10** and **30** bits of security on forgery.

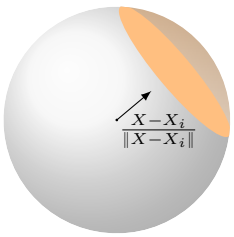


Fig. 4: Visual illustration of the Death Star algorithm. The orange cap is the rejection body.

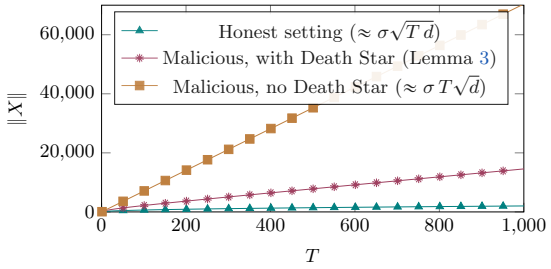


Fig. 5: Norm of $X = \sum_{i \in [T]} X_i$, for X_i being Gaussians of parameter $\sigma = 1$, dimension 4096, 128 bits of security, and $T \leqslant 1000$.

3 Preliminaries

General notations. For an integer $N > 0$, we note $[N] = \{1, \ldots, N\}$. To denote the assign operation, we use $y := f(x)$ when f is deterministic and $y \leftarrow f(x)$ when randomized. When S is a finite set, we note $\mathcal{U}(S)$ the uniform distribution over S, and shorthand $x \xleftarrow{\$} S$ for $x \leftarrow \mathcal{U}(S)$. For a fixed power-of-two d, we note $\mathcal{K} = \mathbb{Q}[x]/(x^d + 1)$ and $\mathcal{R} = \mathbb{Z}[x]/(x^d + 1)$, and $\mathcal{R}_q = \mathcal{R}/(q\mathcal{R})$. Given $\mathbf{x} \in \mathcal{K}^\ell$, we abusively note $\|\mathbf{x}\|$ the Euclidean norm of the $(d\,\ell)$-dimensional vector of the coefficients of $\mathbf{x}$. Unless specified otherwise, vectors are treated as *column*

vectors. We assume familiarity with Landau's asymptotic notations: $O(n)$, $o(n)$, $\omega(n)$, $\Omega(n), a(n) \sim b(n)$, and Knuth's Theta notation $\Theta(n)$.

We make use of functions $\mathbf{req}(b)$ in individual algorithms and $\mathbf{assert}(b)$ within games for checking a boolean statement. If b is false, **req** makes the algorithm immediately return 0, **assert** aborts the entire game and returns 0.

Definition 1 (Discrete Gaussians). *Given a positive definite $\Sigma \in \mathbb{R}^{m \times m}$, we note $\rho_{\sqrt{\Sigma}}$ the Gaussian function defined over $\mathbb{R}^m$ as $\rho_{\sqrt{\Sigma}}(\mathbf{x}) = \exp\left(-\frac{\mathbf{x}^t \cdot \Sigma^{-1} \cdot \mathbf{x}}{2}\right)$. The above definition may be extended to $(\Sigma, \mathbf{x}) \in \mathcal{K}^{m \times m} \times \mathcal{K}^m$ by treating Σ as a block-circulant matrix in $\mathbb{R}^{nm \times nm}$ and $\mathbf{x}$ as a vector in $\mathbb{R}^{nm}$.*

We may note $\rho_{\sqrt{\Sigma},\mathbf{c}}(\mathbf{x}) = \rho_{\sqrt{\Sigma}}(\mathbf{x} - \mathbf{c})$. When Σ is of the form $\sigma \cdot \mathbf{I}_m$, where $\sigma \in \mathcal{K}^{++}$ and $\mathbf{I}_m$ is the identity matrix, we note $\rho_{\sigma,\mathbf{c}}$ as shorthand for $\rho_{\sqrt{\Sigma},\mathbf{c}}$.

For any countable set $S \subset \mathcal{K}^m$, we note $\rho_{\sqrt{\Sigma},\mathbf{c}}(S) = \sum_{\mathbf{x} \in \mathcal{K}^m} \rho_{\sqrt{\Sigma},\mathbf{c}}(\mathbf{x})$ whenever this sum converges. Finally, when $\rho_{\sqrt{\Sigma},\mathbf{c}}(S)$ converges, the discrete Gaussian distribution $D_{S,\mathbf{c},\sqrt{\Sigma}}$ is defined over S by its probability distribution function $D_{S,\sqrt{\Sigma},\mathbf{c}}(\mathbf{x}) = \frac{\rho_{\sqrt{\Sigma},\mathbf{c}}(\mathbf{x})}{\rho_{\sqrt{\Sigma},\mathbf{c}}(S)}$.

Modulus Rounding Let $\nu \in \mathbb{N} \backslash \{0\}$. Any integer $x \in \mathbb{Z}$ can be *uniquely* decomposed as: $x = 2^\nu \cdot x_\top + x_\bot, \quad (x_\top, x_\bot) \in \mathbb{Z} \times [-2^{\nu-1}, 2^{\nu-1} - 1]$, which consists essentially of separating the lower-order bits from the higher-order ones. We define the function $\lfloor \cdot \rceil_\nu : \mathbb{Z} \to \mathbb{Z}$ such that $\lfloor x \rceil_\nu = \lfloor x/2^\nu \rceil = x_\top$, where $\lfloor \cdot \rceil : \mathbb{R} \mapsto \mathbb{Z}$ denotes the rounding operator. More precisely the "rounding half-up" method $\lfloor x \rceil = \lfloor x + \frac{1}{2} \rfloor$ where half-way values are rounded up: e.g. $\lfloor 2.5 \rceil = 3$ and $\lfloor -2.5 \rceil = -2$. With a slight overload of notation, when $q > 2^\nu$, we extend $\lfloor \cdot \rceil_\nu$ to take inputs in $\mathbb{Z}_q$, in which case, we assume the output is an element in $\mathbb{Z}_{q_\nu}$ where $q_\nu = \lfloor q/2^\nu \rceil$. Formally, we define:

$$\lfloor \cdot \rceil_\nu : \mathbb{Z}_q \mapsto \mathbb{Z}_{q_\nu} = \mathbb{Z}_{\lfloor q/2^\nu \rceil} \quad \text{s.t.} \quad \lfloor x \rceil_\nu = \lfloor \bar{x}/2^\nu \rceil + q_\nu \mathbb{Z} = (\bar{x})_\top + q_\nu \mathbb{Z},$$

where $\bar{x}$ is the lifting of x to $[0, q-1]$. The function $\lfloor \cdot \rceil_\nu$ naturally extends to vectors coefficient-wise.

Hardness Assumptions We rely on the SelfTargetMSIS problem [DKL+18], [KLS18]. In this variant of MSIS, the target is defined relative to a hash function. SelfTargetMSIS is shown to be as hard as MSIS in the random oracle model when the range of the hash function is exponentially large with a standard proof using the forking lemma [FS87, BN06]. We will directly work with SelfTargetMSIS, and will estimate its concrete security based on the best known attacks.

Definition 2 (SelfTargetMSIS). *Let k, ℓ, q be integers and $B_{\mathsf{stmsis}} > 0$ be a real number. Let $\mathcal{C}$ be a subset of $\mathcal{R}_q$, and let $G : \mathcal{R}_q^k \times \{0,1\}^{2\kappa} \mapsto \mathcal{C}$ be a cryptographic hash function modeled as a random oracle. The advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{SelfTargetMSIS}}(\kappa)$ of an adversary $\mathcal{A}$ against $\mathsf{SelfTargetMSIS}_{q,k,\ell,B_{\mathsf{stmsis}}}$ is defined as:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{SelfTargetMSIS}}(\kappa) = \Pr\Big[\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{k \times \ell}, (\mathsf{msg}, \mathbf{z}) \leftarrow \mathcal{A}^G(\mathbf{A}), (\mathsf{msg}, \mathbf{z}) \in \{0,1\}^{2\kappa} \times \mathcal{R}_q^{\ell+k} :$$

$$\left(\mathbf{z} = \begin{bmatrix}\mathbf{z}'\\ c\end{bmatrix}\right) \wedge (\|\mathbf{z}\|_2 \leqslant B_{\mathsf{stmsis}}) \wedge G(\left[\mathbf{I}\ \mathbf{A}\right] \cdot \mathbf{z}, \mathsf{msg}) = c\Big].$$

The $\mathsf{SelfTargetMSIS}_{q,k,\ell,\beta}$ *assumption states that any efficient adversary* $\mathcal{A}$ *has a negligible advantage* $\mathsf{Adv}^{\mathsf{SelfTargetMSIS}}_{\mathcal{A}}(\kappa)$.

We also rely on the Hint-MLWE assumption [KLSS23], a variant of MLWE where the adversary additionally obtains some *noisy leakage* of the MLWE sample. We recall below a slightly generalized statement for Hint-MLWE introduced in [ENP24], where the noise distribution may be distinct for each hint.

Definition 3 (Hint-MLWE). *Let* k, ℓ, q, Q *be integers,* $\mathcal{D}_{\mathbf{s}}, (\mathcal{D}^{(i)}_{\mathbf{r}})_{i\in[Q]}$ *be probability distributions over* $\mathcal{R}^{k+\ell}$*, and* $\mathcal{C}$ *be a distribution of challenges over* $\mathcal{R}^{Q}$*. The advantage* $\mathsf{Adv}^{\mathsf{Hint\text{-}MLWE}}_{\mathcal{A}}(\kappa)$ *of an adversary* $\mathcal{A}$ *against the Hint Module Learning with Errors problem* $\mathsf{Hint\text{-}MLWE}_{q,k,\ell,Q,\mathcal{D}_{\mathbf{s}},(\mathcal{D}^{(i)}_{\mathbf{r}})_i,\mathcal{C}}$ *is defined as:*

$$\left|\Pr\left[1 \leftarrow \mathcal{A}\left(\mathbf{A}, \left[\mathbf{I}_k\ \mathbf{A}\right] \cdot \mathbf{s}, (c_i, \mathbf{z}_i)_{i\in[Q]}\right)\right] - \Pr\left[1 \leftarrow \mathcal{A}\left(\mathbf{A}, \mathbf{u}, (c_i, \mathbf{z}_i)_{i\in[Q]}\right)\right]\right|,$$

where $(\mathbf{A}, \mathbf{u}) \xleftarrow{\$} \mathcal{R}_q^{k\times\ell} \times \mathcal{R}_q^{k}$, $\mathbf{s} \leftarrow \mathcal{D}_{\mathbf{s}}^{k+\ell}$ *and for* $i \in [Q]$*:* $c_i \leftarrow \mathcal{C}$, $\mathbf{r}_i \leftarrow \left(\mathcal{D}^{(i)}_{\mathbf{r}}\right)^{k+\ell}$, *and* $\mathbf{z}_i = c_i \cdot \mathbf{s} + \mathbf{y}_i$*. The* $\mathsf{Hint\text{-}MLWE}_{q,k,\ell,Q,\mathcal{D}_{\mathbf{s}},(\mathcal{D}^{(i)}_{\mathbf{r}})_i,\mathcal{C}}$ *assumption states that any efficient adversary* $\mathcal{A}$ *has a negligible advantage.*

We recover the classical MLWE *problem by taking* $Q = 0$*:* $\mathsf{MLWE}_{q,k,\ell,\mathcal{D}_{\mathbf{s}}} = \mathsf{Hint\text{-}MLWE}_{q,k,\ell,Q=0,\mathcal{D}_{\mathbf{s}},\varnothing,\mathcal{C}}$.

Kim et al. [KLSS23] proved that for discrete Gaussians, Hint-MLWE is as hard as MLWE in some regimes. Espitau et al. [ENP24] extended that result to discrete Gaussians with varying standard deviations. This reduction is essentially optimal and the noise standard deviation has to grow with $\sqrt{Q}$ to ensure security as demonstrated in [ASY22, Section 3.3]. We recall this result below.

Theorem 1 (Hardness of Hint-MLWE with varying noises). *Let* k, ℓ, Q, q*, be positive integers,* $\mathcal{C}$ *be a distribution over* $\mathcal{R}^{Q}$*,* $\sigma_{\mathbf{s}}$ *the standard deviation of the secret vector, and* $(\sigma^{(i)}_{\mathbf{r}})_{i\in[Q]}$ *noise standard deviations. Let* $\sigma > 0$ *be a real number defined as* $\frac{1}{\sigma^2} = 2 \cdot \left(\frac{1}{\sigma_{\mathbf{s}}^2} + \sum_{i\in[Q]} \frac{s_1(c_i^{*} c_i)}{(\sigma^{(i)}_{\mathbf{r}})^2}\right)$*. If* $\sigma \geqslant \sqrt{2} \cdot \eta_{\varepsilon}(\mathbb{Z}^{d(k+\ell)})$ *for some* $\varepsilon \in (0, 1/2]$*, then there exists an efficient reduction from* $\mathsf{MLWE}_{q,k,\ell,\sigma}$ *to* $\mathsf{Hint\text{-}MLWE}_{q,k,\ell,Q,\sigma_{\mathbf{s}},(\sigma^{(i)}_{\mathbf{r}})_{i\in[Q]},\mathcal{C}}$ *that reduces the advantage by at most* 2ε.

The Coupon Collector's Problem

Definition 4. *Let* $S = [n]$*. Starting from* $i = 1$*, we sample with replacement* $x_i \leftarrow S$*, until every element in* S *has been sampled. Let us note* T_n *the random variable that gives the number of iterations of this sampling process. We refer to the mathematical analysis of* T_n *as the coupon collector's problem.*

Lemma 1 (Lemmas 1.9.2 and 1.9.3 of [Doe18]). *For all $\varepsilon_{\min}, \varepsilon_{\max} \geqslant 0$:*

$$\mathbb{P}[T_n \leqslant (1-\varepsilon_{\min})\,(n-1)\,\ln n] \leqslant \exp\left(-n^{\varepsilon_{\min}}\right) \tag{1}$$

$$\mathbb{P}[T_n \geqslant (1+\varepsilon_{\max})\,n \ln n] \leqslant n^{-\varepsilon_{\max}} \tag{2}$$

In particular, for positive constants κ, m and C, setting $\varepsilon_{\min} = \frac{\ln C}{\ln n}$ and $\varepsilon_{\max} = \frac{C}{\ln n}$ ensures that the rightmost terms in Eqs. (1) and (2) are equal to e^{-C}.

4 Distributed Key Generation with Short Shares

In this section, we introduce our notion of Distributed Key Generation with Short Shares (sDKG). This refers to an extension to the notion of Distributed Key Generation (DKG) that constrains the output of DKG to: (i) have short shares, (ii) have binary reconstruction coefficients, and (iii) allow one to perform functional evaluation over the honest shares. To allow for these constrained shares, we allow a partial leakage of the secret. Our definition additionally removes the need for an honest majority in the definition of a robust DKG.

4.1 Definition

We recall the notion of Distributed Key Generation (DKG), notably formalized in [KGS23]. We extend it to allow each user to possess several shares of the secret, and simplify it for the case where the reconstruction is a linear function.

Definition 5 (Distributed Key Generation). *A Distributed Key Generation (DKG) is defined for a reconstruction threshold $T \leqslant N$ as a tuple of algorithms* $(\mathsf{Keygen}, \mathsf{Recover}, \mathsf{Verify})$ *where:*

- $\mathsf{Keygen}(1^\kappa, N, T) \to (\mathsf{pk}, \mathsf{aux}, (\mathsf{sk}_i := (S_i, \cdot), \mathsf{status}_i)_{i\in[N]})$ *is an interactive protocol between N participants producing:*
  - *A public key* pk.
  - *Auxiliary information* aux *that may be useful for a particular application.*
  - *A partial private key* sk_i *for each participant, containing of a pool of shares S_i. We consider that secrets are indexed and secret u of user i is accessed with $S_i[u]$.*
  - *A termination status* status_i *for each user indicating whether the protocol succeeded for user i. It can be either* **abort** *(with a set of identified malicious parties) or* **accept**.
- $\mathsf{Recover}(\mathsf{pk}, \mathsf{aux}, \mathsf{act}) \to (\nu_{i,u})_{i\in\mathsf{act}, u\in[|S_i|]}$ *is the algorithm providing coefficients allowing the reconstruction of a common secret trapdoor from individual shares for any given set* act *of at least T parties.*
- $\mathsf{Verify}(\mathsf{pk}, \mathsf{aux}, (\mathsf{sk}_i)_{i\in\mathsf{honest}}) \to \{0,1\}$ *verifies that the public key, auxiliary information and partial private keys produced by a* DKG *are consistent.*

Formally, we define the interactive protocol Keygen *as a synchronous round-based protocol following the syntax of [KGS23]. It is decomposed into a setup algorithm* Setup, $\mathsf{rnd}_{\mathsf{Keygen}}$ *algorithms* $(\mathsf{ShareKeygen}_i)_{i\in\{1,\dots,\mathsf{rnd}_{\mathsf{Keygen}}\}}$, *and two aggregation algorithms* $\mathsf{CombinePrivKey}, \mathsf{CombineKey}$. *For the status report of each party, and the identification of malicious behavior, we adopt the syntax of [IOZ14]. In detail, these algorithms are:*

- $\mathsf{Setup} : \varnothing \to (\mathsf{st}_i)_{i\in[N]}$ *sets up the initial state of all the parties.*
- $\mathsf{ShareKeygen}_1 : (\mathsf{st}_i, N, T) \to (\mathsf{st}_i, \mathsf{pm}_1^i$ *or* $(\mathbf{abort}, \mathcal{S}_\perp^i))$ *generates the message for the first round of the key generation for party* i, *or aborts and identifies misbehaving parties in* $\mathcal{S}_\perp^i$.
- $\mathsf{ShareKeygen}_j : (\mathsf{st}_i, \mathsf{pm}_{j-1}^i) \to (\mathsf{st}_i, \mathsf{pm}_j^i$ *or* $(\mathbf{abort}, \mathcal{S}_\perp^i))$ *generates the message for the* j*-th round of the key generation for party* i, *or identifies misbehaving parties.*
- $\mathsf{CombinePrivKey} : \mathsf{st}_i \to \mathsf{sk}_i$ *combines the private key of party* i.
- $\mathsf{CombineKey} : (\mathsf{pm}_j^i)_{i\in[N],j\in[\mathsf{rnd}_{\mathsf{Keygen}}]} \to (\mathsf{pk}, \mathsf{aux})$ *or* $(\mathbf{abort}, \mathcal{S}_\perp)$ *takes as input all the messages of the protocol execution and outputs a public key, and auxiliary information, or aborts with a set of misbehaving parties.*

A DKG typically requires several properties:

1. **Robustness (or IA):** If up to $T_2 < T$ parties behave dishonestly, the DKG guarantees that honest parties terminate with the same status (always **accept** for robustness, possibly **abort** for IA, but with an identified set of misbehaving parties), and if they accept, we have:
   - Honest parties agree on a public key pk and auxiliary information aux.
   - Given any set $\mathsf{act} \subseteq \mathsf{honest}$ of cardinality T, and noting $(\nu_{i,u})_{i\in\mathsf{act},u} := \mathsf{DKG.Recover}(\mathsf{pk}, \mathsf{aux}, \mathsf{act})$, it holds w.o.p. that $\sum_{i\in\mathsf{act},u\in[|S_i|]} \nu_{i,u} \cdot S_i[u] = \mathbf{s}$., i.e. honest shares are able to reconstruct the secret $\mathbf{s}$.
   - The honest shares pass the keypair verification of the DKG, i.e. $\mathsf{DKG.Verify}(\mathsf{pk}, \mathsf{aux}, (\mathsf{sk}_i)_{i\in\mathsf{honest}}) = 1$.
2. **Zero-Knowledge:** If up to $T_2 < T$ parties are corrupted, their view in the DKG execution can be simulated from only genuine public key material.
3. **Indistinguishability:** The keypair output by the DKG must be indistinguishable from the output of a perfect target key generation algorithm.

Note that achieving robustness/IA requires a strong communication model to tackle issues such as drop of messages, or inconsistent messages sent to parties, and still allow parties to reliably identify cheating parties. We thus adopt a synchronous round-based model, ensuring that messages from a round are reliably and consistently sent to all parties in the next round. Concretely, this can be implemented using an authenticated synchronous broadcast channel, or a public bulletin board. We now introduce in Definitions 6 and 7 the security notions underlying a DKG with short shares (sDKG).

Definition 6 (sDKG robustness/IA). *Let* $K, \eta_1, \eta_2 \in \mathbb{N}$ *be parameters. We say that a* sDKG *is* (K, η_1, η_2)*-robust (resp.* (K, η_1, η_2)*-IA) if the* sDKG *is robust (resp. IA) as defined above and the final honest shares* $\mathsf{pk}, \mathsf{aux}, (S_i)_{i\in\mathsf{honest}}$ *have the following properties:*

– ***Short shares***, *i.e. small cardinality* $|S_i| \leqslant K$, *and small norm* $\|S_i[u]\| \leqslant \eta_1$ *for* $i \in \mathsf{honest}, u \in [|S_i|]$.
– ***Small reconstruction coefficients***, *i.e. for any set* act *of cardinality* T *and* $(\nu_{i,u})_{i\in\mathsf{act},u} := \mathsf{Recover}(\mathsf{act})$, *then* $|\nu_{i,u}| \leqslant \eta_2$.

We also want a notion of simulatability for the secrets produced by a sDKG, to prove that the keys produced by an sDKG remain protected against an adversary. In particular, we want to cover the leakage induced by having short shares – intrinsically leaking the full secret, and of evaluating leaky functions over the honest shares – which are used to preserve their shortness. The zero-knowledge security property of a DKG is however not compatible with such leakages. Therefore, we shall require a weakened notion of zero-knowledge to support the leakage induced by having small secret shares. Additionally, the DKG notion of *indistinguishability* is not required for many lattice-based schemes, and we will deem shortness requirements in the robustness key verification check to be sufficient.

We identify two requirements to capture the leakage resilience of a sDKG: (i) the public key should remain secure and appear as sampled from an ideal distribution for an adversary corrupting up to T parties, and (ii) it is later possible to simulate the evaluation of a function over the honest shares without impacting the security of the public key. We cover this by introducing *functional simulatability*, which simulates an adversary's transcript in the sDKG, as well as evaluations over honest shares only from an honestly generated public key.

Definition 7 (sDKG functional simulatability). *We say that a (distributed)* sDKG *is* T_{corrupt}*-simulatable for a target key distribution* χ *and a function* $f : (\mathsf{pk}, \mathsf{aux}, (S_i)_{i\in\mathsf{honest}}, \mathsf{in}) \to V$ *if there exist two functions* SimKeygen, SimF *such that any adversary* $\mathcal{A}$ *has advantages in the games* $(\mathsf{Game}^b_{\mathsf{sDKG\text{-}sim}})_{b\in\{0,1\}}$ *that are negligibly close for sets of up to* T_{corrupt} *corrupted users, where:*

– *A simulator* $\mathsf{SimKeygen} : (\mathcal{A}, \mathsf{corrupt}, \mathsf{pk}) \to (\mathsf{aux}, \mathsf{st}, \mathsf{status})$ *that simulates the* sDKG *protocol execution for the adversary given a public key sampled from* χ*, auxiliary information, and computes bias information in* st *for the functional evaluation. It also simulates key generation aborts with* status*. Typically, the public key is sampled from a perfect uniform distribution, while the private shares are potentially biased by the adversary.*
– $\mathsf{SimF} : (\mathsf{pk}, \mathsf{aux}, \mathsf{st}, \mathsf{in}) \mapsto V$ *simulates the evaluation of the function* f *with public key information, the bias information from* st *and input* in.

Note: in the random oracle model, we allow the simulator to program the random oracle.

5 Instantiating sDKG for Lattice-Based Schemes

The notion of sDKG is particularly useful for lattice-based schemes where shortness is crucial notably for trapdoor sampling, and in which a bias or partial leakage of the private keys does not necessarily lead to a security issue. Short shares

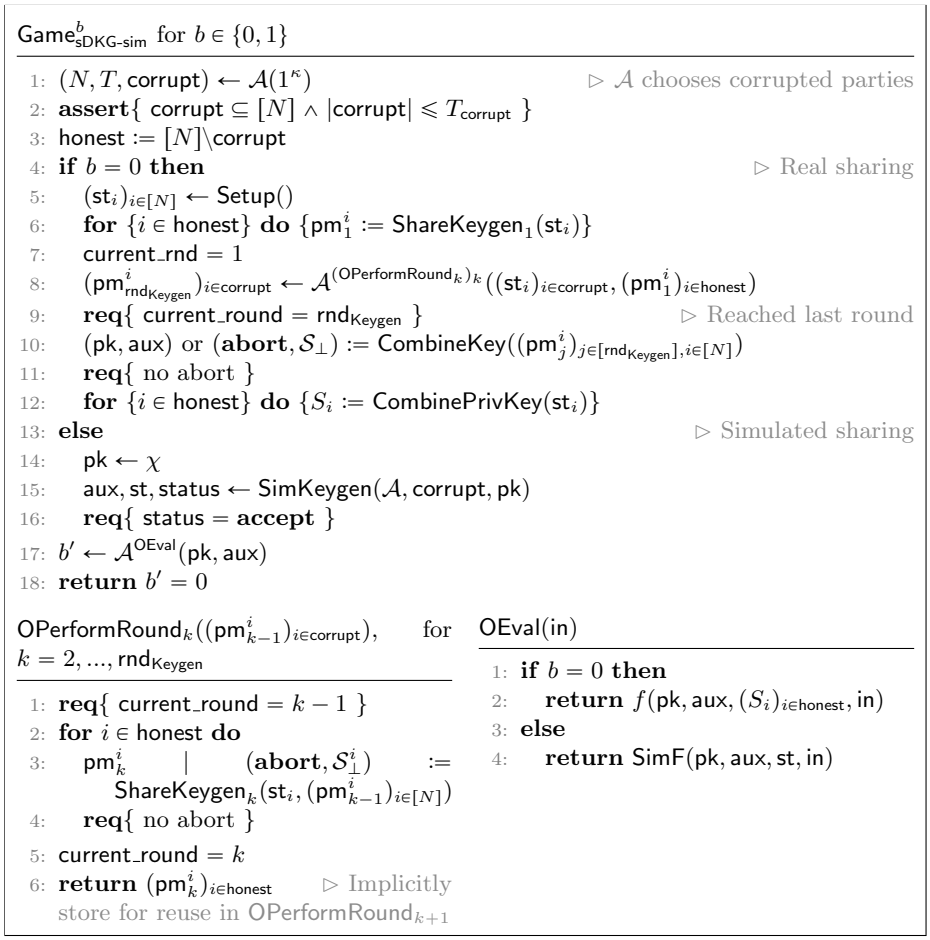


Fig. 6: Simulatability game for a sDKG scheme. The adversary $\mathcal{A}$ wins if the game sDKG-sim returns 1, i.e. if it is guessed correctly whether it is provided with real or simulated shares. Failed assertions stop the game and return a random bit.

also help achieve practical schemes with more advanced properties. Specifically, we are interested in a sDKG that samples public keys perfectly from a target distribution, despite of bias and partial leakage of its (short) private keys. Functions of honest shares (with partial leakage) also appear of interest as will be illustrated in our application section, by allowing one to evaluate over honest partial private keys with leakage without compromising the perfect sampling of the public key. We start by detailing the properties required for a sDKG crafted for lattice-based schemes and then provide two possible instantiations.

5.1 Properties of a sDKG for lattice-based schemes

For lattice-based schemes, the public key typically consists in a $\mathsf{seed} \in \{0,1\}^\kappa$ used to derive a public matrix $\mathbf{A} \in \mathcal{R}_q^{k \times \ell}$ and a polynomial $\mathbf{b} = \lfloor \widehat{\mathbf{b}} \rceil_{\nu_\mathbf{b}} \in \mathcal{R}_{q_{\nu_\mathbf{b}}}^k$, where $\widehat{\mathbf{b}} \in \mathcal{R}_q^k$ is an (indistinguishable from) uniform vector for which the $\nu_\mathbf{b}$ lower bits are rounded. In order to perform operations, schemes rely on a computational assumption such as MLWE or NTRU to sample a short trapdoor $\mathbf{s}$ and compute a vector $\widehat{\mathbf{b}}$ as a function of $\mathbf{s}$ so that $\widehat{\mathbf{b}}$ appear uniformly sampled in $\mathcal{R}_q^k$. We focus here on the case of MLWE, where $\widehat{\mathbf{b}} = \left[\mathbf{A}\ \mathbf{I}\right] \cdot \mathbf{s}$. Interestingly, the MLWE assumption is resilient to a number of biases:

- It is possible to reveal public hints on the secret vector $\mathbf{s}$ of the form $c \cdot \mathbf{s} + \mathbf{r}$ (when $\mathbf{s}$ and $\mathbf{r}$ are Gaussian with proper parameters) without compromising the uniformity of $\mathbf{b}$. This is referred to as the Hint-MLWE assumption [KLSS23], which tightly reduces to MLWE.
- An adversary can introduce an offset in $\widehat{\mathbf{b}}$, and $\widehat{\mathbf{b}}$ still appears uniformly random as long as the offset is independent of $\mathbf{s}$ (conditioned on the value of the hints).

Building on this observation, we show that it is possible to build DKG protocols sampling short shares, which are potentially biased and leak to the adversary, but for which the public key remains provably pseudorandom.

For concrete usability within noise-flooded schemes, we will also require the sDKG to allow one to evaluate hints of the shares, and thus partially leak the honest shares. While a recent technique relying on masking terms [DKM+24] enabled evaluation of a linear function of the secret, with non-leaky intermediary computations, this technique completely loses the shortness property of intermediary computations and prevents leveraging for advanced properties. Our techniques tackle this and open the path for advanced properties. Specifically, we want to instantiate sDKG verifying the below properties:

- **Functional simulatability.** i.e. its output public key $(\mathsf{seed}, \mathbf{b} = \lfloor \widehat{\mathbf{b}} \rceil_{\nu_\mathbf{b}})$ appears as if $(\mathsf{seed}, \widehat{\mathbf{b}})$ are uniformly sampled from $\{0,1\}^\kappa \times \mathcal{R}_q^k$ when up to T_{corrupt} parties are corrupted, and even under evaluations of hints of the honest shares. We thus select as target public key distribution $\chi := \chi_{\mathsf{seed}} \times \chi_\mathbf{b}$ where $\chi_{\mathsf{seed}} := \mathcal{U}(\{0,1\}^\kappa)$ and $\chi_\mathbf{b} := \left\lfloor \mathcal{U}(\mathcal{R}_q^k) \right\rceil_{\nu_\mathbf{b}}$ for $\nu_\mathbf{b}$ parameterizing the public key rounding. We additionally define the hint function f over the honest shares, and set act of T signers:

$f(\mathsf{pk}, \mathsf{aux}, (S_i)_{i \in \mathsf{honest}}, \mathsf{in} := \mathsf{act})$

1: $(\nu_{i,u})_{i \in \mathsf{act}, u \in [|S_i|]} := \mathsf{sDKG.Recover}(\mathsf{pk}, \mathsf{aux}, \mathsf{act})$
2: $c \leftarrow \mathcal{C}$ ▷ Sample a challenge c
3: $(\mathbf{r}_i)_{i \in \mathsf{honest} \cap \mathsf{act}} \leftarrow \mathcal{D}_\mathbf{r}^{(k+\ell) \cdot |\mathsf{honest} \cap \mathsf{act}|}$ ▷ Sample noises in $\mathcal{R}_q^{k+\ell}$
4: **return** $(c \cdot \sum_{u \in [|S_i|]} \nu_{i,u} \cdot S_i[u] + \mathbf{r}_i)_{i \in \mathsf{honest} \cap \mathsf{act}}$

– **Robustness/IA.** We also want a sDKG that achieves robustness or IA, i.e. honest parties simultaneously accept or abort, and the resulting public key and auxiliary information are consistent with honest shares.
For our sDKG, the auxiliary information shall consist of partial public keys corresponding to all the shares: $\mathbf{b}_i[u] := \left[\mathbf{A}\ \mathbf{I}\right] \cdot S_i[u]$. We explicit the sDKG verification procedure underlining the robustness/IA property of the sDKG.

$\mathsf{sDKG.Verify}(\mathsf{pk} = (\mathsf{seed}, \mathbf{b}), \mathsf{aux} = (\mathbf{b}_i[u])_{i\in[N],u}, (S_i)_{i\in\mathsf{honest}})$

1: **for** $i \in \mathsf{honest}, u \in [|S_i|]$ **do**
2: **assert**$\{\ \mathbf{b}_i[u] = \left[\mathbf{A}\ \mathbf{I}\right] \cdot S_i[u]\ \}$
3: **for** $\mathsf{act} \subseteq [N]$ s.t. $|\mathsf{act}| = T$ **do**
4: $(\nu_{i,u})_{i\in\mathsf{act},u\in[|S_i|]} := \mathsf{sDKG.Recover}(\mathsf{pk}, \mathsf{aux}, \mathsf{act})$
5: **assert**$\{\ \mathbf{b} = \left\lfloor \sum_{i\in\mathsf{act}} \sum_{u\in[|S_i|]} \nu_{i,u} \cdot \mathbf{b}_i[u] \right\rceil_{\nu_\mathbf{b}}\ \}$

In both of our sDKG, we will require a Symmetric Key Encryption algorithm (SKE) to implement non-repudiable pairwise communication. We assume that pairwise keys $\mathsf{sk}_i^{\mathsf{SKE}}$ are shared prior to the protocol execution to communicate with the dealer during a Setup phase, along with a signature $\mathsf{sig}_i = \mathsf{Sign}(\mathsf{sk}, (i, \mathsf{sk}_{i,j}))$ signed by the dealer for a verified reveal of the key. Note that requirements on these prior steps could be reduced by using non-repudiable and publicly verifiable signcryption.

5.2 Replicated Secret Sharing

We propose a first sDKG construction that builds on the replicated secret sharing technique used for Multi-Party Computation (MPC). The core idea of this technique lies in distributing a pool of secrets among the parties so that (i) any set of T parties collectively possesses all the secrets, and (ii) any set of less than T parties misses at least one secret. We extend it by using a pool of *short* secrets and introduce new techniques to sample (in a distributive way) such sharings securely and robustly.

In detail, our sDKG relies on a pool of secrets S, and individual sets of secrets $S_i \subset S$, for $i \in [N]$. The sDKG secret is the sum of all the secrets in S. We can implement properties (i) and (ii) by taking a pool of $\binom{N}{N-T+1} = \binom{N}{T-1}$ secrets. Each secret is sampled from a distribution $\mathcal{D}_\mathbf{s}$, and has a related set $I \subset [N]$ of cardinality $N - T + 1$, we note this secret $\mathbf{s}_I$. Secret $\mathbf{s}_I$ is simply distributed to all parties $i \in I$.

The secrets distribution verifies (i): given a set act of T parties, then $\mathsf{act} \cap I$ must be nonempty for any set I, so at least one party in act knows $\mathbf{s}_I$, and (ii): given a set corrupt of cardinality less than T, there exists at least one set I such that $\mathsf{corrupt} \subset [N]\backslash I$, i.e. no corrupted party is in I, and thus knows $\mathbf{s}_I$. We show in Section D.1 that if we want a pool of secrets S with the two properties mentioned above, then this construction is optimal.

To help the reader build an intuition, we first present Repl.TrustedKeygen which samples such a sDKG relying on a trusted party.

Alg. 1 $\mathsf{Repl.TrustedKeygen}(N, T) \to (\mathsf{pk}, \mathsf{ak}, \mathbf{sk})$

$\mathsf{seed} \leftarrow \{0,1\}^\kappa$
$\mathbf{A} := \mathsf{ExpandA}(\mathsf{seed})$
for $I \subset [N]$ s.t. $|I| = N - T + 1$ **do**
  ▷ List sets of $N - T + 1$ parties
  $\mathbf{s}_I \leftarrow \mathcal{D}_{\mathbf{s}}^{k+\ell}$
  $\mathbf{b}_I := \left[\mathbf{A}\ \mathbf{I}\right] \cdot \mathbf{s}_I$
  for $i \in I$ **do**
    $\mathsf{sk}_i = \mathsf{sk}_i \cup \{\mathbf{s}_I\}$
$\mathbf{b} := \left\lfloor \sum_I \mathbf{b}_I \right\rceil_{\nu_{\mathbf{b}}}$
$\mathsf{pk} := (\mathsf{seed}, \mathbf{b})$ ▷ Verification key
$\mathsf{ak} := (\mathbf{b}_I)_I$ ▷ Auxiliary key
$\mathbf{sk} := (\mathsf{sk}_i)_{i \in [N]}$
return $(\mathsf{pk}, \mathsf{ak}, \mathbf{sk})$

Alg. 2 $\mathsf{Repl.Recover}(\mathsf{act}) \to \{0,1\}^{S_1 \times S_2 \times \ldots \times S_N}$

for $i \in \mathsf{act}$ **do**
  for $I \subset [N]$ of cardinality $N - T + 1$ such that $i \in I$ **do**
    if $i = \max(I \cap \mathsf{act})$ **then**
      $\nu_{i,I} = 1$
    else
      $\nu_{i,I} = 0$
return $\mathbf{m} := (m_i)_{i \in \mathsf{act}, I \text{ s.t. } i \in I}$

In order to reconstruct the sDKG secret $\mathbf{s} = \sum_I \mathbf{s}_I$, all users in the acting set act need to agree on a partition of the pool of secrets S among T parties, so that each secret is manipulated by exactly one party in act. Alg. 2 provides such a (non-interactive) consensus mechanism.

Distributing the key generation. It is possible to distribute our key generation simply by having one party in I to sample $\mathbf{s}_I$ for each set I. By accompanying it with shortness checks, and consistency verifications of the partial public key reconstructions, we additionally implement identifiable aborts within our sDKG.

At a high level, our sDKG works as follows:

1. Each party i samples a seed $\mathsf{seed}_i \leftarrow \{0,1\}^\kappa$. It then sends to other parties a hash $\mathsf{cmt}_i := H_{\mathsf{cmt}}(\mathsf{seed}_i)$.
2. In a second round, party i reveals seed_i.
3. Then, each party i computes $\mathsf{seed} = H_{\mathsf{seed}}((\mathsf{seed}_i)_{i \in [N]})$, and a public matrix $\mathbf{A} = \mathsf{ExpandA}(\mathsf{seed})$. It samples secrets $\mathbf{s}_I \leftarrow \mathcal{D}_{\mathbf{s}}^{k+\ell}$ for each set I of cardinality $N - T + 1$ such that $i = \max(I)$, and computes the partial public key $\mathbf{b}_I = \left[\mathbf{A}\ \mathbf{I}\right] \cdot \mathbf{s}_I$. It sends $\mathbf{s}_I$ over a private channel to all parties $j \in I$, and publicly a commitment $\mathsf{cmt}'_I := H_{\mathsf{cmt}}(\mathbf{b}_I)$ for each $\mathbf{b}_I$.
4. Finally, all the parties reveal the $\mathbf{b}_I$ they sampled in the previous round. The parties also verify the consistency of the $\mathbf{s}_I$ they received with the hashes cmt'_I, in case of inconsistency, they complain against the party that sent it, i.e. they reveal the value they received so that the inconsistency can be publicly verified.

Key aggregation consists in choosing as public key $\mathsf{pk} := \mathbf{b} := \sum_I \mathbf{b}_I$, and the auxiliary information is the set of all the partial keys $\mathsf{aux} := (\mathbf{b}_I)_I$. In case there were complaints in the last round, aggregation verifies them and in case they are valid, malicious parties are identified. We provide a formal construction in Fig. 7. Key aggregation is formalized in Alg. 7.

Alg. 3 $\mathsf{ShareKeygen}_1(\mathsf{st}_i)$

$\mathsf{seed}_i \xleftarrow{\$} \{0,1\}^\kappa$
$\mathsf{st}_i.\mathsf{session} := \{\mathsf{seed}_i\}$
$\mathsf{cmt}_i = H_{\mathsf{cmt}}(\mathsf{seed}_i)$
return $\mathsf{pm}_1^i := \mathsf{cmt}_i$

Alg. 4 $\mathsf{ShareKeygen}_2(\mathsf{st}_i, \mathsf{pm}_1)$

Fetch (seed_i) from $\mathsf{st}_i.\mathsf{session}$
$\mathsf{st}_i.\mathsf{session} := \{\mathsf{pm}_1\}$
return $\mathsf{pm}_2^i := \mathsf{seed}_i$

Alg. 5 $\mathsf{ShareKeygen}_3(\mathsf{st}_i, \mathsf{pm}_2)$

Fetch (pm_1) from $\mathsf{st}_i.\mathsf{session}[\mathsf{sid}]$
Parse $\mathsf{pm}_1 = (\mathsf{cmt}_j)_j$, $\mathsf{pm}_2 = (\mathsf{seed}_j)_j$
for $j \in [N]$ **do**
  if $\mathsf{cmt}_j \neq H_{\mathsf{cmt}}(\mathsf{seed}_j)$ **then**
    return $(\mathbf{abort}, \{j\})$
      ▷ Check hash commit.
$\mathsf{seed} := H_{\mathsf{seed}}((\mathsf{seed}_j)_j)$
$\mathbf{A} := \mathsf{ExpandA}(\mathsf{seed})$
$\mathsf{msgs}_i = \varnothing$
for $|I| = N - T + 1 \wedge i = \max(I)$ **do**
  $\mathbf{s}_I \leftarrow \mathcal{D}_{\mathbf{s}}^{k+\ell}$ ▷ Sample secret share
  $\mathbf{b}_I \leftarrow [\mathbf{A}\ \mathbf{I}] \cdot \mathbf{s}_I$
  $\mathsf{cmt}'_I := H_{\mathsf{cmt}}(\mathbf{b}_I)$
  for $j \in I \backslash \{i\}$ **do**
    $\mathsf{msgs}_i := \mathsf{msgs}_i \cup \{(I, j, \mathsf{SKE.Encrypt}(K_{i\to j}^{\mathsf{SKE}}, \mathbf{s}_I))\}$
$\mathsf{st}_i.\mathsf{session} := \{\mathsf{seed}, (\mathbf{s}_I)_I\}$
return $\mathsf{pm}_3^i := (\mathsf{msgs}_i, (\mathsf{cmt}'_I)_I)$

Alg. 6 $\mathsf{ShareKeygen}_4(\mathsf{st}_i, \mathsf{pm}_3)$

Fetch $(\mathsf{seed}, (\mathbf{s}_I)_I)$ from $\mathsf{st}_i.\mathsf{session}$
Parse $\mathsf{pm}_3 = (\mathsf{msgs}_j, (\mathsf{cmt}'_I)_{I \text{ s.t. } \max(I)=j})_j$
$\mathsf{seed} := H_{\mathsf{seed}}((\mathsf{seed}_i)_i)$
$\mathbf{A} := \mathsf{ExpandA}(\mathsf{seed})$
$\mathsf{complaints}_i = \varnothing$
for $|I| = N - T + 1 \wedge i \neq \max(I)$ **do**
  $\mathbf{s}_I := \mathsf{SKE.Decrypt}$ from $\mathsf{msgs}_{\max(I)}$
      ▷ Recover $\mathbf{s}_I$
  $\mathbf{b}_I \leftarrow [\mathbf{A}\ \mathbf{I}] \cdot \mathbf{s}_I$
  if $\|\mathbf{s}_I\| > \eta_1 \vee \mathsf{cmt}'_I \neq H_{\mathsf{cmt}}(\mathbf{b}_I)$ **then**
    $\mathsf{complaints}_i := \mathsf{complaints}_i \cup \{I, j := \max(I), K_{j\to i}^{\mathsf{SKE}}, \mathsf{sig}_{j\to j}^{\mathsf{SKE}}\}$
return $\mathsf{pm}_4^i := (\mathsf{complaints}_i, (\mathbf{b}_I)_I)$

Fig. 7: Algorithms for our sDKG relying on replicated secret sharing. For conciseness, we omit aborts due to parsing errors: if an unexpected message is received, parties abort and report the sender as malicious.

Security properties. We now prove that this sDKG is functionally simulatable, and correctly identifies aborts.

Theorem 2. *Consider the* $\mathsf{Hint\text{-}MLWE}_{q,k,\ell,Q_f,\mathcal{D}_{\mathbf{s}},(\mathcal{D}_{\mathbf{r}})_i,\mathcal{C}^{Q_f}}$ *assumption with* Q_f *hints of the form* $c \cdot \mathbf{s} + \mathbf{r}$*, where the noises are sampled from* $\mathcal{D}_{\mathbf{r}}^{k+\ell}$*. Assume that* η_1 *is an overwhelming bound on honest samples from* $\mathcal{D}_{\mathbf{s}}$*.*

Formally, there exist simulators $\mathsf{SimKeygen}, \mathsf{SimF}$*, such that for any adversary* $\mathcal{A}$ *against the functional simulatability of* Repl sDKG *(with key distribution* χ *and function* f *from Section 5.1) making at most* Q_f *calls to the evaluation oracle* OEval*, and* Q_H *random oracle queries, we can derive adversaries* $\mathcal{B}_1$ *against the* IND-CPA *security of the* SKE*,* $\mathcal{B}_2$ *against* $\mathsf{MLWE}_{q,k,\ell,\mathcal{D}_{\mathbf{s}}}$*, and* $\mathcal{B}_3$ *against* $\mathsf{Hint\text{-}MLWE}_{q,k,\ell,Q_f,\mathcal{D}_{\mathbf{s}},(\mathcal{D}_{\mathbf{r}})_i,\mathcal{C}^{Q_f}}$ *running in time* $\forall i, \mathcal{T}_{\mathcal{B}_i} \approx \mathcal{T}_{\mathcal{A}}$*, we have:*

$$|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}^1_{\mathsf{sDKG\text{-}sim}}} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}^0_{\mathsf{sDKG\text{-}sim}}}| \leqslant \frac{N+2}{2^\kappa} + \frac{Q_H}{q^k} + \frac{2(Q_H+N)Q_H}{2^{2\kappa}} + \mathsf{negl}(\kappa)$$

Alg. 7 $\mathsf{CombineKey}(\mathsf{pm}_2, \mathsf{pm}_3, \mathsf{pm}_4)$

Parse $\mathsf{pm}_2 = (\mathsf{seed}_i)_i$, $\mathsf{pm}_3 = (\mathsf{msgs}_i, \cdot)_i$, and $\mathsf{pm}_4 = (\mathsf{complaints}_i, (\mathbf{b}_I)_I)_i$
for set I of cardinality $N - T + 1$ **do**
  if $H_{\mathsf{cmt}}(\mathbf{b}_I) \neq \mathsf{cmt}'_I$ **then return** $(\mathbf{abort}, \{\max(I)\})$
$\mathsf{invalid} = \varnothing$
for $i \in [N]$ **do**
  for $(I, j, K^{\mathsf{SKE}}_{j\to i}, \mathsf{sig}^{\mathsf{SKE}}_{j\to i}) \in \mathsf{complaints}_i$ **do**
    if $\neg\mathsf{SIG.Verify}(\mathsf{sig}^{\mathsf{SKE}}_{j\to i}, K^{\mathsf{SKE}}_{j\to i})$ **then** $\mathsf{invalid} := \mathsf{invalid} \cup \{i\}$
    else
      $\mathbf{s}_I :=$ SKE.Decrypt using $K^{\mathsf{SKE}}_{i\to j}$ from msgs_j; $\mathbf{b}_I \leftarrow \left[\mathbf{A}\ \mathbf{I}\right] \cdot \mathbf{s}_I$
      if $\|\mathbf{s}_I\| > \eta_1 \vee \mathsf{cmt}'_I \neq H_{\mathsf{cmt}}(\mathbf{b}_I)$ **then**
        $\mathsf{invalid} := \mathsf{invalid} \cup \{j\}$
if $\mathsf{invalid} \neq \varnothing$ **then return** $(\mathbf{abort}, \mathsf{invalid})$
$\mathsf{seed} := H_{\mathsf{seed}}((\mathsf{seed}_i)_i)$
$\mathbf{b} := \left\lfloor \sum_I \mathbf{b}_I \right\rceil_{\nu_{\mathbf{b}}}$
return $\mathsf{pk} := (\mathsf{seed}, \mathbf{b}), \mathsf{aux} := (\mathbf{b}_I)_I$

$$+ N^2 \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{B}_1} + \mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}_2} + \mathsf{Adv}^{\mathsf{Hint\text{-}MLWE}}_{\mathcal{B}_3}$$

The intuition of this proof lies in the fact that replicated secret sharing ensures that at least one of the secrets $\mathbf{s}_I$ remains hidden from the adversary. The only leakage on this safe secret is through the evaluation oracle of the hint function f. This safe secret ensures by relying on MLWE that the public key pk is properly uniformly sampled. We defer the full proof of Theorem 2 to Section A.

Theorem 3. *Assuming the unforgeability of* Sign*, our* Repl sDKG *is* (K, η_1, η_2)*-identifiable abort for* $K = \binom{N}{T-1}$, $\eta_1 = \tau \cdot \sigma_{\mathbf{s}}\sqrt{d(k+\ell)}$, $\eta_2 = 1$ *when* $\mathcal{D}_{\mathbf{s}} = \mathcal{D}_{\sigma_{\mathbf{s}}}$, *and* $\left(\tau \geqslant 1 + \frac{\kappa \log 2}{3d}\right)$ *or* $\left(\tau \geqslant 1 + \sqrt{\frac{4\,\kappa \log 2}{d}}\right)$.

Proof. The intuition for this proof is that honest parties are never reported as malicious as they send correctly built messages to other parties, secrets $\mathbf{s}_I$ smaller than η_1 w.o.p., and malicious parties can't forge a valid complaint against them thanks to the unforgeability of Sign. In case a malicious party sends an invalid secret $\mathbf{s}_I$ to an honest party, the honest party will detect it and complain to other parties. Due to the determinism of the verification of complaints, all the honest parties will accept that complaint and identify the malicious party, ensuring agreement of the honest parties. Finally, by construction of the key aggregation algorithm, the partial public keys always reconstruct to the public key, and reconstruction coefficients are binary. □

5.3 Ramp Secret Sharing with the Coupon's collector problem

In this section, we describe a second sDKG construction that relies on the coupon collector problem. The core idea is to share the secret key $\mathbf{s}$ as $\mathbf{s} = \mathbf{s}_1 + \ldots + \mathbf{s}_n$, and each user $i \in [N]$ is given the secret $\mathbf{s}_{\boldsymbol{\pi}(i)}$ where $\boldsymbol{\pi}$ is a PRF from $[N]$ to $[n]$.

The coupon collector problem tells us that a random set of $T_1 \ll n \log n$ users is, w.o.p., unable to reconstruct the secret, while a set of $T_2 \gg n \log n$ users can.

While this idea does provide a ramp TSS, its parameters are far from satisfactory. Using Eqs. (1) and (2) we see that to have a probability $2^{-\kappa}$ that T_1 users cannot recover $\mathbf{s}$ and that T_2 users can, we obtain these bounds:

$$T_1 \leqslant (\ln n - \ln(\kappa \ln 2)) \cdot (n-1) \quad \text{and} \quad T_2 \geqslant (\ln n + \kappa \ln 2) \cdot n$$

While the ratio $\frac{T_2}{T_1} \underset{n\to\infty}{\sim} 1 + \frac{\kappa \ln 2 + \ln(\kappa \ln 2)}{\ln n}$ tends towards 1 as n grows, this convergence rate is very slow.

Amplifying correctness. To avoid this issue, we use a constant bound for the probability that T_2 parties can reconstruct $\mathbf{s}$, and use amplification to make it overwhelming. We secret share *the same secret* $\mathbf{s}$, $m = \kappa$ times in parallel:

$$\mathbf{s} = (\mathbf{s}_1^1 + \ldots + \mathbf{s}_n^1) = \ldots = (\mathbf{s}_1^m + \ldots + \mathbf{s}_n^m) \tag{3}$$

For each $j \in [m]$, this is efficiently done by sampling small $\mathbf{s}_1^j, \ldots, \mathbf{s}_{n-1}^j \leftarrow \mathcal{D}_{\mathsf{share}}^{k+\ell}$ then setting $\mathbf{s}_n^j = \mathbf{s} - \sum_{i=1}^{n-1} \mathbf{s}_i^j$. It is immediate that for any j, $\sum_i \mathbf{s}_i^j = \mathbf{s}$. Now consider the example where the probability that T_2 parties possess all the shares to reconstruct a given sharing of $\mathbf{s}$ is $1/2$ instead of $1 - 2^{-\kappa}$, then the probability that a set of T_2 users knows all the shares in at least one of the sharings of $\mathbf{s}$ is $\geqslant 1 - 2^{-m}$. To reconstruct $\mathbf{s}$, a set of T_2 users needs to agree on a sharing where all the shares are known (which can be done efficiently) and then sum these shares. Note that giving multiple equations on the same secret has a minimal impact on security as these can be cast as hints for Hint-MLWE.

Remark 1. Consider the following process: For each sharing $(\mathbf{s}_i^j)_i$ of $\mathbf{s}$, sample $v \leftarrow [n]$ and reveal $\mathbf{s}_v^j$. We want to know the probability that a full sharing has been revealed after T iterations of this process. Let:

$$T_1(n, m) = \left(1 - \frac{\ln(\kappa \ln(2) + \ln m)}{\ln n}\right)(n-1)\ln n \tag{4}$$

$$T_2(n, m) = \left(1 + \frac{\ln 2}{\ln n} \cdot \frac{\kappa}{m}\right) n \ln n \tag{5}$$

It follows from Lemma 1 that with probability $\geqslant 1 - 2^{-\kappa}$:

1. If $T \leqslant T_1(n, m)$, there exists no sharing of $\mathbf{s}$ of which we learnt all the entries;
2. If $T \geqslant T_2(n, m)$, there exists a sharing of $\mathbf{s}$ of which we learnt all entries.

For asymptotics, we take $m = \kappa$ and write $T_1(n) = T_1(n, \kappa)$, $T_2(n) = T_2(n, \kappa)$. The refined asymptotic ramp ratio now roughly satisfies $\frac{T_2}{T_1} \underset{n\to\infty}{\sim} 1 + \frac{\ln(\kappa \ln 2)}{\ln n}$.

In order to turn Remark 1 into an effective way of sharing keys, we introduce a function $\boldsymbol{\pi}_n : (\mathsf{str}, u) \in \{0,1\}^\kappa \times \mathbb{N}^* \mapsto \mathbf{v} = (v_0, \ldots, v_{m-1}) \in [n]^m$. In other words, $\boldsymbol{\pi}_n$ takes as input a bitstring str and a user index u, and deterministically

maps them to a m-tuple $\mathbf{v}$ of indices in $[n]$. We require that $\boldsymbol{\pi}_n$ is a PRF. $\boldsymbol{\pi}_n$ is easily constructed from SHAKE, similarly to ExpandA in ML-DSA.

We first formalize this ramp sDKG for a trusted setup in Ramp.Keygen, which is simulatable up to $T_1(n,m)$ corrupted parties, and correct with at least $T_2(n,m)$ signers, where T_1 and T_2 are defined in Eqs. (4) and (5). As for the Repl sDKG, we require a reconstruction function for concrete use, which consists of agreeing on a common sharing of the secret and partitioning its shares among the participants. We provide a non-interactive procedure for this consensus in Ramp.Recover.

Alg. 8 Ramp.Keygen(κ, N, n, m) $\to (\mathsf{pk}, \mathsf{ak}, \mathsf{sk})$

$\mathsf{seed} \leftarrow \{0,1\}^\kappa$; $\mathbf{A} := \mathsf{ExpandA}(\mathsf{seed})$
$\mathbf{s} \leftarrow \mathcal{D}_{\mathbf{s}}^{k+\ell}$
$\widehat{\mathbf{b}} := \left[\mathbf{A}\ \mathbf{I}\right] \cdot \mathbf{s}$; $\mathbf{b} := \lfloor \widehat{\mathbf{b}} \rceil_{\nu_{\mathbf{b}}}$
for $j \in [m]$ **do**
  for $v \in [n-1]$ **do**
    $\mathbf{s}_j^v \leftarrow \mathcal{D}_{\mathsf{share}}^{k+\ell}$
    $\mathbf{b}_j^v := \left[\mathbf{A}\ \mathbf{I}\right] \cdot \mathbf{s}_j^v$
  $\mathbf{s}_j^n \leftarrow \mathbf{s} - \sum_{v=1}^{n-1} \mathbf{s}_j^v$
    ▷ Last share chosen to sum to $\mathbf{s}$
  $\mathbf{b}_j^n := \widehat{\mathbf{b}} - \sum_{v=1}^{n-1} \mathbf{b}_j^v$
for $u \in [N]$ **do**
  $(i_0, \ldots, i_{m-1}) := \boldsymbol{\pi}_T(\mathsf{seed}, u)$
  $\mathsf{sk}_u := \left(\mathbf{s}_0^{i_0}, \mathbf{s}_1^{i_1}, \ldots, \mathbf{s}_{m-1}^{i_{m-1}}\right)$
    ▷ sk_u is the signing key of u
return $(\mathsf{pk} := (\mathsf{seed}, \mathbf{b}), \mathsf{ak} := ((\mathbf{b}_j^i)_{i,j \in [n] \times [m]}), \mathbf{sk} := (\mathsf{sk}_i)_{i \in [N]})$

Alg. 9 Ramp.Recover$(\mathsf{pk} = (\mathsf{seed}, \cdot), \mathsf{aux}, \mathsf{act}) \to (\nu_{i,(v,j)})_{i \in \mathsf{act}, v, j}$ or $\perp$

$\mathbf{M} := (m_{v,j})_{(v,j) \in [n] \times [m]} = (\varnothing)^{n \times m}$
    ▷ Buckets to determine which users in act possess secret (v,j)
for $(u \in \mathsf{act})$ **do**
  $(v_0, \ldots, v_{m-1}) := \boldsymbol{\pi}_n(\mathsf{seed}, u)$
  for $(j \in [m])$ **do**
    $M_{v_j,j} := M_{v_j,j} \sqcup \{u\}$
for $j \in [m]$ **do**
  if $\forall v \in [n], \mathbf{M}_{v,j} \neq \varnothing$ **then**
    ▷ Column j is covered by act
    $(\nu_{i,(v,j)})_{i \in \mathsf{act}, v, j} = (0)$
    for $v \in [n]$ **do**
      $i_0 = \max(M_{v,j})$; $\nu_{i_0,(v,j)} = 1$
    ▷ Secret (v,j) is processed by user i_0
    return $(\nu_{i,(v,j)})_{i \in \mathsf{act}, v, j}$
return $\perp$

Distributing the key generation. Designing a distributed version for our ramp sDKG appears more challenging than for the Repl one, as a priori no party knows concretely the underlying secret, and could thus provide sharings to the others. We solve this by having each party sample one short secret $\mathbf{s}_i \leftarrow \mathcal{D}_{\mathbf{s}}^{k+\ell}$, so that the underlying secret will be $\mathbf{s} = \sum_{i \in [N]} \mathbf{s}_i$. The DKG proceeds in several rounds:

1. After sampling a secret $\mathbf{s}_i \leftarrow \mathcal{D}_{\mathbf{s}}^{k+\ell}$, party i samples a sharing (with short shares) of $\mathbf{s}_i$ for each index $j \in [m]$:
   (a) $[\![\mathbf{s}_i]\!]_j^v \leftarrow \mathcal{D}_{\mathsf{share}}^{k+\ell}$ for $1 \leqslant v < n$
   (b) $[\![\mathbf{s}_i]\!]_j^n := \mathbf{s}_i - \sum_{v=1}^{n-1} [\![\mathbf{s}_i]\!]_j^v$
   It relies on the distribution function π to determine which party should receive the shares of $[\![\mathbf{s}_i]\!]_j$.
2. Using a commit-and-reveal technique, party i additionally reveals the partial public keys $\mathbf{b}_i := \mathbf{A} \cdot \mathbf{s}_i$, and $\mathbf{A} \cdot [\![\mathbf{s}_i]\!]_j^k$. Anyone can then verify the consistency equations $\mathbf{b}_i = \sum_{k \in [n]} \mathbf{A} \cdot [\![\mathbf{s}_i]\!]_j^k$.
   If they all pass, parties select as final shares the values $\mathbf{s}_j^k := \sum_{i \in [N]} [\![\mathbf{s}_i]\!]_j^k$, for $j \in [m], k \in [n]$. We are then guaranteed that for any reconstruction set, the partial public keys correctly reconstruct to $\mathbf{b} := \sum_{i \in [N]} \mathbf{b}_i$.

Due to lack of space, we include its formal description in Section B. We state below the functional simulatability and identifiable aborts of our Ramp sDKG.

Theorem 4. *Assume that η_1^{ind} is a bound holding w.o.p. on the norm of shares of a sharing of a $\mathbf{s}_i$. Our* Repl sDKG *is functionally simulatable up to $T_{\mathsf{corrupt}} = T_1(n,m)$ corrupted parties for the key distribution χ and hint function f of Section 5.1, assuming at most Q_f calls to the evaluation oracle* OEval*, and at most Q_H random oracle queries, if the* Hint-MLWE $_{q,k,\ell,Q_f+m,\mathcal{D}_{\mathbf{s}},(\mathcal{D}_{\mathbf{r}})_i,\{1\}^m\times\mathcal{C}^{Q_f}}$ *assumption with m hints of the form $\mathbf{s}+\mathbf{s}'$, and Q_f hints of the form $c\cdot\mathbf{s}+\mathbf{r}$, where $\mathbf{r}$ (resp. $\mathbf{s}'$) is sampled from $\mathcal{D}_{\mathbf{r}}^{k+\ell}$ (resp.$\mathcal{D}_{\mathsf{share}}^{k+\ell}$), and the* MLWE $_{q,k,\ell,\mathcal{D}_{\mathbf{s}}}$ *assumption hold, and if* SKE *used is* IND-CPA *secure.*

Although a bit more involved, the core intuition for proving Theorem 4 is analogous to Theorem 2, i.e. we rely on the fact that at least one honest secret $\mathbf{s}_{i_0}$ will remain unknown to the adversary. We provide here a high-level overview, and defer a more extended analysis to Section B.1.

$\mathbf{s}_{i_0}$ may now partially leak in two ways to the adversary: (i) through the shares that are corrupted, and (ii) through the evaluations of the hint function f. Interestingly, we can capture both as hints on the secret $\mathbf{s}_{i_0}$:

- For every sharing $(\llbracket\mathbf{s}_{i_0}\rrbracket_j^v)_{1\leqslant v\leqslant n}$, the bound T_1 ensures that at least one share v remains hidden. In case $v=n$ remains hidden, there is no leakage as the other shares are independent of $\mathbf{s}_{i_0}$. If the share $v=n$ leaks, we can see this as a hint on $\mathbf{s}_{i_0}$, as at least one share remains safe in the sharing $\llbracket\mathbf{s}_{i_0}\rrbracket_j$. In total, we require up to m hints on $\mathbf{s}_{i_0}$ of the form $\mathbf{s}_{i_0}+\mathbf{s}'$.
- The evaluations of f partially leak the honest shares of $\llbracket\mathbf{s}_{i_0}\rrbracket_j$. Interestingly, we can reexpress these directly from hints on $\mathbf{s}_{i_0}$. As for the Replicated Secret Sharing, this requires up to Q_f hints on $\mathbf{s}_{i_0}$ of the form $c\cdot\mathbf{s}_{i_0}+\mathbf{r}$.

Theorem 5. *Assuming the unforgeability of* Sign*, our ramp* sDKG *is (K,η_1,η_2)-identifiable abort for $T=T_2(n,m)$ signers, $K=m$, $\eta_1=N\cdot\eta_1^{\mathsf{ind}},\eta_1^{\mathsf{ind}}=\tau\cdot(\sigma_{\mathbf{s}}+\sqrt{n}\cdot\sigma_{\mathsf{share}})\sqrt{d(k+\ell)}$, $\eta_2=1$ when $\mathcal{D}_{\mathbf{s}}=\mathcal{D}_{\sigma_{\mathbf{s}}},\mathcal{D}_{\mathsf{share}}=\mathcal{D}_{\sigma_{\mathsf{share}}}$, and $\left(\tau\geqslant 1+\frac{\kappa\log 2}{3d}\right)$ or $\left(\tau\geqslant 1+\sqrt{\frac{4\,\kappa\log 2}{d}}\right)$.*

Optimizing further: amplifying privacy. Although we do not include this optimization in our formal descriptions due to a lack of space, we show that it would also be possible to amplify the privacy of this ramp scheme to improve parameters. We will consider $p=\kappa$ *distinct secrets* $\mathbf{s}_i$, each secret shared m times, and set the secret key as $\mathbf{s}=\mathbf{s}_1+\ldots+\mathbf{s}_p$. In other words, we have a two-dimension secret sharing of $\mathbf{s}$. A signing set needs to be able to reconstruct all $\mathbf{s}_i$ to reconstruct entirely $\mathbf{s}$, and assuming for instance that T_1 can reconstruct a given $\mathbf{s}_i$ with probability $1/2$, then the probability that a set of T_1 parties can reconstruct $\mathbf{s}$ is once again 2^{-m}. In detail, we obtain the following bounds:

$$T_1'(n,m,p)=\left(1-\frac{\ln\left(\kappa/p\ln(2)+\ln m\right)}{\ln n}\right)(n-1)\,\ln n \tag{6}$$

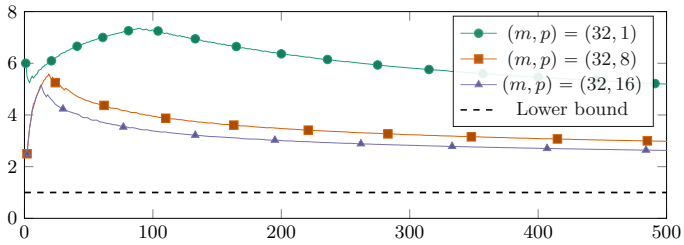


Fig. 8: Ratio T_2/T_1 achieved by our construction as a function of T_1 for $\kappa = 128$. For reference, we show the line $T_2/T_1 = 1$ (- - -) corresponding to a "perfect" threshold.

$$T_2'(n, m, p) = \left(1 + \frac{\ln 2}{\ln n} \cdot \frac{\kappa + \log p}{m}\right) n \ln n \tag{7}$$

While the above formulas provide a good understanding of the asymptotic behavior of our sharing, we can go further for concrete parameters and explicitly compute the probabilities of the Coupon Collector's problem [jh16]. By selecting (m, p) and the corresponding probabilities carefully, we obtain a ramp threshold scheme where the gap factor is at most 8 and quickly decays, as shown in Fig. 8.

6 Application: TSS and IA

In this section, we introduce a variant of Threshold Raccoon [DKM+24] relying on our sDKG to provide IA. We will first formally define TSS with IA, before moving to our concrete instantiation.

6.1 Definitions

TSS with Identifiable Abort are a special type of TSS that provides a mechanism for identifying misbehaving parties.

As for our sDKG definition, we describe TSS as *synchronous protocols*, i.e. round-based protocols where messages sent in some round are delivered by the start of the next round. We adopt the syntax of [DKM+24] and extend it with [IOZ14] to cover abort identification. Formally, a rnd-round TSS with identifiable abort is defined as a tuple $(\mathsf{Keygen}, (\mathsf{ShareSign}_k)_{k \in \mathsf{rnd}}, \mathsf{Combine}, \mathsf{Verify})$, where Keygen is a DKG as defined in Definition 5, $\mathsf{ShareSign}_k$ is the algorithm run by parties for round k, Combine aggregates all the rounds' contributions into a signature, and Verify verifies the signature validity. All of these functions except Verify may abort and return a set of identified malicious parties in case of error. We are interested in two properties in this work:

- **Unforgeability:** After successful termination of the DKG, it must be difficult for an adversary to forge a signature for a new message.
- **Identifiable aborts:** When running the signature protocol, it must either complete and output a valid signature, or honest parties abort and agree on a set of identified malicious parties.

Parameter	Explanation
$\mathcal{R}_q$	Polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^d+1)$
(k, ℓ)	Dimension of public matrix $\mathbf{A} \in \mathcal{R}_q^{k \times \ell}$
$(\mathcal{D}_{\mathbf{s}}, \sigma_{\mathbf{s}})$	Gaussian distribution with width $\sigma_{\mathbf{s}}$ used in the sDKG for sampling the secret
$(\mathcal{D}_{\mathsf{share}}, \sigma_{\mathsf{share}})$	Gaussian distribution with width σ_{share} used in the Ramp sDKG for sampling shares
$\nu_{\mathbf{b}}$	Amount of bit dropping performed on verification key in the sDKG
(K, η_1, η_2)	Shortness properties of sDKG (see Definition 6)
$(\mathcal{D}_{\mathbf{r}}, \sigma_{\mathbf{r}})$	Gaussian distribution with width $\sigma_{\mathbf{r}}$ used for the commitment $\mathbf{w}$
$\nu_{\mathbf{w}}$	Amount of bit dropping performed on (aggregated) commitment
$(\mathcal{C} \subset \mathcal{R}_q, \omega)$	Challenge set $\{c \in \mathcal{R}_q \mid \|c\|_\infty = 1 \wedge \|c\|_1 = \omega\}$ s.t. $\lvert\mathcal{C}\rvert \geqslant 2^\kappa$
B_2^{ind}	Two-norm bound for partial verification $B_2^{\mathsf{ind}} = \omega \cdot K\eta_1\eta_2 + \tau\sqrt{d(k+\ell)} \cdot \sigma_{\mathbf{r}}$
B_2	Two-norm bound on the signature $B_2 = T \cdot B_2^{\mathsf{ind}} + (3 \cdot 2^{\nu_{\mathbf{w}}-1} + \omega \cdot 2^{\nu_{\mathbf{b}}}) \cdot \sqrt{dk}$

Table 1: Overview of parameters used in our TSS. We assume τ is such that $\left(\tau \geqslant 1 + \frac{\kappa \log 2}{3d}\right)$ or $\left(\tau \geqslant 1 + \sqrt{\frac{4\,\kappa \log 2}{d}}\right)$.

We provide fully formalized definitions for TSS with identifiable aborts, and the above security properties in Section C.

6.2 Our construction

We now introduce our construction of a TSS with IA, based on the sDKG introduced in Section 5. At a high level, during each signing session party i individually signs the message using its partial secret $\sum_{u \in [|S_i|]} \nu_{i,u} \cdot S_i[u]$. The sDKG ensures the shortness of this partial secret key which in turn ensures the shortness of partial signatures and allows one to verify honest behavior. To optimize signature size, we consider that sDKG secrets $S_i[u]$ are divided in two parts $\mathbf{s}_i[u] \in \mathcal{R}^\ell$ and $\mathbf{e}_i[u] \in \mathcal{R}^k$ for $i \in [N], u \in [|S_i|]$. We provide a formal description in Fig. 9. Verification procedures are described separately in PartialVerify and Verify. Parameter sets are given in Table 1.

Alg. 14 PartialVerify$(\mathsf{act}, \mathsf{ak}, c, (\mathbf{z}_i, \mathbf{w}_i)_i)$

Fetch all the $(\mathbf{b}_i[u])_{i \in [N], u}$ from ak
$(\nu_{i,u})_{i \in \mathsf{act}, u} :=$ Recover$(\mathsf{pk}, \mathsf{ak}, \mathsf{act})$
▷ Determine the keys in use for act
invalid $= \varnothing$
for $i \in$ act **do**
  $\mathbf{b}_i^{\mathsf{part}} := \sum_{u \in [|S_i|]} \nu_{i,u} \cdot \mathbf{b}_i[u]$
  $\mathbf{y}_i := \left(\mathbf{A} \cdot \mathbf{z}_i - c \cdot \mathbf{b}_i^{\mathsf{part}}\right) - \mathbf{w}_i$
  if $\{\|(\mathbf{z}_i, \mathbf{y}_i)\| > B_2^{\mathsf{ind}}\}$ **then**
    invalid := invalid $\sqcup \{i\}$
return invalid

Alg. 15 Verify$(\mathsf{vk} := (\mathsf{seed}, \mathbf{b}), \mathsf{sig})$

Parse sig $:= (c, \mathbf{z}, \mathbf{h})$
$\mathbf{u} := \lfloor \mathbf{A} \cdot \mathbf{z} - 2^{\nu_{\mathbf{b}}} \cdot c \cdot \mathbf{b} \rceil_{\nu_{\mathbf{w}}}$
$\mathbf{w} = \mathbf{u} + \mathbf{h}$
if $\{H(\mathsf{vk}, \mathsf{msg}, \mathbf{w}) = c\}$ **and** $\{\|(\mathbf{z}, 2^{\nu_{\mathbf{w}}}\mathbf{h})\| \leqslant B_2\}$ **then**
  return true
return false

Unforgeability. We reduce the unforgeability of our scheme to the simulatability property of the sDKG for a uniform public key distribution, and the SelfTargetMSIS assumption. We include the full proof in Section C.3.

Alg. 10 $\mathsf{ShareSign}_1(\mathsf{st}_i, \mathsf{sid}, \mathsf{act}, \mathsf{msg})$

1: **req**{ $\nexists\, \mathsf{st}_i.\mathsf{session}[\mathsf{sid}]$ }
2: $(\mathbf{r}_i, \mathbf{e}'_i) \leftarrow \mathcal{D}_{\mathbf{r}}^{\ell} \times \mathcal{D}_{\mathbf{r}}^{k}$
3: $\mathbf{w}_i := \mathbf{A} \cdot \mathbf{r}_i + \mathbf{e}'_i$
4: $\mathsf{st}_i.\mathsf{session}[\mathsf{sid}] :=$
$\quad \{\mathsf{rnd} = 1, \mathsf{act}, \mathsf{msg}, (\mathbf{r}_i, \mathbf{e}'_i), \mathbf{w}_i, \varnothing\}$
5: $\mathsf{cmt}_i = H_{\mathsf{cmt}}(\mathsf{vk}, i, \mathbf{w}_i)$
6: **return** $\mathsf{pm}_1^i := \mathsf{cmt}_i$

Alg. 11 $\mathsf{ShareSign}_3(\mathsf{st}_i, \mathsf{sid}, \mathsf{pm}_2)$

1: Fetch $(\mathsf{rnd}, \mathsf{act}, \mathsf{msg}, (\mathbf{r}_i, \mathbf{e}'_i), \mathbf{w}_i, \mathsf{pm}_1)$ from $\mathsf{st}_i.\mathsf{session}[\mathsf{sid}]$
2: **req**{ $\mathsf{rnd} = 2$ }
3: Parse $\mathsf{pm}_2 = (\mathbf{w}_i)_i$
4: **for** $j \in [N]$ **do**
5: **if** $\mathsf{cmt}_j \neq H_{\mathsf{cmt}}(\mathsf{vk}, j, \mathbf{w}_j)$ **then**
6: **return** (**abort**, $\{j\}$)
7: $\mathbf{w} := \left\lfloor \sum_{j \in [N]} \mathbf{w}_i \right\rceil_{\nu_{\mathbf{w}}}$
8: $c := H_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$
9: $(\nu_{i,u})_{i \in \mathsf{act}, u} := \mathsf{Recover}(\mathsf{pk}, \mathsf{ak}, \mathsf{act})$
10: $\mathbf{z}_i := c \cdot \sum_{u \in [|S_i|]} \nu_{i,u} \cdot \mathbf{s}_i[u] + \mathbf{r}_i$
▷ Individual response in $\mathcal{R}_q^{\ell}$
11: **return** $\mathsf{pm}_3^i := \mathbf{z}_i$

Alg. 12 $\mathsf{ShareSign}_2(\mathsf{st}_i, \mathsf{sid}, \mathsf{pm}_1)$

1: Fetch $(\mathsf{rnd}, \mathsf{msg}, (\mathbf{r}_i, \mathbf{e}'_i), \mathbf{w}_i)$ from $\mathsf{st}_i.\mathsf{session}[\mathsf{sid}]$
2: **req**{ $\mathsf{rnd} = 1$ }
3: $\mathsf{st}_i.\mathsf{session}[\mathsf{sid}] :=$
$\quad \{\mathsf{rnd} = 2, \mathsf{msg}, (\mathbf{r}_i, \mathbf{e}'_i), \mathbf{w}_i, \mathsf{pm}_1\}$
4: **return** $\mathsf{pm}_2^i := \mathbf{w}_i$

Alg. 13 $\mathsf{Combine}(\mathsf{vk} = (\mathsf{seed}, \mathbf{b}), \mathsf{ak}, \mathsf{msg}, (\mathsf{pm}_k)_{k \in \{1,2,3\}})$

1: Parse $\mathsf{pm}_2 = (\mathbf{w}_i)_i$ and $\mathsf{pm}_3 = (\mathbf{z}_i)_i$
2: $\mathbf{w} := \left\lfloor \sum_{j \in \mathsf{act}} \mathbf{w}_i \right\rceil_{\nu_{\mathbf{w}}}$
3: $c := H_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$
4: $\mathsf{invalid} :=$
$\quad \mathsf{PartialVerify}\left(\mathsf{act}, \mathsf{ak}, c, (\mathbf{z}_i, \mathbf{w}_i)_{i \in [N]}\right)$
5: **if** $\mathsf{invalid} \neq \varnothing$ **then**
6: **return** (**abort**, invalid)
▷ Verify partial signatures
7: $\mathbf{z} = \sum_{i \in [N]} \mathbf{z}_i$
8: $\mathbf{u} := \lfloor \mathbf{A} \cdot \mathbf{z} - 2^{\nu_{\mathbf{b}}} \cdot c \cdot \mathbf{b} \rceil_{\nu_{\mathbf{w}}}$
9: $\mathbf{h} := \mathbf{w} - \mathbf{u}$ ▷ Hint in $\mathcal{R}_{q_{\mathbf{w}}}^{k}$
10: **return** $\mathsf{sig} := (c, \mathbf{z}, \mathbf{h})$

Fig. 9: Algorithms (part 1/2) for a distributed signing with IA. For conciseness, we omit aborts due to parsing errors: if an unexpected message is received, parties abort and report the sender as malicious.

Theorem 6. *Let* $B_{\mathsf{stmsis}} = B_2 + \sqrt{\omega} + (\omega \cdot 2^{\nu_{\mathbf{b}}} + 2^{\nu_{\mathbf{w}}+1}) \cdot \sqrt{dk}$*, where* B_2 *is set as in Table 1. Let* $\mathcal{A}$ *be an adversary against the* TH-UF *security of our signature scheme with* sDKG *making at most* Q_{Sign} *signing queries, and at most* Q_H *random oracle queries. There exist adversaries* $\mathcal{B}_1$ *against the* sDKG-sim *game with uniform key distribution* χ *and hint function* f *from Section 5.1 (calling up to* Q_{Sign} *times the evaluation oracle* OEval *for* f*), and* $\mathcal{B}_2$ *against* $\mathsf{SelfTargetMSIS}_{q,k,\ell+1,\mathcal{C},B_{\mathsf{stmsis}}}$ *running in time* $\mathcal{T}_{\mathcal{B}_1} \approx \mathcal{T}_{\mathcal{B}_2} \approx \mathcal{T}_{\mathcal{A}}$ *such that:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{TH\text{-}UF}} \leqslant \frac{N(Q_H+1)Q_{\mathsf{Sign}}}{2^{d-2}} + \frac{Q_H(Q_H+T)}{2^{2\kappa}} + \mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{sDKG\text{-}sim}} + \mathsf{Adv}_{\mathcal{B}_2}^{\mathsf{SelfTargetMSIS}}$$

Identifiable aborts (IA). We prove that when the protocol aborts, all honest acting parties can agree on a set of guilty corrupted parties. Since no private information is used to determine who caused the abort we could also have defined a notion of IA in which any third party can trace guilty parties from the

protocol's transcript. We chose to use the definition of Fig. 19 as it is closer to existing notions. We defer the proof of Theorem 7 to Section C.5, including proofs for the overwhelming bounds B_2^{ind}, B_2 and a fix of [DKM+24, Lem C.1].

Theorem 7. *Our TSS identifies abort if the* sDKG *is* (K, η_1, η_2)*-robust/IA for parameters as in Table 1.*

Let $\mathcal{A}$ *be a PPT adversary against* $\mathsf{Game}_{\mathsf{TH\text{-}ID}}$ *of Fig. 19. Let the* act *validity predicate be* $\mathcal{P}(\mathsf{vk}, \mathsf{ak}, \mathsf{act}) = \{\mathsf{sDKG.Recover}(\mathsf{vk}, \mathsf{ak}, \mathsf{act}) \neq \bot\}$*. There exists an adversary* $\mathcal{B}_1$ *against the* sDKG *robustness/IA with* $\mathcal{T}_{\mathcal{B}_1} \approx \mathcal{T}_{\mathcal{A}}$ *such that:* $\mathsf{Adv}^{\mathsf{TH\text{-}ID}}_{\mathcal{A}}(1^\kappa) \leqslant \mathsf{Adv}^{\mathsf{sDKG\text{-}robust/id\ abort}}_{\mathcal{B}_1}$.

Parameter selection. Parameter sets are provided in Table 2. Parameter selection is fairly standard and detailed in Section E.

Table 2: Parameters sets of our TSS (usable with either Repl or Ramp sDKG), valid for any $1 \leqslant T_1 \leqslant T_2 \leqslant N \leqslant N_{\max}$ verifying: (i) $T_1 = T_2 - 1$ for Repl, (ii) while for Ramp the gap is larger and possible values are plotted in Fig. 8. σ_{share} is only used within the Ramp sDKG. Sizes are in kilobytes (kB).

κ	Q_{Sign}	$N_{\max}$ Repl	$N_{\max}$ Ramp	$\lceil\log_2 q\rceil$	d	k	ℓ	ω	$\sigma_{\mathbf{s}}$	σ_{share}	$\sigma_{\mathbf{r}}$	$\nu_{\mathbf{b}}$	$\nu_{\mathbf{w}}$	$\lvert\mathsf{vk}\rvert$ (kB)	$\lvert\mathsf{sig}\rvert$ (kB)	Comm. (kB/party)
128	2^{59}	16	16	45	512	4	4	18	2^{10}	2^{10}	2^{33}	30	34	3.9	10.3	20.0
128	2^{58}	16	1024	48	512	5	5	18	2^{8}	2^{10}	2^{30}	37	43	3.5	11.9	25.0
256	2^{60}	16	16	45	512	7	7	44	2^{10}	2^{10}	2^{34}	30	25	6.8	18.4	35.5

7 The Death Star Algorithm: Tighter Identifiable Aborts

In this section of independent interest, we provide a geometric perspective on the identifiable aborts problem for lattice schemes and introduce a simple algorithmic technique to detect anomalous correlations between random Gaussian vectors. This approach is based on a basic observation: the norm of the sum of independent vectors should neither be too large nor too small. We now turn into an algorithm the intuition developed in Section 2.6. Remark that if some positive correlation occurred between a subset of the vectors, then they all lie in some spherical cap, which also contains (after proper normalization) the sum X of all vectors. Hence, we want to detect such correlation by checking whether vectors anomalously correlate with the global sum X. Geometrically, this is essentially equivalent to excluding some cap around X (orange cap in Fig. 4). Therefore a vector X_i is only deemed to be valid if it lies on the gray spherical cap (in Fig. 4), which can be tested by a simple inner product testing of the shape $\langle X_i, X - X_i\rangle / \|X - X_i\| \geqslant \alpha$ for a properly chosen α. We can write the following procedure Identify (Alg. 16), parametrized by some bounds $I_{\max}$,

$J_{\max}$ depending on the parameters and the accuracy desired for the test. We call *corrupted* vectors the vectors not passing, as they will correspond to corrupted parties during the signing session.

Alg. 16 Identify$((X_i)_{i\in[T]}) \to \mathsf{corrupt} \subseteq [T]$

$X = \sum_{i\in[T]} X_i$, $\mathsf{corrupt} := \varnothing$
for $i \in [T]$ **do**
  if $\{\|X_i\| > I_{\max}\}$ **then** {add i to corrupt}
  if $\{\langle X_i, X - X_i\rangle / \|X - X_i\| > J_{\max}\}$ **then** { add i to corrupt }
return corrupt

Correctness for discrete Gaussians follows from Lemma 2, which proves that honestly generated vectors are accepted w.o.p. Shortness of the aggregated vector follows from Lemma 3. This new bound now entails a new (st)MSIS bound for the Verify function. Lemmas 2 and 3 are proven in Section F.

Lemma 2 (Correctness). *For $i \in \{1, \dots, T-1\}$, let $X_i \in \mathbb{Z}^d$ be arbitrary vectors. Let X_0 be (honestly) sampled from a discrete Gaussian: $X_0 \xleftarrow{\$} D^d_\sigma$. For $I_{\max} = \sigma\sqrt{d}\,(1+\tau)$, $J_{\max} = \sigma\sqrt{2\,(\kappa+1)\,\ln 2}$, $\tau = \sqrt{\frac{2\ln(2)\kappa}{d}}$, then with probability at least $1 - 2^{-\kappa+1}$, $0 \notin \mathsf{corrupt} :=$ Identify$((X_i)_{i\in[T]})$.*

Lemma 3 (Shortness). *For any set of vectors $(X_i)_{i\in[T]}$ such that* Identify$((X_i)_{i\in[T]}) = \varnothing$*, it holds that:*

$$\left\|\sum_i X_i\right\|^2 \leqslant \sigma^2 T\left(T\kappa\ln 2\left(1+\sqrt{1+\frac{2d(1+\tau)^2}{\kappa T\ln 2}}\right)+d(1+\tau)^2\right)$$

The bound we find this way is (loosely) of the shape $\|X\| \leqslant \sigma\sqrt{Td} + \sigma T\alpha$ for some constant $1 < \alpha < \sqrt{d}$. This is better than the worst-case bound obtained by a naive application of triangular inequality on the individual bounds on the X_i (i.e., $\|X\| \leqslant \sigma\, T\sqrt{d}$), but it is of course slightly worse than the norm we expect on average: $\sqrt{\mathbb{E}[\|X\|^2]} = \sigma\sqrt{T\,d}$.

Dealing with non-centered distributions One detail we omitted in the previous analysis is that partial signatures will actually be of the form $X_i = Y_i + S_i$, where $Y_i \sim D_{\mathbb{Z}^d,\sigma}$ is fresh randomness and S_i is some multiple of a long-term (short) private key share such that $\forall i, \|S_i\| \leqslant C$ for some constant C. The algorithm and its analysis are carried exactly in the same way with:

$$I_{\max} = \sigma\sqrt{d}\,(1+\tau) + C \quad \text{and} \quad J_{\max} = \sigma\sqrt{2\,(\kappa+1)\,\ln 2} + C.$$

Remark 2. Alg. 16 can be used *verbatim* with other distributions (rounded Gaussians, sums of uniforms, etc.) showcasing a concentration of norm phenomenon, even though the analysis has to be adapted accordingly. Alg. 16 may also be applied (non-exhaustively) to (variations) of the threshold signatures and multi-signatures from [DOTT21, DOTT22, BTT22, Che23, GKS24] and [CATZ24, ADP24], we leave this as a direction for future work. Interestingly, it does not seem straightforwardly applicable to Threshold Raccoon [DKM+24] or Pelican [ENP24], since they output uniform-looking (as opposed to short) partial signatures: only the final aggregated signature is short. For the parameters of Table 2 this translates in gaining *for free* between **10** and **30** bits of security on forgery.

References

ADP24. Nabil Alkeilani Alkadri, Nico Döttling, and Sihang Pu. Practical lattice-based distributed signatures for a small number of signers. In Christina Pöpper and Lejla Batina, editors, *ACNS 24International Conference on Applied Cryptography and Network Security, Part I*, volume 14583 of *LNCS*, pages 376–402. Springer, Cham, March 2024.

APS15. Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *J. Math. Cryptol.*, 9(3):169–203, 2015.

ASY22. Shweta Agrawal, Damien Stehlé, and Anshu Yadav. Round-optimal lattice-based threshold signatures, revisited. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *ICALP 2022*, volume 229 of *LIPIcs*, pages 8:1–8:20. Schloss Dagstuhl, July 2022.

BD10. Rikke Bendlin and Ivan Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 201–218. Springer, Berlin, Heidelberg, February 2010.

BGG+18. 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 Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 565–596. Springer, Cham, August 2018.

BKL+24. Cecilia Boschini, Darya Kaviani, Russell W. F. Lai, Giulio Malavolta, Akira Takahashi, and Mehdi Tibouchi. Ringtail: Practical two-round threshold signatures from learning with errors. Cryptology ePrint Archive, Paper 2024/1113, 2024.

BKP13. Rikke Bendlin, Sara Krehbiel, and Chris Peikert. How to share a lattice trapdoor: Threshold protocols for signatures and (H)IBE. In Michael J. Jacobson, Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *ACNS 13International Conference on Applied Cryptography and Network Security*, volume 7954 of *LNCS*, pages 218–236. Springer, Berlin, Heidelberg, June 2013.

BN06. Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, October / November 2006.

BP23. Luís T. A. N. Brandão and Rene Peralta. Nist first call for multi-party threshold schemes. NIST Internal Report (IR) 8214C, National Institute of Standards and Technology, January 2023. Initial Public Draft.

BTT22. Cecilia Boschini, Akira Takahashi, and Mehdi Tibouchi. MuSig-L: Lattice-based multi-signature with single-round online phase. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 276–305. Springer, Cham, August 2022.

CATZ24. Rutchathon Chairattana-Apirom, Stefano Tessaro, and Chenzhi Zhu. Partially non-interactive two-round lattice-based threshold signatures. Cryptology ePrint Archive, Report 2024/467, 2024.

CCL+23. Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Bandwidth-efficient threshold EC-DSA revisited: Online/offline extensions, identifiable aborts proactive and adaptive security. *Theor. Comput. Sci.*, 939:78–104, 2023.

CGG+20. Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1769–1787. ACM Press, November 2020.

Che23. Yanbo Chen. DualMS: Efficient lattice-based two-round multi-signature with trapdoor-free simulation. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 716–747. Springer, Cham, August 2023.

CKM23. Elizabeth C. Crites, Chelsea Komlo, and Mary Maller. Fully adaptive Schnorr threshold signatures. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 678–709. Springer, Cham, August 2023.

CPS+20. Chitchanok Chuengsatiansup, Thomas Prest, Damien Stehlé, Alexandre Wallet, and Keita Xagawa. ModFalcon: Compact signatures based on module-NTRU lattices. In Hung-Min Sun, Shiuh-Pyng Shieh, Guofei Gu, and Giuseppe Ateniese, editors, *ASIACCS 20*, pages 853–866. ACM Press, October 2020.

dEK+23. Rafael del Pino, Thomas Espitau, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, Mélissa Rossi, and Markku-Juhani Saarinen. Raccoon. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures.

DKL+18. Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR TCHES*, 2018(1):238–268, 2018.

DKLS25. Antonín Dufka, Semjon Kravtšenko, Peeter Laud, and Nikita Snetkov. Trilithium: Efficient and universally composable distributed ML-DSA signing. Cryptology ePrint Archive, Paper 2025/675, 2025.

DKM+24. Rafaël Del Pino, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, and Markku-Juhani O. Saarinen. Threshold raccoon: Practical threshold signatures from standard lattice assumptions. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 219–248. Springer, Cham, May 2024.

Doe18. Benjamin Doerr. Probabilistic tools for the analysis of randomized optimization heuristics. *CoRR*, abs/1801.06733, 2018.

DOTT21. Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment

from lattices. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 99–130. Springer, Cham, May 2021.

DOTT22. Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. *Journal of Cryptology*, 35(2):14, April 2022.

dPKN+25. Rafael del Pino, Shuichi Katsumata, Guilhem Niot, Michael Reichle, and Kaoru Takemure. Unmasking TRaccoon: Identifiable aborts for lattice-based threshold signature. *CRYPTO 2025, to appear*, 2025.

dPN25. Rafael del Pino and Guilhem Niot. Finally! a compact lattice-based threshold signature. In Tibor Jager and Jiaxin Pan, editors, *Public-Key Cryptography – PKC 2025*, pages 169–199, Cham, 2025. Springer Nature Switzerland.

EEN+24. Muhammed F. Esgin, Thomas Espitau, Guilhem Niot, Thomas Prest, Amin Sakzad, and Ron Steinfeld. Plover: Masking-friendly hash-and-sign lattice signatures. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VII*, volume 14657 of *LNCS*, pages 316–345. Springer, Cham, May 2024.

EKT24. Thomas Espitau, Shuichi Katsumata, and Kaoru Takemure. Two-round threshold signature from algebraic one-more learning with errors. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 387–424. Springer, Cham, August 2024.

ENP24. Thomas Espitau, Guilhem Niot, and Thomas Prest. Flood and submerse: Distributed key generation and robust threshold signature from lattices. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 425–458. Springer, Cham, August 2024.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987.

GHKR08. Rosario Gennaro, Shai Halevi, Hugo Krawczyk, and Tal Rabin. Threshold RSA for dynamic and ad-hoc groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 88–107. Springer, Berlin, Heidelberg, April 2008.

GKS24. Kamil Doruk Gür, Jonathan Katz, and Tjerand Silde. Two-round threshold lattice-based signatures from threshold homomorphic encryption. In Markku-Juhani Saarinen and Daniel Smith-Tone, editors, *Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part II*, pages 266–300. Springer, Cham, June 2024.

IOZ14. Yuval Ishai, Rafail Ostrovsky, and Vassilis Zikas. Secure multi-party computation with identifiable abort. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 369–386. Springer, Berlin, Heidelberg, August 2014.

ISN87. M. Ito, A. Saito, and Takao Nishizeki. Secret sharing schemes realizing general access structure. In *Proc. IEEE Global Telecommunication Conf. (Globecom'87)*, pages 99–102, 1987.

jh16. joriki (https://math.stackexchange.com/users/6622/joriki). Probability distribution in the coupon collector's problem. Mathematics Stack Exchange, 2016. URL:https://math.stackexchange.com/q/1454749 (version: -03-27).

KG20. Chelsea Komlo and Ian Goldberg. FROST: Flexible round-optimized Schnorr threshold signatures. In Orr Dunkelman, Michael J. Jacobson,

Jr., and Colin O'Flynn, editors, *SAC 2020*, volume 12804 of *LNCS*, pages 34–65. Springer, Cham, October 2020.

KGS23. Chelsea Komlo, Ian Goldberg, and Douglas Stebila. A formal treatment of distributed key generation, and new constructions. Cryptology ePrint Archive, Report 2023/292, 2023.

KLS18. Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 552–586. Springer, Cham, April / May 2018.

KLSS23. Duhyeong Kim, Dongwon Lee, Jinyeong Seo, and Yongsoo Song. Toward practical lattice-based proof of knowledge from hint-MLWE. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 549–580. Springer, Cham, August 2023.

KRT24. Shuichi Katsumata, Michael Reichle, and Kaoru Takemure. Adaptively secure 5 round threshold signatures from MLWE/MSIS and DL with rewinding. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 459–491. Springer, Cham, August 2024.

LDK+22. Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

Lyu12. Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, Berlin, Heidelberg, April 2012.

TPCZ23. Guofeng Tang, Bo Pang, Long Chen, and Zhenfeng Zhang. Efficient lattice-based threshold signatures with functional interchangeability. *IEEE Trans. Inf. Forensics Secur.*, 18:4173–4187, 2023.

A Omitted proof of functional simulatability of Repl sDKG

In this section, we present the proof of functional simulatability of our Repl sDKG. At a high level, it consists of observing that the Repl sDKG ensures that at least one of the secrets $\mathbf{s}_I$ remains unknown to the adversary. Given that the other secrets are chosen independently from it, we prove the indistinguishability of the public key from uniform assuming an Hint-MLWE instance over the safe secret $\mathbf{s}_I$.

Proof (of Theorem 2). We prove the functional simulatability of our Repl sDKG with a series of hybrid games, starting from the case $b = 0$ (real key distribution) of the $\mathsf{Game}_{\mathsf{sDKG\text{-}sim}}$ game in the random oracle model, and finishing with the case $b = 1$ of the $\mathsf{Game}_{\mathsf{sDKG\text{-}sim}}$ game (simulated key distribution). Let $\mathcal{A}$ be an adversary against the $\mathsf{Game}_{\mathsf{sDKG\text{-}sim}}$ security game.

Hybrid_1. This is the original case $b = 0$ of the $\mathsf{Game}_{\mathsf{sDKG\text{-}sim}}$ game from Fig. 6, where key distribution is executed using real algorithms $(\mathsf{ShareKeygen}_i)_i$.

Hybrid_2. In this hybrid, we replace the hash commitment on seed_i in the first round of honest parties by random values $\mathsf{cmt}_i \xleftarrow{\$} \{0,1\}^{2\kappa}$. The random oracle is later programmed in round 2 to remain consistent. We also program H_{seed} and $\mathsf{ExpandA}$ to ensure true random sampling of $\mathsf{seed} \leftarrow \chi_{\mathsf{seed}}$ and of the public matrix $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{k \times \ell}$. This is formalized in Fig. 10.

Hybrid_2

1: $\mathsf{ProgrammedSeed} = \bot$
▷▷▷ Lines 1-7 are identical ◁◁◁
2: **for** $i \in$ honest **do**
3: $\quad \mathsf{cmt}_i \xleftarrow{\$} \{0,1\}^{2\kappa}$
▷▷▷ Other lines are left unchanged ◁◁◁

$\mathsf{OPerformRound}_2((\mathsf{pm}_1^j)_{j \in \mathsf{corrupt}} = (\mathsf{cmt}_j)_{j \in \mathsf{corrupt}})$

1: **req**{ current_round $= 1$ }
2: **for** $i \in$ honest **do**
3: $\quad \mathsf{seed}_i \xleftarrow{\$} \{0,1\}^{\kappa}$
4: $\quad H_{\mathsf{cmt}}(\mathsf{seed}_i) := \mathsf{cmt}_i$ ▷ Program random oracle
5: **if** $\forall j \in \mathsf{corrupt}, \exists \mathsf{seed}_j, \mathsf{cmt}_j = H(\mathsf{seed}_j)$ **then**
6: $\quad \mathsf{seed} \xleftarrow{\$} \chi_{\mathsf{seed}}, \mathbf{A} \xleftarrow{\$} \mathcal{R}_q^k$
7: $\quad H_{\mathsf{seed}}((\mathsf{seed}_i)_{i \in [N]}) := \mathsf{seed}$
8: $\quad \mathsf{ExpandA}(\mathsf{seed}) := \mathbf{A}$ ▷ Program random oracle
9: $\quad \mathsf{ProgrammedSeed} = (\mathsf{seed}_i)_{i \in [N]}$
10: **return** $(\mathsf{pm}_2^i)_{i \in \mathsf{honest}} = (\mathsf{seed}_i)_{i \in \mathsf{honest}}$

$\mathsf{OPerformRound}_3((\mathsf{pm}_2^j)_{j \in \mathsf{corrupt}} = (\mathsf{seed}_j)_{j \in \mathsf{corrupt}})$

1: Retrieve $(\mathsf{seed}_j)_{j \in \mathsf{honest}}$
2: **for** $i \in$ honest **do**
3: $\quad$ Parse $\mathsf{pm}_1 = (\mathsf{cmt}_j)_j$, $\mathsf{pm}_2 = (\mathsf{seed}_j)_j$
4: $\quad$ **for** $j \in [N]$ **do**
5: $\quad\quad$ **if** $\{\mathsf{cmt}_j \neq H_{\mathsf{cmt}}(\mathsf{seed}_j)\}$ **then return** $(\mathbf{abort}, \{j\})$
6: **assert**{ $\mathsf{ProgrammedSeed} = (\mathsf{seed}_i)_{i \in [N]}$ }
▷▷▷ Other lines are left unchanged ◁◁◁

Fig. 10: The second hybrid of the security proof of the functional simulatability of our Repl sDKG scheme. Difference with the previous hybrid are highlighted. We omit state updates for conciseness. In case the game programs the random oracle on an existing index, we consider that the adversary wins the game.

The view of the adversary differs in several cases:

- If $\mathcal{A}$ queries the random oracle on one of $H_{\mathsf{cmt}}(\mathsf{seed}_i)$, $H_{\mathsf{seed}}((\mathsf{seed}_i)_i)$, or $\mathsf{ExpandA}(\mathsf{seed})$ before their programming. This happens with probability at most:
$$N \cdot 2^{-\kappa} + 2^{-\kappa} + 2^{-\kappa} = (N+2) \cdot 2^{-\kappa}$$
- If the adversary finds a pre-image for one of its commitment $(\mathsf{cmt}_i)_{i \in \mathsf{corrupt}}$ after round 2 is performed. This happens with probability at most $N \cdot Q_H \cdot 2^{-2\kappa}$.
- If the adversary calls round 3 with a different sets of $(\mathsf{cmt}_i)_{i \in \mathsf{corrupt}}$ than what was used for programming in round 2, i.e. if $\mathcal{A}$ finds a collision in H_{cmt}. This happens with probability bounded by $Q_H^2 \cdot 2^{-2\kappa}$.

Thus, the advantage difference with Hybrid_1 is:

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_1} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_2}\right| \leqslant (N+2) \cdot 2^{-\kappa} + (Q_H + N) Q_H \cdot 2^{-2\kappa}$$

Hybrid_3. In this hybrid, we ensure that no complaints are raised by honest parties against honest parties. In particular, it ensures that the private keys of SKE remain hidden. This is formalized in Fig. 11.

The only case where an honest party may complain about another is when a share is larger than the bound η_1. However, this bound is chosen so that this event happens with negligible probability.

Thus,

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_2} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_3}\right| = \mathsf{negl}(\kappa)$$

Hybrid_4. In this hybrid, we replace SKE encryptions to other honest parties, with encryptions of 0, ensuring that honest shares do not leak on transit to the adversary. This is formalized in Fig. 12.

As we made sure no decryptions are performed using the private key of the channel between $i, j \in \mathsf{honest}$, and this private key is only used for the encryption of the shares $\mathbf{s}_I$, we can reduce the advantage loss of Hybrid_4 to the $\mathsf{IND\text{-}CPA}$ security of the SKE. We need to replace up to N^2 encryptions, adding a factor loss N^2.

Formally, there exists an adversary $\mathcal{B}_1$ against the $\mathsf{IND\text{-}CPA}$ security of the SKE such that:

Thus,

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_3} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_4}\right| \leqslant N^2 \cdot \mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{IND\text{-}CPA}}$$

Hybrid_5. In this hybrid, we fix an index I_0 such that $I_0 \subseteq \mathsf{honest}$, i.e. the secret $\mathbf{s}_{I_0}$ is not shared with the adversary. We defer the sampling of $\mathbf{b}_{I_0}$ to the fourth round by programming the random oracle to tackle rushing adversaries. This is formalized in Fig. 13.

First, such an index I_0 exists by the *threshold security* property of replicated secret sharing, as we assume that at most $T-1$ parties are corrupted.

Then, similarly to Hybrid_2, several bad events may happen that make the view of $\mathcal{A}$ differ:

Hybrid_3

▷▷▷ Remains identical ◁◁◁

$\mathsf{OPerformRound}_4((\mathsf{pm}_3^j)_{j\in\mathsf{corrupt}})$

1: **for** $i \in \mathsf{honest}$ **do**
2: Fetch $(\mathsf{seed}, (\mathbf{s}_I)_I)$ from $\mathsf{st}_i.\mathsf{session}$
3: Parse $\mathsf{pm}_3 = (\mathsf{msgs}_j, (\mathsf{cmt}'_I)_{I \text{ s.t. } \max(I)=j})_j$
4: $\mathsf{seed} := H_{\mathsf{seed}}((\mathsf{seed}_i)_i)$; $\mathbf{A} := \mathsf{ExpandA}(\mathsf{seed})$
5: $\mathsf{complaints}_i = \varnothing$
6: **for** I s.t. $|I| = N - T + 1 \wedge i \neq \max(I)$ **do**
7: **if** $\max(I) \in \mathsf{honest}$ **then**
8: **continue** ▷ Do not complain against honest senders
9: $\mathbf{s}_I := \mathsf{SKE.Decrypt}$ from $\mathsf{msgs}_{\max(I)}$
10: $\mathbf{b}_I \leftarrow \left[\mathbf{A}\ \mathbf{I}\right] \cdot \mathbf{s}_I$
11: **if** $\|\mathbf{s}_I\| > \eta_1 \vee \mathsf{cmt}'_I \neq H_{\mathsf{cmt}}(\mathbf{b}_I)$ **then**
12: $\mathsf{complaints}_i := \mathsf{complaints}_i \cup \{I, j := \max(I), K^{\mathsf{SKE}}_{j\to i}, \mathsf{sig}^{\mathsf{SKE}}_{j\to j}\}$
13: $\mathsf{pm}_4^i := (\mathsf{complaints}_i, (\mathbf{b}_I)_I)$
14: **return** $(\mathsf{pm}_4^i)_{i\in\mathsf{honest}}$

Fig. 11: The third hybrid of the security proof of the functional simulatability of our Repl sDKG scheme. Difference with the previous hybrid are highlighted. We omit state updates for conciseness.

– If $\mathcal{A}$ queries H_{cmt} on $\mathbf{b}_{I_0}$ before it is programmed in round 4. Consider two probabilities: (i) p_1 the probability that this event occurs when $\mathbf{b}_{I_0} = \left[\mathbf{A}\ \mathbf{I}\right] \cdot \mathbf{s}_{I_0}$, and (ii) p_2 that it occurs if we had instead sampled $\mathbf{b}_{I_0}$ at random in $\mathcal{R}_q^k$. We can design an adversary $\mathcal{B}_2$ against the problem $\mathsf{MLWE}_{q,k,\ell,\sigma_{\mathbf{s}}}$ that aborts when round 4 starts (before $\mathbf{s}_{I_0}$ is used anywhere), and tries to distinguish the two above scenarios. Then $|p_1 - p_2| \leqslant \mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}_2}$.
Additionally, we have $p_2 \leqslant Q_H \cdot q^k$. Hence the advantage loss for this case is $\mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}_2} + Q_H \cdot q^{-k}$.
– If $\mathcal{A}$ breaks the pre-image or collision resistance of the function H_{cmt}, this happens with probability $Q_H(Q_H + N) \cdot 2^{-2\kappa}$.

We deduce that the advantage difference is,

$$\left|\mathsf{Adv}^{\mathsf{Hybrid}_4}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Hybrid}_5}_{\mathcal{A}}\right| \leqslant \mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}_2} + Q_H \cdot q^{-k} + Q_H(Q_H + N) \cdot 2^{-2\kappa}$$

Hybrid_6. In this hybrid, we reexpress the function f so that it can be evaluated only from hints on $\mathbf{s}_{I_0}$ and from $(\mathbf{s}_I)_{I\neq I_0}$. This is formalized in Fig. 14.

This expression is equivalent and there is no advantage difference.

$$\mathsf{Adv}^{\mathsf{Hybrid}_5}_{\mathcal{A}} = \mathsf{Adv}^{\mathsf{Hybrid}_6}_{\mathcal{A}}$$

Hybrid_4

▷▷▷ Remains identical ◁◁◁

$\mathsf{OPerformRound}_3((\mathsf{pm}_2^j)_{j\in\mathsf{honest}} = (\mathsf{seed}_j)_{j\in\mathsf{corrupt}})$

Retrieve $(\mathsf{seed}_j)_{j\in\mathsf{honest}}$
for $i \in \mathsf{honest}$ **do**
  Parse $\mathsf{pm}_1 = (\mathsf{cmt}_j)_j$, $\mathsf{pm}_2 = (\mathsf{seed}_j)_j$
  for $j \in [N]$ **do**
    if $\{\mathsf{cmt}_j \neq H_{\mathsf{cmt}}(\mathsf{seed}_j)\}$ **then return** $(\mathbf{abort}, \{j\})$
assert$\{$ $\mathsf{ProgrammedSeed} = (\mathsf{seed}_i)_{i\in[N]}$ $\}$
for $i \in \mathsf{honest}$ **do**
  $\mathsf{seed} := H_{\mathsf{seed}}((\mathsf{seed}_j)_j)$
  $\mathbf{A} := \mathsf{ExpandA}(\mathsf{seed})$
  $\mathsf{msgs}_i = \varnothing$
  for I s.t. $|I| = N - T + 1 \wedge i = \max(I)$ **do**
    $\mathbf{s}_I \leftarrow \mathcal{D}_{\mathbf{s}}^{k+\ell}$ ▷ Sample secret share
    $\mathbf{b}_I \leftarrow \left[\mathbf{A}\ \mathbf{I}\right] \cdot \mathbf{s}_I$
    $\mathsf{cmt}'_I := H_{\mathsf{cmt}}(\mathbf{b}_I)$
    for $j \in I \backslash \{i\}$ **do**
      if $j \in \mathsf{honest}$ **then**
        $\mathsf{msgs}_i := \mathsf{msgs}_i \cup \{(I, j, \mathsf{SKE.Encrypt}(K^{\mathsf{SKE}}_{i\to j}, \mathbf{0}))\}$
      else
        $\mathsf{msgs}_i := \mathsf{msgs}_i \cup \{(I, j, \mathsf{SKE.Encrypt}(K^{\mathsf{SKE}}_{i\to j}, \mathbf{s}_I))\}$
  $\mathsf{st}_i.\mathsf{session} := \{\mathsf{seed}, (\mathbf{s}_I)_I\}$
  $\mathsf{pm}_3^i := (\mathsf{msgs}_i, (\mathsf{cmt}'_I)_I)$
return $(\mathsf{pm}_3^i)_{i\in\mathsf{honest}}$

Fig. 12: The fourth hybrid of the security proof of the functional simulatability of our Repl sDKG scheme. Difference with the previous hybrid are highlighted. We omit state updates for conciseness.

Hybrid_7. In this final hybrid, we replace the public key vector $\mathbf{b}$ by a sample from the rounded distribution $\chi_{\mathbf{b}}$. We also sample $\widehat{\mathbf{b}} \xleftarrow{\$} \mathcal{R}_q^k$ conditioned on $\mathbf{b} = \left\lfloor \widehat{\mathbf{b}} \right\rceil_{\nu_{\mathbf{b}}}$, and express $\mathbf{b}_{I_0}$ as a function of $\widehat{\mathbf{b}}$ and $(\mathbf{b}_I)_{I\neq I_0}$. This is formalized in Fig. 15.

Observe that when $\mathbf{b}$ is sampled from $\chi_{\mathbf{b}}$, then $\widehat{\mathbf{b}}$ is uniformly sampled from $\mathcal{R}_q^k$. This implies that $\mathbf{b}_{I_0}$ is uniformly distributed in $\mathcal{R}_q^k$ in Hybrid_7. As $\mathbf{s}_{I_0}$ is only used to compute $\mathbf{b}_{I_0}$ and for hints in OEval, we deduce that distinguishing Hybrid_7 from Hybrid_6 reduces to the problem $\mathsf{Hint\text{-}MLWE}_{q,k,\ell,Q_f,\mathcal{D}_{\mathbf{s}},\mathcal{D}_{\mathbf{r}},\mathcal{C}^{Q_f}}$, where we assume up to Q_f hints on the secret $\mathbf{s}_{I_0} \leftarrow \mathcal{D}_{\mathbf{s}}$ of the form $c \cdot \mathbf{s}_{I_0} + \mathbf{r}$, for $\mathbf{r} \leftarrow \mathcal{D}_{\mathbf{r}}^{k+\ell}$.

Hybrid_5

1: $\mathsf{ProgrammedB} = \bot$

▷▷▷ Rest is identical ◁◁◁

$\mathsf{OPerformRound}_3((\mathsf{pm}_2^j)_{j\in\mathsf{honest}} = (\mathsf{seed}_j)_{j\in\mathsf{corrupt}})$

1: Retrieve $(\mathsf{seed}_j)_{j\in\mathsf{honest}}$
2: **for** $i \in \mathsf{honest}$ **do**
3: Parse $\mathsf{pm}_1 = (\mathsf{cmt}_j)_j$, $\mathsf{pm}_2 = (\mathsf{seed}_j)_j$
4: **for** $j \in [N]$ **do**
5: **if** $\{\mathsf{cmt}_j \neq H_{\mathsf{cmt}}(\mathsf{seed}_j)\}$ **then return** $(\mathbf{abort}, \{j\})$
6: **assert**$\{$ $\mathsf{ProgrammedSeed} = (\mathsf{seed}_i)_{i\in[N]}$ $\}$
7: Fix index I_0 such that $I_0 \subset \mathsf{honest}$
8: $\mathsf{cmt}'_{I_0} \xleftarrow{\$} \{0,1\}^{2\kappa}$
9: **for** $i \in \mathsf{honest}$ **do**
10: $\mathsf{seed} := H_{\mathsf{seed}}((\mathsf{seed}_j)_j)$
11: $\mathbf{A} := \mathsf{ExpandA}(\mathsf{seed})$
12: $\mathsf{msgs}_i = \varnothing$
13: **for** I s.t. $|I| = N - T + 1 \wedge i = \max(I) \wedge I \neq I_0$ **do**

▷▷▷ Rest is identical ◁◁◁

$\mathsf{OPerformRound}_4((\mathsf{pm}_3^j)_{j\in\mathsf{corrupt}})$

1: Parse $\mathsf{pm}_3 = (\mathsf{msgs}_j, (\mathsf{cmt}'_I)_{I \text{ s.t. } \max(I)=j})_j$
2: **if** $\forall I \neq I_0, \exists H_{\mathsf{cmt}}(\mathbf{b}_I) = \mathsf{cmt}'_I$ **then**
3: $\mathsf{ProgrammedB} = (\mathbf{b}_I)_{I\neq I_0}$
4: $\mathbf{s}_{I_0} \leftarrow \mathcal{D}_{\mathbf{s}}$
5: $\mathbf{b}_{I_0} := \left[\mathbf{A}\ \mathbf{I}\right] \cdot \mathbf{s}_{I_0}$
6: $H(\mathbf{b}_{I_0}) := \mathsf{cmt}'_{I_0}$ ▷ Program random oracle
7: **for** $i \in \mathsf{honest}$ **do**

▷▷▷ Other lines are identical ◁◁◁

$\mathsf{CombineKey}(\mathsf{pm}_2, \mathsf{pm}_3, \mathsf{pm}_4)$

1: Parse $\mathsf{pm}_2 = (\mathsf{seed}_i)_i$, $\mathsf{pm}_3 = (\mathsf{msgs}_i, \cdot)_i$, and $\mathsf{pm}_4 = (\mathsf{complaints}_i, (\mathbf{b}_I)_I)_i$
2: **for** set I of cardinality $N - T + 1$ **do**
3: **if** $\mathbf{b}_I \neq \mathsf{cmt}'_I$ **then**
4: **return** $(\mathbf{abort}, \{\max(I)\})$
5: **assert**$\{$ $\mathsf{ProgrammedB} = (\mathbf{b}_I)_{I\neq I_0}$ $\}$

▷▷▷ Other lines are identical ◁◁◁

Fig. 13: The fifth hybrid of the security proof of the functional simulatability of our Repl sDKG scheme. Difference with the previous hybrid are highlighted. We omit state updates for conciseness.

Hybrid_6

▷▷▷ Identical ◁◁◁

$\mathsf{OEval}(\mathsf{in} := \mathsf{act})$

1: $(\nu_{i,I})_{i\in\mathsf{act},I \text{ s.t. } i\in I} := \mathsf{Repl.Recover}(\mathsf{pk}, \mathsf{aux}, \mathsf{act})$
2: $(\mathbf{r}_i)_{i\in\mathsf{honest}\cap\mathsf{act}} \leftarrow \mathcal{D}_{\mathbf{r}}^{(k+\ell)\cdot|\mathsf{honest}\cap\mathsf{act}|}$
3: $c \leftarrow \mathcal{C}$
4: $\mathbf{h} := c \cdot \mathbf{s}_{I_0} + \mathbf{r}_{\max(I_0\cap\mathsf{act})}$
5: $\mathbf{h}_{\max(I_0\cap\mathsf{act})} := \mathbf{h} + c \cdot \sum_{I\neq I_0} \nu_{i,I}$
6: **for** $i \neq \max(I_0 \cap \mathsf{act})$ **do**
7: $\quad \mathbf{h}_i := c \cdot \sum_I \nu_{i,I} \cdot S_i[I] + \mathbf{r}_i$
8: **return** $(\mathbf{h}_i)_{i\in\mathsf{honest}\cap\mathsf{act}}$

Fig. 14: The sixth hybrid of the security proof of the functional simulatability of our Repl sDKG scheme. Difference with the previous hybrid are highlighted. We omit state updates for conciseness.

Hybrid_7

▷▷▷ Identical ◁◁◁

$\mathsf{OPerformRound}_4((\mathsf{pm}_3^j)_{j\in\mathsf{corrupt}})$

1: Parse $\mathsf{pm}_3 = (\mathsf{msgs}_j, (\mathsf{cmt}'_I)_{I \text{ s.t. } \max(I)=j})_j$
2: **if** $\forall I \neq I_0, \exists H_{\mathsf{cmt}}(\mathbf{b}_I) = \mathsf{cmt}'_I$ **then**
3: $\quad \mathsf{ProgrammedB} = (\mathbf{b}_I)_{I\neq I_0}$
4: $\mathbf{s}_{I_0} \leftarrow \mathcal{D}_{\mathbf{s}}$
5: $\mathbf{b} \leftarrow \chi_{\mathbf{b}}$
6: Sample $\widehat{\mathbf{b}} \xleftarrow{\$} \mathcal{R}_q^k$ conditioned on $\mathbf{b} = \left\lfloor \widehat{\mathbf{b}} \right\rceil_{\nu_{\mathbf{b}}}$
7: $\mathbf{b}_{I_0} := \widehat{\mathbf{b}} - \sum_{I\neq I_0} \mathbf{b}_I$
8: $H(\mathbf{b}_{I_0}) := \mathsf{cmt}'_{I_0}$ ▷ Program random oracle
9: **for** $i \in$ honest **do**
▷▷▷ Other lines are identical ◁◁◁

Fig. 15: The seventh hybrid of the security proof of the functional simulatability of our Repl sDKG scheme. Difference with the previous hybrid are highlighted. We omit state updates for conciseness.

Formally, there exists an adversary $\mathcal{B}_3$ against the $\mathsf{Hint\text{-}MLWE}_{q,k,\ell,Q_f,\mathcal{D}_{\mathbf{s}},\mathcal{D}_{\mathbf{r}},\mathcal{D}^{Q_f}}$ problem such that:

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_6} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_7}\right| \leqslant \mathsf{Adv}_{\mathcal{B}_3}^{\mathsf{Hint\text{-}MLWE}}$$

We observe that this last hybrid corresponds to the case $b = 1$ of $\mathsf{Game}_{\mathsf{sDKG\text{-}sim}}$, where we define the simulators $\mathsf{SimKeygen}, \mathsf{SimF}$ based on the hybrid.

We conclude the proof of Theorem 2 by summing all the advantage differences. □

B Formal construction of the distributed ramp sDKG

Due to a lack of space, we defer the formal description of our ramp sDKG to the appendices. We concretize the construction described in Section 5.3, in the Fig. 16 and Fig. 17.

B.1 Proofs of security

Proofs of simulatability and identifiable aborts are analogous to our Repl sDKG and we only provide proof overviews here.

Proof (of Theorem 4). The proof is analogous to the proof of Theorem 2. It consists of observing that at least one of the partial secret $\mathbf{s}_{i_0}$ will remain unknown to the adversary. Two type of leakages appear on $\mathbf{s}_{i_0}$:

- Through the shares that are leaked to the adversary $[\![\mathbf{s}_{i_0}]\!]_j^v$. In case $v = n$ remains hidden, this is fine as the other shares are independent of $\mathbf{s}_{i_0}$. If the share $v = n$ leaks, we can see this as a hint on $\mathbf{s}_{i_0}$, as at least one share remains safe in the sharing $[\![\mathbf{s}_{i_0}]\!]_j$. This requires up to m hints on $\mathbf{s}_{i_0}$ of the form $\mathbf{s}_{i_0} + \mathbf{s}'$.
- Through the evaluations of f. We can reexpress hints on the honest shares of $[\![\mathbf{s}_{i_0}]\!]_j$ as hints on $\mathbf{s}_{i_0}$. This requires up to Q_f hints on $\mathbf{s}_{i_0}$ of the form $c \cdot \mathbf{s}_{i_0} + \mathbf{r}$.
  In more detail, assuming the share v_0 remains hidden in the sharing of $\mathbf{s}_{i_0}$:
  - if $v_0 < n$, we have:

$$c \cdot [\![\mathbf{s}_{i_0}]\!]_j^{v_0} + \mathbf{y}_i = c \cdot (\underbrace{\sum_{v<n} [\![\mathbf{s}_{i_0}]\!]_j^v - \mathbf{s}_{i_0})}_{=[\![\mathbf{s}_{i_0}]\!]_j^v} + (c \cdot \mathbf{s}_{i_0} + \mathbf{y}_i) - \sum_{v<n \wedge v \neq j} \underbrace{[\![\mathbf{s}_{i_0}]\!]_j^v}_{\text{leaked}},$$

    i.e. we reexpress a hint on $[\![\mathbf{s}_{i_0}]\!]_j$ as a function of a hint on $\mathbf{s}_{i_0}$, and of the leaked shares.
  - if $v_0 = n$, we have:

$$c \cdot [\![\mathbf{s}_{i_0}]\!]_j^{v_0} + \mathbf{y}_i = c \cdot \sum_{v<n} \underbrace{[\![\mathbf{s}_{i_0}]\!]_j^v}_{\text{leaked}} - (c \cdot \mathbf{s}_{i_0} + \mathbf{y}_i).$$

    Again, a hint on $[\![\mathbf{s}_{i_0}]\!]_j^{v_0}$ can be expressed as a function of a hint on $\mathbf{s}_{i_0}$, and of leaked shares.

□

Proof (of Theorem 5). The proof is analogous to Theorem 3:

Alg. 17 $\mathsf{ShareKeygen}_1(\mathsf{st}_i)$

$\mathsf{seed}_i \xleftarrow{\$} \{0,1\}^\kappa$
$\mathsf{st}_i.\mathsf{session} := \{\mathsf{seed}_i\}$
$\mathsf{cmt}_i = H_{\mathsf{cmt}}(\mathsf{seed}_i)$
return $\mathsf{pm}_1^i := \mathsf{cmt}_i$

Alg. 18 $\mathsf{ShareKeygen}_2(\mathsf{st}_i, \mathsf{pm}_1)$

Fetch (seed_i) from $\mathsf{st}_i.\mathsf{session}$
$\mathsf{st}_i.\mathsf{session} := \{\mathsf{pm}_1\}$
return $\mathsf{pm}_2^i := \mathsf{seed}_i$

Alg. 19 $\mathsf{ShareKeygen}_3(\mathsf{st}_i, \mathsf{pm}_2)$

Fetch (pm_1) from $\mathsf{st}_i.\mathsf{session}[\mathsf{sid}]$
Parse $\mathsf{pm}_1 = (\mathsf{cmt}_j)_j$, $\mathsf{pm}_2 = (\mathsf{seed}_j)_j$
for $j \in [N]$ **do**
  if $\{\mathsf{cmt}_j \neq H_{\mathsf{cmt}}(\mathsf{seed}_j)\}$ **then return** $(\mathbf{abort}, \{j\})$ ▷ Check hash commit.
$\mathsf{seed} := H_{\mathsf{seed}}((\mathsf{seed}_j)_j)$; $\mathbf{A} := \mathsf{ExpandA}(\mathsf{seed})$
$\mathbf{s}_i \leftarrow \mathcal{D}_{\mathbf{s}}^{k+\ell}$; $\mathbf{b}_i := \left[\mathbf{A}\ \mathbf{I}\right] \cdot \mathbf{s}_i$ ▷ Sample partial secret $\mathbf{s}_i$
$\mathsf{cmt}_{\mathbf{b}}^i := H_{\mathsf{cmt}}(\mathbf{b}_i)$
$\mathsf{msgs}_i = \varnothing$
for $j = 1, ..., m$ **do**
  for $v = 1, ..., n$ **do**
    $[\![\mathbf{s}_i]\!]_j^v \leftarrow \begin{cases} \mathcal{D}_{\mathsf{share}}^{k+\ell} & \text{if } v < n \\ \mathbf{s}_i - \sum_{v=1}^{n-1} [\![\mathbf{s}_i]\!]_j^v & \text{otherwise} \end{cases}$
    $[\![\mathbf{b}_i]\!]_j^v \leftarrow \left[\mathbf{A}\ \mathbf{I}\right] \cdot [\![\mathbf{s}_i]\!]_j^v$
    $\mathsf{cmt}'_{i,v,j} := H_{\mathsf{cmt}}([\![\mathbf{b}_i]\!]_j^v)$
    for u such that $\pi_n(\mathsf{seed}, u)_j = v$ **do** ▷ Party u knows secret (v, j)
      $\mathsf{msgs}_i := \mathsf{msgs}_i \cup \{(u, (v, j), \mathsf{SKE.Encrypt}(K_{i\to u}^{\mathsf{SKE}}, [\![\mathbf{s}_i]\!]_j^v))\}$
$\mathsf{st}_i.\mathsf{session} := \{\mathsf{seed}\}$
return $\mathsf{pm}_3^i := (\mathsf{msgs}_i, (\mathsf{cmt}'_{i,v,j})_{j,v})$

Alg. 20 $\mathsf{ShareKeygen}_4(\mathsf{st}_i, \mathsf{pm}_3)$

Fetch (seed) from $\mathsf{st}_i.\mathsf{session}$
Parse $\mathsf{pm}_3 = (\mathsf{msgs}_j, (\mathsf{cmt}'_I)_{I \text{ s.t. } \max(I)=j})_j$
$\mathsf{seed} := H_{\mathsf{seed}}((\mathsf{seed}_i)_i)$; $\mathbf{A} := \mathsf{ExpandA}(\mathsf{seed})$
$\mathsf{complaints}_i = \varnothing$
for $j = 1, ..., m$ **do**
  for $u \in [N]$ **do**
    $[\![\mathbf{s}_u]\!]_j^{\pi_n(\mathsf{seed},i)_j} := \mathsf{SKE.Decrypt}$ from msgs_u ▷ Recover $\mathbf{s}_I$
    if $\|[\![\mathbf{s}_u]\!]_j^{\pi_n(\mathsf{seed},i)_j}\| > \eta_1^{\mathsf{ind}} \vee \mathsf{cmt}'_I \neq H_{\mathsf{cmt}}(\left[\mathbf{A}\ \mathbf{I}\right] \cdot [\![\mathbf{s}_u]\!]_j^{\pi_n(\mathsf{seed},i)_j})$ **then**
      $\mathsf{complaints}_i := \mathsf{complaints}_i \cup \{u, j, K_{j\to i}^{\mathsf{SKE}}, \mathsf{sig}_{j\to j}^{\mathsf{SKE}}\}$
return $\mathsf{pm}_4^i := (\mathsf{complaints}_i, \mathbf{b}_i, ([\![\mathbf{b}_i]\!]_j^v)_{j,v})$

Fig. 16: Algorithms for our ramp sDKG. For conciseness, we omit aborts due to parsing errors: if an unexpected message is received, parties abort and report the sender as malicious.

Alg. 21 $\mathsf{CombineKey}(\mathsf{pm}_2, \mathsf{pm}_3, \mathsf{pm}_4)$

Parse $\mathsf{pm}_2 = (\mathsf{seed}_i)_i$ and $\mathsf{pm}_3 = (\mathsf{msgs}_i, \cdot)_i$
Parse $\mathsf{pm}_4 = (\mathsf{complaints}_i, \mathbf{b}_i, (\llbracket \mathbf{s}_u \rrbracket_j^{\pi_n(\mathsf{seed},i)_j})_{j,v})_i$
for $i \in [N]$ **do**
  if $H_{\mathsf{cmt}}(\mathbf{b}_i) \neq \mathsf{cmt}_{\mathbf{b}}^i$ **then**
    return $(\mathbf{abort}, \{i\})$
$\mathsf{seed} := H_{\mathsf{seed}}((\mathsf{seed}_i)_i)$
$\mathsf{invalid} = \varnothing$
for $i \in [N]$ **do**
  if $\forall j \in [m], \mathbf{b}_i \neq \sum_{v \in [n]} \llbracket \mathbf{b}_i \rrbracket_j^v$ **then**
    $\mathsf{invalid} := \mathsf{invalid} \cup \{i\}$
  for $(u, j, K_{u \to i}^{\mathsf{SKE}}, \mathsf{sig}_{u \to i}^{\mathsf{SKE}}) \in \mathsf{complaints}_i$ **do**
    if $\neg \mathsf{SIG.Verify}(\mathsf{sig}_{u \to i}^{\mathsf{SKE}}, K_{u \to i}^{\mathsf{SKE}})$ **then**
      $\mathsf{invalid} := \mathsf{invalid} \cup \{i\}$
    else
      $\llbracket \mathbf{s}_u \rrbracket_j^{\pi_n(\mathsf{seed},i)_j} := \mathsf{SKE.Decrypt}$ using $K_{i \to u}^{\mathsf{SKE}}$ from msgs_u
      $\llbracket \mathbf{b}_u \rrbracket_j^{\pi_n(\mathsf{seed},i)_j} \leftarrow [\mathbf{A}\ \mathbf{I}] \cdot \llbracket \mathbf{s}_u \rrbracket_j^{\pi_n(\mathsf{seed},i)_j}$
      if $\| \llbracket \mathbf{s}_u \rrbracket_j^{\pi_n(\mathsf{seed},i)} \| > \eta_1^{\mathsf{ind}} \vee \mathsf{cmt}'_{i,j,\pi_n(\mathsf{seed},i)_j} \neq H_{\mathsf{cmt}}(\llbracket \mathbf{b}_u \rrbracket_j^{\pi_n(\mathsf{seed},i)_j})$ **then**
        $\mathsf{invalid} := \mathsf{invalid} \cup \{j\}$
if $\mathsf{invalid} \neq \varnothing$ **then**
  return $(\mathbf{abort}, \mathsf{invalid})$
$\mathbf{b} := \left\lfloor \sum_{i \in [N]} \mathbf{b}_i \right\rceil_{\nu_{\mathbf{b}}}$
return $\mathsf{pk} := (\mathsf{seed}, \mathbf{b}), \mathsf{aux} := (\mathbf{b}_j^v := \sum_{i \in [N]} \llbracket \mathbf{b}_i \rrbracket_j^v)_{j \in [m], v \in [n]}$

Alg. 22 $\mathsf{CombinePrivKey}(\mathsf{st}_i, \mathsf{pm}_4)$

Parse $\mathsf{pm}_4 = (\mathsf{complaints}_i, \mathbf{b}_i, (\llbracket \mathbf{s}_u \rrbracket_j^{\pi_n(\mathsf{seed},i)_j})_{j,v})_i$
$\mathsf{seed} := H_{\mathsf{seed}}((\mathsf{seed}_i)_i)$
$v_1, ..., v_m = \pi_n(\mathsf{seed}, i)$
for $j \in [m]$ **do**
  $\mathbf{s}_j^{v_j} := \sum_{i \in [N]} \llbracket \mathbf{s}_i \rrbracket_j^{v_j}$
return $\mathsf{sk}_i := (\mathbf{s}_j^{v_j})_{j \in [m]}$

Fig. 17: Aggregation for our ramp sDKG. For conciseness, we omit aborts due to parsing errors: if an unexpected message is received, parties abort and report the sender as malicious.

– We show that honest parties can never be identified as malicious thanks to the unforgeability of the signature scheme ensuring the authenticity of the encryptions in case of complaints, and as honest parties provide consistently built elements to other parties.

– In case malicious parties provide honest parties with (i) large or (ii) inconsistent shares, they are detected during round 4 for (i) by having honest parties explicitly check for a norm bound, and during key aggregation for (ii) by checking that partial public keys of the sharings sampled by a malicious party correctly reconstruct to this party full partial public key.

□

C Omitted Definitions and Security Proofs of Threshold Signature with Identifiable Aborts

In this section, we formally define threshold signatures with identifiable aborts and provide proofs for our scheme that were omitted in the main body of this paper.

C.1 Definitions

We first provide complete definitions for a threshold signature scheme with identifiable abort, as was introduced in Section 6.1.

We use the definition of security with identifiable aborts of [IOZ14]. Namely, we describe our protocols as *synchronous protocols*, i.e. round-based protocols where messages sent in some round are delivered by the start of the next round, and, at the end of each round, parties are allowed to *abort* under the condition that they identify a misbehaving party.

We first define TSS with identifiable aborts, by adapting the definition from [DKM+24] to cope with the identifiable aborts as in [IOZ14]. We also cover the case of *ramp TSS*, for which there is a gap between the corruption threshold for unforgeability, and the minimal number of parties required to sign a message.

Definition 8 (Threshold signature with identifiable aborts). *Consider* $1 \leqslant T \leqslant N$. *A* rnd*-round* (T, N) *TSS with identifiable aborts is a tuple of algorithms* $(\mathsf{Keygen}, (\mathsf{ShareSign}_k)_{k\in[\mathsf{rnd}]}, \mathsf{Combine}, \mathsf{Verify})$ *such that:*

$\mathsf{Keygen}(N, T) \to (\mathsf{vk}, \mathsf{ak}, (\mathsf{sk}_i)_{i\in\mathsf{honest}}, (\mathsf{status}_i)_{i\in[N]})$**.** *The (distributed) key generation is an interactive protocol, as defined in Definition 5. In case of success, it returns a full public key* vk*, public information* ak *allowing abort identification, and secret key shares* $(\mathsf{sk}_1, ..., \mathsf{sk}_N)$*. For conciseness, we leave implicit the setup, and algorithms for each round defined in Definition 5.*

$\mathsf{ShareSign}_1(\mathsf{st}_i, \mathsf{sid}, \mathsf{act}, \mathsf{msg}) \to (\mathsf{st}_i, \mathsf{pm}_1^i$ **or** $(\mathbf{abort}, \mathsf{S}_\perp^{\mathsf{i}}))$**.** *The first round signing algorithm run by user* i*, takes as input an internal state* st_i *(including the secret key share of* i*), a session id* sid*, a signing set* $\mathsf{act} \subseteq [N]$ *of size* T*, and a message* msg*. It returns an updated internal state* st_i*, and either a response* pm_1^i*, or an* **abort** *signal with a set* $\mathsf{S}_\perp^{\mathsf{i}}$ *of identified malicious users.*

$\mathsf{ShareSign}_k(\mathsf{st}_i, \mathsf{sid}, (\mathsf{pm}_{k-1}^j)_{j\in[N]}) \to (\mathsf{st}_i, \mathsf{pm}_k^i$ **or** $(\mathbf{abort}, \mathsf{S}_\perp^{\mathsf{i}}))$**.** *The* k*-th round signing algorithm for* $2 \leqslant k \leqslant \mathsf{rnd}$ *run by user* i*, takes as input an internal state* st_i*, a session id* sid*, and responses from all users from round* $k-1$*. It returns an updated internal state* st_i*, and either a response* pm_j^i*, or an* **abort** *command with a set* $\mathsf{S}_\perp^{\mathsf{i}}$ *of identified malicious users.*

$\mathsf{Combine}(\mathsf{vk}, \mathsf{ak}, \mathsf{sid}, \mathsf{msg}, (\mathsf{pm}_k^i)_{k\in[\mathsf{rnd}],i\in[N]}) \to \mathsf{sig}$ **or** $(\mathbf{abort}, \mathsf{S}_\perp)$**.** *The aggregation algorithm combines the contribution from all rounds to produce a signature* sig *for the message* msg *under the public key* vk*, or an* **abort** *with a set of identified dishonest parties.*

$\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}) \to \{0,1\}$**.** *The verification algorithm checks that a signature* sig *is deemed valid with regard to a public key* vk *for a message* msg*.*

We now define security notions for TSS with identifiable aborts: *correctness*, *unforgeability*, and *identifiable aborts*.

As for sDKG, we require a communication model reliably exchanging messages in order to achieve identifiable aborts. Although we will use this notion for all our notions for simplicity, we note that the unforgeability of our scheme could be proved in a stronger communication model, where the adversary fully owns the communication layer as in [DKM+24].

Unforgeability We start by introducing the notion of unforgeability for a TSS with identifiable aborts. We cover *ramp TSS* in our unforgeability definition by introducing a maximal number of corrupted parties T_{corrupt}.

Definition 9 (Unforgeability). *A (N,T)-TSS with idenfiable aborts is unforgeable up to T_{corrupt} corrupted parties, if for any efficient adversary $\mathcal{A}$, its advantage in the game $\mathsf{Game}_{\mathsf{TH\text{-}UF}}$ from Fig. 18 is negligible, i.e.*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{TH\text{-}UF}}(1^\kappa) = \mathsf{negl}(\kappa)$$

Note that a TSS may support up to $T_{\mathsf{corrupt}} = T-1$ corrupted parties. When $T_{\mathsf{corrupt}} < T-1$, the TSS is called a ramp TSS*.*

Identifiable aborts The main novelty of our scheme lies in the notion we present now, which relates to the soundness of identifiable aborts. Definition 10 states that when the protocol is executed over an authenticated broadcast channel, or over an authenticated bulletin board, we want one of the two following scenarios to occur:

1. either the protocol terminates and produces a valid signature.
2. or, all honest parties abort at some round $k \in [\mathsf{rnd}]$, and they agree on a set of cheating parties $\mathsf{S}_\perp \subseteq \mathsf{corrupt}$.

Definition 10 (Identifiable aborts). *A (N,T)-TSS can* identify aborts *for valid signer sets* act *verifying a predicate $\mathcal{P}(\mathsf{vk},\cdot)$, if the advantage of any efficient $\mathcal{A}$ in the game $\mathsf{Game}_{\mathsf{TH\text{-}ID}}$ from Fig. 19 is negligible:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{TH\text{-}ID}}(1^\kappa) = \mathsf{negl}(\kappa)$$

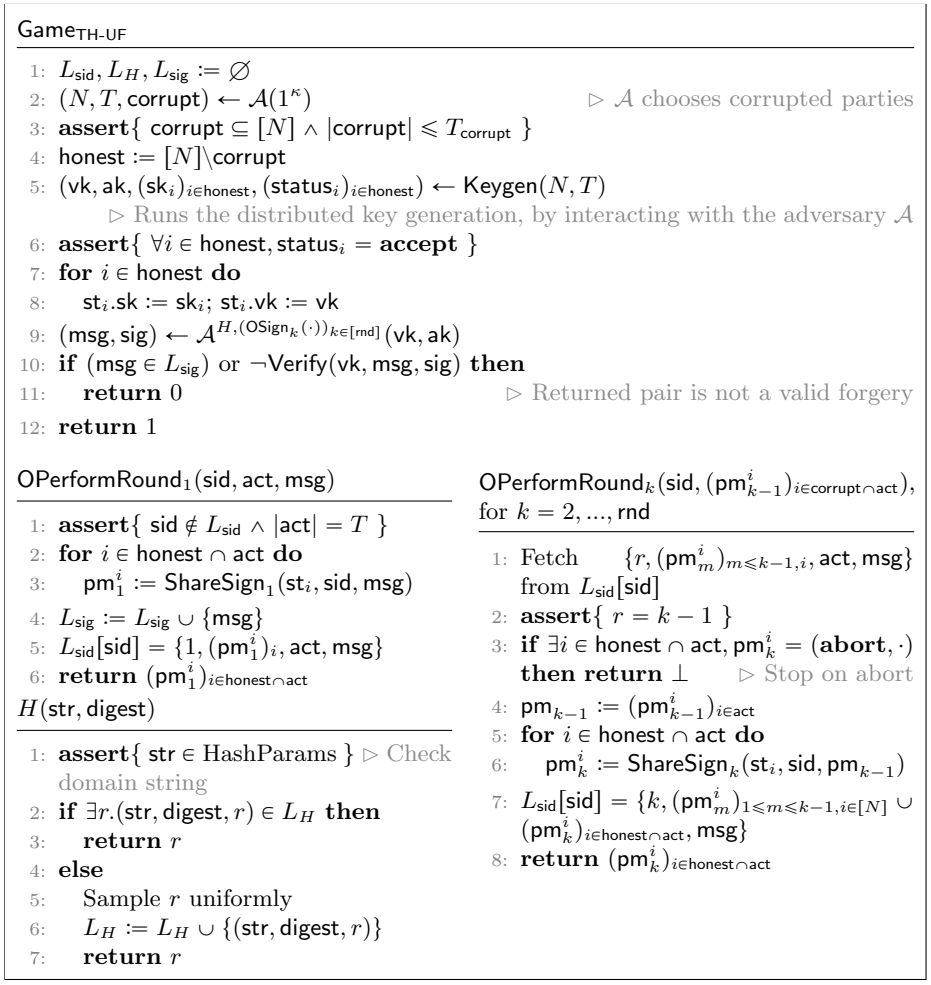


Fig. 18: Unforgeability game for a TSS with identifiable aborts. The adversary $\mathcal{A}$ wins if the game TH-UF returns 1, i.e. if it forged a new signature.

Remark 3. The predicate $\mathcal{P}$ is introduced to support ramp schemes, for which a portion of signer sets act of cardinality T are not supported. If unspecified, $\mathcal{P} = 1$, i.e. the protocol works with any set act of cardinality T. If $\mathcal{P}$ is 1 w.o.p. for any given act, we keep some notion of abort identification. A failure of the protocol could be attributed to whoever chose the signer set.

The standard notion of correctness is straightforwardly implied by abort identification when the predicate $\mathcal{P}$ passes with overwhelming probability for any set act of cardinality T over the randomness of vk. We will thus limit our proofs

$\mathsf{Game}_{\mathsf{TH\text{-}ID}}$

$L_H, L_{\mathsf{sid}}, L_{\mathsf{sig}} := \varnothing$
$(N, T, \mathsf{corrupt}) \leftarrow \mathcal{A}(1^\kappa)$
assert$\{\ \mathsf{corrupt} \subseteq [N] \wedge |\mathsf{corrupt}| < T\ \}$
$\mathsf{honest} := [N] \backslash \mathsf{corrupt}$
$(\mathsf{vk}, \mathsf{ak}, (\mathsf{sk}_i)_{i \in \mathsf{honest}}, (\mathsf{status}_i)_{i \in \mathsf{honest}}) \leftarrow \mathsf{Keygen}(N, T)$
assert$\{\ \forall i \in \mathsf{honest}, \mathsf{status}_i = \mathbf{accept}\ \}$
for $i \in \mathsf{honest}$ **do**
  $\mathsf{st}_i.\mathsf{sk} := \mathsf{sk}_i$; $\mathsf{st}_i.\mathsf{vk} := \mathsf{vk}$
$\mathsf{sid}, (\mathsf{pm}^i_{\mathsf{rnd}})_{i \in \mathsf{corrupt} \cap \mathsf{act}} \leftarrow \mathcal{A}^{H, (\mathsf{OPerformRound}_k(\cdot))_{k \in [\mathsf{rnd}]}}(\mathsf{vk}, \mathsf{ak})$
Fetch $L_{\mathsf{sid}}[\mathsf{sid}] = \{k, (\mathsf{pm}^i_m)_{m \leqslant k, i \in \mathsf{act}}, \mathsf{act}, \mathsf{msg}\}$ ▷ Return 0 if invalid
if $\exists i \in \mathsf{honest} \cap \mathsf{act}, \mathsf{pm}^i_r = (\mathbf{abort}, \mathsf{S}_\perp)$ **then**
  return $\exists i' \in \mathsf{honest} \cap \mathsf{act}, \mathsf{pm}^{i'}_k \neq (\mathbf{abort}, \mathsf{S}_\perp) \vee \mathsf{S}_\perp = \varnothing \vee \mathsf{S}_\perp \nsubseteq \mathsf{corrupt}$
    ▷ $\mathcal{A}$ wins if honest parties did not agree on a valid set of cheating parties
if $k = \mathsf{rnd}$ **then** ▷ In case the protocol succesfully terminated
  $\mathsf{sig} := \mathsf{Combine}(\mathsf{vk}, \mathsf{ak}, \mathsf{sid}, \mathsf{msg}, (\mathsf{pm}^i_{k'})_{k' \in [\mathsf{rnd}], i \in \mathsf{act}})$
  if $\mathsf{sig} = (\mathbf{abort}, \mathsf{S}_\perp)$ **then** ▷ Abort during aggregation
    return $\mathsf{S}_\perp \neq \varnothing \wedge \mathsf{S}_\perp \subseteq \mathsf{corrupt}$ ▷ Identified parties are corrupted
  return $\neg\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$ ▷ If no abort, signature must be valid
return 0

$\mathsf{OPerformRound}_1(\mathsf{sid}, \mathsf{act}, \mathsf{msg})$

assert$\{\ \mathsf{sid} \notin L_{\mathsf{sid}} \wedge |\mathsf{act}| = T\ \}$
assert$\{\ \mathcal{P}(\mathsf{vk}, \mathsf{act})\ \}$
  ▷ Set of signers act is valid
for $i \in \mathsf{honest} \cap \mathsf{act}$ **do**
  $\mathsf{pm}^i_1 := \mathsf{ShareSign}_1(\mathsf{st}_i, \mathsf{sid}, \mathsf{msg})$
$L_{\mathsf{sig}} := L_{\mathsf{sig}} \cup \{\mathsf{msg}\}$
$L_{\mathsf{sid}}[\mathsf{sid}] = \{1, (\mathsf{pm}^i_1)_i, \mathsf{act}, \mathsf{msg}\}$
return $(\mathsf{pm}^i_1)_{i \in \mathsf{honest} \cap \mathsf{act}}$

$H(\mathsf{str}, \mathsf{digest})$

▷▷▷ Same as in Fig. 18 ◁◁◁

$\mathsf{OPerformRound}_k(\mathsf{sid}, (\mathsf{pm}^i_{k-1})_{i \in \mathsf{corrupt} \cap \mathsf{act}})$, for $k = 2, ..., \mathsf{rnd}$

Fetch $\{r, (\mathsf{pm}^i_m)_{m \leqslant k-1, i}, \mathsf{act}, \mathsf{msg}\}$ from $L_{\mathsf{sid}}[\mathsf{sid}]$
assert$\{\ r = k - 1\ \}$
if $\exists i \in \mathsf{honest} \cap \mathsf{act}, \mathsf{pm}^i_k = (\mathbf{abort}, \cdot)$ **then return** $\perp$ ▷ Stop on abort
$\mathsf{pm}_{k-1} := (\mathsf{pm}^i_{k-1})_{i \in \mathsf{act}}$
for $i \in \mathsf{honest} \cap \mathsf{act}$ **do**
  $\mathsf{pm}^i_k := \mathsf{ShareSign}_k(\mathsf{st}_i, \mathsf{sid}, \mathsf{pm}_{k-1})$
$L_{\mathsf{sid}}[\mathsf{sid}] = \{k, (\mathsf{pm}^i_m)_{1 \leqslant m \leqslant k-1, i \in [N]} \cup (\mathsf{pm}^i_k)_{i \in \mathsf{honest} \cap \mathsf{act}}, \mathsf{msg}\}$
return $(\mathsf{pm}^i_k)_{i \in \mathsf{honest} \cap \mathsf{act}}$

Fig. 19: Identifiable abort game for a TSS. The adversary $\mathcal{A}$ wins if the game TH-ID returns 1, i.e. if the protocol produced an invalid signature, or in case of abort if honest parties did not agree on a set of cheating parties.

to abort identification. For completeness, we include the formal definition of correctness in Section C.2.

C.2 Correctness of a threshold signature

For completeness, we include the formal definition of the correctness of a threshold signature.

Definition 11 (Correctness). *A (T, N)-threshold signature scheme with identifiable aborts is correct if for any set* act *of T parties, message* msg*, and session id* sid*, the game* $\mathsf{Game}_{\mathsf{correct}}$ *from Fig. 20 returns 1 w.o.p.*

$\mathsf{Game}_{\mathsf{correct}}(N, T, \mathsf{act}, \mathsf{sid})$

$(\mathsf{vk}, \mathsf{ak}, (\mathsf{sk}_i)_{i\in[N]}) \leftarrow \mathsf{Keygen}(N, T)$
for $i \in [N]$ **do**
  $\mathsf{st}_i.\mathsf{sk} := \mathsf{sk}_i$; $\mathsf{st}_i.\mathsf{vk} := \mathsf{vk}$
for $i \in \mathsf{act}$ **do** $\mathsf{pm}_1^i = \perp$;
for $k = 1, ..., \mathsf{rnd}$ **do**
  for $i \in [N]$ **do**
    $\mathsf{st}_i, \mathsf{pm}_k^i \leftarrow \mathsf{ShareSign}_k(\mathsf{st}_i, \mathsf{sid}, \mathsf{pm}_{k-1}, \mathsf{msg})$
    if $\mathsf{pm}_k^i = (\mathbf{abort}, \cdot)$ **then** ▷ Unexpected early termination
      return 0
$\mathsf{sig} := \mathsf{Combine}(\mathsf{vk}, \mathsf{ak}, \mathsf{sid}, \mathsf{msg}, (\mathsf{pm}_k^i)_{k\in[\mathsf{rnd}],i\in[N]})$
if $\mathsf{sig} = (\mathbf{abort}, \cdot)$ **then**
  return 0
return $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$

Fig. 20: Correctness game for a threshold signature with identifiable aborts. The threshold signature is correct if the above game returns 1 w.o.p.

As remarked before, correctness is straightforwardly implied by abort identification as we can simply define an adversary $\mathcal{B}$ against the abort identification property, that corrupts no user, and honestly executes the signature protocol. If $\mathcal{B}$ has a negligible advantage against the TH-ID game, it ensures that no **abort** will occur, and that a valid signature will be produced w.o.p.

C.3 Proof of Unforgeability of our TSS scheme

We prove that our TSS scheme is unforgeable (Theorem 6).

The following property from [DKM+24] will be useful in the security proof.

Definition 12. *For any $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q k \times \ell$, let $\mathcal{D}^{\mathsf{bd-MLWE}}_{q,l,\ell,\sigma,\nu}(\mathbf{A})$ be the distribution defined as $\left\{\lfloor \mathbf{A}\cdot\mathbf{s}+\mathbf{e}\rceil_\nu \mid (\mathbf{s},\mathbf{e}) \leftarrow \mathcal{D}_\sigma^\ell \times \mathcal{D}_\sigma^k\right\}$. That is, it samples an $\mathsf{MLWE}_{q,k,\ell}$ instance with ν bits dropped.*

Lemma 4. *For any $\sigma > 2d \cdot q^{\frac{1}{k+\ell}+\frac{2}{d\ell}}$ and $\nu < \log(q) - 2$, we have:*

$$\Pr_{\mathbf{A}\xleftarrow{\$}\mathcal{R}^{k\times\ell}}\left[H_\infty(\mathcal{D}^{\mathsf{bd-MLWE}}_{q,l,\ell,\sigma,\nu}(\mathbf{A})) \geqslant d-1\right] \geqslant 1-2^{-d+1}$$

We refer to [DKM+24, Lemma 3.8] for the proof of Lemma 4.

We now prove Theorem 6, i.e. the unforgeability of our threshold signature scheme with identifiable abort.

Proof (of Theorem 6). We prove the unforgeability of our threshold signature scheme with identifiable aborts with a series of hybrid games, starting from the TH-UF game against our signature scheme in the random oracle model, and finishing with a game where we can build an adversary $\mathcal{B}_2$ against SelfTargetMSIS. Let $\mathcal{A}$ be an adversary against the TH-UF security game.

Hybrid_1. This is the original TH-UF game from Fig. 18.

Hybrid_2. In this hybrid, we replace the hash commitment in the first round of honest parties by random values $\mathsf{cmt}_i \xleftarrow{\$} \{0,1\}^{2\kappa}$. The random oracle is later programmed in round 2 to remain consistent. This is formalized in Fig. 21.

Hybrid_2

▷▷▷ Remains identical ◁◁◁

$\mathsf{OPerformRound}_1(\mathsf{sid}, \mathsf{act}, \mathsf{msg})$

1: **req**{ $\mathsf{sid} \notin L_{\mathsf{sid}} \wedge |\mathsf{act}| = T$ }
2: **for** $i \in \mathsf{honest} \cap \mathsf{act}$ **do**
3: $\quad \mathsf{pm}_1^i := \mathsf{cmt}_i \xleftarrow{\$} \{0,1\}^{2\kappa}$ ▷ Replace commit with a random value
4: $L_{\mathsf{sig}} := L_{\mathsf{sig}} \cup \{\mathsf{msg}\}$
5: $L_{\mathsf{sid}}[\mathsf{sid}] = \{1, (\mathsf{pm}_1^i)_{i \in \mathsf{honest} \cap \mathsf{act}}, \mathsf{act}, \mathsf{msg}\}$
6: **return** $(\mathsf{pm}_1^i)_{i \in \mathsf{honest} \cap \mathsf{act}}$

$\mathsf{OPerformRound}_2(\mathsf{sid}, (\mathsf{pm}_1^i)_{i \in \mathsf{corrupt} \cap \mathsf{act}})$

1: Fetch $\{r, (\mathsf{pm}_1^i)_{i \in \mathsf{honest} \cap \mathsf{act}}, \mathsf{act}, \mathsf{msg}\}$ from $L_{\mathsf{sid}}[\mathsf{sid}]$
2: **req**{ $r = k - 1$ }
3: Parse $(\mathsf{cmt}_i)_{i \in \mathsf{act}} = (\mathsf{pm}_1^i)_{i \in \mathsf{act}}$
4: **if** $\exists i \in \mathsf{honest} \cap \mathsf{act}, \mathsf{pm}_k^i = (\mathbf{abort}, \cdot)$ **then return** $\perp$ ▷ Stop on abort
5: **for** $i \in \mathsf{honest} \cap \mathsf{act}$ **do**
6: $\quad (\mathbf{r}_i, \mathbf{e}'_i) \leftarrow \mathcal{D}_{\mathbf{r}}^{\ell} \times \mathcal{D}_{\mathbf{r}}^{k}$
7: $\quad \mathbf{w}_i := \mathbf{A} \cdot \mathbf{r}_i + \mathbf{e}'_i$
8: $\quad H_{\mathsf{cmt}}(\mathsf{vk}, i, \mathbf{w}_i), \cdot := \mathsf{cmt}_i$ ▷ Program random oracle
9: $\quad \mathsf{pm}_2^i := \mathbf{w}_i$
10: $L_{\mathsf{sid}}[\mathsf{sid}] = \{2, (\mathsf{pm}_1^i)_{i \in \mathsf{act}}, (\mathsf{pm}_2^i)_{i \in \mathsf{honest} \cap \mathsf{act}}, \mathsf{act}, \mathsf{msg}\}$
11: **return** $(\mathsf{pm}_2^i)_{i \in \mathsf{honest} \cap \mathsf{act}}$

Fig. 21: The second hybrid of the security proof of the unforgeability of our ramp scheme. Difference with the previous hybrid are highlighted. We omit state updates for conciseness. In case the game programs the random oracle on an existing index, we consider that the adversary wins the game.

The adversary's view differs only if they queried the random oracle on $\mathbf{w}_i$ before the second round started in Hybrid_1. Except with probability at most $Q_{\mathsf{Sign}} \cdot 2^{-d+1}$, $\mathcal{D}_{\mathbf{r}}^k$ has high min-entropy by applying Lemma 4 and for our parameter selection. This then ensures that the programmation of an existing entry happens with probability at most $Q_H \cdot 2^{-H_\infty(\mathcal{D}_{\mathbf{r}}^k)}$ for each party and each signing query. Union bound ensures that the advantage loss is lower than $\frac{N(Q_H+1)Q_{\mathsf{Sign}}}{2^{d-1}}$, where d is the dimension of the ring $\mathcal{R}$.

Hence,

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_1} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_2}\right| \leqslant \frac{N(Q_H+1)Q_{\mathsf{Sign}}}{2^{d-1}}$$

Hybrid_3. In this hybrid, we program responses of the third round in the second round. This is formalized in Fig. 22.

Concretely, the challenger first tries to retrieve challenges $\mathbf{w}_i$ corresponding to each hash $\mathsf{cmt}_i = H(\mathsf{vk}, \mathsf{act}, i, \mathbf{w}_i)$. Then, we have two cases:

1. in case of success, we are able to compute $\mathbf{w} = \lfloor \sum_i \mathbf{w}_i \rceil_{\nu_{\mathbf{w}}}$. We can derive $c = H(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$, and program a partial response $\mathbf{z}_i = c \cdot \mathbf{s}_j^{i_j^{\mathsf{honest}}} + \mathbf{r}_i$.
2. otherwise, we make no change.

In round 3, after the hash commitments are verified, we require responses to have been previously programmed and return them. If they are not, the game aborts and the adversary loses.

There are two scenarios where the view of $\mathcal{A}$ differs:

- in case OSign_3 aborts because a partial response was not programmed. This happens if the adversary found a pre-image for some cmt_i after the partial response programmation in OSign_2, and passed it to OSign_3. It however requires breaking the pre-image resistance of the hash function H_{cmt} which happens with probability at most $T \cdot Q_H / 2^{2\kappa}$.
- in case OSign_3 is called with a different set of commitments $(\mathbf{w}_i)$ than what was used for programmation. This requires breaking the collision resistance of the hash function H_{cmt} and thus happens with probability at most $Q_H^2 / 2^{2\kappa}$.

Overall, we can bound the advantage difference between Hybrid_3 and Hybrid_2:

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_2} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_3}\right| \leqslant (Q_H + T)Q_H / 2^{2\kappa}$$

Hybrid_4. In this hybrid, whenever $\mathcal{A}$ has correctly committed to $(\mathbf{w}_i)_{i \in \mathsf{corrupt} \cap \mathsf{act}}$ and honest partial responses are about to be programmed in $\mathsf{OPerformRound}_2$, we sample a challenge c at random in the challenge space $\mathcal{C}$, and program the random oracle H_c on $\mathbf{w}$ before programming the partial responses. This is formalized in Fig. 23

The adversary's view is identically distributed in Hybrid_4 and Hybrid_3 unless we program H_c on a previously queried $\mathbf{w}$. As in Threshold Raccoon [DKM+24], $\mathbf{w}$ comes from the contribution of a party as a $\mathcal{D}_{q,\ell,k,\mathcal{D}_{\mathbf{r}},\nu_{\mathbf{w}}}^{\mathsf{bd-MLWE}}(\mathbf{A})$ sample which has

Hybrid_3

$L_{\mathsf{cmt}}[\cdot]$ Programmed $[\cdot] = \varnothing$

▷▷▷ Other lines are identical ◁◁◁

$\mathsf{OPerformRound}_2(\mathsf{sid}, (\mathsf{pm}_1^i)_{i \in \mathsf{corrupt} \cap \mathsf{act}})$

Fetch $\{r, (\mathsf{pm}_1^i)_{i \in \mathsf{honest} \cap \mathsf{act}}, \mathsf{act}, \mathsf{msg}\}$ from $L_{\mathsf{sid}}[\mathsf{sid}]$
req$\{\ r = k - 1\ \}$
Parse $(\mathsf{cmt}_i)_{i \in \mathsf{act}} = (\mathsf{pm}_1^i)_{i \in \mathsf{act}}$
if $\exists i \in \mathsf{honest} \cap \mathsf{act}, \mathsf{pm}_k^i = (\mathbf{abort}, \cdot)$ **then return** $\perp$
for $i \in \mathsf{honest} \cap \mathsf{act}$ **do**
  $(\mathbf{r}_i, \mathbf{e}_i') \leftarrow \mathcal{D}_{\mathbf{r}}^{\ell} \times \mathcal{D}_{\mathbf{r}}^{k}$
  $\mathbf{w}_i := \mathbf{A} \cdot \mathbf{r}_i + \mathbf{e}_i'$
  $H_{\mathsf{cmt}}(\mathsf{vk}, i, \mathbf{w}_i), \cdot := \mathsf{cmt}_i$
  $\mathsf{pm}_2^i := \mathbf{w}_i$
if $\forall j \in \mathsf{act} \cap \mathsf{corrupt}, \exists \mathbf{w}_j$ s.t $H_{\mathsf{cmt}}(\mathsf{vk}, j, \mathbf{w}_j) = \mathsf{cmt}_j$ **then**
▷ Recover corrupted contributions
  $\mathbf{w} := \left\lfloor \sum_{j \in \mathsf{act}} \mathbf{w}_j \right\rceil_{\nu_{\mathbf{w}}}$
  $c := H_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$
  $(\nu_{i,u})_{i \in \mathsf{act}, u} := \mathsf{Recover}(\mathsf{pk}, \mathsf{ak}, \mathsf{act})$
  for $i \in \mathsf{honest} \cap \mathsf{act}$ **do**
    $\mathbf{s}_i^{\mathsf{part}} := \sum_{u \in [|S_i|]} S_i[u]$
    $\mathbf{z}_i := c \cdot \mathbf{s}_i^{\mathsf{part}} + \mathbf{r}_i$ ▷ Partial response
  $\mathsf{Programmed}[\mathsf{sid}, \mathbf{w}] := (\mathbf{z}_i)_{i \in \mathsf{honest} \cap \mathsf{act}}$
$L_{\mathsf{sid}}[\mathsf{sid}] = \{2, (\mathsf{pm}_1^i)_{i \in \mathsf{act}}, (\mathsf{pm}_2^i)_{i \in \mathsf{honest} \cap \mathsf{act}}, \mathsf{act}, \mathsf{msg}\}$
return $(\mathsf{pm}_2^i)_{i \in \mathsf{honest} \cap \mathsf{act}}$

$\mathsf{OPerformRound}_3(\mathsf{sid}, (\mathsf{pm}_2^i)_{i \in \mathsf{corrupt} \cap \mathsf{act}})$

req$\{\ \mathsf{rnd} = 2\ \}$
for $(j \in \mathsf{act})$ **do**
  req$\{\ \mathsf{cmt}_j = H_{\mathsf{cmt}}(\mathsf{vk}, j, \mathbf{w}_j)\ \}$
assert$\{\ \mathsf{Programmed}[\mathsf{sid}, \mathbf{w}] \neq \perp\ \}$
return $\mathsf{Programmed}[\mathsf{sid}, \mathbf{w}]$

Fig. 22: The third hybrid of the security proof of the unforgeability of our ramp scheme. Difference with the previous hybrid are highlighted. We omit state updates for conciseness.

high min-entropy – applying Lemma 4 – guaranteeing that programming fails with probability at most $Q_H \cdot 2^{-d+1}$ for each signing query.

Hybrid_4

▷▷▷ Identical to Hybrid_3 ◁◁◁

$\mathsf{OPerformRound}_2(\mathsf{sid}, (\mathsf{pm}_1^i)_{i\in\mathsf{corrupt}\cap\mathsf{act}})$

Fetch $\{r, (\mathsf{pm}_1^i)_{i\in\mathsf{honest}\cap\mathsf{act}}, \mathsf{act}, \mathsf{msg}\}$ from $L_{\mathsf{sid}}[\mathsf{sid}]$
req$\{\ r = k-1\ \}$
Parse $(\mathsf{cmt}_i)_{i\in\mathsf{act}} = (\mathsf{pm}_1^i)_{i\in\mathsf{act}}$
if $\exists i \in \mathsf{honest}\cap\mathsf{act}, \mathsf{pm}_k^i = (\mathbf{abort}, \cdot)$ **then return** $\perp$
for $i \in \mathsf{honest}\cap\mathsf{act}$ **do**
  $(\mathbf{r}_i, \mathbf{e}'_i) \leftarrow \mathcal{D}_{\mathbf{r}}^{\ell} \times \mathcal{D}_{\mathbf{r}}^{k}$
  $\mathbf{w}_i := \mathbf{A}\cdot\mathbf{r}_i + \mathbf{e}'_i$
  $H_{\mathsf{cmt}}(\mathsf{vk}, i, \mathbf{w}_i), \cdot := \mathsf{cmt}_i$
  $\mathsf{pm}_2^i := \mathbf{w}_i$
if $\forall j \in \mathsf{act}\cap\mathsf{corrupt}, \exists \mathbf{w}_j$ s.t $H_{\mathsf{cmt}}(\mathsf{vk}, j, \mathbf{w}_j) = \mathsf{cmt}_j$ **then**
  ▷ Recover corrupted contributions
  $\mathbf{w} := \left\lfloor \sum_{j\in\mathsf{act}} \mathbf{w}_j \right\rceil_{\nu_{\mathbf{w}}}$
  $c \xleftarrow{\$} \mathcal{C}$
  $H_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w}) := c$ ▷ Program random oracle
  $(\nu_{i,u})_{i\in\mathsf{act},u} := \mathsf{Recover}(\mathsf{pk}, \mathsf{ak}, \mathsf{act})$
  for $i \in \mathsf{honest}\cap\mathsf{act}$ **do**
    $\mathbf{s}_i^{\mathsf{part}} := \sum_{u\in[|S_i|]} S_i[u]$
    $\mathbf{z}_i := c\cdot\mathbf{s}_i^{\mathsf{part}} + \mathbf{r}_i$ ▷ Partial response
$\mathsf{Programmed}[\mathsf{sid}, \mathbf{w}] := (\mathbf{z}_i)_{i\in\mathsf{honest}\cap\mathsf{act}}$
$L_{\mathsf{sid}}[\mathsf{sid}] = \{2, (\mathsf{pm}_1^i)_{i\in\mathsf{act}}, (\mathsf{pm}_2^i)_{i\in\mathsf{honest}\cap\mathsf{act}}, \mathsf{act}, \mathsf{msg}\}$
return $(\mathsf{pm}_2^i)_{i\in\mathsf{honest}\cap\mathsf{act}}$

Fig. 23: The fourth hybrid of the security proof of the unforgeability of our ramp scheme. Difference with the previous hybrid are highlighted. We omit state updates for conciseness.

We conclude that the advantage difference is bounded by:

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_3} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_4}\right| \leqslant \frac{Q_H Q_{\mathsf{Sign}}}{2^{d-1}}$$

Hybrid_5. In this hybrid, when a partial response is programmed, we exchange the sampling of $\mathbf{z}_i$ and $\mathbf{w}_i$. We express $\mathbf{w}_i$ as a function of $\mathbf{z}_i = c\cdot\mathbf{s}_i^{\mathsf{part}} + \mathbf{r}_i$ and $\mathbf{y}_i = c\cdot\mathbf{e}_i^{\mathsf{part}} + \mathbf{e}'_i$, i.e. $\mathbf{w}_i = \mathbf{A}\cdot\mathbf{z}_i + \mathbf{y}_i - c\cdot\mathbf{b}_i^{\mathsf{part}}$, where the partial values are reconstructed using the reconstruction coefficients $(\nu_{i,u})_{u\in[|S_u|]}$ given by $\mathsf{Recover}(\mathsf{pk}, \mathsf{ak}, \mathsf{act})$. We directly write $\mathbf{z}_i$ and $\mathbf{y}_i$ as being sampled by the function f of the sDKG. This is formalized in Fig. 24.

Hybrid_5

▷▷▷ Identical to Hybrid_3 ◁◁◁

$\mathsf{OPerformRound}_2(\mathsf{sid}, (\mathsf{pm}_1^i)_{i\in\mathsf{corrupt}\cap\mathsf{act}})$

1: Fetch $\{r, (\mathsf{pm}_1^i)_{i\in\mathsf{honest}\cap\mathsf{act}}, \mathsf{act}, \mathsf{msg}\}$ from $L_{\mathsf{sid}}[\mathsf{sid}]$
2: **req**$\{\ r = k-1\ \}$
3: Parse $(\mathsf{cmt}_i)_{i\in\mathsf{act}} = (\mathsf{pm}_1^i)_{i\in\mathsf{act}}$
4: **if** $\exists i \in \mathsf{honest}\cap\mathsf{act}, \mathsf{pm}_k^i = (\mathbf{abort}, \cdot)$ **then return** $\perp$
5: **if** $\forall j \in \mathsf{act}\cap\mathsf{corrupt}, \exists \mathbf{w}_j$ s.t $H_{\mathsf{cmt}}(\mathsf{vk}, j, \mathbf{w}_j) = \mathsf{cmt}_j$ **then**
6: $c, \mathbf{z}_i, \mathbf{y}_i \leftarrow f(\mathsf{pk}, \mathsf{ak}, (\mathsf{sk}_i)_{i\in\mathsf{honest}}, \mathsf{act})$
▷ Use the sDKG function to compute hints on the secret
▷ $\mathbf{z}_i := c \cdot \sum_{u\in[|S_i|]} \nu_{i,u} \cdot \mathbf{s}_i[u] + \mathbf{r}_i$ and $\mathbf{y}_i := c \cdot \sum_{u\in[|S_i|]} \nu_{i,u} \cdot \mathbf{e}_i[u] + \mathbf{e}'_i$
7: **for** $i \in \mathsf{honest}\cap\mathsf{act}$ **do**
8: $\mathbf{b}_i^{\mathsf{part}} := \sum_{u\in[|S_i|]} \nu_{i,u} \cdot \mathbf{b}_i[u]$
9: $\mathbf{w}_i := \begin{bmatrix}\mathbf{A}\ \mathbf{I}\end{bmatrix} \cdot \begin{bmatrix}\mathbf{z}_i \\ \mathbf{y}_i\end{bmatrix} - c \cdot \mathbf{b}_i^{\mathsf{part}}$
10: $\mathbf{w} := \left\lfloor \sum_{j\in\mathsf{act}} \mathbf{w}_j \right\rceil_{\nu_{\mathbf{w}}}$
11: $H_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w}) := c$ ▷ Program random oracle
12: $\mathsf{Programmed}[\mathsf{sid}, \mathbf{w}] := (\mathbf{z}_i)_{i\in\mathsf{honest}\cap\mathsf{act}}$
13: **else**
14: **for** $i \in \mathsf{honest}\cap\mathsf{act}$ **do**
15: $(\mathbf{r}_i, \mathbf{e}'_i) \leftarrow \mathcal{D}_{\mathbf{r}}^{\ell} \times \mathcal{D}_{\mathbf{r}}^{k}$
16: $\mathbf{w}_i := \mathbf{A} \cdot \mathbf{r}_i + \mathbf{e}'_i$
17: **for** $i \in \mathsf{honest}\cap\mathsf{act}$ **do**
18: $H_{\mathsf{cmt}}(\mathsf{vk}, i, \mathbf{w}_i), \cdot := \mathsf{cmt}_i$
19: $\mathsf{pm}_2^i := \mathbf{w}_i$
20: $L_{\mathsf{sid}}[\mathsf{sid}] = \{2, (\mathsf{pm}_1^i)_{i\in\mathsf{act}}, (\mathsf{pm}_2^i)_{i\in\mathsf{honest}\cap\mathsf{act}}, \mathsf{act}, \mathsf{msg}\}$
21: **return** $(\mathsf{pm}_2^i)_{i\in\mathsf{honest}\cap\mathsf{act}}$

Fig. 24: The fifth hybrid of the security proof of the unforgeability of our ramp scheme. Difference with the previous hybrid are highlighted. We omit state updates for conciseness.

All values are identically distributed, hence,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_5} = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_4}$$

Hybrid_6. In this hybrid, we apply the simulatability property of the sDKG to replace the public key $(\mathsf{seed}, \mathbf{b})$ by an independent sample from the distribution $\chi = \chi_{\mathsf{seed}} \times \chi_{\mathbf{b}}$.

Formally, we can define an adversary $\mathcal{B}_1$ against the game $\mathsf{Game}_{\mathsf{sDKG\text{-}sim}}$ with key distribution χ and hint function f from Section 5.1, simulating the view of $\mathcal{A}$ in Hybrid_5 and Hybrid_6 by replacing the public key pk, identifiable abort key ak,

Hybrid_6

$L_{\mathsf{sid}}, L_H, L_{\mathsf{sig}} := \varnothing$
$(N, T, \mathsf{corrupt}) \leftarrow \mathcal{A}(1^\kappa)$
req$\{\ \mathsf{corrupt} \subseteq [N] \wedge |\mathsf{corrupt}| \leqslant T_{\mathsf{corrupt}}\ \}$
$\mathsf{honest} := [N] \backslash \mathsf{corrupt}$
$(\mathsf{vk}, \mathsf{ak}, \mathsf{st}, \mathsf{status}) \leftarrow \mathsf{SimKeygen}(\mathcal{A}, \mathsf{corrupt})$
▷ Runs the distributed key generation, by interacting with the adversary $\mathcal{A}$
req$\{\ \mathsf{status} = \mathbf{accept}\ \}$
$(\mathsf{msg}, \mathsf{sig}) \leftarrow \mathcal{A}^{H, (\mathsf{OSign}_k(\cdot))_{k \in [\mathsf{rnd}]}}(\mathsf{vk}, \mathsf{ak})$
if $(\mathsf{msg} \in L_{\mathsf{sig}})$ or $\neg\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$ **then**
  return 0
return 1

$\mathsf{OPerformRound}_2(\mathsf{sid}, (\mathsf{pm}_1^i)_{i \in \mathsf{corrupt} \cap \mathsf{act}})$

Fetch $\{r, (\mathsf{pm}_1^i)_{i \in \mathsf{honest} \cap \mathsf{act}}, \mathsf{act}, \mathsf{msg}\}$ from $L_{\mathsf{sid}}[\mathsf{sid}]$
req$\{\ r = k - 1\ \}$
Parse $(\mathsf{cmt}_i)_{i \in \mathsf{act}} = (\mathsf{pm}_1^i)_{i \in \mathsf{act}}$
if $\exists i \in \mathsf{honest} \cap \mathsf{act}, \mathsf{pm}_k^i = (\mathbf{abort}, \cdot)$ **then return** $\perp$
if $\forall j \in \mathsf{act} \cap \mathsf{corrupt}, \exists \mathbf{w}_j$ s.t $H_{\mathsf{cmt}}(\mathsf{vk}, j, \mathbf{w}_j) = \mathsf{cmt}_j$ **then**
  $c, \mathbf{z}_i, \mathbf{y}_i \leftarrow \mathsf{OEval}(\mathsf{pk}, \mathsf{ak}, \mathsf{st}, \mathsf{act})$ ▷ Use the sDKG hint oracle
  ▷▷▷ Rest is identical. ◁◁◁
else
  for $i \in \mathsf{honest} \cap \mathsf{act}$ **do**
    $(\mathbf{r}_i, \mathbf{e}'_i) \leftarrow \mathcal{D}_{\mathbf{r}}^\ell \times \mathcal{D}_{\mathbf{r}}^k$
    $\mathbf{w}_i := \mathbf{A} \cdot \mathbf{r}_i + \mathbf{e}'_i$
  ▷▷▷ Other lines are identical. ◁◁◁

Fig. 25: The sixth hybrid of the security proof of the unforgeability of our ramp scheme. Difference with the previous hybrid are highlighted. We omit state updates for conciseness.

and evaluations of the hint function f by the values provided by the sDKG-sim challenger. Then,

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_5} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_6}\right| \leqslant \mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{sDKG\text{-}sim}}$$

Conclusion. At this stage the verification key of the signature scheme is distributed randomly, and signatures no longer depend on a secret value.

The same reduction used for Raccoon and Threshold Raccoon holds in our case, and we refer to [DKM+24, Lemma C.4] for full details.

Formally, there exists an adversary $\mathcal{B}_2$ against the $\mathsf{SelfTargetMSIS}_{q,k,\ell+1,\mathcal{C},B_{\mathsf{stmsis}}}$ problem with execution time $\mathcal{T}_{\mathcal{B}_2} \approx \mathcal{T}_{\mathcal{A}}$ such that

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_6} \leqslant \mathsf{Adv}_{\mathcal{B}_2}^{\mathsf{SelfTargetMSIS}}$$

We obtain the theorem statement by summing the different advantage differences obtained at each step of the proof. □

C.4 Reduction from Hint-MLWE to MLWE

Theorem 8. *Assume the secrets* $\mathbf{s}$ *and* $\mathbf{s}_j$ *follow Gaussian distributions of standard deviation* $\sigma_{\mathbf{s}}$ *and* σ_{share} *respectively, and perturbations* $\mathbf{r}$ *follow a Gaussian distribution of standard deviation* $\sigma_{\mathbf{r}}$.

The $\mathsf{Hint\text{-}MLWE}_{q,k,\ell,m+Q_{\mathsf{Sign}},\mathcal{D}_{\mathbf{s}},(\mathcal{D}_{\mathbf{r}}^{(i)})_i,\{1\}^m\times\mathcal{C}^{Q_{\mathsf{Sign}}}}$ *problem with* m *hints of the form* $\mathbf{s}+\mathbf{s}_j$, *and* Q_{Sign} *hints of the form* $c\cdot\mathbf{s}+\mathbf{r}$, *where the noises have distributions (i)* $\mathcal{D}_{\mathsf{share}}^{k+\ell}$ *conditioned on the values of* Q_{Sign} *hints of the form* $c\cdot\mathbf{s}_j+\mathbf{r}$ *for the* m *first noises* $\mathbf{s}_j$, *and (ii) the other noises are sampled from* $\mathcal{D}_{\mathbf{r}}^{k+\ell}$ *is hard if* $\mathsf{MLWE}_{q,k,\ell,\sigma_{\mathsf{red}}}$ *is hard, where* σ_{red} *verifies:*

$$\frac{1}{\sigma_{\mathsf{red}}^2}=2\cdot\left(\frac{1}{\sigma_{\mathbf{s}}^2}+m\cdot\frac{1}{\sigma_{\mathsf{share}}^2}+(1+o(1))\cdot\frac{Q_{\mathsf{Sign}}\cdot\omega}{\sigma_{\mathbf{r}}^2}\right)$$

Proof (of Theorem 8). This is a direct application of the MatrixHint-MLWE reduction to MLWE introduced in [ENP24, Theorem 1], which accepts varying noise standard deviations. It is recalled in Theorem 1 for polynomial hints. We can apply Theorem 1 by distinguishing the two types of hints we have:

- we get a term $m\cdot\frac{1}{\sigma_{\mathsf{share}}^2}$ for the m hints of the form $\mathbf{s}+\mathbf{s}_j$, since the spectral norm of the polynomial $c=1$ is 1.
- Del Pino et al proved in [DKM+24, Lemma B.2] that $\sum_{i=1}^{Q_{\mathsf{Sign}}} s_1(c_i^* c_i)$ is upper bounded by $(1+o(1))\cdot Q_{\mathsf{Sign}}\cdot\omega$ w.o.p. when $c_i \xleftarrow{\$} \mathcal{C}$ and $(\kappa\cdot d)^2=o(Q_{\mathsf{Sign}})$. This ensures that the remaining Q_{Sign} hints of the form $c\cdot\mathbf{s}+\mathbf{r}$ introduce a term at most $(1+o(1))\cdot Q_{\mathsf{Sign}}\cdot\omega/\sigma_{\mathbf{r}}^2$.

Applying Theorem 1 concludes the proof. □

C.5 Proof of Identifiable Abort of our TSS scheme

We now prove the Identifiable Abort property (Theorem 7) of our TSS scheme. We noticed a wrongful application of the triangular inequality in the proof of correctness of TRaccoon [DKM+24, Lemma C.1], and we fix their proof using an extended lemma to bound rounding errors. Interestingly, the final bound is slightly better than the one they claimed.

Though it is not made explicit in Fig. 9 (for concision), we consider that when a parsing or an assertion fails, an honest party will output $(\mathbf{abort}, \mathsf{S}_\perp)$ where $\mathsf{S}_\perp$ is the set of users for which the parsing or assertion failed, or $\mathsf{S}_\perp=\varnothing$ if the failure was not related to a message output by another user (e.g. sid).

We first introduce a useful lemma.

Lemma 5 (Lemma 4.4 in [Lyu12], extended). *For any* $\tau>1$ *and any lattice* $\Lambda\subseteq\mathbb{R}^d$,

$$\mathbb{P}\left[\|\mathbf{z}\|>\tau\sigma\sqrt{d};\mathbf{z}\leftarrow D_{\Lambda,\sigma}\right]\leqslant C^d. \tag{8}$$

where $C = \tau \cdot e^{\frac{1}{2}(1-\tau^2)} < 1$. *As a corollary, if* $\left(\tau \geqslant 1 + \frac{\kappa \log 2}{3d}\right)$ *or if* $\left(\tau \geqslant 1 + \sqrt{\frac{4\,\kappa \log 2}{d}}\right)$, *we note that* $C^d \leqslant 2^{-\kappa}$.

We will also rely on an extension of [DKM+24, Lemma 3.2] to bound the error of roundings. This is the core element to fix the proof of [DKM+24, Lemma C.1].

Lemma 6 (extended from Lemma 3.2 [DKM+24]). *Let* $\nu \geqslant 4, q$ *be positive integers such that* $q > 2^\nu$ *and set* $q_\nu = \lfloor q/2^\nu \rceil$. *Moreover, assume* q *and* ν *satisfy* $q_\nu = \lfloor q/2^\nu \rceil$, *that is,* q *can be decomposed as* $q = 2^\nu \cdot q_\nu + q_\perp$, *for* $q_\perp \in [0, 2^{\nu-1} - 1]$. *Then, for any* $\mathbf{x}, \boldsymbol{\delta} \in \mathbb{Z}_q^n$, *we have*

$$\left\| 2^\nu \left(\lfloor \mathbf{x} + \boldsymbol{\delta} \rceil_\nu - \lfloor \mathbf{x} \rceil_\nu \right) - \boldsymbol{\delta} \right\| \leqslant \|1\| \cdot 3 \cdot 2^{\nu-1}$$

Proof. The proof proceeds similarly to [DKM+24, Lemma 3.2]. We will note $\overline{\mathbf{x}}$ the lift of $\mathbf{x}$ to $\mathbb{Z}^n$ with coefficients in $[0, q-1]$.

We start by uniquely writing $\overline{\mathbf{x}} = 2^\nu \cdot \mathbf{x}_\top + \mathbf{x}_\perp \bmod q$, where $(\mathbf{x}_\top, \mathbf{x}_\perp) \in [0, q_\nu]^n \times [-2^{\nu-1}, 2^{\nu-1} - 1]^n$. Similarly, we define $\boldsymbol{\delta}_\top$ and $\boldsymbol{\delta}_\perp$.

Now, remark that $\overline{\mathbf{x}} + \overline{\boldsymbol{\delta}}$ is bounded by $2q$. Hence, we can write,

$$\begin{aligned}
\lfloor \mathbf{x} + \boldsymbol{\delta} \rceil_\nu &= \lfloor 2^\nu \cdot (\mathbf{x}_\top + \boldsymbol{\delta}_\top) + \mathbf{x}_\perp + \boldsymbol{\delta}_\perp - \boldsymbol{\alpha} \cdot q \rceil_\nu \\
&= \lfloor 2^\nu \cdot (\mathbf{x}_\top + \boldsymbol{\delta}_\top - \boldsymbol{\alpha} \cdot q_\nu) + \mathbf{x}_\perp + \boldsymbol{\delta}_\perp - \boldsymbol{\alpha} \cdot q_\perp \rceil_\nu
\end{aligned}$$

where $\boldsymbol{\alpha} \in \{0,1\}^n$ such $\alpha_i = 1$ if $x_i + \delta_i > q - 1$, and $\alpha_i = 0$ otherwise. Put differently, we reduce $\mathbf{x} + \boldsymbol{\delta}$ modulo q.

We can then reexpress the quantity we are interested in

$$\begin{aligned}
2^\nu \cdot \overline{\lfloor \mathbf{x} + \boldsymbol{\delta} \rceil_\nu - \lfloor \mathbf{x} \rceil_\nu} - \boldsymbol{\delta} &= 2^\nu \cdot \overline{(\mathbf{x}_\top + \boldsymbol{\delta}_\top - \boldsymbol{\alpha} \cdot q_\nu + \boldsymbol{\gamma}) - \mathbf{x}_\top} - \boldsymbol{\delta} \\
&= 2^\nu \cdot ((\boldsymbol{\delta}_\top + \boldsymbol{\gamma}) - \boldsymbol{\varepsilon} \cdot q_\nu) - \boldsymbol{\delta} \\
&= 2^\nu \cdot \boldsymbol{\gamma} + \boldsymbol{\varepsilon} \cdot q_\perp - \boldsymbol{\delta}_\perp
\end{aligned}$$

where the last equality follows from $q = 2^\nu \cdot q_\nu + q_\perp$ and $\boldsymbol{\delta} = 2^\nu \cdot \boldsymbol{\delta}_\top + \boldsymbol{\delta}_\perp$, and we define $(\boldsymbol{\gamma}, \boldsymbol{\varepsilon}) \in \{-1, 0, 1\}^n \times \{0, 1\}^n$ as in [DKM+24]:

$$\gamma_i = \begin{cases} -1 & \text{if } x_{\perp,i} + \delta_{\perp,i} - \alpha_i \cdot q_\perp \leqslant -2^{\nu-1} - 1 \\ 0 & \text{if } x_{\perp,i} + \delta_{\perp,i} - \alpha_i \cdot q_\perp \in [-2^{\nu-1}, 2^{\nu-1} - 1] \\ 1 & \text{if } x_{\perp,i} + \delta_{\perp,i} - \alpha_i \cdot q_\perp \geqslant 2^{\nu-1} \end{cases}$$

$$\varepsilon_i = \begin{cases} 0 & \text{if } \delta_{\top,i} + \gamma_i \in \{0, ..., q-1\} \\ 1 & \text{if } (\delta_{\top,i}, \gamma_i) \in \{(q_\nu - 1, 1), (q_\nu, 0), (q_\nu, 1)\} \end{cases}$$

We can then bound,

$$\begin{aligned}
\left\| 2^\nu \cdot \overline{\lfloor \mathbf{x} + \boldsymbol{\delta} \rceil_\nu - \lfloor \mathbf{x} \rceil_\nu} - \boldsymbol{\delta} \bmod q \right\| &= \left\| 2^\nu \cdot \boldsymbol{\gamma} + \boldsymbol{\varepsilon} \cdot q_\perp - \boldsymbol{\delta}_\perp \bmod q \right\| \\
&\leqslant \left\| \boldsymbol{\delta}_\perp \bmod q \right\| + \left\| 2^\nu \cdot \boldsymbol{\gamma} + \boldsymbol{\varepsilon} \cdot q_\perp \right\| \\
&\leqslant 2^{\nu-1} \cdot \|1\| + 2^\nu \cdot \|1\|
\end{aligned}$$

where the last inequality follows from the same analysis on $(\boldsymbol{\gamma}, \boldsymbol{\varepsilon})$ as in [DKM+24], and as coefficients in $\boldsymbol{\delta}_\perp$ are in $[-2^{\nu-1}, 2^{\nu-1} - 1]$.

This concludes this proof. □

We now prove the main theorem for the identifiable abort property of our scheme.

Proof (of Theorem 7). We first remark that in the context of authenticated synchronous broadcast channels (or bulletin boards), all honest parties see the same transcripts and execute the same checks, hence when one honest party outputs an abort message with a corresponding set $\mathsf{S}_\perp$, all other honest parties output the same abort message and set.

We first observe that all honest parties will pass the assert checks on line 1 of $\mathsf{ShareSign}_1$, line 2 of $\mathsf{ShareSign}_2$, and line 2 of $\mathsf{ShareSign}_3$ as they do not depend on information that can be modified by the adversary and by assumption each session uses a new sid.

From this it follows that only the checks on lines 3 and 5 of $\mathsf{ShareSign}_3$ and lines 1, 4, and 5 of Combine can output an abort message for an honest user, and in such a case we have $\mathsf{S}_\perp \neq \emptyset$. Since all honest users output a well-formed message $\mathbf{w}_i$ in $\mathsf{ShareSign}_2$, and a well-formed $\mathbf{z}_i$ in $\mathsf{ShareSign}_3$ (assuming they did not abort in a previous round) the parsing checks of $\mathsf{ShareSign}_3$ line 3 and Combine line 1 can only fail if the messages received from a subset of the corrupted parties is malformed, in which case all honest players will output $(\mathbf{abort}, \mathsf{S}_\perp)$ for some $\mathsf{S}_\perp \subset \mathsf{corrupt}$.

The check $\mathsf{cmt}_j = H_{\mathsf{cmt}}(\mathsf{vk}, j, \mathbf{w}_j)$ of $\mathsf{ShareSign}_3$ will succeed with probability 1 for all $j \in \mathsf{honest} \cap \mathsf{act}$ since H_{cmt} is a deterministic function, so any abort output by honest parties after round 3 will be on a set $\mathsf{S}_\perp \subset \mathsf{corrupt}$.

For each honest party $i \in \mathsf{honest}$, we have that $(\mathbf{z}_i, \mathbf{y}_i) = (c \cdot \mathbf{s}_i^{\mathsf{part}} + \mathbf{r}_i, c \cdot \mathbf{e}_i^{\mathsf{part}} + \mathbf{e}'_i)$, where $\begin{bmatrix}\mathbf{s}_i^{\mathsf{part}} \\ \mathbf{e}_i^{\mathsf{part}}\end{bmatrix} = \sum_{u \in [|S_i|]} \nu_{i,u} \cdot S_i[u]$. This is ensured by the sDKG robustness/identifiable property property which ensures that $\mathbf{b}_i^{\mathsf{part}} = \mathbf{A} \cdot \mathbf{s}_i^{\mathsf{part}} + \mathbf{e}_i^{\mathsf{part}}$. The sDKG additionally guarantees that we have $|\nu_{i,u}| \leqslant \eta_2$, $\|S_i[u]\| \leqslant \eta_1$, and there are at most K secrets $S_i[u]$. We thus deduce that $\left\|\begin{bmatrix}\mathbf{s}_i^{\mathsf{part}} \\ \mathbf{e}_i^{\mathsf{part}}\end{bmatrix}\right\| \leqslant K\eta_1\eta_2$. Besides, $r_i, e'_i \leftarrow \mathcal{D}_{\mathbf{w}}^{\ell} \times \mathcal{D}_{\mathbf{w}}^{k}$, hence by Eq. (8) we have that $\|(\mathbf{z}_i, \mathbf{y}_i)\| \leqslant B_2^{\mathsf{ind}}$ with probability $1 - 2^{-\kappa}$. By a union bound the probability that an honest party triggers an abort is $p \leqslant T \cdot 2^{-\kappa}$.

Finally, we check that the signature output by Combine is valid with probability 1. The hint is such that:

$$
\begin{aligned}
\|2^{\nu_{\mathbf{w}}} \cdot \mathbf{h} - \sum \mathbf{y}_i\| &= \left\| 2^{\nu_{\mathbf{w}}} \cdot \left(\left\lfloor \sum \mathbf{w}_i \right\rceil_{\nu_{\mathbf{w}}} - \left\lfloor \mathbf{A} \sum \mathbf{z}_i - 2^{\nu_{\mathbf{b}}} \cdot c \cdot \mathbf{b} \right\rceil_{\nu_{\mathbf{w}}} \right) - \sum \mathbf{y}_i \right\| \\
&= \left\| 2^{\nu_{\mathbf{w}}} \cdot \left(\left\lfloor \sum \mathbf{w}_i \right\rceil_{\nu_{\mathbf{w}}} - \left\lfloor \sum \left(\mathbf{y}_i + c \cdot \mathbf{b}_j^{m_i} + \mathbf{w}_i \right) - 2^{\nu_{\mathbf{b}}} \cdot c \cdot \mathbf{b} \right\rceil_{\nu_{\mathbf{w}}} \right) - \sum \mathbf{y}_i \right\| \\
&= \left\| 2^{\nu_{\mathbf{w}}} \cdot \left(\left\lfloor \sum \mathbf{w}_i \right\rceil_{\nu_{\mathbf{w}}} - \left\lfloor \sum \mathbf{w}_i + c \left(\sum \mathbf{b}_j^{m_i} - 2^{\nu_{\mathbf{b}}} \cdot \mathbf{b} \right) + \sum \mathbf{y}_i \right\rceil_{\nu_{\mathbf{w}}} \right) - \sum \mathbf{y}_i \right\| \\
&\leqslant 3 \cdot 2^{\nu_{\mathbf{w}}-1} \sqrt{dk} + \omega \left\| \sum \mathbf{b}_j^{m_i} - 2^{\nu_{\mathbf{b}}} \cdot \mathbf{b} \right\| \quad \text{(applying Lemma 6 and the definition of } \mathbf{b}) \\
&\leqslant \sqrt{dk}(3 \cdot 2^{\nu_{\mathbf{w}}-1} + \omega \cdot 2^{\nu_{\mathbf{b}}}) \quad \text{(applying [DKM+24, Lemma 3.2])}
\end{aligned}
$$

where the second equality comes from the fact that all users (even corrupted) passed the partial verification, the robustness of the sDKG ensuring that $\mathbf{b} = \sum_{i \in \mathsf{act}} \mathbf{b}_i^{\mathsf{part}}$, and both inequalities are obtained by applying Lemma 6 and [DKM+24, Lemma 3.2]. From this we have $\|(\mathbf{z}, 2^{\nu_\mathbf{w}} \cdot \mathbf{h})\| \leqslant \sum \|(\mathbf{z}_i, \mathbf{y}_i)\| + \sqrt{dk}(3 \cdot 2^{\nu_\mathbf{w} - 1} + \omega \cdot 2^{\nu_\mathbf{b}}) = T B_2^{\mathsf{ind}} + \sqrt{dk}(3 \cdot 2^{\nu_\mathbf{w} - 1} + \omega \cdot 2^{\nu_\mathbf{b}}) = B_2$.

Finally, $H(\mathsf{vk}, \mathsf{msg}, \lfloor \mathbf{A} \cdot \mathbf{z} - 2^{\nu_\mathbf{b}} \cdot c \cdot \mathbf{b} \rceil_{\nu_\mathbf{w}} + \mathbf{h}) = c$ since this equation is a rewriting of the partial verification equations. □

D Deferred Proofs

D.1 Proof of optimality of Repl secret sharing

Lemma 7. *Let S be a pool of secrets distributed among N parties, where party i knows all the secrets in $S_i \subset S$. Assume that (i) $\forall\, I \subset [N]$ of cardinality T, we have $\bigcup_{i \in I} S_i = S$, and (ii) $\forall\, I \subset [N]$ of cardinality $T-1$, we have $\bigcup_{i \in I} S_i \subsetneq S$. Then $|S| \geqslant \binom{N}{T-1}$.*

Proof. We first observe that no secret can be owned by strictly less than $N-T+1$ parties as this directly entails that the set of parties that do not possess it cannot reconstruct the whole pool S and is of size at least T. We then define a bijection ϕ between each secret and the set of parties that do not possess it (we assume that no two secrets are possessed exactly by the same parties as they would be redundant). Suppose $|S| < \binom{N}{T-1}$, then there exists a set $I \subset [N]$ of cardinality $T-1$ that is not in $\phi(S)$. Additionally since we proved that no secret is owned by strictly less than $N - T + 1$ parties, for any secret $s \in S$ we have $|\phi(s)| \leqslant T - 1$, so the set I is not included in any set that is in the image of ϕ. From this we have that there is no secret in S that is not owned by anyone in the set I which contradicts our hypothesis. □

E More on parameter selection

We derive parameters for our TSS with identifiable aborts for both of the sDKG we introduced. Interestingly, we can use the same parameter sets for both, the number of secrets and size of shares having a limited impact on parameters due to the unusually large randomness used for noise flooding. We obtain sizes competitive with the state-of-the-art TRaccoon [DKM+24]. We provide concrete parameters in Table 2 and explain our selection strategy below.

Unforgeability. The unforgeability of our scheme relies on the SelfTargetMSIS assumption, and on the functional simulatability of the underlying sDKG. The latter reduces to the Hint-MLWE assumption for both of the sDKG we introduced. We study both assumptions separately.

Hint-MLWE. Both of our sDKG are functionally simulatable if the Hint-MLWE problem is hard, for a secret $\mathbf{s} \leftarrow \mathcal{D}_{\sigma_\mathbf{s}}$, and up to Q_{Sign} hints of the form $c \cdot \mathbf{s} + \mathbf{r}$, plus m hints of the form $\mathbf{s} + \mathbf{s}'$ for the ramp sDKG, where $\mathbf{r} \leftarrow \mathcal{D}_{\sigma_\mathbf{r}}, \mathbf{s}' \leftarrow \mathcal{D}_{\sigma_{\mathsf{share}}}$. Following Theorem 8, this problem is at least as hard as $\mathsf{MLWE}_{q,k,\ell,\sigma_{\mathsf{red}}}$, for:

$$\frac{1}{\sigma_{\mathsf{red}}^2} = 2 \cdot \left(\frac{1}{\sigma_\mathbf{s}^2} + (1 + o(1)) \cdot \frac{Q_{\mathsf{Sign}} \cdot \omega}{\sigma_\mathbf{r}^2} + \mathbf{1}_{\mathsf{Ramp}} \cdot \frac{m}{\sigma_{\mathsf{share}}^2} \right) \sim \frac{2 \cdot Q_{\mathsf{Sign}} \cdot \omega}{\sigma_\mathbf{r}^2}, \tag{9}$$

where the $\sim$ asymptotics holds when $\max(\sigma_\mathbf{s}, \sigma_{\mathsf{share}}/\sqrt{m})\sqrt{Q_{\mathsf{Sign}} \cdot \omega} = \omega(\sigma_\mathbf{r})$, and $\mathbf{1}_{\mathsf{Ramp}}$ is 1 only for the Ramp sDKG. We then quantify the hardness of $\mathsf{MLWE}_{q,k,\ell,\sigma_{\mathsf{red}}}$ using the open source Lattice Estimator [APS15].

The additional leakage of the Ramp sDKG is thus negligible when taking σ_{share} sufficiently larger than $\sigma_\mathbf{r}\sqrt{m}/\sqrt{Q_{\mathsf{Sign}} \cdot \omega}$.

Self-Target MSIS. The term $\mathsf{Adv}_{\mathcal{B}_2}^{\mathsf{SelfTargetMSIS}}$ in Theorem 6 corresponds to the hardness of $\mathsf{SelfTargetMSIS}_{q,k,\ell+1,\mathcal{C},B_{\mathsf{stmsis}}}$. We follow the footsteps of Dilithium [LDK+22, §C.3] and Raccoon [dEK+23, §4.3.5] and evaluate its hardness based on the best known attacks. Following [LDK+22, dEK+23], the best known attacks against SelfTargetMSIS are either to (i) break the second preimage resistance of the underlying hash function, or (ii) to find a short vector $\mathbf{z}$ such that:

$$\mathbf{A} \cdot \bar{\mathbf{z}} = \mathbf{w} + \mathbf{b} \cdot c + \alpha, \tag{10}$$

where the term $\alpha = \mathbf{A} \cdot \mathbf{z} - 2^{\nu_\mathbf{w}} \lfloor \mathbf{A} \cdot \mathbf{z} \rceil_{\nu_\mathbf{w}}$ satisfies $\|\alpha\| \leqslant 2^{\nu_\mathbf{w}}\sqrt{k \cdot d}$. Solving (i) is hard as long as $2^\omega \binom{d}{\omega} \geqslant 2^\kappa$. On the other hand, solving (ii) requires solving Eq. (10), which is an instance of the (inhomogeneous) MSIS problem. The best known attack against MSIS is due to Chuengsatiansup et al [CPS+20]. We are secure against this attack as long as:

$$B_2 + 2^{\nu_\mathbf{w}-1}\sqrt{kd} \leqslant \min_{\ell d \leqslant m \leqslant (k+\ell)d} \left(q^{\frac{k\,d}{m}} \cdot \delta_{B_{\mathsf{RSIS}}}^m \right), \tag{11}$$

Since $B_2^{\mathsf{ind}} \leqslant B_2$, it is always easier to forge a signature than a partial signature.

Correctness and Identifiable aborts. Correctness is mainly subsumed to the bounds provided in Theorem 7. Interestingly, due to the noise flooding technique, there exists a large gap between the secret standard deviations $\sigma_\mathbf{s}, \sigma_{\mathsf{share}}$ and $\sigma_\mathbf{r}$ to guarantee unforgeability. Looking at the Hint-MLWE reduction, we can select this gap to be as high as $\sqrt{Q_{\mathsf{Sign}}/m}$ with no security loss. This ensures for both of our sDKG that the number and size of shares has a small influence on the signature size and $B_2 \approx T \cdot \tau\sqrt{d(k+\ell)} + (2^{\nu_\mathbf{w}} + \omega \cdot 2^{\nu_\mathbf{b}}) \cdot \sqrt{dk}$.

In particular, for the Ramp sDKG, one can observe that the last share is $\sqrt{n}$ times larger than the other shares. Again due the aforementioned gap, this has almost no impact on correctness and on the signature size. Concrete values of (T_1, T_2) for the Ramp sDKG can be computed explicitly with Coupon Collector's bounds, as plotted in Fig. 8. Typical values are $(8, 36)$, $(17, 80)$, $(127, 412)$, etc.

F Appendix of Death Star algorithm

We provide the omitted proof of Lemmas 2 and 3.

Proof (of Lemma 2). Any honestly generated X_i will pass (w.o.p.) Alg. 16 using classical concentration results on the geometry of discrete Gaussians.

– Alg. 16: from the tailbound [Lyu12, Lemma 4.4] we have:

$$\Pr\left[\|X_i\| > \sqrt{d}\sigma(1+\tau)\right] \leqslant (1+\tau)^d e^{\frac{d}{2}(1-(1+\tau)^2)} \leqslant e^{-\frac{d\tau^2}{2}} = 2^{-\kappa}.$$

– Alg. 16: by [Lyu12, Lemma 4.3] with probability:

$$\Pr\left[\langle X_i, \frac{X - X_i}{\|X - X_i\|}\rangle > \sigma\sqrt{2\ln 2(\kappa+1)}\right] \leqslant 2e^{-\frac{\sigma^2(\kappa+1)2\ln 2}{2\sigma^2}} = 2^{-\kappa},$$

the condition will pass.

□

Proof (of Lemma 3). Given vectors of the previous setting, assume we now only keep the T vectors which successfully passed the tests of Alg. 16, and call once again their sum X. Then we have by reordering sums:

$$\begin{aligned}\|X\|^2 &= \sum_{i=1}^{T}\langle X_i, X_i\rangle + \sum_{i=1}^{T}\langle X_i, X - X_i\rangle \\ &\leqslant T\cdot I_{\max}^2 + \sum_{i\,|\,\langle X_i, X-X_i\rangle \geqslant 0}\langle X_i, X - X_i\rangle\end{aligned}$$

Remark that when $\langle X_i, X - X_i\rangle > 0$, then $\|X\|^2 \geqslant \|X - X_i\|^2 + \|X_i\|^2 \geqslant \|X - X_i\|^2$, so that we have $\|X - X_i\|^2 \leqslant \|X\|^2 - \|X_i\|^2 \leqslant \|X\|^2$ by condition of Alg. 16 of Alg. 16. Hence, we find:

$$\begin{aligned}\|X\|^2 &\leqslant T\cdot I_{\max}^2 + \sum_{i\,|\,\langle X_i, X-X_i\rangle \geqslant 0}\|X - X_i\|\cdot J_{\max} \\ &\leqslant T\cdot I_{\max}^2 + T\cdot\|X\|\cdot J_{\max}\end{aligned}$$

Write $x = \|X\|^2$, so that if $x \geqslant T\cdot I_{\max}^2$:

$$\begin{aligned}\left(x - T\cdot I_{\max}^2\right)^2 &\leqslant T^2\cdot x\cdot J_{\max}^2 \\ x^2 - \left(2\cdot T\cdot I_{\max}^2 + T^2 J_{\max}^2\right)\cdot x + T^2\cdot I_{\max}^4 &\leqslant 0\end{aligned}$$

Writing $B = 2\cdot T\cdot I_{\max}^2 + T^2 J_{\max}^2$ and $C = T^2\cdot I_{\max}^4$, we can rewrite it as the quadratic inequation of shape $x^2 - Bx + C \leqslant 0$, whose discriminant is:

$$\begin{aligned}\Delta = B^2 - 4C &= 4T^2 I_{\max}^4 + T^4 J_{\max}^4 + 4T^3 I_{\max}^2 J_{\max}^2 - 4T^2 I_{\max}^4 \\ &= T^2 J_{\max}^2\left(T^2 J_{\max}^2 + 4T I_{\max}^2\right)\end{aligned}$$

We observe that $\Delta > 0$, which in turn asserts that this inequation entails $x \in \frac{1}{2}\left[B - \sqrt{\Delta}, B + \sqrt{\Delta}\right]$. In particular, the upper bound writes:

$$\begin{aligned}
\|X\|^2 &\leqslant \frac{1}{2}(2 \cdot T \cdot I_{\max}^2 + T^2 J_{\max}^2) + \frac{1}{2} T J_{\max} \sqrt{T^2 J_{\max}^2 + 4 T I_{\max}^2} \\
&\leqslant \frac{T^2 J_{\max}^2}{2} \left(1 + \sqrt{1 + \frac{4 I_{\max}^2}{J_{\max}^2 T}}\right) + I_{\max}^2 T \\
&\leqslant \sigma^2 T \left(T \kappa \ln 2 \left(1 + \sqrt{1 + \frac{2d(1+\tau)^2}{\kappa T \ln 2}}\right) + d(1+\tau)^2\right)
\end{aligned}$$

□