BMuSig-L: Blind Multi-Signature from Lattice

Anirban Chakraborty[1], Mriganka Dey[2,3] Nikita Dey[2,4], Avijit Dutta[2,4], and Subhabrata Samajder[2]

[1] University of Warsaw, Poland
[2] Institute for Advancing Intelligence, TCG CREST, Kolkata
[3] Ramakrishna Mission Vivekananda Educational Research Institute, Belur
[4] Academy of Scientific and Innovative Research (AcSIR), Ghaziabad, India
anirban2chakrabarti@gmail.com, mriganka.dey.86@tcgcrest.org, nikita.dey.127@tcgcrest.org, avijit.dutta@tcgcrest.org, subhabrata.samajder@tcgcrest.org

Abstract. Blind signatures and multi-signatures are two fundamental cryptographic primitives for privacy-preserving decentralized protocols. A blind signature enables a user to obtain a signature on a message from a signer while keeping the message hidden from the signer. Moreover, the signer cannot subsequently link the resulting signature to the corresponding signing session. In contrast, a multi-signature scheme allows a group of signers to jointly generate a single signature on a message. Blind multi-signatures, a primitive recently introduced by Karantaidou et al. (AsiaCCS 2024), combine the advantages of both techniques. They enable a user to obtain a joint signature from a group of signers while ensuring that the message remains hidden from every signer and that colluding signers cannot later link the final signature to the corresponding signing session. Recently, Bisht et al. proposed BMuSig2 (AsiaCCS 2026), a Schnorr-compatible blind multi-signature scheme whose security is based on the hardness of the discrete logarithm problem. In this paper, we present BMuSig-L, a lattice-based, two-round, concurrently secure blind multi-signature scheme. Our construction builds upon the lattice-based multi-signature scheme MuSig-L (CRYPTO 2022), the concurrently secure blind signature scheme of Fuchsbauer and Wolf (CRYPTO 2024), which employs non-interactive zero-knowledge arguments and public-key encryption, and a knowledge-sound non-interactive argument system. We prove that BMuSig-L is one-more unforgeable under the assumptions of the unforgeability of MuSig-L, the computational soundness of the underlying non-interactive zero-knowledge proof system, and the knowledge soundness of the employed non-interactive argument of knowledge. Finally, we present a prototype implementation of BMuSig-L and evaluate its performance in terms of signature size and verification time.

Keywords: Blind Signature, Multi-Signature, Blind Multi-Signature, MuSig-L, LaBRADOR.

1 Introduction

Blind Signature. Blind signatures, introduced by Chaum in 1983 [17], are a fundamental cryptographic primitive that enables a user to obtain a signature on a message from a signer while keeping the message hidden from the signer. At the same time, they ensure that the signer cannot subsequently associate the resulting signature with the corresponding signing session. Blind signatures have found numerous applications, ranging from electronic voting [18] and anonymous electronic cash [16] to a wide variety of modern privacy-preserving systems, including anonymous credential frameworks such as U-Prove [48] and Idemix [44], as well as anonymous token systems such as Privacy Pass [35]. More recently, blind signatures have also been integrated into blockchain-based systems to enable the anonymous signing of smart contracts and support privacy-preserving transactions [33,34].

Multi-Signature. Multi-signature [5], on the other hand, is a distributed signing protocol that enables a group of signers to jointly produce a signature on a message M. Rather than simply concatenating individual signatures on the same message, a multi-signature scheme combines them into a single compact signature, thereby significantly reducing both storage and communication overhead. Although multi-signatures share some similarities with group signatures [19], they differ fundamentally in that group signature schemes rely on a dedicated group manager to control user enrollment and revocation, whereas multi-signature schemes do not require such a centralized authority. Over the years, multi-signatures have found numerous applications in distributed systems, particularly in blockchain protocols, where consensus among multiple participants is required and the resulting signature serves as proof of collective approval. Furthermore, in applications such as digital wallets and cryptocurrencies, multi-signatures act as a drop-in replacement for conventional digital signatures, ensuring that a transaction is authorized by a designated group of entities or validators before it is recorded on the blockchain.

Blind Multi-Signatures. Blind Multi-Signature (BMS) combines the functionalities of both blind signatures and multi-signatures, allowing a user to obtain a signature on a message from a group of signers while ensuring that the message remains hidden from all signers and that none of them can later link the signature to the corresponding signing session. In other words, BMSprovides message anonymity together with collective authentication of the message. The notion of blind multi-signature was introduced in a recent work by Karantaidou et al. [36] for the decentralized issuance of anonymized tokens. They proposed two BMSconstructions: one based on the BLS signature scheme, called BM_BLS [36], and another based on the hardness of the discrete logarithm assumption in groups without pairings, called BM_SB [36].

In [10], Bisht et al. proposed BMuSig2 [10], a Schnorr-compatible blind multi-signature scheme that non-trivially integrates a concurrently secure Schnorr-compatible blind signature scheme [28] with the recently proposed multi-signature scheme MuSig2 [45]. Despite requiring additional cryptographic

mechanisms to achieve blindness and concurrent security, BMuSig2 produces a single compact multi-signature that can be verified using an aggregated public key. Consequently, the verification cost remains independent of the number of participating signers, making the scheme highly scalable.

A secure blind multi-signature scheme satisfies two fundamental security notions: *one-more unforgeability* and *blindness*. One-more unforgeability guarantees that an adversary interacting with the signers for l signing sessions cannot generate more than l valid signatures. On the other hand, blindness ensures that even colluding signers cannot link a final signature to any of their previous interactions with the user. Functionally, a blind multi-signature scheme is similar to an (n, n)-threshold blind signature scheme [20]. However, in a threshold setting, the signing key is distributed among the signers by a trusted dealer using a secret-sharing mechanism. In contrast, in a blind multi-signature scheme, each signer can independently generate its own public-private key pair, making it more suitable for decentralized applications.

A blind multi-signature scheme can be applied to any application of blind signatures that naturally extends to a multi-authority setting, including e-voting, e-auctions, and related privacy-preserving applications. In the context of e-voting, where the election authority may consist of a group of entities rather than a single authority, it may be desirable for these entities to jointly verify the identity of each voter. Using BMS, multiple authorities can collectively authenticate a voter's eligibility without gaining access to the voter's choice. Subsequently, the authorities can jointly sign the blinded ballot using BMS, and the voter can unblind the resulting signature and submit the authenticated ballot. During the counting phase, only those ballots for which the BMSverification succeeds are considered valid and counted. Beyond e-voting, BMScan serve as a useful cryptographic primitive in several decentralized applications, including: (i) *Decentralized Autonomous Organizations* (DAOs), where multiple governors or council members jointly issue anonymous voting tokens to eligible participants; (ii) *Decentralized Identity Systems* (DIDs), where multiple authorities issue anonymous credentials certifying a user's identity or attributes; and (iii) *Federated e-cash systems*, where multiple financial institutions jointly issue anonymous digital tokens representing different denominations, thereby preventing unilateral control and ensuring distributed governance of the monetary system.

The security of the existing BMSschemes, namely BM_SB [36] and BMuSig2 [10], relies entirely on the hardness of the discrete logarithm problem in prime-order groups. Consequently, these schemes are vulnerable to quantum attacks due to Shor's algorithm [52]. Since anonymous tokens are expected to serve as long-lived infrastructure components, it is essential that they are designed to be quantum-resistant. Moreover, given the increasing maturity of lattice-based signature schemes, such as Dilithium [25] and Falcon [50], which have already been standardized by NIST [43], along with the recent development of lattice-based multi-signature schemes such as MuSig-L [14], designing a lattice-based BMSscheme is a natural and important research direction. However, any lattice-based BMS construction must overcome the following three difficulties simultaneously.

1. Blinding a lattice signature without leaking secret-key information through rejection sampling or through the shape of the short vectors involved.
2. Aggregating N independently keyed partial signatures into an output whose size does not grow linearly with N, where N denotes the group size.
3. Proving one more unforgeability against a concurrent adversary that can initiate several concurrent signing sessions to mount a so-called ROS attack [7], under a standard lattice assumption such as MLWE or MSIS, without relying on heuristic security arguments.

1.1 Our Contribution

In this paper, we propose BMuSig-L, the first lattice-based blind multi-signature construction. BMuSig-L is a two-round, four-move protocol between a user and a group of signers. It is obtained through a novel combination of the lattice-based multi-signature scheme MuSig-L and the recently proposed concurrently secure blind signature scheme of Fuchsbauer et al. [28], which leverages non-interactive zero-knowledge arguments and a public-key encryption scheme. The final signature generated by our proposed scheme is a proof π that establishes a simple linear lattice relation through a non-interactive argument of knowledge. Specifically, we employ the LaBRADOR [9] proof system for this purpose. As a result, we require two non-interactive argument of knowledge systems: (i) one with the zero knowledge property that is required by the underlying blind signature scheme and (ii) the one without zero knowledge property that is used in the generation of the final signature of the protocol. We prove that BMuSig-L achieves one-more unforgeability under the standard lattice assumptions, namely Module Learning With Errors (MLWE) and Module Short Integer Solution (MSIS), together with the soundness of the underlying non-interactive zero knoweledge argument system and the knowledge soundess of the underlying non-interactive argument of knowledge system. Furthermore, the blindness property of our construction is reduced to the zero-knowledge property of the underlying non-interactive zero-knowledge argument system and the IND-CPA security of the underlying public-key encryption scheme.

Construction	BM_SB	BMuSig2	BMRaccoon	BMuSig-L
Assumption	DL	DL	AOM-MISIS + MSIS	MLWE + MSIS
# rounds	3	2	2	2
Key aggregation	No	Yes	No	Yes
Concurrent security	Yes	Yes	Yes	Yes
Verification cost	$O(N)$	$O(1)$	$O(N)$	$O(1)$
Signature Size	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Security model	KOSK	PPK	KOSK	PPK

Table 1: Comparison with Existing Blind Multi-Signature Schemes.

As shown in Table 1, BMuSig-L is the only round-optimal, concurrently secure lattice-based blind multi-signature scheme that supports key aggregation in the *plain public key* (PPK) model, while providing constant-time verification and constant-size signatures.

Concurrent Security. The concurrent security of our construction is inherited from the security guarantees of the underlying multi-signature scheme MuSig-L [14] and the blind signature scheme proposed by Fuchsbauer et al. [28]. In MuSig-L, the authors achieve security against concurrent signing sessions by requiring each signer to sample multiple nonces and compute the final aggregated nonce as a linear combination of these nonces. The coefficient used for this linear combination is publicly computable from the hashes of the individual nonces generated by all signers. This mechanism prevents the ROS attack [7] against MuSig-L. In particular, if an adversary could control the aggregated nonce by selecting its nonce after observing the nonce of an honest signer, the resulting nonce coefficient would also change accordingly. Consequently, the effective contribution of the honest signer's nonce would vary across different signing sessions. This variability is crucial for launching a ROS attack [7], and therefore the above mechanism effectively prevents such an attack.

To transform MuSig-L into a concurrently secure BMSscheme, we employ the non-interactive zero-knowledge arguments (ZK-NARK) introduced by Fuchsbauer and Wolf [28] in the context of concurrently secure blind signatures. The main idea is to prevent the user from adaptively choosing the blinding randomness after observing the signers' nonces. In the first round, the user commits to the message and the secret randomness, including the blinding factors, before receiving the nonces from the signers. In the second round, the user provides a ZK-NARK proof certifying that the challenge sent to the signers is correctly generated from the previously committed values. Therefore, the challenge is bound to the user's committed message and randomness and cannot be adaptively modified based on the signers' nonces. This prevents an adversary from exploiting nonce-dependent randomness to mount a ROS attack, thereby ensuring concurrent security in the blind signature setting.

Implementation and Evaluation. We report a reference implementation of BMuSig-L instantiated at ring degree $n = 256$ and module ranks $\kappa = m = 6$, chosen so the underlying MSIS instance reaches a smaller root Hermite factor ($\delta = 1.00330$) than Dilithium2's ($\delta = 1.00425$) — a relative statement, not a bit-security claim — yielding a packed signature of 15.0 KiB and verification in 5.6 ms. Signing costs 21.7 ms per signer in round 1 — the dominant phase, and 91% of it is the ν products $\bar{\mathbf{A}}\vec{\mathbf{r}}_{i,j}$ rather than the $\nu(m+\kappa)n$ Gaussian coefficients, which account for only 2.0 ms; with $q \equiv 5 \pmod 8$ ruling out a negacyclic NTT, schoolbook multiplication is the binding constraint — against 7.3 ms in round 2 and 4.2 ms for key generation. The key observation is that aggregation makes both size and verification effectively independent of the number of signers N (a 5% growth in signature size and flat verification from $N = 1$ to 16), while the expected cost of one *successful* signature still rises 17×, entirely because session completion decays as $K_N = K^N$, from 95.5% at $N = 1$ to $\approx 39\%$ at $N = 16$.

The bottleneck in the multi-signer setting is therefore the abort rate rather than bandwidth or verification; we note that the reported signature omits the zero-knowledge argument, which remains a placeholder, and instantiating it at the cost published for comparable lattice proofs would put the total nearer 43 kB.

Concurrent Work. Recently, Varshini et al. [54] have proposed a two round concurrently secure lattice based blind multi signature BMRaccoon. Their scheme is based on Raccoon [23] based multisignature. However, the proposed scheme is proven secure under Algebraic One-more MISIS assumption [55] and the MSIS assumption in the random oracle model. Moreover, the proposed scheme works in the KOSK model and does not support key aggregation. Instead the verification algorithm takes the full list of signer keys and hence the verification key size becomes accountable. It has been shown that BMRaccoon generates 68.56 kB signature size with verification key size 16.13 kB when the number of signers is 64. Although, authors have mentioned that reducing the one-more unforgeability of BMRaccoon to the strong unforgeability notion of the underlying multisignature scheme MRaccoon does not require rewinding technique, but proving the strong unforgeability of MRaccoon from the hardness of Algebraic one-more ISIS assumption requires rewinding and hence incurs a quadratic secrurity loss. Compared to their scheme, one-more unforgeability of our scheme is proven secure under the natural standard hardness assumption MLWE and MSIS. Our scheme works in the plain-public key model and supports key aggregation and hence the verification key size becomes constant (i.e., independent of the number of signers). Moreover, the signature size of our scheme is much less (15 kiB) compared to the BMRaccoon scheme.

2 Preliminaries

In this section, we introduce the notation, algorithms, and security notions of blind signatures and multi-signatures that will be used throughout the paper.

2.1 Notation

Given positive integers a and b with $a < b$, we denote by $[a, b]$ the integer interval $\{a, a+1, \ldots, b\}$. We use $[b]$ to represent $[1, b]$. $\mathbf{A}$ denotes a matrix and $\mathbb{I}_{\mathsf{ord}}$ is the identity matrix of order ord. For a set S, $s \xleftarrow{\$} S$ means s chosen uniformly at random from S. If D is a probability distribution, we will write $s \leftarrow D$ to denote a sample s drawn from D. We write $s \leftarrow D^{(r)}$ to denote a sample s drawn from D when the random coin r is externally provided. If A is a probabilistic (resp. deterministic) algorithm, then $s \leftarrow \mathsf{A}$ (resp. $s := \mathsf{A}$) denotes setting s equal to an output of A. If A is a probabilitstic algorithm that runs in time polynomial of the input size, then we call A to be a probabilistic polynomial time, in short, PPT algorithm. For a set S, $\langle S \rangle$ is the unique representation of the set S, for example, the lexicographically sorted list of its elements.

$\mathbb{Q}(\zeta_n)$ is the pth cyclotomic number field, where ζ_n is an pth root of unity and $n = \varphi(p)$, the *Euler-totient function*. We will denote the ring of integers by $\mathcal{R}$.

Let $\Phi_p(X)$ denote the pth cyclotomic polynomial. Then, $\Phi_p(\zeta_n) = 0$, $\mathbb{Q}(\zeta_n) \cong \mathbb{Q}[X]/\langle\Phi_p(X)\rangle$ and $\mathbb{Z}(\zeta_n) \cong \mathbb{Z}[x]/\langle\Phi_p(X)\rangle$, where $\mathbb{Z}[X]$ and $\mathbb{Q}[X]$ respectively denote the ring of polynomials over $\mathbb{Z}$ and $\mathbb{Q}$, and $\langle\Phi_p(X)\rangle$ denotes the ideal generated by $\Phi_p(X)$. We denote $L_{\mathbb{R}} := \mathbb{R}[X]/\langle\Phi_p(X)\rangle$.

Let $\mathcal{R} = \mathbb{Z}[X]/\langle\Phi_p(X)\rangle$ be the underlying polynomial ring, where $\Phi_p(X) = X^n + 1$, with $p = 2n$. For a modulus q (for this work we have taken $q \equiv 5 \mod 8$), we define $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. The infinity norm and L_2 norm of a polynomial $f = \sum_{i=0}^{n-1} f_i X^i \in \mathcal{R}$ are respectively

$$\|f\|_\infty := \max_{0 \le i < n} |f_i|, \qquad \|f\|_2 := \sqrt{\sum_{i=0}^{n-1} |f_i|^2}.$$

For a vector $\vec{\mathbf{v}} = (\sum_{i=0}^{n-1} v_{i,1}X^i, \sum_{i=0}^{n-1} v_{i,2}X^i, \ldots, \sum_{i=0}^{n-1} v_{i,k}X^i)^T \in \mathcal{R}^k$, L_2 norm is defined by $\|\vec{\mathbf{v}}\|_2 := \sqrt{\sum_{j=1}^{k}\sum_{i=0}^{n-1} |v_{i,j}|^2}$. The notations and the terminologies that will be followed throughout the paper are given in Table 2.

Parameter	Description
N	Number of signers.
τ	Number of parallel repetitions.
$n = \text{poly}(\lambda)$	A power of 2 defining the degree of $f(X)$.
$f(X) = X^n + 1$	The $2n$-th cyclotomic polynomial.
$q \equiv 5 \pmod 8$	Prime modulus.
$w = \lceil \log_2 q \rceil$	Logarithm of the modulus.
$\mathcal{R} = \mathbb{Z}[X]/(f(X))$	Cyclotomic ring.
$\mathcal{R}_q = \mathbb{Z}_q[X]/(f(X))$	Ring modulo q.
κ	Number of rows of matrix $\bar{\mathbf{A}}$.
m	Number of columns of matrix $\mathbf{A}$.
$B = \sigma_1\sqrt{n(m+\kappa)}$	Maximum ℓ_2-norm of a signature share $\vec{\mathbf{s}}_i \in \mathcal{R}^{m+\kappa}$.
$B_N = \sqrt{N}\,B$	Maximum ℓ_2-norm of the combined signature $\vec{\mathbf{s}} \in \mathcal{R}^{m+\kappa}$.
κ_1	Maximum ℓ_1-norm of the challenge vector $\mathbf{c}$.
$\mathcal{C} = \{\mathbf{c} \in \mathcal{R} : \|\mathbf{c}\|_\infty = 1 \wedge \|\mathbf{c}\|_1 = \kappa_1\}$	Challenge space with $\lvert\mathcal{C}\rvert = \binom{n}{\kappa_1} 2^{\kappa_1}$.
η	Maximum ℓ_∞-norm of the secrets.
$S_\eta = \{s \in \mathcal{R} : \|s\|_\infty \le \eta\}$	Set of small secrets.
$T = \kappa_1^2\eta\sqrt{n(m+\kappa)}$	Chosen to satisfy the hypotheses of Rejection Sampling.
$\sigma_1 = \sigma_b\tilde{\sigma}\sqrt{n(2\kappa w+1)(m+\kappa)}$	Standard deviation of the Gaussian distribution.
$\tilde{\sigma} = \frac{2^9}{\pi^{3/2}} 2^{2/n\kappa} q^{\kappa/m+\kappa} n^2\sqrt{(\kappa w+1)(2+n+\log((m+\kappa)n))}$	Standard deviation of the Gaussian distribution.
$\sigma_b = \frac{2^{5/2}}{\sqrt{\pi}} 2^{2/n\kappa} n^{3/2}\sqrt{\kappa w+1}$	Standard deviation of the Gaussian distribution.
$\widehat{\Sigma} = \text{diag}(\sigma_1, \ldots, \sigma_1)$	Covariance matrix of the target Gaussian distribution.
$\alpha = \frac{\sigma_1 - 1}{T}$	Parameter defining M.
$t = \sqrt{\frac{n}{(\pi-1)\log_2 e}}$	Parameter defining M.
$K = e^{t/\alpha + (1/(2\alpha^2))}$	Expected number of restarts until a single party can proceed.
$K_N = K^N$	Expected number of restarts until all N parties proceed simultaneously.

Table 2: Notation and Parameters.

Discrete Gaussian Distribution. Let $\Sigma \in L_{\mathbb{R}}^{d\times d}$ be a symmetric positive definite matrix, and let $\sqrt{\Sigma} \in L_{\mathbb{R}}^{d\times d}$ be a nonsingular matrix such that $\Sigma = \sqrt{\Sigma}\sqrt{\Sigma}^\star$. For a lattice $\Lambda \subseteq \mathcal{R}^d$, center $\vec{\mathbf{c}} \in \mathcal{R}^d$, and Gaussian parameter Σ, the

discrete Gaussian distribution $\mathcal{D}^d_{\sqrt{\Sigma},\vec{\mathbf{c}},\Lambda}$ over $\vec{\mathbf{z}} \in \Lambda$ is defined by,

$$\mathcal{D}^d_{\sqrt{\Sigma},\vec{\mathbf{c}},\Lambda}(\vec{\mathbf{z}}) := \frac{\rho_{\sqrt{\Sigma},\vec{\mathbf{c}}}(\vec{\mathbf{z}})}{\sum_{\vec{\mathbf{x}}\in\Lambda}\rho_{\sqrt{\Sigma},\vec{\mathbf{c}}}(\vec{\mathbf{x}})}, \rho_{\sqrt{\Sigma},\vec{\mathbf{c}}}(\vec{\mathbf{z}}) := \exp\left(-\pi \left\|\sqrt{\Sigma}^{-1}(\vec{\mathbf{z}}-\vec{\mathbf{c}})\right\|_2^2\right).$$

When $\Sigma = \sigma^2\mathbb{I}_d$ for some $\sigma > 0$, we write $D_{\sigma,\vec{\mathbf{c}},\Lambda}$ for the corresponding spherical discrete Gaussian distribution. We write $\mathcal{D}^d_{\sqrt{\Sigma},\vec{\mathbf{c}}}$ to denote discrete Gaussian distribution over $\mathcal{R}^d$ and omit the center $\vec{\mathbf{c}}$ when $\vec{\mathbf{c}} = \vec{\mathbf{0}}$.

Cryptographic Hard Problems. We will follow the definitions of *module short integer solution* (MSIS) and *module learning with error* (MLWE) problems from [14].

Definition 1. *Let $\lambda \in \mathbb{N}$ be the security parameter. For a prime $q = q(\lambda)$, a bound $\beta = \beta(\lambda) > 0$, and positive integers $k = k(\lambda)$ and $\ell = \ell(\lambda)$, the $\mathsf{MSIS}_{q,\kappa,m,\beta}$ assumption holds if, for every PPT algorithm A, the following advantage is negligible in λ:*

$$\mathsf{Adv}^{\mathsf{MSIS}}_{q,\kappa,m,\beta,\mathsf{A}}(\lambda) := \Pr\left[\begin{array}{c} 0 < \|\vec{\mathbf{x}}\|_2 \le \beta \\ \wedge [\mathbf{A} \mid \mathbb{I}_\kappa]\cdot\vec{\mathbf{x}} = \vec{\mathbf{0}} \mod q \end{array} \middle| \begin{array}{c} \mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{\kappa\times m}, \\ \vec{\mathbf{x}} \leftarrow \mathsf{A}(\mathbf{A}) \end{array}\right].$$

Definition 2. *Let $\lambda \in \mathbb{N}$ be the security parameter. For a prime $q = q(\lambda)$, and positive integers $\kappa = \kappa(\lambda)$, $m = m(\lambda)$, $\eta = \eta(\lambda)$ the $\mathsf{MLWE}_{q,\kappa,m,\eta}$ assumption holds if, for every PPT algorithm D, the following advantage is negligible in λ:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{MLWE}}_{q,\kappa,m,\eta,\mathsf{D}}(\lambda) := \Bigg| &\Pr\left[b = 1 \middle| \begin{array}{c} \mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{\kappa\times m}, \vec{\mathbf{s}} \xleftarrow{\$} S_\eta^{m+\kappa}, \\ \mathbf{t} := [\mathbf{A} \mid \mathbb{I}_\kappa]\vec{\mathbf{s}} \mod q, \\ b \leftarrow \mathsf{D}(\mathbf{A},\mathbf{t}) \end{array}\right] \\ &- \Pr\left[b = 1 \middle| \begin{array}{c} \mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{\kappa\times m}, \vec{\mathbf{s}} \xleftarrow{\$} S_\eta^{m+\kappa}, \\ \mathbf{t} \xleftarrow{\$} R_q^\kappa, b \leftarrow \mathsf{D}(\mathbf{A},\mathbf{t}) \end{array}\right]\Bigg|.\end{aligned}$$

As defined in Table 2, S_η denotes the set of small polynomials parameterized by $\eta \ge 0$. Since our construction is defined over a module of dimension $m+\kappa$, every secret signing key belongs to the set $S_\eta^{m+\kappa}$. We define the challenge space as

$$\mathcal{C} = \{c \in \mathcal{R} : \|c\|_\infty = 1,\ \|c\|_1 = \kappa_1\},$$

where κ_1 denotes the challenge weight. Furthermore, we also define

$$\overline{\mathcal{C}} = \{c - c' : c, c' \in \mathcal{C},\ c \ne c'\}$$

such that every element of $\overline{\mathcal{C}}$ is invertible in the ring $\mathcal{R}_q$.

Lemma 1 (Rejection Sampling). *[14, Lemma 3.1] Let $\Lambda \subseteq \mathcal{R}^{m+\kappa}$ be a lattice, with smoothing parameter $\eta_\varepsilon(\Lambda^\perp)$. Let $\alpha, T, \nu > 0$ and $\varepsilon \le \frac{1}{2}$. Define $\sigma_1, \sigma_b, \tilde{\sigma} > 0$ such that $\tilde{\sigma} > \eta_\varepsilon(\Lambda^\perp)$, $\sigma_b > \eta_\varepsilon(\mathcal{R})$, and $\sigma_1 \ge$*

$\max\left\{\alpha T, \tilde{\sigma}\sigma_b\sqrt{n\nu(m+\kappa)}\right\}$. *We consider the following set* $V \subseteq \mathcal{R}^{1\times\nu} \times \mathcal{R}^{\kappa} \times \mathcal{R}^{m+\kappa}$. *Let* $h : V \to [0,1]$ *be the composition of three probability distributions* $\mathcal{D}_b \times \mathcal{D}_u \times \mathcal{D}_v$, *where* $\mathcal{D}_b$ *returns*

$$\left(\mathbf{1}, \mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_\nu\right) \text{ such that } \mathbf{b}_j \leftarrow \mathcal{D}_{\sigma_b} \forall j \in [\nu],$$

$\mathcal{D}_u$ *returns a vector* $\vec{\mathbf{u}} \in \mathcal{R}^{\kappa}$, *and* $\mathcal{D}_v$ *returns a vector* $\vec{\mathbf{v}} \in \mathcal{R}^{m+\kappa}$ *such that* $\|\vec{\mathbf{v}}\|_2 \leq T$. *Then, we define*

$$\Sigma = \left(\sigma_1^2 + \tilde{\sigma}^2 \sum_{j=2}^{\mu} \mathbf{b}_j^{\star}\mathbf{b}_j\right) \mathbb{I}_{m+\kappa}, \widehat{\Sigma} = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_1^2).$$

Then, for any $t > 0$, *let* $G = \exp(\pi/\alpha^2 + \pi t/\alpha)$ *and* $\epsilon = 2(1+\varepsilon)/(1-\varepsilon)\exp(-t^2(\pi-1))$ *the distribution of the following algorithm*

RejSamp

1. $(\mathbf{1}, \mathbf{b}_2, \ldots, \mathbf{b}_\nu, \vec{\mathbf{u}}, \vec{\mathbf{v}}) \leftarrow h$.
2. $\vec{\mathbf{z}} \leftarrow \mathcal{D}^{m+\kappa}_{\sqrt{\Sigma}, \vec{\mathbf{v}}, \Lambda^{\perp}_{\vec{\mathbf{u}}}}$.
3. *With probability* $1 - \min\left(1, \frac{\mathcal{D}^{m+\kappa}_{\sqrt{\widehat{\Sigma}}}(\vec{\mathbf{z}})}{K \cdot \mathcal{D}^{m+\kappa}_{\sqrt{\Sigma}, \vec{\mathbf{v}}}(\vec{\mathbf{z}})}\right)$, *set* $\vec{\mathbf{z}} := \perp$.
4. *Output* $(\vec{\mathbf{z}}, \mathbf{1}, \mathbf{b}_2, \ldots, \mathbf{b}_\nu, \vec{\mathbf{u}}, \vec{\mathbf{v}})$.

is within statistical distance $\frac{\epsilon'}{2K} + \frac{2\varepsilon}{K}$ *from the following distribution,*

SimRS

1. $(\mathbf{1}, \mathbf{b}_2, \ldots, \mathbf{b}_\nu, \vec{\mathbf{u}}, \vec{\mathbf{v}}) \leftarrow h$.
2. $\vec{\mathbf{z}} \leftarrow \mathcal{D}^{m+\kappa}_{\sqrt{\widehat{\Sigma}}, \vec{\mathbf{0}}, \Lambda^{\perp}_{\vec{\mathbf{u}}}}$.
3. *With probability* $1 - \frac{1}{K}$, *set* $z := \perp$.
4. *Output*
$$(\vec{\mathbf{z}}, \mathbf{1}, \mathbf{b}_2, \ldots, \mathbf{b}_\nu, \vec{\mathbf{u}}, \vec{\mathbf{v}}).$$

Moreover, RejSamp *outputs non-*$\perp$ *with probability* $\geq \frac{1-\epsilon}{K}\left(1 - \frac{4\varepsilon}{(1+\varepsilon)^2}\right)$.

2.2 Non-Interactive Zero-Knowledge Argument

Let $\mathrm{R} : \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ be a parameterized relation, where pp_{R} denotes the relation parameters, θ is a statement, and w is a witness such that $\mathrm{R}(pp_{\mathrm{R}}, \theta, w) = 1$. Let $\mathcal{L}$ be a language that denote the set of all $\theta \in \{0,1\}^*$ such that there exists at least one $w \in \{0,1\}^*$ for which $\mathrm{R}(pp_{\mathrm{R}}, \theta, w) = 1$ holds

true. A Non-Interactive Argument of Knowledge (NARK) for the relation R, denoted by $\mathsf{NARK}[\mathrm{R}]$ [5], consists of the following PPT algorithms:

- $\mathsf{NARK.Rel}(1^\lambda) \to pp_\mathrm{R}$: On input the security parameter λ, outputs the relation parameters pp_R.
- $\mathsf{NARK.Setup}(pp_\mathrm{R}) \to \mathsf{crs}$: On input pp_R, outputs a common reference string crs.
- $\mathsf{NARK.Prove}(\mathsf{crs}, \theta, w) \to \pi$: On input crs, a statement θ, and a witness w, outputs a proof π.
- $\mathsf{NARK.Ver}(\mathsf{crs}, \theta, \pi) \to \{0,1\}$: On input crs, a statement θ, and a proof π, outputs 1 if π is accepted and 0 otherwise.

Correctness: A NARK is said to satisfy the correctness condition, if for every (θ, w) pair such that $\mathrm{R}(pp_\mathrm{R}, \theta, w) = 1$, it holds that

$$\Pr\left[\mathsf{NARK.Ver}(\mathsf{crs}, \theta, \mathsf{NARK.Prove}(\mathsf{crs}, \theta, w)) = 1\right] = 1,$$

where $\mathsf{crs} \leftarrow \mathsf{NARK.Setup}(pp_\mathrm{R})$.

Computational Soundness. A NARK is said to satisfy the computational soundness condition, if for every PPT adversary A, $\mathsf{Adv}^{\mathsf{SND}}_{\mathsf{NARK}[\mathrm{R}],\mathsf{A}}(\lambda)$, defined as follows

$$\Pr\left[\begin{array}{c|c} \mathsf{NARK.Ver}(\mathsf{crs}, \theta, \pi) = 1 \\ \wedge \mathrm{R}(pp_\mathrm{R}, \theta, w) = 0 \end{array} \middle| \begin{array}{c} pp_\mathrm{R} \leftarrow \mathsf{NARK.Rel}(1^\lambda); \\ \mathsf{crs} \leftarrow \mathsf{NARK.Setup}(pp_\mathrm{R}); \\ (\theta, \pi) \leftarrow \mathsf{A}(\mathsf{crs}); \end{array}\right]$$

is at most $\mathrm{negl}(\lambda)$, where $\mathrm{negl}(\lambda)$ is a negligible function with respect to the input size λ. In other words, computational soundness of a NARK says that a computationally bounded adversary A should not be able to produce an invalid proof that passes the verification algorithm of the NARK except with negligible probability.

Knowledge Soundness. A NARK is said to satify the knowledge soundness condition, if for every PPT adversary A, there exists a PPT extractor $\mathcal{E}$ such that $\mathsf{Adv}^{\mathsf{KS}}_{\mathsf{NARK}[\mathrm{R}],\mathsf{A}}(\lambda)$, defined as follows

$$\Pr\left[\begin{array}{c|c} \mathsf{NARK.Ver}(\mathsf{crs}, \theta, \pi) = 1 \\ \wedge \mathrm{R}(pp_\mathrm{R}, \theta, w) = 1 \end{array} \middle| \begin{array}{c} pp_\mathrm{R} \leftarrow \mathsf{NARK.Rel}(1^\lambda); \\ \mathsf{crs} \leftarrow \mathsf{NARK.Setup}(pp_\mathrm{R}); \\ (\theta, \pi) \leftarrow \mathsf{A}(\mathsf{crs}); \\ w \leftarrow \mathcal{E}^{\mathsf{A}}(\mathsf{crs}, \theta, \pi) : \end{array}\right]$$

is at least $1 - \mathrm{negl}(\lambda)$, where $\mathrm{negl}(\lambda)$ is a negligible function with respect to the input size λ. In other words, whenever an adversary A, without having the knowledge of the witness, is able to produce an accepting proof, then a PPT algorithm $\mathcal{E}^{\mathsf{A}}$ exists, which depends on the specific adversary A, can extract a valid witness for the corresponding statement.

[5] We write NARK instead of $\mathsf{NARK}[\mathrm{R}]$ whenever the relation R is understood from the context.

Zero-Knowledge Non-Interactive Argument of Knowledge. A Zero-Knowledge Non-Interactive Argument of Knowledge (ZK-NARK) for a relation R, denoted by $\mathsf{ZK\text{-}NARK}[\mathrm{R}]$, is a $\mathsf{NARK}[\mathrm{R}]$ with an additional PPT algorithm called *simulator* that takes as input a common reference string crs, a simulation trapdoor τ and a statement θ, and outputs a proof π_{sim}, i.e.,

$$\mathsf{ZK\text{-}NARK.SimProve}(\mathsf{crs}, \tau, \theta) \rightarrow \pi_{\mathsf{sim}},$$

where the setup algorithm $\mathsf{ZK\text{-}NARK.Setup}(pp_{\mathrm{R}}) \rightarrow \mathsf{crs}$ additionally outputs a simulation trapdoor τ along with a common reference string crs. We call the ZK-NARK is computationally zero-knowledge if, for every PPT algorithm A and every (θ, w) such that $\mathrm{R}(pp_{\mathrm{R}}, \theta, w) = 1$ holds true, and

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{ZK\text{-}NARK}[\mathrm{R}],\mathsf{A}}(\lambda) := \Big| &\Pr\left[\mathsf{A}(\mathsf{crs}, \mathsf{ZK\text{-}NARK.Prove}(\mathsf{crs}, \theta, w)) = 1\right] \\ &- \Pr\left[\mathsf{A}(\mathsf{crs}, \mathsf{ZK\text{-}NARK.SimProve}(\mathsf{crs}, \tau, \theta)) = 1\right] \Big|\end{aligned}$$

is upper bounded by $\mathrm{negl}(\lambda)$. Note that, a $\mathsf{ZK\text{-}NARK}$ satisfies the correctness and the computational soundness properties of the underlying $\mathsf{NARK}[\mathrm{R}]$, together with the above computational zero-knowledge property.

2.3 Blind Signature

A blind signature (BS) scheme consists of the following PPT algorithms:

- $pp \leftarrow \mathsf{BS.Setup}(1^{\lambda})$: it takes as input the security ppameter λ and outputs the public parameters pp.
- $(sk, vk) \leftarrow \mathsf{BS.KeyGen}(pp)$: it takes as input the public parameter pp and generates the signing and verification key pair (sk, vk).
- $\sigma / \perp \leftarrow \mathsf{BS.Sign}(\mathcal{U}(M, vk), \mathcal{S}(sk))$: An interactive protocol between the user $\mathcal{U}$ with input the message M and the verification key vk and the signer $\mathcal{S}$ with input the signing key, where the user obtains a signature σ on the message M without revealing M to the signer. The protocol outputs a valid signature σ upon successful completion and $\perp$ otherwise.
- $b \leftarrow \mathsf{BS.Verify}(vk, M, \sigma)$: it takes as input the verification key vk, the message M and the purported signature σ and returns a bit $b \in \{0, 1\}$.

A blind signature scheme BS is said to satisfy the correctness condition if for every security parameter λ and for every message M, an honest execution of $\mathsf{BS.Sign}(\mathcal{U}(M, vk), \mathcal{S}(sk))$ between the signer and the user with honestly generated keys (sk, vk) outputs a signature σ such that

$$\Pr\left[\mathsf{BS.Verify}(vk, M, \sigma) = 1\right] = 1 - \mathsf{negl}(\lambda).$$

Security Notion. A blind signature scheme is said to be secure if it satisfies the blindness property and the one-more unforgeablity property, which are defined as follows:

Blindness. Let A be a PPT malicious signer that chooses a pair of messages (M_0, M_1) and the honest user chooses a bit $b \in \{0,1\}$ uniformly at random. Then the malicious signer and the user participates in the concurrent signing process on both the messages. If the signature of any of the two messages happens to produce $\perp$, then the entire experiment terminates with $\perp$, otherwise A receives the signatures for both the messages. Finally, A has to output a bit b'. A wins the game if b' matches with b. Thus, for any PPT malicious signer A and for any two messages M_0, M_1, the blindness advantage of BS for A is defined as

$$\mathsf{Adv}^{\mathsf{Blind}}_{\mathsf{BS},\mathsf{A}}(\lambda) := \left| \Pr\left[\mathsf{Blind}^{\mathsf{A}}_{\mathsf{BS}}(\lambda) = 1\right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{Blind}^{\mathsf{A}}_{\mathsf{BS}}(\lambda)$ is defined in Fig. 1.

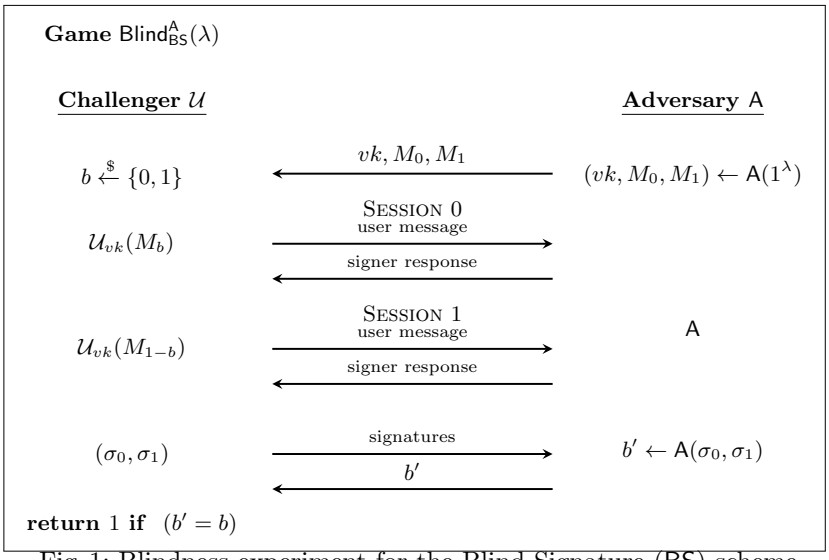

Fig. 1: Blindness experiment for the Blind Signature (BS) scheme.

One-More Unforgeability. The notion of one-more unforgeability (OMUF) says that for any PPT adversary A that interacts with an honest signer in at most ℓ signing sessions, the probability that A outputs $\ell+1$ distinct message-signature pairs $\{(M_i, \sigma_i)\}_{i=1}^{\ell+1}$ such that $\mathsf{BS.Verify}(vk, M_i, \sigma_i) = 1$, for all $i \in [\ell+1]$, is negligible i.e

$$\mathsf{Adv}^{\mathsf{OMUF}}_{\mathsf{BS},\mathsf{A}}(\lambda) := \Pr\left[\mathsf{OMUF}^{\mathsf{A}}_{\mathsf{BS}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda),$$

where the game $\mathsf{OMUF}^{\mathsf{A}}_{\mathsf{BS}}$ is defined in Fig. 2. The game returns 1 if A outputs $\ell+1$ distinct message-signature pairs $\{(M_i, \sigma_i)\}_{i=1}^{\ell+1}$ such that $\mathsf{BS.Verify}(vk, M_i, \sigma_i) = 1$ for all $i \in [\ell+1]$.

Game $\mathsf{OMUF}^{\mathsf{A}}_{\mathsf{BS}}(\lambda)$

1. $pp \leftarrow \mathsf{BS.Setup}(1^\lambda)$
2. $(sk, vk) \leftarrow \mathsf{BS.KeyGen}(pp)$
3. Initialise q_s to 0
4. Run $\{(M^*_\ell, \sigma^*_\ell)\}_{\ell \in [f]} \leftarrow \mathsf{A}^{\mathsf{SIGN}}(vk)$.
5. **return** 1 iff each of the following holds:
   (a) $f > q_s$,
   (b) $\forall \ell \in [f],\ \mathsf{BS.Ver}(M^*_\ell, \sigma^*_\ell) = 1$.
6. Otherwise, **return** 0.

Oracle SIGN(pp,M)

1. $q_s \leftarrow q_s + 1$ // increment session counter
2. $\sigma \leftarrow \mathsf{BS.Sign}(pp, sk, M)$
3. **return** σ

Fig. 2: One-more Unforgeability Game of Blind Signature scheme

2.4 Two Round Multi-Signature

Following [45], a two-round multi-signature scheme MS for a group of N signers on a message M in the plain public key model [5] consists of the following PPT algorithms:

- $pp \leftarrow \mathsf{MS.Setup}(1^\lambda)$: it takes the security parameter λ as input and output the publiuc parameters pp.
- $(sk_i, vk_i) \leftarrow \mathsf{MS.KeyGen}(pp)$: This algorithm is executed by every signers in the group. It takes the public parameter as input and generates the signing and verification key pair of individual users in the group.
- $avk \leftarrow \mathsf{MS.KeyAgg}(L = \{vk_i\}_{i=1}^N)$: This algorithm is executed by every signers in the group. iIt takes the verification keys of all the signers in the group and produces an aggregated single verification key avk.

To sign a message M by all the N signers in the group, the following interactive signature algorithm ($\mathsf{MS.Sign1}, \mathsf{MS.SignAgg1}, \mathsf{MS.Sign2}, \mathsf{MS.SignAgg2}, \mathsf{MS.Sign}$) is run by each signer $i \in [N]$ in a sequence of two communication rounds

- $(out_i, state_i)/\perp \leftarrow \mathsf{MS.Sign1}(pp)$: This algorithm is executed by every signers in the group. It takes the public paramater as input and produces the first round output out_i, and some first round secret state $state_i$ for the i-th signer or it outputs $\perp$ on failure. Every signer broadacasts their individual first round output out_i to all the other co-signers.
- $out/\perp \leftarrow \mathsf{MS.SignAgg1}(pp, out_1, \ldots, out_N)$: This algorithm is executed by every signers in the group. It takes the public parameter pp and the first round output from all signers $(out_1, \ldots, out_N)$ and aggregates them to a single first round output out and returns out or $\perp$ on failure.

- $(out'_i, state'_i)/\perp \leftarrow \mathsf{MS.Sign2}(pp, sk_i, M, \{vk_j\}_{\substack{j=1 \\ j\neq i}}^{N}, out, state_i)$: This algorithm is executed by every signers in the group. It takes the public parameter pp, the secret key of the i-th signer, the mesage and the set of public keys of all the other co-signers, aggregated first round output out and the first round secret state $state_i$ of the i-th signer. It produces the second round output out'_i, and some second round secret state $state'_i$ for the i-th signer or or it outputs $\perp$ on failure. Every signer broadacasts their individual second round output out'_i to all the other co-signers.
- $out'/\perp \leftarrow \mathsf{MS.SignAgg2}(out'_1, \ldots, out'_N)$: This algorithm is executed by every signers in the group. It takes second round output from all signers $(out'_1, \ldots, out'_N)$ and aggregates them to a single second round output out' and returns out' or $\perp$ on failure
- $\sigma/\perp \leftarrow \mathsf{MS.Sign}(pp, out', state'_i)$: This algorithm is executed by every signers in the group. It takes the public parameter, aggregated second round output out' and the second round secret state $state'_i$ of the i-th signer. It outputs a signature σ or $\perp$ on failure
- $b \leftarrow \mathsf{MS.Ver}(pp, avk, M, \sigma)$: it takes as input the public parameter pp, aggregated verification key avk, message M and a purported signature σ and outputs a bit $b \in \{0, 1\}$.

Game $\mathsf{Correct}^{\mathsf{MS}}_{M,N,i}(\lambda)$

$pp \leftarrow \mathsf{MS.Setup}(1^\lambda)$
for $j = 1, \ldots, N$ **do**
  $(sk_j, vk_j) \leftarrow \mathsf{MS.KeyGen}(pp)$
  $(out_j, state_j) \leftarrow \mathsf{MS.Sign1}(pp)$
end for
$out \leftarrow \mathsf{MS.SignAgg1}(pp, out_1, \ldots, out_N)$
for $j = 1, \ldots, N$ **do**
  $(out'_j, state'_j) \leftarrow \mathsf{MS.Sign2}(pp, sk_j, M, \{vk_t\}_{\substack{t=1 \\ t\neq j}}^{N}, out, state_j)$
end for
$out' \leftarrow \mathsf{MS.SignAgg2}(out'_1, \ldots, out'_N)$
$\sigma \leftarrow \mathsf{MS.Sign}(pp, out', state'_i)$
$avk \leftarrow \mathsf{MS.KeyAgg}(vk_1, \ldots, vk_N)$
return $\mathsf{MS.Ver}(pp, avk, M, \sigma)$

Fig. 3: Correctness of the MS scheme.

A multi-signature scheme is said to satisfy the correctness condition if for every security parameter λ and for every message M, an honest execution of the following interactive signature algorithms ($\mathsf{MS.Sign1}, \mathsf{MS.SignAgg1}, \mathsf{MS.Sign2}$, $\mathsf{MS.SignAgg2}, \mathsf{MS.Sign}$) in sequence between all the signers in the group with honestly generated keys $(sk_i, vk_i)_{i=1}^N$, outputs a signature σ such that

$$\Pr[\mathsf{MS.Ver}(pp, avk, M, \sigma) = 1] = 1 - \mathrm{negl}(\lambda),$$

where $avk = \mathsf{MS.KeyAgg}(L = \{vk_i\}_{i=1}^{N})$. In other words, correctness requires that for every λ, every message M, every integer N and every $i \in [N]$, it holds that

$$\Pr[\mathsf{Correct}^{\mathsf{MS}}_{M,N,i}(\lambda) = 1] = 1,$$

where the correctness game $\mathsf{Correct}^{\mathsf{MS}}_{M,N,i}$ is given in Fig. 3.

Security Notion. A multi-signature scheme is said to be secure if it satisfies the *unforgeability* condition, which is typically formalized through the following game EUF-CMA, defined in Fig. 4. In this game, a key pair (sk_1, vk_1) is generated for the honest signer and the verification key vk_1 is given to the adversary A. The adversary can engage in any number of concurrent signing sessions with the honest signer by accessing three oracles SIGN1, SIGN2 and SIGN implementing the three steps Sign1, Sign2 and Sign of the signing algorithm respectively with the honest signer's secret key. This in particular means that the adversary can pass the same L, containing vk_1 multiple times, and the same message M to multiple SIGN2 calls, effectively obtaining a signing session in which the honest signer participates multiple times. Finally, A returns a multiset $L = \{vk_1, \ldots, vk_N\}$ of verification keys, a message M, and a signature σ. A wins the game if

- $vk_1 \in L$
- $\mathsf{MS.Ver}(\mathsf{MS.KeyAgg}(L = \{vk_i\}_{i\in[N]}), M, \sigma) = 1$
- A never made a SIGN2 query for multiset of verification keys L and message M.

Thus, for any PPT adversary A, the unforgeability advantage of MS for A is defined as

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MS,A}}(\lambda) := \Pr\left[\mathsf{EUF\text{-}CMA}^{\mathsf{A}}_{\mathsf{MS}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{EUF\text{-}CMA}^{\mathsf{A}}_{\mathsf{MS}}(\lambda)$ is defined in Fig. 4.

MuSig-L: Two Round Lattice Based Multi-Signature. MuSig-L is a lattice based two round multi-signature in the plain public key model that supports key aggregation. Its unforgeable security is based on the hardness of MLWE and MSIS problem. Moreover, unlike previous two round multi-signature schemes [42,4,41], MuSig-L is secure with resepct to the concurrent ROS attack [7]. One can view MuSig-L as a lattice adaptation of the MuSig2 construction [45]. An algorithmic description of the scheme is given in Algorithm 1.

Theorem 1 (EUF-CMA Security of MuSig-L). *[14, Theorem 4.5] Let* $q = 5 \mod 8$. *Then, for any PPT adversary* A *against the* EUF-CMA *security of* MuSig-L *that makes at most* q_h *queries to the random oracles and* q_s *sign queries,*

Game $\mathsf{EUF\text{-}CMA}^{\mathsf{A}}_{\mathsf{MS}}(\lambda)$

1. $pp \leftarrow \mathsf{MS.Setup}(1^\lambda)$
   // Honest signer idex is 1
2. $(sk^*, vk^*) \leftarrow \mathsf{MS.KeyGen}(pp)$; $(sk_1, vk_1) \leftarrow (sk^*, vk^*)$
3. Initialize ctr to 0 //session counter
4. $S \leftarrow \emptyset$, $S' \leftarrow \emptyset$ // sets of ongoing signing sessions for SIGN1 and SIGN2, respectively.
5. $\mathcal{S} \leftarrow \emptyset$, $\mathcal{S}' \leftarrow \emptyset$
6. $Q \leftarrow \emptyset$ // set of SIGN2 queries
7. $(L, M, \sigma) \leftarrow \mathsf{A}^{\mathsf{SIGN1},\mathsf{SIGN2},\mathsf{SIGN}}(vk_1)$
8. **return** 1 iff each of the following holds:
   (a) $vk_1 \in L$
   (b) $(L, M) \notin Q$
   (c) $\mathsf{Ver}(\mathsf{KeyAgg}(L), M, \sigma) = 1)$
9. Otherwise **return** 0

Oracle $\mathsf{SIGN1}(pp)$

1. $ctr \leftarrow ctr + 1$ // increment session counter
2. $k \leftarrow ctr$; $S \leftarrow S \cup \{k\}$ // open session k
3. $(out_{1,k}, state_{1,k}) \leftarrow \mathsf{MS.Sign1}(pp)$
4. $\mathcal{S} \leftarrow \mathcal{S} \cup \{state_{1,k}\}$
5. **return** $out_{1,k}$

Oracle $\mathsf{SIGN2}(pp, k, out, M, \{vk_2, \ldots, vk_n\})$

1. **if** $k \notin S$ **then return** $\perp$
2. Retrieve $state_{1,k}$ from $\mathcal{S}$
3. $(out'_{1,k}, state'_{1,k}) \leftarrow \mathsf{MS.Sign2}(pp, sk_1, M, \{vk_2, \ldots, vk_n\}, out, state_{1,k})$
4. $L \leftarrow \{vk_1, \ldots, vk_n\}$
5. $Q \leftarrow Q \cup \{L, M\}$
6. $S \leftarrow S \cup \{k\}$; $S' \leftarrow S' \cup \{k\}$
7. $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{state'_{1,k}\}$
8. **return** $out'_{1,k}$

Oracle $\mathsf{SIGN}(pp, k, out')$

1. **if** $k \notin S'$ **then return** $\perp$
2. Retrieve $state'_{1,k}$ from $\mathcal{S}'$
3. $\sigma \leftarrow \mathsf{MS.Sign}(pp, out', state'_{1,k})$
4. $S' \leftarrow S' \setminus \{k\}$
5. **return** σ

Fig. 4: EUF-CMA security game for two round Multi-Signature (MS) scheme.

Algorithm 1 Multi-Signature (MuSig-L)

Setup(1^λ):

1: $\bar{\mathbf{A}} \xleftarrow{\$} \mathcal{R}_q^{\kappa \times m}$
2: $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbb{I}_\kappa]$
3: $\mathsf{H}_{\mathsf{agg}} : \{0,1\}^* \to \mathcal{C}$, $\mathsf{H}_{\mathsf{sig}} : \{0,1\}^* \to \mathcal{C}$, $\mathsf{H}_{\mathsf{non}} : \{0,1\}^* \to \mathcal{C}^{\nu-1}$
4: $sp := (\mathbf{A}, q, \mathsf{H}_{\mathsf{agg}}, \mathsf{H}_{\mathsf{non}}, \mathsf{H}_{\mathsf{sig}})$
5: $crs_1 \leftarrow \mathsf{NARK.Setup}(sp)$
6: **return** $pp : (sp, crs_1)$

KeyGen(pp):

1: Sample $\vec{\mathbf{x}} \xleftarrow{\$} S_\eta^{m+\kappa}$ and compute $\vec{\mathbf{vk}} = \mathbf{A}\vec{\mathbf{x}} \mod q$
2: **return** $(\vec{\mathbf{vk}}, \vec{\mathbf{sk}}) := (\mathbf{A}\vec{\mathbf{x}}, \vec{\mathbf{x}})$

SignAgg2($out'_1, \ldots, out'_N$):

1: **if** $\exists\, i \in [1, N], \vec{\mathbf{s}}_i = \perp$ **then**
2: **return** $\perp$
3: **end if**
4: $\vec{\mathbf{s}} = \sum_{j=1}^{N} \vec{\mathbf{s}}_j$
5: **return** $out' \leftarrow \vec{\mathbf{s}}$

Sign($pp, out', state'_i$):

1: $state'_i = \vec{\mathbf{R}}$
2: $out' = \vec{\mathbf{s}}$
3: **return** $\sigma = (\vec{\mathbf{R}}, \vec{\mathbf{s}})$

RejSamp($\vec{\mathbf{v}}, \vec{\mathbf{s}}, (\mathbf{b}_j)_{j=1}^{\nu}$)

1: $\Sigma := \sigma_1^2 + \tilde{\sigma}^2 \cdot \sum_{i=2}^{\nu} (\mathbf{b}_j^\star \mathbf{b}_j) \cdot \mathbb{I}_{m+\kappa}$
2: $\gamma \xleftarrow{\$} [0,1]$
3: **if** $\gamma \geq \min\left(1, \frac{\mathcal{D}^{m+\kappa}_{\sqrt{\Sigma}}(\vec{\mathbf{s}})}{K \cdot \mathcal{D}^{m+\kappa}_{\sqrt{\Sigma}, \vec{\mathbf{v}}}(\vec{\mathbf{s}})}\right)$ **then**
4: **return** 0
5: **else**
6: **return** 1
7: **end if**

VERIFY($pp, \vec{\mathbf{X}}, M, \sigma$):

1: Compute $\vec{\mathbf{c}} \leftarrow \mathsf{H}_{\mathsf{sig}}(\vec{\mathbf{X}}, \vec{\mathbf{R}}, M)$
2: $\sigma = (\vec{\mathbf{R}}, \vec{\mathbf{s}})$
3:
4: **if** $\mathbf{A}\vec{\mathbf{s}} - \vec{\mathbf{R}} = \mathbf{c}\vec{\mathbf{X}}$ **then**
5: **return** 1
6: **else**
7: **return** 0
8: **end if**

Samp($\mathbf{r}$):

1: Sample $\mathbf{b} \leftarrow \mathcal{D}_{\sigma_b}^{(\mathbf{r})}$.
2: **return b**

KeyAgg($\mathbf{L}$):

1: $\mathbf{L} := \{\vec{\mathbf{vk}}_1, \ldots, \vec{\mathbf{vk}}_N\}$
2: **for** $i = 1, \ldots, N$ **do**
3: $\vec{\mathbf{a}}_i \leftarrow \mathsf{H}_{\mathsf{agg}}(\mathbf{L}, \vec{\mathbf{vk}}_i)$
4: **end for**
5: **return** $\vec{\mathbf{X}} = \sum_{i=1}^{N} \vec{\mathbf{a}}_i \vec{\mathbf{vk}}_i$

Sign1(pp):

1: Sample $\vec{\mathbf{r}}_{i,1} \leftarrow \mathcal{D}_{\sigma_1}^{m+\kappa}, \vec{\mathbf{r}}_{i,2} \ldots, \vec{\mathbf{r}}_{i,\nu} \leftarrow \mathcal{D}_{\tilde{\sigma}}^{m+\kappa}$
2: Compute $\vec{\mathbf{R}}_{i,1} = \mathbf{A}\vec{\mathbf{r}}_{i,1}, \ldots, \vec{\mathbf{R}}_{i,\nu} = \mathbf{A}\vec{\mathbf{r}}_{i,\nu}$
3: $out_i := (\vec{\mathbf{R}}_{i,1}, \ldots, \vec{\mathbf{R}}_{i,\nu})$
4: $state_i := (\vec{\mathbf{R}}_{i,1}, \ldots, \vec{\mathbf{R}}_{i,\nu})$
5: **return** $(out_i, state_i)$

SignAgg1($pp, out_1, \ldots, out_N$):

1: **if** $\exists\, j \in [N] \setminus \{i\}$, $\vec{\mathbf{vk}}_j = \vec{\mathbf{vk}}_i$ **then**
2: **return** $\perp$
3: **end if**
4: $\mathbf{L} := \{\vec{\mathbf{vk}}_1, \ldots, \vec{\mathbf{vk}}_n\}$
5: Compute $\mathbf{a}_i \leftarrow \mathsf{H}_{\mathsf{agg}}(\mathbf{L}, \vec{\mathbf{vk}}_i)$
6: $\vec{\mathbf{X}} \leftarrow \mathsf{KeyAgg}(\mathbf{L})$
7: Compute $\vec{\mathbf{R}}_j = \sum_{i=1}^{N} \vec{\mathbf{R}}_{i,j}$ where $j = 1$ to ν
8: **return** $out \leftarrow (\vec{\mathbf{R}}_1, \ldots, \vec{\mathbf{R}}_\nu)$

Sign2($pp, \vec{\mathbf{sk}}_i, M, (\vec{\mathbf{vk}}_j)_{j=1, j\neq i}^{N}, out, state_i$):

1: $out = (\vec{\mathbf{R}}_1, \ldots, \vec{\mathbf{R}}_\nu)$
2: $(\mathbf{r}_{(2)}, \ldots, \mathbf{r}_{(\nu)}) := \mathsf{H}_{\mathsf{non}}(\vec{\mathbf{X}}, \vec{\mathbf{R}}_1, \ldots, \vec{\mathbf{R}}_\nu, M)$
3: $\mathbf{b}_1 := 1$
4: For $j \in [2, \nu]$, $\mathbf{b}_j := \mathsf{Samp}(\mathbf{r}_{(j)})$
5: Compute $\vec{\mathbf{R}} = \sum_{j=1}^{\nu} \mathbf{b}_j \vec{\mathbf{R}}_j$
6: $\mathbf{c} := \mathsf{H}_{\mathsf{sig}}(\vec{\mathbf{X}}, \vec{\mathbf{R}}, M)$
7: Compute $\mathbf{a}_i \leftarrow \mathsf{H}_{\mathsf{agg}}(\mathbf{L}, \vec{\mathbf{vk}}_i)$
8: compute $\vec{\mathbf{v}}_i = \mathbf{c}\mathbf{a}_i \vec{\mathbf{x}}_i$
9: compute $\vec{\mathbf{s}}_i = \vec{\mathbf{v}}_i + \sum_{t=1}^{\nu} \mathbf{b}_t \vec{\mathbf{r}}_{i,t}$
10: **if** $\mathsf{RejSamp}(\vec{\mathbf{v}}_i, \vec{\mathbf{s}}_i, (\mathbf{b}_j)_{j=1}^{\nu}) = 0$ **then**
11: **return** $\vec{\mathbf{s}}_i = \perp$
12: **else**
13: **return** $\vec{\mathbf{s}}_i$
14: **end if**
15: **return** $(out'_i, state'_i) \leftarrow (\vec{\mathbf{s}}_i, \vec{\mathbf{R}})$

there exist PPT adversaries B *and* C *such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MuSig\text{-}L},\mathsf{A}}(\lambda) \leq & 2(q_h+q_s)^2 \cdot \left(\frac{1+2^{-\Omega(n)}}{q^{\kappa n}}\right)^{\nu} + \frac{(2q_h+q_s)^2}{\sigma_{\rho_b}(\mathcal{R})} \\ & + e(q_s+1)\left(q_s\epsilon_s + \frac{q_h(2q_h+3)}{|\mathcal{C}|} + \frac{2^{\kappa+1}}{q^{\kappa n/2}} + \right. \\ & \left. \mathrm{Adv}^{\mathsf{MLWE}}_{q,\kappa,m,\eta,\mathsf{B}} \sqrt{\frac{q_h^2}{|\mathcal{C}|} + q_h\sqrt{q_h \cdot \mathsf{Adv}^{\mathsf{MSIS}}_{q,\kappa,m+1,\beta,\mathsf{C}}}}\right),\end{aligned}$$

where recall that $\mathcal{C}$ *is the challenge space and* $\beta = 8\kappa_1\sqrt{B_N^2+\kappa_1^3}$ *where* $B_N = \sqrt{N}B$, N *is the number of signers in the group and* $B = \sigma_1\sqrt{n(m+\kappa)}$. *Moreover,* ϵ_s *is the statistical distance of the simulator and the original honest party protocols. Remaining parameters are described in Table 2.*

3 Two Round Blind Multi-Signature

In this section, we combine the definition of blind signature and the two round multi-signature to realize two round blind multi-signature (BMS) scheme. A two round blind multi-signature (BMS) scheme for a group of N signers on a message M consists of the following PPT algorithms.

- $pp \leftarrow \mathsf{BMS.Setup}(1^\lambda)$: it takes the security parameter λ as input and output the publiuc parameters pp.
- $(sk_i, vk_i) \leftarrow \mathsf{BMS.KeyGen}(pp)$: This algorithm is executed by every signers in the group. It takes the public parameter as input and generates the signing and verification key pair of individual users in the group.
- $avk \leftarrow \mathsf{BMS.KeyAgg}(L = \{vk_i\}_{i=1}^N)$: This algorithm is executed by every signers in the group. iIt takes the verification keys of all the signers in the group and produces an aggregated single verification key avk.

To obtain a blind multi-signature on a message M from all N signers in the group, the following interactive signature algorithms ($\mathsf{BMS.User1}, \mathsf{BMS.Sign1}$, $\mathsf{BMS.UserAgg1}, \mathsf{BMS.User2}, \mathsf{BMS.Sign2}, \mathsf{BMS.User}$) is executed in sequence between the user $\mathcal{U}$ and the N signers $\{\mathcal{S}_i\}_{i\in[N]}$ in two communication rounds.

- $(out_0, state_0)/\perp \leftarrow \mathsf{BMS.User1}(pp, L = \{vk_i\}_{i=1}^N, M)$: This algorithm is executed by the user only. It takes the public parameter pp, the set of public keys of all the signers L and the message to be signed M as input, and it returns user's public output out_0 and user's internal secret state $state_0$ or $\perp$ on failure. After the execution, the user broadcasts its public output out_0 to all the signers.
- $(out_i, state_i)/\ \perp \leftarrow \mathsf{BMS.Sign1}(pp)$: This algorithm is executed by every signers in the group. It takes the public parameter pp as input and generates the first round output out_i and some first round secret state $state_i$ for the i-th signer, or outputs $\perp$ on failure. Every signer sends their individual first round output out_i to the user.

- $out/\perp \leftarrow \mathsf{BMS.UserAgg1}(pp, out_1, \ldots, out_N)$: This algorithm is executed by the user only. It takes the public parameter pp and the first round output from the all the signers $(out_1, \ldots, out_N)$ and aggregates them into a single output out, or outputs $\perp$ if aggregation fails.
- $(out'_0, state'_0)/\perp \leftarrow \mathsf{BMS.User2}(pp, out, state_0)$: This algorithm is again executed by the user only. It takes the public parameter pp, the aggregated output out from $\mathsf{BMS.UserAgg}$ and the user's internal secret state $state_0$. It returns the second-round user's public output out'_0 and an updated user's internal secret state $state'_0$ or returns $\perp$ on failure. After the execution, the user broadcasts its second round public output out'_0 to all the signers.
- $out'_i/\perp \leftarrow \mathsf{BMS.Sign2}(pp, sk_i, state_i, out_0, out'_0)$: This algorithm is executed by every signers in the group. It takes the public parameter pp, the secret key of the i-th signer sk_i, the secret state of the i-th signer $state_i$ from the first round, and user's old and new public output out_0, out'_0 respectively. It returns the second round output out'_i for the i-th signer or returns $\perp$ on failure. Every signer sends their individual second round output out'_i to the user.
- $\sigma/\perp \leftarrow \mathsf{BMS.User}(pp, state'_0, out'_1, \ldots, out'_N)$: This algorithm is executed by the user only. It takes the public parameter pp, the updated user's internal secret state $state'_0$ and the second-round outputs from all signers. It generates the blind multi-signature σ, or outputs $\perp$ on failure.

- $b \leftarrow \mathsf{BMS.Ver}(pp, avk, M, \sigma)$: it takes as input the public parameter pp, aggregated verification key avk, the message M and a purported signature σ and outputs a bit $b \in \{0, 1\}$.

A blind multi-signature (BMS) scheme is said to satisfy the correctness condition if for every security parameter λ and for every message M, an honest execution of the following interactive signature algorithms ($\mathsf{BMS.User1BMS.Sign1}$, $\mathsf{BMS.UserAgg1}$, $\mathsf{BMS.User2}$, $\mathsf{BMS.Sign2}$, $\mathsf{BMS.User}$) in sequence between all the signers in the group and the user with honestly generated keys $(sk_i, vk_i)_{i\in[N]}$, outputs a signature σ such that

$$\Pr[\mathsf{BMS.Ver}(pp, avk, M, \sigma) = 1] = 1 - \mathrm{negl}(\lambda),$$

where $avk = \mathsf{MS.KeyAgg}(L = \{vk_i\}_{i=1}^N)$. In other words, correctness requires that for every λ, every message M, and every integer N, it holds that

$$\Pr[\mathsf{Correct}^{\mathsf{BMS}}_{M,N}(\lambda) = 1] = 1,$$

where the correctness game $\mathsf{Correct}^{\mathsf{BMS}}_{M,N}$ is given in Fig. 5.

Security Notion. A blind multi-signature scheme is said to be secure if it satisfies the *one-more unforgeability* game and the *blindness* game as defined below:

One-More Unforgeability. We consider security in the *plain public-key (PPK)* model, meaning that the signers are only required to have public keys, but do

Game $\mathsf{Correct}^{\mathsf{BMS}}_{M,N}(\lambda)$
1: $pp \leftarrow \mathsf{BMS.Setup}(1^\lambda)$
2: **for** $j = 1, \ldots, N$ **do**
3: $\quad (sk_j, vk_j) \leftarrow \mathsf{BMS.KeyGen}(pp)$
4: **end for**
5: $(out_0, state_0) \leftarrow \mathsf{BMS.User1}(pp, L = \{vk_i\}_{i=1}^N, M)$
6: **for** $j = 1, \ldots, N$ **do**
7: $\quad (out_j, state_j) \leftarrow \mathsf{BMS.Sign1}(pp)$
8: **end for**
9: $out \leftarrow \mathsf{BMS.UserAgg1}(pp, out_1, \ldots, out_N)$
10: $(out'_0, state'_0) \leftarrow \mathsf{BMS.User2}(pp, out, state_0)$
11: **for** $j = 1, \ldots, N$ **do**
12: $\quad out'_j \leftarrow \mathsf{BMS.Sign2}(pp, sk_j, state_j, out_0, out'_0)$
13: **end for**
14: $\sigma \leftarrow \mathsf{BMS.User}(pp, state'_0, out'_1, \ldots, out'_N)$
15: $avk \leftarrow \mathsf{BMS.KeyAgg}(vk_1, \ldots, vk_N))$
16: **return** $\mathsf{BMS.Ver}(pp, avk, M, \sigma)$

Fig. 5: Correctness of the BMS scheme.

not have to prove knowledge of the private keys corresponding to the public keys to a certification authority or to other signers before engaging the protocol.

In the one-more unforgeability game, the adversary A corrupts all signers except one honest signer. The challenger generates a key pair (sk_1, vk_1) for the honest signer and provides the verification key vk_1 to A. The adversary may initiate an arbitrary number of concurrent signing sessions with the honest signer. To model these interactions, A is given oracle access to SIGN1 and SIGN2, which simulate the honest signer's behavior in the first and second rounds of the interactive signing protocol, respectively, using the secret key sk_1. Eventually, the adversary outputs a non-empty forgery set

$$\{(L^*_\ell, M^*_\ell, \sigma^*_\ell = (R^*_\ell, s^*_\ell))\}_{\ell \in [f]}\,.$$

The adversary wins the OMUF game if the following conditions are satisfied:

(i) the number of valid forgeries satisfies $f > q_s$, where q_s denotes the number of completed SIGN2 queries;
(ii) for every $\ell \in [f]$, the verification key set L^*_ℓ contains the honest signer's verification key vk_1;
(iii) every forgery is valid, i.e.,

$$\forall\, \ell \in [f], \quad \mathsf{BMS.Ver}(\mathsf{BMS.AggKey}(L^*_\ell), M^*_\ell, \sigma^*_\ell) = 1.$$

Fig. 6 presents the corresponding security games for OMUF, which formally capture the unforgeability requirements of BMS. Observe that the adversary may invoke the SIGN1 oracle without ever completing the corresponding protocol execution via SIGN2. Consequently, any valid forgery obtained after interacting

only with SIGN1 still counts as a forgery. Therefore, the winning condition compares the number of forgeries against the number of completed SIGN2 queries, rather than the total number of initiated signing sessions. A blind multi-signature scheme BMS is *one-more unforgeable* if, for every probabilistic polynomial-time adversary A,

$$\mathsf{Adv}^{\mathsf{OMUF}}_{\mathsf{BMS},\mathsf{A}}(\lambda) := \Pr\Big[\mathsf{OMUF}^{\mathsf{A}}_{\mathsf{BMS}}(\lambda) = 1\Big] \leq \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{OMUF}^{\mathsf{A}}_{\mathsf{BMS}}$ is defined in Fig. 6.

Game $\mathsf{OMUF}^{\mathsf{A}}_{\mathsf{BMS}}(\lambda)$

1. $pp \leftarrow \mathsf{BMS.Setup}(1^\lambda)$.
2. $(sk_1, vk_1) \leftarrow \mathsf{BMS.KeyGen}(pp)$ //signer 1 is honest.
3. Initialize q_s to 0.
4. $S \leftarrow \emptyset$, $\mathcal{S} \leftarrow \emptyset$
5. Run $\{(L^*_\ell, M^*_\ell, \sigma^*_\ell)\}_{\ell\in[f]} \leftarrow \mathsf{A}^{\mathsf{SIGN1},\mathsf{SIGN2}}(vk_1)$.
6. **return** 1 iff each of the following holds:
   (a) $f > q_s$,
   (b) $\forall \ell \in [f],\ vk_1 \in L^*_\ell$,
   (c) $\forall \ell \in [f],\ \mathsf{BMS.Ver}(\mathsf{BMS.KeyAgg}(L^*_\ell), M^*_\ell, \sigma^*_\ell) = 1$.
7. Otherwise, **return** 0.

Oracle $\mathsf{SIGN1}(pp, out_0)$

1. $k \leftarrow k + 1$ // increment session counter
2. $S \leftarrow S \cup \{k\}$ // open session k
3. $(out_{1,k}, state_{1,k}) \leftarrow \mathsf{BMS.Sign1}(pp, ct)$
4. $\mathcal{S} \leftarrow \mathcal{S} \cup \{state_{1,k}\}$
5. **return** $out_{1,k}$

Oracle $\mathsf{SIGN2}(pp, k, out_0, out'_0)$

1. **if** $k \notin S$ **then return** $\perp$
2. Retrieve $state_{1,k}$ from $\mathcal{S}$
3. $out'_{1,k} \leftarrow \mathsf{BMS.Sign2}(pp, sk_1, state_{1,k}, out_0.out'_o)$
4. $q_s \leftarrow q_s + 1$, $S \leftarrow S \cup \{k\}$
5. **return** $out'_{1,k}$

Fig. 6: One-More Unforgeability Game of Blind Multi-Signature Scheme.

Blindness. In this experiment, the adversary A controls all-but-one signers in the group and is modeled by a pair of probabilistic polynomial-time algorithms $(\mathsf{A}_1, \mathsf{A}_2)$, i.e., $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$. In the first phase, A_1 outputs a verification key set L together with two challenge messages (M_0, M_1). The challenger samples a uniformly random bit $b \xleftarrow{\$} \{0, 1\}$ and initiates two concurrent signing sessions indexed by $k \in \{0, 1\}$. The first session is executed on message M_b, while the

second session is executed on message M_{1-b}. In each session, the honest user's actions are simulated by the oracles USER_1, USER_2, and USER. If either signing session aborts, the experiment returns $(\perp, \perp)$. Otherwise, after both sessions terminate successfully, the challenger returns the resulting signatures (σ_0, σ_1) on messages (M_0, M_1), respectively.

Finally, A_2 outputs a guess bit b'. The adversary wins the blindness experiment if $b' = b$. Intuitively, blindness requires that the adversary cannot determine which signing session produced which signature. Therefore, any probabilistic polynomial-time adversary can succeed only with negligible advantage over random guessing. We depict the game in Figure 7.

A blind multi-signature scheme BMS is *blind* if, for every probabilistic polynomial-time adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$,

$$\mathsf{Adv}^{\mathsf{Blind}}_{\mathsf{BMS},\mathsf{A}}(\lambda) := \left| \Pr\left[\mathsf{Blind}^{\mathsf{A}}_{\mathsf{BMS}}(\lambda) = 1\right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{Blind}^{\mathsf{A}}_{\mathsf{BMS}}$ is defined in Figure 7.

4 BMuSig-L: A Lattice Based Blind Multi-Signature

In this section, we propose the first lattice based blind multi-signature scheme $\mathsf{BMuSig\text{-}L}$. To get a message M blindly signed by a group of signers, the user first samples blinding factors $\vec{\boldsymbol{\alpha}}$, $\boldsymbol{\beta}$ and $\vec{\boldsymbol{\xi}}$, then encrypts them along with the message M using an IND-CPA secure [6] public key encryption scheme $\mathsf{PKE.Enc}$ with randomness ρ to obtain the ciphertext ct, i.e., $ct \leftarrow \mathsf{PKE.Enc}\big(ek, (M, \vec{\boldsymbol{\alpha}}, \boldsymbol{\beta}, \vec{\boldsymbol{\xi}}); \rho\big)$ and then the ciphertext is sent to all the signers. Next, each signer sends a nonce tuple $(\vec{\mathbf{R}}_{i,1}, \ldots, \vec{\mathbf{R}}_{i,\nu})$ of length ν, to the user. The user collects them from all signers and computes the aggregated nonce $\vec{\mathbf{R}}_t := \sum_{i=1}^{N} \vec{\mathbf{R}}_{i,t}$ for each $t \in [\nu]$ to yield an aggregated nonce tuple $(\vec{\mathbf{R}}_1, \ldots, \vec{\mathbf{R}}_\nu)$. The user also computes the nonce coefficients as $(\mathbf{r}_{(2)}, \ldots, \mathbf{r}_{(\nu)})$ as the hash of the aggregated verification key $\vec{\mathbf{X}}$, the aggregated nonce tuple $(\vec{\mathbf{R}}_1, \ldots, \vec{\mathbf{R}}_\nu)$, and the message M, i.e.,

$$(\mathbf{r}_{(2)}, \ldots, \mathbf{r}_{(\nu)}) \leftarrow \mathsf{H}_{\mathsf{non}}\big(\vec{\mathbf{X}}, \vec{\mathbf{R}}_1, \ldots, \vec{\mathbf{R}}_\nu, M\big) + \vec{\boldsymbol{\xi}},$$

where $\vec{\mathbf{X}}$ is the aggregated verification key, computed as $\vec{\mathbf{X}} \leftarrow \sum_{i=1}^{N} \vec{\mathbf{a}}_i \vec{\mathbf{vk}}_i$, where $\vec{\mathbf{a}}_i \leftarrow \mathsf{H}_{\mathsf{agg}}(\{\vec{\mathbf{vk}}_1, \ldots, \vec{\mathbf{vk}}_N\}, \vec{\mathbf{vk}}_i)$. Then, the user sets $\mathbf{b}_1 := \mathbf{1}$ and samples the remaining $\nu - 1$ many $\mathbf{b}_j$ values following the discrete Gaussian distribution using the corresponding randomness $\mathbf{r}_{(j)}$. The user then combines the aggregated nonce tuples with $(\mathbf{b}_1, \ldots, \mathbf{b}_\nu)$ as $\vec{\mathbf{R}} \leftarrow \sum_{j=1}^{\nu} \mathbf{b}_j \vec{\mathbf{R}}_j$. We call $\vec{\mathbf{R}}$ as aggregated nonce. User subsequently rerandomizes the aggregated nonce as $\vec{\mathbf{R}}' \leftarrow \mathbf{A}\vec{\boldsymbol{\alpha}} + \boldsymbol{\beta}\vec{\mathbf{X}} + \vec{\mathbf{R}}$.

[6] The IND-CPA security definition for a given public key encryption scheme is given in Appendix D.

Game $\mathsf{Blind}^{\mathsf{A}}_{\mathsf{BMS}}(\lambda)$

1. $b \xleftarrow{\$} \{0,1\}$
2. $\mathcal{S} \leftarrow \emptyset$, $\mathcal{S}' \leftarrow \emptyset$
3. $pp \leftarrow \mathsf{BMS.Setup}(1^\lambda)$
4. $(M_0, M_1, L) \leftarrow \mathsf{A}_1(pp)$
5. $(\mathsf{sess}_0, \mathsf{sess}_1) \leftarrow (\mathsf{init}, \mathsf{init})$
6. $b' \leftarrow \mathsf{A}_2^{\mathsf{USER1},\mathsf{USER2},\mathsf{USER}}()$
7. **return** 1 **if** $(b' = b)$

Oracle $\mathsf{USER1}(pp, k, L = \{vk_i\}_{i=1}^N, M)$

1. **if** $\mathsf{sess}_k \neq \mathsf{init}$ **then return** $\perp$
2. $\mathsf{sess}_k \leftarrow \mathsf{open}$ // start session k
3. $(out_{0,k}, state_{0,k}) \leftarrow \mathsf{BMS.User1}(pp, L, M_{k \oplus b})$
4. $\mathcal{S} \leftarrow \mathcal{S} \cup \{state_{0,k}\}$
5. **return** $out_{0,k}$

Oracle $\mathsf{USER2}(pp, k, out_k)$

1. **if** $\mathsf{sess}_k \neq \mathsf{open}$ **then return** $\perp$
2. $\mathsf{sess}_k \leftarrow \mathsf{await}$
3. Retrieve $state_{0.k}$ from $\mathcal{S}$
4. $(out'_{0,k}, state'_{0,k}) \leftarrow \mathsf{BMS.User2}(pp, out_k, state_{0,k})$
5. $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{state'_{0,k}\}$
6. **return** $out'_{0,k}$

Oracle $\mathsf{USER}(pp, k, out'_{1,k}, \ldots, out'_{N,k})$

1. **if** $\mathsf{sess}_k \neq \mathsf{await}$ **then return** $\perp$
2. $\mathsf{sess}_k \leftarrow \mathsf{closed}$ // complete session k
3. Retrieve $state'_{0.k}$ from $\mathcal{S}'$
4. $\sigma_{k \oplus b} \leftarrow \mathsf{BMS.User}_2\big(pp, state'_{0,k}, out'_{1,k}, \ldots, out'_{N,k}\big)$
5. **if** $\mathsf{sess}_0 = \mathsf{closed}$ **and** $\mathsf{sess}_1 = \mathsf{closed}$ **then**
6. **if** $\sigma_0 = \perp$ **or** $\sigma_1 = \perp$ **then return** $(\perp, \perp)$
7. **else return** (σ_0, σ_1)
8. **else return** $\perp$

Fig. 7: Blindness experiment for the Blind Multi-Signature (BMS) scheme. init denotes the initialization state, await denotes the waiting state and closed denotes the closed state of a session.

The user then computes the challenge as $\mathbf{c} \leftarrow \mathsf{H}_{\mathsf{sig}}\big(\vec{\mathbf{X}}, \vec{\mathbf{R}}', M\big) + \boldsymbol{\beta}$. Finally, the user generates a ZK-NARK proof π attesting to the consistency of the values $\vec{\mathbf{b}}$ and $\mathbf{c}$ with the ciphertext ct and sends it to all the signers. In particular, the proof establishes that the encrypted values contained in ct are consistent with the corresponding values $\vec{\mathbf{b}}$ and $\mathbf{c}$ generated during the protocol execution. We define the relation R as follows: $\mathrm{R} = \{(\theta, \omega) : \mathsf{PRED}(\theta, \omega) = 1\}$, where the public statement θ and the witness ω are given by

$$\begin{cases} \theta = \big(\vec{\mathbf{X}}, \vec{\mathbf{R}}_1, \ldots, \vec{\mathbf{R}}_\nu, \vec{\mathbf{b}} := (\mathbf{b}_1, \ldots, \mathbf{b}_\nu), \vec{\mathbf{R}}, \mathbf{c}, ct, ek\big) \\ \omega = \big(M, \vec{\boldsymbol{\alpha}}, \boldsymbol{\beta}, \vec{\boldsymbol{\xi}}; \rho\big) \end{cases}$$

and $\mathsf{PRED}(\theta, \omega) = 1$ holds if and only if

$$\begin{cases} b_1 = \mathbf{1} \\ (\vec{\mathbf{b}})_t = \mathsf{Samp}\Big(\mathsf{H}_{\mathsf{non}}\Big(\vec{\mathbf{X}}, \vec{\mathbf{R}}_1, \ldots, \vec{\mathbf{R}}_\nu, M\Big)_t + (\vec{\boldsymbol{\xi}})_t\Big), \forall 2 \le t \le \nu \\ \mathbf{c} = \mathsf{H}_{\mathsf{sig}}\left(\vec{\mathbf{X}}, \underbrace{\mathbf{A}\vec{\boldsymbol{\alpha}} + \boldsymbol{\beta}\vec{\mathbf{X}} + \vec{\mathbf{R}}}_{\vec{\mathbf{R}}'}, M\right) + \boldsymbol{\beta} \\ ct = \mathsf{PKE.Enc}\Big(ek, (M, \vec{\boldsymbol{\alpha}}, \boldsymbol{\beta}, \vec{\boldsymbol{\xi}}); \rho\Big), \end{cases}$$

where $(\vec{\mathbf{b}})_t$, $\mathsf{H}_{\mathsf{non}}(\vec{\mathbf{X}}, \vec{\mathbf{R}}_1, \ldots, \vec{\mathbf{R}}_\nu, M)_t$ and $(\vec{\boldsymbol{\xi}})_t$ denote the t-th entries of the corresponding tuples, for $2 \le t \le \nu$. Upon receiving the proof π, each signer first verifies the proof π. Upon successful verification, i-th signer generates a partial signature $\vec{\mathbf{s}}_i \leftarrow \mathbf{c}\mathbf{a}_i\vec{\mathbf{x}}_i + \sum_{t=1}^{\nu} \mathbf{b}_t\vec{\mathbf{r}}_{i,t}$ and sends $\vec{\mathbf{s}}_i$ to the user. The user aggregates all the partial signatures as $\vec{\mathbf{s}}' \leftarrow \sum_{i=1}^{N} \vec{\mathbf{s}}_i$. The user then generates a NARK proof π_1 for the relation R_1, defined over public parameters pp_{R_1}, statement $\theta_1 = (\mathbf{A}, \vec{\mathbf{R}}')$, and witness $\omega_1 = \vec{\mathbf{s}}' = \vec{\mathbf{s}} + \vec{\boldsymbol{\alpha}}$, such that $\mathrm{R}_1 = \{(pp, \theta_1 := (\mathbf{A}, \vec{\mathbf{R}}'), \omega_1 := (\vec{\mathbf{s}} + \vec{\boldsymbol{\alpha}})) : \mathbf{A}\vec{\mathbf{s}}' = \vec{\mathbf{R}}' + \mathsf{H}_{\mathsf{sig}}(\vec{\mathbf{X}}, \vec{\mathbf{R}}', M)\vec{\mathbf{X}}\}$. Finally, the user outputs the pair $(\vec{\mathbf{R}}', \pi_1)$ as the final signature. An algorithmic description of our proposed scheme is shown in Algorithm 2.

Note that the signature size or the verification time of our proposed blind multi-signature scheme BMuSig-L basically depends on the proof size and the verification time of the underlying NARK respectively. Since, we do not need zero knowledge property for proving the relation R_1, we consider LaBRADOR [9] as the underlying NARK system that incurs $O(\log \log N)$ proof size and $O(N \log N)$ verification time, where N is the size of the witness. Since the witness size of relation R_1 is $m + \kappa$, where m and κ are system parameters, signature size of BMuSig-L is $O(\log \log(m + \kappa))$, which is independent to the number of signers in the group, as shown in Table 1. Moreover, since the zero-knowledge argument system is invoked only during the signature generation process, it does not affect the final signature size or the signature verification time. Hence, one can use a zero knowledge proof system for general circuit to prove relation R.

Algorithm 2 Blind Multi-Signature (BMuSig-L)

$\underline{\mathsf{Setup}(1^\lambda)}$:

1: $\bar{\mathbf{A}} \xleftarrow{\$} \mathcal{R}_q^{\kappa \times m}$
2: $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbb{I}_\kappa]$
3: $\mathsf{H}_{\mathsf{agg}} : \{0,1\}^* \to \mathcal{C}$, $\mathsf{H}_{\mathsf{sig}} : \{0,1\}^* \to \mathcal{C}$, $\mathsf{H}_{\mathsf{non}} : \{0,1\}^* \to \mathcal{C}^{\nu-1}$
4: $sp := (\mathbf{A}, q, \mathsf{H}_{\mathsf{agg}}, \mathsf{H}_{\mathsf{non}}, \mathsf{H}_{\mathsf{sig}})$
5: $(crs, \tau) \leftarrow \mathsf{ZK\text{-}NARK.Setup}(sp)$
6: $crs_1 \leftarrow \mathsf{NARK.Setup}(sp)$
7: $(ek, dk) \leftarrow \mathsf{PKE.KeyGen}(\lambda)$
8: **return** $pp : (crs, crs_1, ek, sp)$

$\underline{\mathsf{User1}(pp, \mathbf{L} = \{\vec{\mathbf{vk}}_1, \ldots, \vec{\mathbf{vk}}_N\}, M)}$:

1: $\vec{\boldsymbol{\alpha}} = (\alpha_1, \ldots, \alpha_m) \xleftarrow{\$} \mathcal{C}^{m+\kappa}$, $\boldsymbol{\beta} \xleftarrow{\$} \mathcal{C}$, $\vec{\boldsymbol{\xi}} = (\xi_{(2)}, \ldots, \xi_{(\nu)}) \xleftarrow{\$} \mathcal{C}^{\nu-1}$ and ρ for encryption scheme randomness
2: $ct \leftarrow \mathsf{PKE.Enc}(ek, (M, \vec{\boldsymbol{\alpha}}, \boldsymbol{\beta}, \vec{\boldsymbol{\xi}}); \rho)$
3: **return** $(out_0, state_0) = (ct, (M, \vec{\boldsymbol{\alpha}}, \boldsymbol{\beta}, \vec{\boldsymbol{\xi}}; \rho))$

$\underline{\mathsf{UserAgg1}(pp, (out_1, \ldots, out_N))}$:

1: From $\{out_i\}_{i \in [N]}$ get $(\vec{\mathbf{R}}_{i,j})_{i=1,j=1}^{i=N,j=\nu}$
2: $\mathbf{L} := \{\vec{\mathbf{vk}}_1, \ldots, \vec{\mathbf{vk}}_N\}$
3: $\vec{\mathbf{X}} \leftarrow \mathsf{KeyAgg}(\mathbf{L})$
4: Compute $\vec{\mathbf{R}}_j = \sum_{i=1}^N \vec{\mathbf{R}}_{i,j}$ where $j = 1$ to ν
5: $out \leftarrow (\vec{\mathbf{R}}_1, \ldots, \vec{\mathbf{R}}_\nu)$
6: **return** out

$\underline{\mathsf{User2}(pp, out, state_0)}$:

1: $out = (\vec{\mathbf{R}}_1, \ldots, \vec{\mathbf{R}}_\nu)$
2: $(\mathbf{r}_{(2)}, \ldots, \mathbf{r}_{(\nu)}) := \mathsf{H}_{\mathsf{non}}(\vec{\mathbf{X}}, \vec{\mathbf{R}}_1, \ldots, \vec{\mathbf{R}}_\nu, M) + \vec{\boldsymbol{\xi}}$
3: $\mathbf{b}_1 := 1$
4: For $j \in [2, \nu]$, $\mathbf{b}_{(j)} := \mathsf{Samp}(\mathbf{r}_j)$
5: Compute $\vec{\mathbf{R}} = \sum_{j=1}^\nu \mathbf{b}_j \vec{\mathbf{R}}_{(j)}$
6: Compute $\vec{\mathbf{R}}' := \mathbf{A}\vec{\alpha} + \boldsymbol{\beta}\vec{\mathbf{X}} + \vec{\mathbf{R}}$
7: $\mathbf{c} := \mathsf{H}_{\mathsf{sig}}(\vec{\mathbf{X}}, \vec{\mathbf{R}}', M) + \boldsymbol{\beta}$
8: $\theta := (\vec{\mathbf{X}}, \vec{\mathbf{R}}_1, \ldots, \vec{\mathbf{R}}_\nu, \vec{\mathbf{b}} = (\mathbf{b}_1, \ldots, \mathbf{b}_\nu), \vec{\mathbf{R}}, \mathbf{c}, ct, ek)$
9: $\omega := (M, \vec{\boldsymbol{\alpha}}, \boldsymbol{\beta}, \vec{\boldsymbol{\xi}}; \rho)$
10: $\pi \leftarrow \mathsf{ZK\text{-}NARK.Prove}(crs, \theta, \omega)$
11: $out'_0 \leftarrow \vec{\mathbf{R}}_1, \ldots, \vec{\mathbf{R}}_\nu, (\mathbf{b}_j)_{j=1}^\nu, \mathbf{c}, \pi$
12: $state'_0 \leftarrow (\omega, \vec{\mathbf{R}}, \mathbf{c}, \vec{\mathbf{X}}, \vec{\mathbf{R}}')$
13: **return** $(out'_0, state'_0)$

$\underline{\mathsf{User}(pp, state'_0, (out'_1, \ldots, out'_N))}$:

1: $\vec{\mathbf{s}} = \sum_{j=1}^N \vec{\mathbf{s}}_j$
2: **if** $\mathbf{A}\vec{\mathbf{s}} \neq \vec{\mathbf{R}} + \mathbf{c}\vec{\mathbf{X}}$ **then**
3: **return** $\perp$
4: **else**
5: $\vec{\mathbf{s}}' \leftarrow \vec{\mathbf{s}} + \vec{\boldsymbol{\alpha}}$
6: $\theta_1 := (pp, \mathbf{A}, \vec{\mathbf{R}}')$
7: $\omega_1 := \vec{\mathbf{s}}'$
8: $\pi_1 \leftarrow \mathsf{NARK.Prove}(crs_1, \theta_1, \omega_1)$
9: **return** $\sigma = (\vec{\mathbf{R}}', \pi_1)$
10: **end if**

$\underline{\mathsf{KeyGen}(pp)}$:

1: Sample $\vec{\mathbf{x}} \xleftarrow{\$} S_\eta^{m+\kappa}$ and compute $\vec{\mathbf{vk}} = \mathbf{A}\vec{\mathbf{x}} \mod q$
2: **return** $(\vec{\mathbf{vk}}, \vec{\mathbf{sk}}) := (\mathbf{A}\vec{\mathbf{x}}, \vec{\mathbf{x}})$

$\underline{\mathsf{KeyAgg}(\mathbf{L})}$:

1: $\mathbf{L} := \{\vec{\mathbf{vk}}_1, \ldots, \vec{\mathbf{vk}}_N\}$
2: **for** $i = 1, \ldots, N$ **do**
3: $\vec{\mathbf{a}}_i \leftarrow \mathsf{H}_{\mathsf{agg}}(\mathbf{L}, \vec{\mathbf{vk}}_i)$
4: **end for**
5: **return** $avk = \vec{\mathbf{X}} = \sum_{i=1}^N \vec{\mathbf{a}}_i \vec{\mathbf{vk}}_i$

$\underline{\mathsf{Sign1}(pp, out_0)}$:

1: Sample $\vec{\mathbf{r}}_{i,1} \leftarrow \mathcal{D}_{\sigma_1}^{m+\kappa}$, $\vec{\mathbf{r}}_{i,2} \ldots, \vec{\mathbf{r}}_{i,\nu} \xleftarrow{\$} \mathcal{D}_{\tilde{\sigma}}^{m+\kappa}$
2: Compute $\vec{\mathbf{R}}_{i,1} = \mathbf{A}\vec{\mathbf{r}}_{i,1}, \ldots, \vec{\mathbf{R}}_{i,\nu} = \mathbf{A}\vec{\mathbf{r}}_{i,\nu}$
3: $out_i \leftarrow (\vec{\mathbf{R}}_{i,1}, \ldots, \vec{\mathbf{R}}_{i,\nu})$
4: $state_i \leftarrow (\vec{\mathbf{r}}_{i,1}, \ldots, \vec{\mathbf{r}}_{i,\nu})$
5: **return** $(out_i, state_i)$

$\underline{\mathsf{Sign2}(pp, sk_i, state_i, out_0, out'_0)}$:

1: $out'_0 = \vec{\mathbf{R}}_1, \ldots, \vec{\mathbf{R}}_\nu, \vec{\mathbf{b}} = (\mathbf{b}_1 \ldots, \mathbf{b}_\nu), \mathbf{c}, \pi$
2: $\vec{\mathbf{a}}_i \leftarrow \mathsf{H}_{\mathsf{agg}}(\mathbf{L}, \vec{\mathbf{vk}}_i)$
3: $\vec{\mathbf{X}} \leftarrow \mathsf{KeyAgg}(\mathbf{L})$
4: Compute $\vec{\mathbf{R}} = \sum_{j=1}^\nu \mathbf{b}_j \vec{\mathbf{R}}_j$
5: $\theta \leftarrow (\vec{\mathbf{X}}, \vec{\mathbf{R}}_1, \ldots, \vec{\mathbf{R}}_\nu, (\mathbf{b}_1 \ldots, \mathbf{b}_\nu), \vec{\mathbf{R}}, \mathbf{c}, ct)$
6: **if** $\mathsf{ZK\text{-}NARK.Verify}(crs, \theta, \pi) = 0$ **then**
7: **return** $\vec{\mathbf{s}}_i = \perp$
8: **else**
9: compute $\vec{\mathbf{v}}_i = \mathbf{c}\mathbf{a}_i\vec{\mathbf{x}}_i$
10: compute $\vec{\mathbf{s}}_i = \vec{\mathbf{v}}_i + \sum_{t=1}^\nu \mathbf{b}_t \vec{\mathbf{r}}_{i,t}$
11: **end if**
12: **if** $\mathsf{RejSamp}(\vec{\mathbf{v}}_i, \vec{\mathbf{s}}_i, (\mathbf{b}_j)_{j=1}^\nu) = 0$ **then**
13: **return** $\vec{\mathbf{s}}_i = \perp$
14: **else**
15: **return** $\vec{\mathbf{s}}_i$
16: **end if**
17: **return** $out'_i \leftarrow \vec{\mathbf{s}}_\mathbf{i}$

$\underline{\mathsf{Verify}(pp, \vec{\mathbf{X}}, M, \sigma)}$:

1: Compute $\vec{\mathbf{c}}' \leftarrow \mathsf{H}_{\mathsf{sig}}(\vec{\mathbf{X}}, \vec{\mathbf{R}}', M)$
2: $b \leftarrow \mathsf{NARK.Verify}(crs_1, \theta_1, \pi_1)$
3: **return** b

$\underline{\mathsf{Samp}(\mathbf{r})}$:

1: Sample $\mathbf{b} \leftarrow \mathcal{D}_{\sigma_b}$ with randomness r
2: **return** b

$\underline{\mathsf{RejSamp}(\vec{\mathbf{v}}, \vec{\mathbf{s}}, (\mathbf{b}_j)_{j=1}^\nu)}$

1: $\Sigma := \sigma_1^2 + \tilde{\sigma}^2 \cdot \sum_{i=2}^\nu (\mathbf{b}_j^* \mathbf{b}_j) \cdot \mathbb{I}_{m+\kappa}$
2: $\gamma \xleftarrow{\$} [0, 1]$
3: **if** $\gamma \geq \min\left(\frac{\mathcal{D}^{m+\kappa}(\vec{\mathbf{s}})}{1, K \cdot \mathcal{D}^{m+\kappa}_{\sqrt{\Sigma}, \vec{\mathbf{v}}}(\vec{\mathbf{s}})}\right)$ **then**
4: **return** 0
5: **else**
6: **return** 1
7: **end if**

5 Security Results of BMuSig-L

Theorem 2 (Unforgeability of BMuSig-L). *Let* PKE *be a public-key encryption scheme,* MuSig-L *be the underlying lattice-based multi-signature scheme,* ZK-NARK *be a non-interactive argument of knowledge* [7] *for the relation* R *used in the blind signing protocol, and* NARK *be a non-interactive argument of knowledge for the underlying signing relation* R_1*. For any PPT adversary* A *that wins the* OMUF *game against* BMuSig-L *after completing at most* q_s *signing sessions, there exist PPT adversaries* U, S, *and* S′ *such that* U *breaks the* EUF-CMA *security of* MuSig-L, S *breaks the soundness of* ZK-NARK, *and* S′ *breaks the knowledge soundness of* NARK*. In other words,*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{OMUF}}_{\mathsf{BMuSig\text{-}L},\mathsf{A}}(\lambda) \leq & \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{MuSig\text{-}L},\mathsf{U}}(\lambda) \\ & + \mathsf{Adv}^{\mathsf{SND}}_{\mathsf{ZK\text{-}NARK}[\mathrm{R}],\mathsf{S}}(\lambda) + \mathsf{Adv}^{\mathsf{KS}}_{\mathsf{NARK}[\mathrm{R}_1],\mathsf{S}'}(\lambda).\end{aligned}$$

We give here a proof overview, details of the proof is given in Appendix E.

Proof Overview. We prove the unforgeability of BMuSig-L through three games. In G_1, the challenger generates and stores the PKE key pair (ek, dk). Using dk, the SIGN1 oracle decrypts each ciphertext and recovers the tuple $(M, \vec{\boldsymbol{\alpha}}, \boldsymbol{\beta}, \vec{\boldsymbol{\xi}})$. We use the soundness of ZK-NARK to show that an accepting proof must correspond to a valid statement with a consistent witness. Otherwise, the reduction obtains a soundness forgery against ZK-NARK. In G_2, we simulate the blind signing sessions using the signing oracle of the underlying MuSig-L scheme. The blinding values $\vec{\boldsymbol{\alpha}}$ and $\boldsymbol{\beta}$ are absorbed into the commitments and $\vec{\boldsymbol{\xi}}$ is absorbed in signing responses. Since the commitments of MuSig-L are uniformly distributed, this transformation does not change their distribution. Hence, the adversary's views in G_1 and G_2 are identically distributed. Finally, for every accepting forgery in G_2, the knowledge extractor for NARK extracts a valid signing witness for the underlying relation R_1. Thus, except with negligible probability, every valid blind signature yields a valid signature for MuSig-L. Since the adversary produces more valid forgeries than the number q_s of completed signing sessions, at least one extracted signature is not among the signatures obtained from the signing oracle. This signature gives an EUF-CMA forgery against MuSig-L. A simple corollary follows from Theorem 2 and Theorem 1 as follows:

Theorem 3 (Unforgeability of BMuSig-L based on MSIS and MLWE). *Let* PKE *be a public-key encryption scheme, and* $q = 5 \mod 8$*. Let* ZK-NARK *be a non-interactive argument of knowledge for the relation* R *used in the blind signing protocol, and* NARK *be a non-interactive argument of knowledge for the underlying signing relation* R_1*. For any PPT adversary* A *that wins the* OMUF *game against* BMuSig-L *after completing at most* q_s *signing sessions, and* q_h

[7] A formal definition of (zero-knowledge) Non-interactive argument is given in Appendix 2.2

random oracle queries, there exist PPT adversaries B, C, S, *and* S' *such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{OMUF}}_{\mathsf{BMuSig\text{-}L},\mathsf{A}}(\lambda) \leq & 2(q_h+q_s)^2 \cdot \left(\frac{1+2^{-\Omega(n)}}{q^{\kappa n}}\right)^{\nu} + \frac{(2q_h+q_s)^2}{\sigma_{\rho_b}(\mathcal{R})} \\ & + e(q_s+1)\left(q_s\epsilon_s + \frac{q_h(2q_h+3)}{|\mathcal{C}|} + \frac{2^{\kappa+1}}{q^{\kappa n/2}} + \right. \\ & \left. \mathrm{Adv}^{\mathsf{MLWE}}_{q,\kappa,m,\eta,\mathsf{B}} \sqrt{\frac{q_h^2}{|\mathcal{C}|} + q_h\sqrt{q_h \cdot \mathsf{Adv}^{\mathsf{MSIS}}_{q,\kappa,m+1,\beta,\mathsf{C}}}}\right), \\ & + \mathsf{Adv}^{\mathsf{SND}}_{\mathsf{ZK\text{-}NARK}[\mathrm{R}],\mathsf{S}}(\lambda) + \mathsf{Adv}^{\mathsf{KS}}_{\mathsf{NARK}[\mathrm{R}_1],\mathsf{S}'}(\lambda).\end{aligned}$$

where recall that $\mathcal{C}$ *is the challenge space and* $\beta = 8\kappa_1\sqrt{B_N^2+\kappa_1^3}$ *where* $B_N = \sqrt{N}B$, N *is the number of signers in the group and* $B = \sigma_1\sqrt{n(m+\kappa)}$. *Moreover,* ϵ_s *is the statistical distance of the simulator and the original honest party protocols. Remaining parameters are described in Table 2.*

Theorem 4. *Assume that* ZK-NARK *is computationally zero-knowledge,* NARK *is knowledge-sound, and* PKE *is* IND-CPA *secure. For every PPT adversary* A, *there exist PPT adversaries* $\mathsf{Z}_0, \mathsf{Z}_1, \mathsf{C}_0, \mathsf{C}_1$, *and* S' *such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{Blind}}_{\mathsf{BMuSig\text{-}L},\mathsf{A}}(\lambda) \leq & \ \mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{ZK\text{-}NARK}[\mathrm{R}],\mathsf{Z}_0}(\lambda) + \mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{ZK\text{-}NARK}[\mathrm{R}],\mathsf{Z}_1}(\lambda) \\ & + \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE},\mathsf{C}_0}(\lambda) + \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE},\mathsf{C}_1}(\lambda) + \mathsf{Adv}^{\mathsf{KS}}_{\mathsf{NARK}[\mathrm{R}_1],\mathsf{S}'}(\lambda).\end{aligned}$$

We give here a proof overview, details of the proof is given in Appendix F.

Proof Overview. We prove the blindness of BMuSig-L through a sequence of games. Game G_0 is the original blindness game, where the two concurrent signing sessions are executed on M_b and M_{1-b}, respectively. In G_1, we replace all proofs generated during the signing sessions with simulated proofs. By the computational zero-knowledge property of ZK-NARK, G_0 and G_1 are computationally indistinguishable. In G_2, we replace the encrypted tuple containing the message and blinding values with an encryption of a fixed public value. The simulated proofs remain unchanged. By the IND-CPA security of PKE, G_1 and G_2 are computationally indistinguishable. In the final game G_2, the ciphertext and simulated proof do not depend on the challenge bit. By knowledge soundness, an accepting proof gives a valid witness except with negligible probability. Thus, the same transcript can be explained for either M_0 or M_1. Therefore, $G_2^{\mathsf{A},0}$ and $G_2^{\mathsf{A},1}$ are identically distributed up to the knowledge-soundness error.

Discussion. It is evident from Theorem 5.2 that the unforgeability of BMuSig-L incurs quartic security loss. To the best of our knowledge, MuSig-L is currently the best lattice-based multisignature scheme despite incurring a quartic security loss. Existing lattice-based multisignature schemes are either inefficient or also suffer from quartic security loss due to the use of the double-forking technique. Although several multisignature schemes (e.g., Chopsticks [46], Toothpick-I [47], and Toothpick-II [47]) avoid double forking and achieve tighter security reductions, they rely on classical assumptions such as DDH. Moreover, these constructions fundamentally require equivocable commitment schemes, which appear to

be difficult to instantiate in the lattice setting. Consequently, no lattice-based multisignature scheme with a tight security reduction is currently known. Therefore, designing such a scheme remains an important open research problem. In our construction, the quartic security loss is inherited from MuSig-L and is therefore unavoidable.

6 Implementation Benchmarks

We report measured sizes, timings and failure rates for a reference implementation of BMuSig-L, a lattice-based blind multi-signature. Appendix G gives the sampling detail, the full measurement tables, scaling in the ring degree, the measurement platform.

6.1 Parameters

Table 3 gives the instantiated set. Since $q \equiv 5 \pmod 8$, the polynomial $X^n + 1$ splits modulo q into exactly two irreducible factors of degree $n/2$, so every short non-zero element of $\mathcal{R}_q$ is invertible. This rules out a fully split ring, and hence a complete negacyclic NTT, which would need $q \equiv 1 \pmod{2n}$ and therefore $q \equiv 1 \pmod 8$; all polynomial multiplication below is schoolbook $O(n^2)$. The bound B_N occupies 74% of $q/2$. That is the binding constraint on σ_1: the norm bound must fit inside the modulus, or it wraps and certifies nothing.

Symbol	Description	Value
n	ring degree	256
q	modulus, $q \equiv 5 \pmod 8$	1 073 741 789
w	$\lceil \log_2 q \rceil$	30
κ	rows of $\bar{\mathbf{A}}$	6
m	columns of $\mathbf{A}$	6
$m + \kappa$	witness dimension	12
κ_1	challenge weight	60
$\lvert\mathcal{C}\rvert$	challenge space	$\approx 2^{257}$
η	secret bound	2
ν	nonces per signer	4
σ_1	first-nonce width	1.500×10^6
$\tilde{\sigma}$	remaining-nonce width	1.000×10^3
σ_b	combiner width	2
K	rejection constant	1.06
B	$\sigma_1\sqrt{n(m+\kappa)}$	8.314×10^7
B_N	bound checked by VERIFY	3.991×10^8
$q/2$		5.369×10^8

Table 3: Instantiated parameters.

Object	Raw	Packed	Object	Bytes
Signature $\sigma = (\vec{\mathbf{R}}', \pi_1)$			*Keys and preprocessing*	
$\vec{\mathbf{R}}'$, κ polys, $w = 30$ b	6144	5760	public key $\vec{\mathbf{pk}}$	6144
π_1, $m + \kappa$ polys, 25 b	12 288	9600	secret key $\vec{\mathbf{x}}$	12 288
total	18 432	15 360	ciphertext ct	33 792
Per-session traffic				
round 1, per signer	24 576			
round 2, user to signers	29 696			
partial signature $\vec{\mathbf{s}}_i$	12 288			

Table 4: Sizes in bytes, $n = 256$, $N = 2$ signers. π_1 is the placeholder NARK, which carries $\vec{\mathbf{s}}'$ in the clear.

Phase	Time	Phase	Time
KEYGEN (per signer)	4.2 ms	SIGN1 rd. 2 (per signer)	7.3 ms
KEYAGG	0.7 ms	USER finalise	11.2 ms
SIGN1 rd. 1 (per signer)	21.7 ms	VERIFY	5.6 ms
USER round 2	13.9 ms		

Table 5: Mean over 20 sessions, $N = 2$. Platform and its limitations in Appendix G.4.

6.2 Sizes and Performances

In the packed column $\vec{\mathbf{R}}'$ is uniform modulo q and therefore incompressible at w bits per coefficient, while $\vec{\mathbf{s}}'$ is Gaussian and needs only $\lceil \log_2(2 \cdot 4\sqrt{N}\sigma_1) \rceil = 25$ at $N = 2$. Round 1 dominates the timings, but not because of sampling: it draws $\nu(m + \kappa)n$ discrete Gaussian coefficients at 201 ns each (Appendix G.1), which is 2.0 ms of the 21.7 ms, and spends the remaining 20.1 ms on the ν products $\bar{\mathbf{A}}\vec{\mathbf{r}}_{i,j}$. The cost is 91% arithmetic, which is precisely why the absence of an NTT (Section 6.1) is the binding performance constraint. The signature is nearly flat in the number of signers: only the width of $\vec{\mathbf{s}}'$ grows, and only as $\log \sqrt{N}$, from 24 bits at $N = 1$ to 27 at $N = 64$. Going from one signer to sixty-four costs 1152 B, under 8%.

Rejection sampling. RejSamp accepts with probability $1/K$, measured at 0.9435 against a predicted 0.9434. Acceptance collapses as $\alpha = \sigma_1/\|\vec{\mathbf{v}}\|_2$ falls — from 0.94 at $\alpha \approx 1.9 \times 10^4$ to 0.51 at $\alpha \approx 0.8$ — confirming the sampler responds to $\langle \vec{\mathbf{s}}, \vec{\mathbf{v}} \rangle$ rather than accepting unconditionally. Session failure compounds as $K_N = K^N$: completion falls from 95.5% at $N = 1$ to 59.5% at $N = 8$, and $\approx 39\%$ at $N = 16$, i.e. ≈ 2.5 full round trips per signature. Every restart is a fresh round 1 *and* round 2 to every participant, so from Table 5 one attempt costs $N(21.7 + 7.3) + 25.1$ ms of signer and user work; measured end to end, including key aggregation, the ciphertext and verification, this is $\approx (29N + 40)$ ms, and the expected cost of a signature is that times K_N. Appendix G.2 gives the full tables.

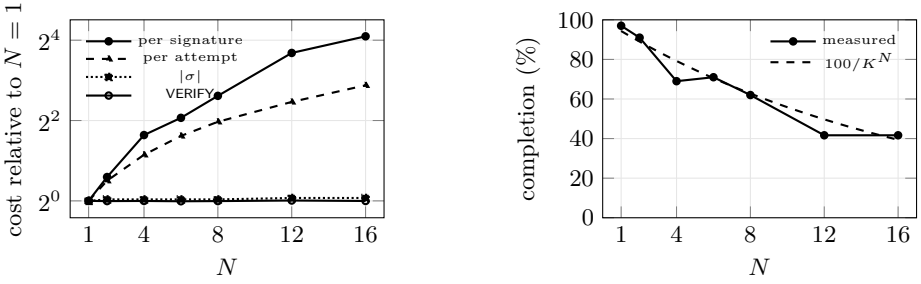

Fig. 8: Scaling in the number of signers. Left: signature size and verification are flat, per-attempt cost is linear, and expected cost per successful signature is their product. Right: session completion follows $K_N = K^N$. These points are a separate, smaller experiment from Table 10 — 100 sessions per point, 60 for $N = 12, 16$ against that table's 200 — so they carry several points of sampling error and the two need not agree exactly; the inversion between $N = 4$ and $N = 6$ is that error, not structure.

Scheme	**Primitive**	**Signers**	$\lvert\sigma\rvert$
Hauck et al. [32]	blind sig.	1	$\approx 7.73\,\mathrm{MB}$
Lyubashevsky et al. [40]	blind sig.	1	$\approx 150\,\mathrm{kB}$
AKSY [2]	blind sig.	1	$45.19\,\mathrm{kB}$
MuSig-L [14]	multi-sig.	N	—
BMRaccoon [54]	*blind* multi-sig.	64	$68.56\,\mathrm{kB}$
BMuSig-L	*blind* multi-sig.	N	$15.0\,\mathrm{KiB}^{\dagger}$

Table 6: Lattice-based blind and multi-signature schemes. † Not like-for-like: our π_1 is a placeholder, so this is $(\vec{\mathbf{R}}', \vec{\mathbf{s}}')$ *without* a zero-knowledge argument.

6.3 Scaling in the Number of Signers

Figure 8 separates the quantities that key aggregation keeps constant from the one that compounds. The signature grows from 14 976 B at $N = 1$ to 15 744 B at $N = 16$, some 5%: only the width of $\vec{\mathbf{s}}'$ moves, and only as $\log \sqrt{N}$. Verification is flat at 5.5 ms — the verifier sees one aggregated key and one response whatever N is — and so is the round-2 message, since the user sends the aggregated $\vec{\mathbf{R}}_j$ rather than per-signer nonces. What does grow is the session. Every signer runs both rounds, so the cost of one attempt is linear in N (69 ms to 509 ms from $N = 1$ to 16, a slope of 29 ms per signer that matches the two SIGN1 rows of Table 5), and completion decays as $K_N = K^N$. The product is the expected cost of one *successful* signature, which rises from 71 ms to 1.22 s, a factor of 17 across a 16× increase in N.

Instance	κ, m	$\log_2 q$	$\log_2 2B_N$	δ
Dilithium2	4	23	24.0	1.004247
ranks 4 (rejected)	4	30	29.3	1.004848
as implemented	6	30	29.6	**1.003295**
ranks 8	8	30	29.8	1.002504

Table 7: Root Hermite factor an MSIS attacker must reach, from $\log_2 \beta = 2\sqrt{\kappa n \cdot \log_2 q \cdot \log_2 \delta}$ with $\beta = 2B_N$, where κn is the lattice dimension, not the module rank κ. Smaller is harder.

6.4 Comparison and Security Positioning

AKSY's 45.19 kB is a complete signature: of that, the proof alone accounts for 37.18 kB. Ours contains no argument at all. Instantiating π_1 at comparable cost would give $\approx 5.6\,\text{kB} + 37\,\text{kB} \approx 43\,\text{kB}$, the same order as AKSY but carrying N signers rather than one. That, not the 15.0 KiB, is the figure to take away until a proof system is chosen. AKSY also report an expected 10.4 repetitions of their argument against our $K = 1.06$; the comparison flatters us, since they are round-optimal and their repetitions are local, whereas ours cost a full round trip to every signer.

At module ranks 4 the instance needs $\delta = 1.00485$ against Dilithium2's 1.00425 — a larger δ, hence an easier problem — and MLWE agrees: at identical $\eta = 2$, moving the modulus from 2^{23} to 2^{30} widens $\log_2(q/\sigma)$ from 22.5 to 29.5, making key recovery easier. Both axes placed the original set below Dilithium2. Raising the ranks to 6 brings δ to 1.00330, so MSIS is now the harder of the two, at the cost of 50% in signature size and roughly doubled signing time. Ranks 8 would reach 1.00250, but B_N then takes 86% of $q/2$. This is a *relative* statement, not a bit-security claim. Note also the three-way tension: security wants a smaller q, the norm bound wants $q > 2B_N$, and acceptance wants a large σ_1, which forces B_N up.

7 Conclusion

In this paper, we have proposed the first lattice-based two-round blind multi-signature construction that achieves the key aggregation property. We have proven that our construction is unforgeable, blind and concurrent secure in the plain public-key model under the standard lattice assumptions, namely the MSIS and MLWE problems, along with the soundness property of the underlying non-interactive zero-knowledge protocol. However, the security bound of our construction is not tight due to the double-forking technique employed in the underlying multi-signature scheme MuSig-L. In particular, our proposed construction suffers from a *quartic* security loss in the reduction. Similarly, the security bound of BMuSig2 is also not tight due to the double-forking technique used in its underlying multi-signature scheme MuSig2. Therefore, designing a two-

round classically secure or post-quantum secure blind multi-signature scheme that achieves a tight security reduction remains an important open problem.

References

1. Masayuki Abe. A secure three-move blind signature scheme for polynomially many signatures. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT '01, page 136–151, Berlin, Heidelberg, 2001. Springer-Verlag.
2. Shweta Agrawal, Elena Kirshanova, Damien Stehlé, and Anshu Yadav. Practical, round-optimal lattice-based blind signatures. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 39–53, New York, NY, USA, 2022. Association for Computing Machinery.
3. Shweta Agrawal, Damien Stehlé, and Anshu Yadav. Round-optimal lattice-based threshold signatures, revisited. In *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.
4. Ali Bagherzandi, Jung Hee Cheon, and Stanislaw Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, October 27-31, 2008*, pages 449–458. ACM, 2008.
5. 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, *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, October 30 - November 3, 2006*, pages 390–399. ACM, 2006.
6. Rikke Bendlin, Sara Krehbiel, and Chris Peikert. How to share a lattice trapdoor: Threshold protocols for signatures and (h)ibe. *IACR Cryptol. ePrint Arch.*, 2013:419, 2013.
7. Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. On the (in)security of ROS. *Journal of Cryptology*, 35(4):25, 2022.
8. Ward Beullens, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Lattice-based blind signatures: Short, efficient, and round-optimal. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, CCS '23, page 16–29, New York, NY, USA, 2023. Association for Computing Machinery.
9. Ward Beullens and Gregor Seiler. LaBRADOR: Compact proofs for R1CS from module-SIS. Cryptology ePrint Archive, Paper 2022/1341, 2022.
10. Kanchan Bisht, Keerthi Aiswarya Varshini, Shivam Sethi, Maria Francis, and R. Kabaleeshwaran. Bmusig2: Schnorr-compatible blind multi-signatures. In Manindra Agrawal, Ian Molloy, Vinayaka Pandit, Debdeep Mukhopadhyay, and Kenny Paterson, editors, *Proceedings of the ACM Asia Conference on Computer and Communications Security, ASIA CCS 2026, Bangalore, India, June 1-5, 2026*, pages 326–342. ACM, 2026.
11. Dan Boneh, Rosario Gennaro, Steven Goldfeder, and Sam Kim. Threshold cryptosystems from threshold fully homomorphic encryption. In *Advances in Cryptology – CRYPTO 2018*, volume 10991 of *Lecture Notes in Computer Science*, pages 565–596. Springer, Heidelberg, 2018.

12. Dan Boneh and Sam Kim. One-time and interactive aggregate signatures from lattices. 2020.
13. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. ASIACRYPT '01, page 514–532, Berlin, Heidelberg, 2001. Springer-Verlag.
14. 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, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 276–305. Springer, 2022.
15. Katharina Boudgoust and Adeline Roux-Langlois. Overfull: Too large aggregate signatures based on lattices. *The Computer Journal*, 67(2):719–727, 02 2024.
16. David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982*, pages 199–203. Plenum Press, New York, 1982.
17. David Chaum. Blind signature system. In David Chaum, editor, *Advances in Cryptology, Proceedings of CRYPTO '83, Santa Barbara, California, USA, August 21-24, 1983*, page 153. Plenum Press, New York, 1983.
18. David Chaum. Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. In Christoph G. Günther, editor, *Advances in Cryptology - EUROCRYPT '88, Workshop on the Theory and Application of of Cryptographic Techniques, Davos, Switzerland, May 25-27, 1988, Proceedings*, volume 330 of *Lecture Notes in Computer Science*, pages 177–182. Springer, 1988.
19. David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer, 1991.
20. Elizabeth C. Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. Snowblind: A threshold blind signature in pairing-free groups. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part I*, volume 14081 of *Lecture Notes in Computer Science*, pages 710–742. Springer, 2023.
21. Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. *Two-Round n-out-of-n and Multi-signatures and Trapdoor Commitment from Lattices*, pages 99–130. 05 2021.
22. Rafael del Pino and Shuichi Katsumata. A new framework for more efficient round-optimal lattice-based (partially) blind signature via trapdoor sampling. In *Advances in Cryptology – CRYPTO 2022: 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15–18, 2022, Proceedings, Part II*, page 306–336, Berlin, Heidelberg, 2022. Springer-Verlag.
23. Rafaël del Pino, Shuichi Katsumata, Thomas Prest, and Mélissa Rossi. Raccoon: A masking-friendly signature proven in the probing model. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part I*, volume 14920 of *Lecture Notes in Computer Science*, pages 409–444. Springer, 2024.

24. Rafael del Pino, Vadim Lyubashevsky, and Gregor Seiler. Lattice-based blind signatures and anonymous credentials from standard assumptions. In *EUROCRYPT*, 2019.
25. Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018.
26. Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. CRYPTO'06, page 60–77, Berlin, Heidelberg, 2006. Springer-Verlag.
27. Nils Fleischhacker, Mark Simkin, and Zhenfei Zhang. Squirrel: Efficient synchronized multi-signatures from lattices. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 1109–1123, New York, NY, USA, 2022. Association for Computing Machinery.
28. Georg Fuchsbauer and Mathias Wolf. Concurrently secure blind schnorr signatures. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part II*, volume 14652 of *Lecture Notes in Computer Science*, pages 124–160. Springer, 2024.
29. Sanjam Garg, Vanishree Rao, Amit Sahai, Dominique Schröder, and Dominique Unruh. *Round Optimal Blind Signatures*, volume 6841, pages 630–648. 08 2011.
30. François Garillot, Yashvanth Kondi, Payman Mohassel, and Valeria Nikolaenko. *Threshold Schnorr with Stateless Deterministic Signing from Standard Assumptions*, pages 127–156. 08 2021.
31. Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 20(1):51–83, 2007.
32. Eduard Hauck, Eike Kiltz, Julian Loss, and Ngoc Khanh Nguyen. Lattice-based blind signatures, revisited. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 500–529. Springer, 2020.
33. Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. Tumblebit: An untrusted bitcoin-compatible anonymous payment hub. In *24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017*. The Internet Society, 2017.
34. Ethan Heilman, Foteini Baldimtsi, and Sharon Goldberg. Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions. In Jeremy Clark, Sarah Meiklejohn, Peter Y. A. Ryan, Dan S. Wallach, Michael Brenner, and Kurt Rohloff, editors, *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*, volume 9604 of *Lecture Notes in Computer Science*, pages 43–60. Springer, 2016.
35. Scott Hendrickson, Jana Iyengar, Tommy Pauly, Steven Valder, and Christopher A Wood. Privacy access tokens. In *Technical Report. Internet-Draft draft-private-access-tokens-01, Internet Engineering Task Force*, 2021.

36. Ioanna Karantaidou, Omar Renawi, Foteini Baldimtsi, Nikolaos Kamarinakis, Jonathan Katz, and Julian Loss. Blind multisignatures for anonymous tokens with decentralized issuance. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, pages 1508–1522. ACM, 2024.
37. Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Round-optimal blind signatures in the plain model from classical and quantum standard assumptions. *IACR Cryptol. ePrint Arch.*, 2021:306, 2021.
38. Shuichi Katsumata, Michael Reichle, and Yusuke Sakai. Practical round-optimal blind signatures in the rom from standard assumptions. In *Advances in Cryptology – ASIACRYPT 2023: 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4–8, 2023, Proceedings, Part II*, page 383–417, Berlin, Heidelberg, 2023. Springer-Verlag.
39. Chelsea Komlo and Ian Goldberg. *FROST: Flexible Round-Optimized Schnorr Threshold Signatures*, pages 34–65. 07 2021.
40. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 71–101. Springer, 2022.
41. Changshe Ma, Jian Weng, Yingjiu Li, and Robert H. Deng. Efficient discrete logarithm based multi-signature scheme in the plain public key model. *Des. Codes Cryptogr.*, 54(2):121–133, 2010.
42. Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple schnorr multi-signatures with applications to bitcoin. *Des. Codes Cryptogr.*, 87(9):2139–2164, 2019.
43. National Institute of Standards and Technology. Post-quantum cryptography, 2026. Accessed: November 14, 2026.
44. Gregory Neven. Ibm identity mixer (idemix). In *NIST Meeting on Privacy Enhanching Technology, Zurich, Switzerland, December 8-9, 2011*, 2011.
45. Jonas Nick, Tim Ruffing, and Yannick Seurin. Musig2: Simple two-round schnorr multi-signatures. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021*, pages 189–221, Cham, 2021. Springer International Publishing.
46. Jiaxin Pan and Benedikt Wagner. Chopsticks: Fork-free two-round multi-signatures from non-interactive assumptions. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 597–627. Springer, 2023.
47. Jiaxin Pan and Benedikt Wagner. Toothpicks: More efficient fork-free two-round multi-signatures. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part I*, volume 14651 of *Lecture Notes in Computer Science*, pages 460–489. Springer, 2024.
48. Christian Paquin. U-prove technology overview v1.1. In *Microsoft Corporation Draft Revision 1*, 2011.
49. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *J. Cryptol.*, 13(3):361–396, January 2000.

50. Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyuba-shevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon. In *Technical Report. National Institute of Standards and Technology*, 2020.
51. Markus Rückert. Lattice-based blind signatures. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010*, pages 413–430, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
52. Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.
53. Douglas R. Stinson and Reto Strobl. Provably secure distributed schnorr signatures and a (t, n) threshold scheme for implicit certificates. In *Information Security and Privacy (ACISP 2001)*, volume 2119 of *Lecture Notes in Computer Science*, pages 417–434, Heidelberg, 2001. Springer.
54. Keerthi Aiswarya Varshini, Kanchan Bisht, Palkesh Jain, Maria Francis, and R. Kabaleeshwaran. BMRaccoon: Blind multi-signatures from lattices. Cryptology ePrint Archive, Paper 2026/2870, 2026.
55. Chenzhi Zhu and Stefano Tessaro. The algebraic one-more MISIS problem and applications to threshold signatures. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part I*, volume 16000 of *Lecture Notes in Computer Science*, pages 548–581. Springer, 2025.

A Open Science Appendix

The reference implementation, the test suite and the measurement data behind every figure and table in this paper are available, anonymised for review, at

`https://anonymous.4open.science/r/BMUSIGL-537C/`

B Generative AI Usage

We utilized Claude Sonnet 4.6 (Anthropic) strictly to assist in drafting an initial code template. The authors independently formulated all algorithmic logic, conducted thorough code reviews, and implemented extensive structural modifications. Additionally, all unit tests, experimental evaluations, and reported results were developed and executed exclusively by the authors.

C Related Work

C.1 Blind Signature

Blind signature was introduced by Chaum [17], enabling a user to obtain a signature on a message without revealing the message itself to the signer. Following Chaum's seminal work, a large amount of research has focused on constructing

efficient blind signature schemes based on RSA, discrete logarithm, and pairing-based assumptions. Pointcheval and Stern [49] formalized the fundamental security notions of blindness and one-more unforgeability and provided rigorous security analyses for blind signature schemes. Subsequently, Schnorr-based blind signatures attracted considerable attention due to their efficiency and compatibility with existing public-key infrastructures [1].

With the advent of post-quantum cryptography, significant effort has been devoted to constructing blind signatures from lattice assumptions. Rückert [51] proposed one of the earliest lattice-based blind signature schemes based on the hardness of the SIS problem. Subsequently, several works improved both efficiency and security by employing lattice trapdoors and rejection-sampling techniques [24].

On the other hand, Fischlin's generic framework [26] also inspired the design of round-optimal blind signature protocols, reducing the interaction required by traditional three-round Schnorr-type constructions. [37,29,38] Among recent lattice-based constructions, the scheme of [22] achieves security from standard lattice assumptions and produces signatures of approximately 100 KB. The scheme of [2] further reduces the signature size to about 45 KB and lowers the communication complexity, albeit under the non-standard one-more-ISIS assumption. More recently, Beullens *et al.* [8] significantly improved the state of the art by constructing a blind signature scheme from standard lattice assumptions with substantially smaller signatures. The same work also introduced a keyed-verification blind signature scheme whose signatures are only 48 bytes long.

C.2 Multi-Signature

Multi-signatures belong to a broader family of signature schemes that support compact signature generation, similar to aggregate and threshold signatures. Several works have studied Schnorr-style threshold signatures [31,30,39,53], whose techniques are closely related to those used in multi-signature schemes. In particular, FROST [39] employs a random linear-combination technique, reminiscent of MuSig2, to achieve a two-round threshold signing protocol. Threshold signatures can also be instantiated from lattice assumptions. However, existing T-out-of-N lattice-based constructions require either threshold secret sharing of the signing key of a GPV signature scheme [6] or fully homomorphic encryption (FHE) [11,3]. The lattice-based multi-signature scheme of [21] can also be viewed as an N-out-of-N threshold signature scheme, and the authors showed that essentially the same techniques apply in both security models. The landscape of lattice-based multi-signatures is similar. Boneh and Kim [12] proposed a three round construction that requires interactive aggregation, closely following the approach of the Bellare-Neven Schnorr-based aggregate signature scheme. More recently, Boudgoust and Roux-Langlois [15] constructed an aggregate signature scheme that eliminates interaction among the signers. However, its asymptotic signature size grows linearly with the number of signers. Fleischhacker, Simkin and Zhang [27] proposed lattice-based multi-signature scheme

Squirrel, which was provably secure against rogue key attacks in the classical ROM and under the standard lattice assumptions. Their scheme combines the one time signature [12] with their improved homomorphic Merkle tree commitment. Concurrently, Boschini et al. proposed MuSig-L [14], a lattice-based multi-signature scheme based on the Fiat-Shamir with aborts paradigm. For the comparison of these two lattice-based multi-signature schemes, we refer to [14, Section 1.3]. In this work we have used the framework of MuSig-L.

C.3 Blind Multi-Signature

To the best of our knowledge, only three BMS schemes have been proposed in the literature, namely BM_BLS, BM_SB, and BMuSig2. The first two schemes were introduced by Karantaidou et al. [36]. Specifically, BM_BLS is a pairing-based construction derived from the BLS signature scheme [13], while BM_SB is a pairing-free blind multi-signature scheme obtained from the Snowblind threshold blind signature protocol [20]. The latter requires three rounds of interaction between the user and the signer.

Recently, Bisht et al. [10] have proposed BMuSig2, a pairing-free BMS scheme based on the MuSig2 [45] framework. Their construction requires the interaction complexity to two rounds in four moves as each round requires two move from user to signer and then from signer to user.

However, all existing BMSschemes rely on the hardness of the discrete logarithm problem or its variants. Consequently, their security is vulnerable to quantum adversaries due to Shor's algorithm [52]. To the best of our knowledge, no BMS scheme based on post-quantum assumptions has been proposed so far. This gap motivates the study of lattice-based blind multi-signatures that can provide security against quantum attacks while preserving the privacy and functionality guarantees of classical BMSschemes.

D IND-CPA Secure Public Key Encryption.

Let $\Pi = (\mathsf{KGEN}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption algorithm that consists of three probabilistic polynomial time algorithms: KEYGEN takes 1^λ as input, and outputs a pair of public key and private key (ek, dk). Enc is a probabilistic polynomial time algorithm that takes a message M and encrypts it with the public key ek along with some randomness ρ and outputs a ciphertex. Dec is a deterministic polynomial time algorithm that takes a ciphertext and decrypts it with the secret key dk to obtain a corresponding plaintext ct.

Definition 3. *Let $\Pi = (\mathsf{KGEN}, \mathsf{Enc}, \mathsf{Dec})$ be a Public Key Encryption scheme. Let* A *be a PPT algorithm that has access to the public key ek, and thus can make finite number of encryption queries. Then* A *has to submit a pair of distinct messages M_0 and M_1 of equal length. The challenger randomly encrypts one of them and sends the ciphertext to* A*. We say that* A *wins if* A *can correctly guess*

which of the two messages have been encrypted. Therefore, Π is said to be IND-CPA *secure if the winning probability of any PPT algorithm* A *in the above game is upper bounded by* $\mathrm{negl}(\lambda)$.

E Proof of Theorem 2: Unforgeability of BMuSig-L

The proof proceeds through a sequence of games.

Game G_0. This is the original OMUF experiment. The system parameters and key pairs are generated using BMuSig-L.Setup and BMuSig-L.KeyGen, respectively. The adversary A is given the verification key of the honest signer, and is granted access to signing oracles. Through these oracles, A may concurrently initiate and participate in multiple signing sessions to obtain blind multi-signatures on messages of its choice. At the end of the game, A outputs a set of f' candidate forgeries

$$\{(L_\ell^*,\, M_\ell^*,\, \sigma_\ell^* := (R_\ell^*, s_\ell^*))\}_{\ell \in [f']}\,.$$

The adversary wins if the following conditions hold:

1. $f > q_s \geq 0$, where q_s denotes the number of signing sessions successfully completed through queries to the SIGN2 oracle;
2. for every $\ell \in [f']$, the honest signer's verification key is included in L_ℓ^*; and
3. all output forgeries are valid.

Game G_1. We construct a reduction to the soundness of ZK-NARK. To this end, we maintain four lists, denoted by $\mathcal{L}_M$, $\mathcal{L}_{\vec{\boldsymbol{\alpha}}}$, $\mathcal{L}_{\boldsymbol{\beta}}$, and $\mathcal{L}_{\vec{\boldsymbol{\xi}}}$. We modify the SIGN1 oracle as follows. The challenger generates the PKE key pair $(ek, dk) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ and stores the decryption key dk. Upon receiving an input $(pp, out_0 := ct)$, the oracle uses dk to decrypt ct and recovers the tuple $(M, \vec{\boldsymbol{\alpha}}, \boldsymbol{\beta}, \vec{\boldsymbol{\xi}})$. and stores the corresponding values in $\mathcal{L}_M$, $\mathcal{L}_{\vec{\boldsymbol{\alpha}}}$, $\mathcal{L}_{\boldsymbol{\beta}}$, and $\mathcal{L}_{\vec{\boldsymbol{\xi}}}$ respectively. Whenever the adversary invokes the SIGN2 oracle on input (pp, k, out_0, out_0'), the oracle retrieves the values $(M, \vec{\boldsymbol{\alpha}}, \boldsymbol{\beta}, \vec{\boldsymbol{\xi}})$ from the four lists at index k and computes

$$\vec{\mathbf{R}}' = \mathbf{A}\vec{\boldsymbol{\alpha}} + \boldsymbol{\beta}\vec{\mathbf{X}} + \vec{\mathbf{R}},\ \vec{\mathbf{R}}_1' = \mathbf{A}\vec{\boldsymbol{\alpha}} + \boldsymbol{\beta}\vec{\mathbf{X}} + \vec{\mathbf{R}}_1,\ \vec{\mathbf{R}}_j' = \vec{\mathbf{R}}_j,\ \forall\, j \in \{2, \ldots, \nu\}.$$

It then checks whether the following event, denoted by E_1, occurs:

$$\mathbf{c} \neq \mathsf{H}_{\mathsf{sig}}(\vec{\mathbf{X}}, \vec{\mathbf{R}}', M) + \boldsymbol{\beta}\ \vee\ \mathbf{b}_1 \neq 1\ \vee\ \mathbf{b}_j \neq \mathsf{Samp}(\mathbf{r}_{(j)}).$$

with $\mathbf{r}_{(j)} = (\mathsf{H}_{\mathsf{non}}(\vec{\mathbf{X}}, \vec{\mathbf{R}}_1', \ldots, \vec{\mathbf{R}}_\nu', M_k))_{(j)} + (\vec{\boldsymbol{\xi}})_{(j)},\ \forall j \in \{2, \ldots, \nu\}$. The challenger aborts the game and outputs 0.

Reduction to the Computational Soundness of ZK-NARK. We show that the difference between the success probabilities in Games G_0 and G_1 is bounded by the advantage of an adversary S against the soundness of ZK-NARK instantiated with the relation R.

$$|\Pr[G_0^{\mathsf{A}}(\lambda) = 1] - \Pr[G_1^{\mathsf{A}}(\lambda) = 1]| \leq \mathsf{Adv}^{\mathsf{SND}}_{\mathsf{ZK\text{-}NARK[R],S}}(\lambda).$$

The reduction S receives $\mathsf{crs} \leftarrow \mathsf{NARK.Setup}(pp)$, where pp is generated by $\mathsf{BMuSig\text{-}L.Setup}$ algorithm. Since S knows all public parameters, it can perfectly simulate Game G_1 for the adversary A. During an invocation of the $\mathsf{SIGN2}$ oracle on input (pp, k, out_0, out_0'), let

$$\theta = (\vec{\mathbf{X}}, \vec{\mathbf{R}}_1, \ldots, \vec{\mathbf{R}}_\nu, \mathbf{b}, \vec{\mathbf{R}}, \mathbf{c}, ct, ek)$$

be the corresponding statement. The reduction first verifies the proof π. If π is valid, it extracts the witness $\omega = (M, \vec{\boldsymbol{\alpha}}, \boldsymbol{\beta}, \vec{\boldsymbol{\xi}}; \rho)$ and reconstructs

$$\vec{\mathbf{R}}' = \mathbf{A}\vec{\boldsymbol{\alpha}} + \boldsymbol{\beta}\vec{\mathbf{X}} + \vec{\mathbf{R}},\ \vec{\mathbf{R}}_1' = \mathbf{A}\vec{\boldsymbol{\alpha}} + \boldsymbol{\beta}\vec{\mathbf{X}} + \vec{\mathbf{R}}_1,\ \vec{\mathbf{R}}_j' = \vec{\mathbf{R}}_j,\ \forall\, j \in \{2, \ldots, \nu\}.$$

It then checks whether the following event, denoted by E_1, occurs:

$$\mathbf{c} \neq \mathsf{H}_{\mathsf{sig}}(\vec{\mathbf{X}}, \vec{\mathbf{R}}', M) + \boldsymbol{\beta}\ \vee\ \mathbf{b}_1 \neq 1\ \vee\ \mathbf{b}_j \neq \mathsf{Samp}(\mathbf{r}_{(j)}).$$

with $\mathbf{r}_{(j)} = (\mathsf{H}_{\mathsf{non}}(\vec{\mathbf{X}}, \vec{\mathbf{R}}_1', \ldots, \vec{\mathbf{R}}_\nu', M))_{(j)} + (\vec{\boldsymbol{\xi}})_{(j)},\ \forall j \in \{2, \ldots, \nu\}$. If either condition holds, then S outputs the pair (θ, π) as its forgery in the soundness game. We now show that whenever either of the conditions occurs, the reduction S wins the soundness game. Since π is accepted for the statement θ, it suffices to prove that θ is not a valid statement. Assume, towards a contradiction, that θ is valid. Then there exists a witness

$$\omega' = (M', \vec{\boldsymbol{\alpha}}', \boldsymbol{\beta}', \vec{\boldsymbol{\xi}}'; \rho')$$

such that $(\theta, \omega') \in \mathrm{R}$ and $\mathsf{PRED}(\theta, \omega') = 1$. Consequently, we have

$$\begin{cases} (1)\ \mathbf{r}_{(j)} = (\mathsf{H}_{\mathsf{non}}(\vec{\mathbf{X}}, \vec{\mathbf{R}}_1', \ldots, \vec{\mathbf{R}}_\nu', M'))_{(j)} + (\vec{\boldsymbol{\xi}}')_{(j)},\ \mathbf{b}_1 = 1,\ \mathbf{b}_j = \mathsf{Samp}(\mathbf{r}_{(j)}) \\ (2)\ \mathbf{c} = \mathsf{H}_{\mathsf{sig}}(\vec{\mathbf{X}}, \vec{\mathbf{R}}', M') + \boldsymbol{\beta}', \\ (3)\ ct = \mathsf{PKE.Enc}(ek, (M', \vec{\boldsymbol{\alpha}}', \boldsymbol{\beta}', \vec{\boldsymbol{\xi}}'); \rho'). \end{cases}$$

By the perfect correctness of the public-key encryption scheme and Eqn. (3), the extracted witness must satisfy $(M', \vec{\boldsymbol{\alpha}}', \boldsymbol{\beta}', \vec{\boldsymbol{\xi}}') = (M, \vec{\boldsymbol{\alpha}}, \boldsymbol{\beta}, \vec{\boldsymbol{\xi}})$. Substituting these equations into Eqn. (1) and Eqn. (2) implies

$$\mathbf{r}_{(j)} = (\mathsf{H}_{\mathsf{non}}(\vec{\mathbf{X}}, \vec{\mathbf{R}}_1', \ldots, \vec{\mathbf{R}}_\nu', M))_{(j)} + (\vec{\boldsymbol{\xi}})_{(j)} \quad \text{and} \quad \mathbf{c} = \mathsf{H}_{\mathsf{sig}}(\vec{\mathbf{X}}, \vec{\mathbf{R}}', M) + \boldsymbol{\beta}$$

which contradicts the condition under which S outputs (θ, π). Therefore, θ cannot be a valid statement, implying that (θ, π) is a successful soundness forgery. Hence,

$$|\Pr[G_0^{\mathsf{A}}(\lambda) = 1] - \Pr[G_1^{\mathsf{A}}(\lambda) = 1]| \leq \mathsf{Adv}^{\mathsf{SND}}_{\mathsf{ZK\text{-}NARK[R],S}}(\lambda),$$

or equivalently,

$$\mathsf{Adv}^{\mathsf{OMUF}}_{\mathsf{BMuSig\text{-}L,A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{SND}}_{\mathsf{ZK\text{-}NARK[R],S}}(\lambda) + \Pr[G_1^{\mathsf{A}}(\lambda) = 1]. \tag{1}$$

Game G_2. In this game, we reduce the advantage of the adversary A in winning the game G_2 to the existential unforgeability game of $\mathsf{MuSig\text{-}L}$. The reduction maintains a counter table

$$Q_{\mathsf{ctr}}[(\vec{\mathbf{X}}, M)],$$

which records the number of completed signing sessions corresponding to the aggregate public key $\vec{\mathbf{X}}$ and message M. To answer a SIGN1 query, the reduction decrypts the ciphertext to recover $(M, \vec{\boldsymbol{\alpha}}, \boldsymbol{\beta})$. It then queries the MuSig-L signing oracle to obtain the honest signer's commitments $(\vec{\mathbf{R}}_1, \ldots, \vec{\mathbf{R}}_\nu)$. The reduction sets

$$\vec{\mathbf{R}}_1 = \vec{\mathbf{R}}_1 - \mathbf{A}\vec{\boldsymbol{\alpha}} - \boldsymbol{\beta}\vec{\mathbf{X}},\ \vec{\mathbf{R}}_t = \vec{\mathbf{R}}_t, \forall t = 2, \ldots, \nu.$$

Since $(\vec{\mathbf{R}}_1, \ldots, \vec{\mathbf{R}}_\nu)$ is uniformly distributed and $\mathbf{A}\vec{\boldsymbol{\alpha}} + \boldsymbol{\beta}\vec{\mathbf{X}}$ is a fixed value, the resulting tuple $(\vec{\mathbf{R}}_1, \ldots, \vec{\mathbf{R}}_\nu)$ has the same distribution as in the real protocol. To answer a SIGN2 query, the reduction forwards the query to the MuSig-L signing oracle and obtains the partial signature $\vec{\mathbf{s}}_1$. It returns

$$\vec{\mathbf{s}}_1 = \vec{\mathbf{s}}_1 - \boldsymbol{\alpha},$$

and updates

$$Q_{\mathsf{ctr}}[(\vec{\mathbf{X}}, M)] \leftarrow Q_{\mathsf{ctr}}[(\vec{\mathbf{X}}, M)] + 1.$$

Therefore, the view of the adversary A in Game G_2 is identically distributed to its view in Game G_1. Hence,

$$\Pr[G_2^{\mathsf{A}}(\lambda) = 1] = \Pr[G_1^{\mathsf{A}}(\lambda) = 1]. \tag{2}$$

Reduction to the EUF-CMA security of MuSig-L and the Knowledge Soundess of NARK. We construct two PPT adversaries U and S′. The adversary U is constructed against the EUF-CMA security of MuSig-L, while S′ is constructed against the knowledge soundness of the underlying NARK for relation R_1. The adversary U receives the public parameters pp, the aggregate public key $\vec{\mathbf{X}}$, and access to the signing oracles of MuSig-L. It simulates the signing queries of A by forwarding them to the corresponding signing oracles. Suppose that A outputs f valid forgeries

$$\left\{(\vec{\mathbf{X}}_\ell, M_\ell, \vec{\mathbf{R}}'_\ell, \pi_{1,\ell})\right\}_{\ell \in [f']}.$$

For each forgery, S′ invokes the knowledge extractor of NARK on $\pi_{1,\ell}$ and obtains a witness $\vec{\mathbf{s}}'_\ell$ satisfying

$$\mathbf{A}\vec{\mathbf{s}}'_\ell = \vec{\mathbf{R}}'_\ell + \mathbf{c}_\ell \vec{\mathbf{X}}_\ell.$$

If the extractor fails for any valid proof, S′ breaks the knowledge soundness of NARK. Otherwise, S′ obtains a valid witness $\vec{\mathbf{s}}'_\ell$ for every forgery. From each extracted witness, S′ recovers the corresponding signing response and hence a valid signature for the underlying MuSig-L scheme.

Suppose that A completes at most q_s signing sessions and outputs $f' > q_s$ valid forgeries. For every pair $(\vec{\mathbf{X}}, M)$, let $Q_{\mathsf{ctr}}[(\vec{\mathbf{X}}, M)]$ denote the number of completed signing queries, and let $F_{\mathsf{ctr}}[(\vec{\mathbf{X}}, M)]$ denote the number of extracted valid signatures corresponding to $(\vec{\mathbf{X}}, M)$.

If $F_{\mathsf{ctr}}[(\vec{\mathbf{X}}, M)] \leq Q_{\mathsf{ctr}}[(\vec{\mathbf{X}}, M)]$ for every $(\vec{\mathbf{X}}, M)$, then the total number of extracted signatures is at most q_s, contradicting $f' > q_s$. Hence, there exists a pair $(\vec{\mathbf{X}}^*, M^*)$ such that

$$F_{\mathsf{ctr}}[(\vec{\mathbf{X}}^*, M^*)] > Q_{\mathsf{ctr}}[(\vec{\mathbf{X}}^*, M^*)].$$

Therefore, at least one extracted signature on $(\vec{\mathbf{X}}^*, M^*)$ was never returned by the signing oracle. The adversary U outputs this signature as a forgery against the MuSig-L scheme. Therefor,

$$\Pr[G_2^{\mathsf{A}}(\lambda) = 1] \leq \mathsf{Adv}^{\mathsf{EUF-CMA}}_{\mathsf{MuSig\text{-}L},\mathsf{U}}(\lambda) + \mathsf{Adv}^{\mathsf{KS}}_{\mathsf{NARK}[\mathrm{R}_1],\mathsf{S}'}(\lambda). \tag{3}$$

Finally, the result follows by combining Eqn. (1), Eqn. (2) and Eqn. (3).

F Proof of Theorem 4: Blindness of BMuSig-L

Proof. The proof proceeds through a sequence of games.

Game G_0. This is the original blindness game. The adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ acts as the N signers. First, A_1 outputs a set of verification keys L and two messages (M_0, M_1). The challenger then chooses a random bit $b \leftarrow \{0,1\}$ and concurrently executes two signing sessions with A_2: session $k = 0$ signs M_b, while session $k = 1$ signs M_{1-b}. If either session does not produce a valid signature, the game outputs $(\perp, \perp)$. Otherwise, let (σ_0, σ_1) be the resulting signatures. Finally, A outputs a bit b' and wins if $b' = b$.

Game G_1. Game G_1 is identical to Game G_0, except that every proof generated in the two concurrent signing sessions is simulated. That is, instead of generating

$$\pi \leftarrow \mathsf{ZK\text{-}NARK.Prove}(crs, \theta, \omega),$$

the challenger generates

$$\pi \leftarrow \mathsf{ZK\text{-}NARK.SimProve}(crs, \theta).$$

All other parts of the game remain unchanged. We show that G_0 and G_1 are computationally indistinguishable under the zero-knowledge property of ZK-NARK. For each $b \in \{0,1\}$, we construct a PPT adversary Z_b against the zero-knowledge security of ZK-NARK. The adversary Z_b simulates the blindness experiment for A and uses its zero-knowledge challenger to obtain either real or simulated proofs. It then forwards these proofs to A_2. When A_2 outputs its decision bit b', Z_b outputs the same bit. Thus, when Z_b receives real proofs, its simulation is identical to Game G_0 with challenge bit b, whereas when it receives simulated proofs, its simulation is identical to Game G_1 with challenge bit b. Hence,

$$\left|\Pr[G_0^{\mathsf{A},b}(\lambda) = 1] - \Pr[G_1^{\mathsf{A},b}(\lambda) = 1]\right| \leq \mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{ZK\text{-}NARK}[\mathrm{R}],\mathsf{Z}_b}(\lambda).$$

Applying the triangle inequality for $b = 0$ and $b = 1$, we obtain

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{Blind}}_{\mathsf{BMuSig\text{-}L},\mathsf{A}}(\lambda) \leq\ & \left|\Pr[G_0^{\mathsf{A},1}(\lambda) = 1] - \Pr[G_0^{\mathsf{A},0}(\lambda) = 1]\right| \\
& \leq \mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{ZK\text{-}NARK}[\mathrm{R}],\mathsf{Z}_1}(\lambda) \\
& + \left|\Pr\left[G_1^{\mathsf{A},1}(\lambda) = 1\right] - \Pr\left[G_1^{\mathsf{A},0}(\lambda) = 1\right]\right| \\
& + \mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{ZK\text{-}NARK}[\mathrm{R}],\mathsf{Z}_0}(\lambda).
\end{aligned} \tag{4}$$

Game G_2**.** Game G_2 is identical to G_1, except that, in each session $k \in \{0,1\}$, the challenger encrypts a fixed public value $(\overline{M}, \vec{\mathbf{0}}, \mathbf{0}, \vec{\mathbf{0}})$ instead of $(M_{k\oplus b}, \vec{\boldsymbol{\alpha}}_k, \boldsymbol{\beta}_k, \vec{\boldsymbol{\xi}}_k)$. However, the proofs are still generated using ZK-NARK.SimProve. Now, we reduce the indistinguishability of G_1 and G_2 to the IND-CPA security of PKE. For each $b \in \{0,1\}$, let C_b be a CPA adversary that submits

$$M_0 = (\overline{M}, \vec{\mathbf{0}}, \mathbf{0}, \vec{\mathbf{0}}), \qquad M_1 = (M_{k\oplus b}, \vec{\boldsymbol{\alpha}}_k, \boldsymbol{\beta}_k, \vec{\boldsymbol{\xi}}_k)$$

to its CPA challenger and forwards the resulting ciphertext to A_2. If M_1 is encrypted, the simulation is $G_1^{\mathsf{A},b}$; if M_0 is encrypted, it is $G_2^{\mathsf{A},b}$. Hence,

$$\left|\Pr[G_1^{\mathsf{A},b}(\lambda) = 1] - \Pr[G_2^{\mathsf{A},b}(\lambda) = 1]\right| \leq \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE},\mathsf{C}_b}(\lambda).$$

By applying the triangle inequality,

$$\begin{aligned} &\left|\Pr[G_1^{\mathsf{A},1}(\lambda) = 1] - \Pr[G_1^{\mathsf{A},0}(\lambda) = 1]\right| \leq \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE},\mathsf{C}_1}(\lambda) \\ &+ \left|\Pr[G_2^{\mathsf{A},1}(\lambda) = 1] - \Pr[G_2^{\mathsf{A},0}(\lambda) = 1]\right| + \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE},\mathsf{C}_0}(\lambda). \end{aligned} \tag{5}$$

Here, we assume that there are 2 concurrent signing sessions. For each session $k \in \{0,1\}$ and each signer $j \in [N]$, the transcript is

$$\mathbf{T}_k = \left\{ \begin{array}{c} ct_k, (\vec{\mathbf{R}}^{(k)}_{j,1}, \ldots, \vec{\mathbf{R}}^{(k)}_{j,\nu}), (\vec{\mathbf{R}}^{(k)}_1, \ldots, \vec{\mathbf{R}}^{(k)}_\nu), \\ \vec{\mathbf{R}}_k, \mathbf{b}^{(k)}_1, \mathbf{b}^{(k)}_\nu, \vec{\mathbf{c}}_k, \pi^{(k)}, \vec{\mathbf{s}}^{(k)}_i \end{array} \right\}$$

Also, signer can set the signature of the corresponding messages $(M_0, \sigma_0 = (\vec{\mathbf{R}}'_0, \pi'_{1,0}))$, $(M_1, \sigma_1 = (\vec{\mathbf{R}}'_1, \pi'_{1,1}))$. Since ct is encryption of a fixed message with randomness ρ_k and π is generated by the zero-knowledge simulator of ZK-NARK, the distributions of ct and π_k are independent of the challenge bit b. Therefore, neither ct nor π reveals any information about b. Therefore, we omit ct and π from the subsequent analysis and consider only the remaining components of the signing transcripts. Now, the view of the adversary (signers) is $V_{k,i} = (T_k, M_i, \sigma_i)$. First, we show that for every $k, i \in \{0,1\}$, the signer's view $V_{k,i}(\mathsf{A})$ is consistent with a valid user state

$$st_{k,i} = (\vec{\boldsymbol{\alpha}}_{k,i}, \boldsymbol{\beta}_{k,i}, \vec{\boldsymbol{\xi}}_{k,i}),$$

where $(\vec{\boldsymbol{\alpha}}_{k,i}, \boldsymbol{\beta}_{k,i}, \vec{\boldsymbol{\xi}}_{k,i})$ are blinding values satisfying the protocol equations for the transcript T_k and the signature $\sigma_i = (\mathbf{R}'_i, \pi_{1,i})$. Since $\mathsf{NARK.Verify}(\theta_1, \pi_1) = 1$, knowledge soundness of NARK guarantees the existence of an expected-polynomial-time extractor $\mathcal{E}$ that, given the adversary's proof π_1, extracts a valid witness $\vec{\mathbf{s}}''_i$ except with $\mathsf{Adv}^{\mathsf{KS}}_{\mathsf{NARK}[\mathrm{R}_1],\mathsf{S}'}(\lambda)$. More precisely, if the required witness cannot be extracted, then S' can use the successful execution of A to violate the knowledge soundness of NARK. Therefore, except with probability $\mathsf{Adv}^{\mathsf{KS}}_{\mathsf{NARK}[\mathrm{R}_1],\mathsf{S}'}(\lambda)$, there exists a witness $\vec{\mathbf{s}}''$ satisfying the underlying signing

relation. Therefore, we have

$$\vec{\boldsymbol{\alpha}}_{k,i} = \vec{\mathbf{s}}'_i - \sum_{i=1}^{N} \vec{\mathbf{s}}_i^{(k)},\ \boldsymbol{\beta}_{k,i} = \mathbf{c}_k - \mathsf{H}_{\mathsf{sig}}(\vec{\mathbf{X}}, \vec{\mathbf{R}}'_i, M_i) \tag{6}$$

Now, we define $\vec{\mathbf{R}}'_{k,i} := \sum_{t=1}^{\nu} \mathbf{b}_t^{(k)} \vec{\mathbf{R}}_t^{(k)} + \mathbf{A}\vec{\boldsymbol{\alpha}}_{k,i} + \boldsymbol{\beta}_{k,i}\vec{\mathbf{X}}$, $\vec{\mathbf{s}}_{k,i} := \sum_{t=1}^{\nu} \mathbf{b}_t^{(k)} - \vec{\boldsymbol{\alpha}}_{k,i}$, $\mathbf{c}_{k,i} := \mathsf{H}_{\mathsf{sig}}(\vec{\mathbf{X}}, \vec{\mathbf{R}}'_{k,i}, M_i)$. Finally, we show that for all $k, i \in \{0,1\}$, the user state defines a valid signature $\vec{\mathbf{R}}'_i, \underline{\mathbf{s}}'_i$ and a hash value $\mathbf{c}'_i$. As the signature is valid we have

$$\mathbf{A}\vec{\mathbf{s}}'_i = \vec{\mathbf{R}}'_i + \mathbf{c}'_i\vec{\mathbf{X}} \implies \vec{\mathbf{R}}'_i = \mathbf{A}\vec{\mathbf{s}}'_i - \mathbf{c}'_i\vec{\mathbf{X}}, \tag{7}$$

where $\mathbf{c}'_i = \mathsf{H}_{\mathsf{sig}}(\vec{\mathbf{X}}, \vec{\mathbf{R}}'_i, M_i)$. By substituting $\vec{\mathbf{s}}'_i = \vec{\boldsymbol{\alpha}}_{k,i} + \sum_{i=1}^{N} \vec{\mathbf{s}}_i^{(k)}$ and $\mathbf{c}'_i = \mathbf{c}_k - \boldsymbol{\beta}_{k,i}$ into Eqn. (6) and Eqn. (7), we get

$$\vec{\mathbf{R}}'_i = \mathbf{A}(\vec{\boldsymbol{\alpha}}_{k,i} + \sum_{i=1}^{N} \vec{\mathbf{s}}_i^{(k)}) - (\mathbf{c}_k + \boldsymbol{\beta}_{k,i})\vec{\mathbf{X}} \tag{8}$$

We assume that A closes both signing sessions $k \in \{0,1\}$ successfully, and in this case T_k is a valid transcript. Therefore, we have

$$\mathbf{A}(\sum_{i=1}^{N} \vec{\mathbf{s}}_i^{(k)}) = \vec{\mathbf{R}}_k + \mathbf{c}_k\vec{\mathbf{X}},\ \text{where } \vec{\mathbf{R}}_k = \sum_{t=1}^{\nu} \mathbf{b}_t^{(k)} \vec{\mathbf{R}}_t^{(k)}. \tag{9}$$

By substituting Eqn. (9) into Eqn. (8), we get

$$\begin{aligned}
\vec{\mathbf{R}}'_i &= \mathbf{A}\vec{\boldsymbol{\alpha}}_{k,i} + \vec{\mathbf{R}}_k + \mathbf{c}_k\vec{\mathbf{X}} - (\mathbf{c}_k + \boldsymbol{\beta}_{k,i})\vec{\mathbf{X}} \\
&= \mathbf{A}\vec{\boldsymbol{\alpha}}_{k,i} + \vec{\mathbf{R}}_k + \boldsymbol{\beta}_{k,i}\vec{\mathbf{X}} = \sum_{t=1}^{\nu} \mathbf{b}_t^{(k)} \vec{\mathbf{R}}_t^{(k)} + \mathbf{A}\vec{\boldsymbol{\alpha}}_{k,i} + \vec{\boldsymbol{\beta}}_{k,i}\vec{\mathbf{X}} \\
&= \vec{\mathbf{R}}'_{k,i}
\end{aligned}$$

Note that that $\mathbf{R}'_i$ is determined by the user state. Consequently, $\vec{\mathbf{s}}'_i = \sum_{i=1}^{N} \vec{\mathbf{s}}_i^{(k)} - \vec{\boldsymbol{\alpha}}_{k,i}$, $\mathbf{c}_{k,i} = \mathsf{H}_{\mathsf{sig}}(\vec{\mathbf{X}}, \vec{\mathbf{R}}'_{k,i}, M_i)$. This means both transcripts T_0 and T_1 can define the signature $(\vec{\mathbf{R}}'_i, \pi'_{1,i})$. Hence, except with $\mathsf{Adv}^{\mathsf{KS}}_{\mathsf{NARK}[\mathrm{R}_1],\mathsf{S}'}(\lambda)$, the views generated in $G_2^{\mathsf{A},0}$ and $G_2^{\mathsf{A},1}$ are identically distributed. Therefore,

$$\left|\Pr\left[G_2^{\mathsf{A},1}(\lambda) = 1\right] - \Pr\left[G_2^{\mathsf{A},0}(\lambda) = 1\right]\right| \leq \mathsf{Adv}^{\mathsf{KS}}_{\mathsf{NARK},\mathsf{S}'}(\lambda). \tag{10}$$

The result follows by combining Eqn. (4), Eqn. (5), and Eqn. (10).

Object	Distribution	Method	Target	Meas.
$\mathbf{A}$	unif. $\mathcal{R}_q^{\kappa\times m}$	XOF, 30-bit rej.	—	—
$\vec{\mathbf{x}}$	unif. $S_\eta^{m+\kappa}$	XOF	—	1.412
$\mathbf{a}_i$	$\mathcal{C}$, wt. κ_1	SampleInBall	—	—
$\mathbf{c}$	$\mathcal{C}$, wt. κ_1	SampleInBall	—	—
$\vec{\mathbf{r}}_{i,1}$	$\mathcal{D}_{\sigma_1}$	rej., 12σ tail	1.500e6	1.498e6
$\vec{\mathbf{r}}_{i,j\geq 2}$	$\mathcal{D}_{\tilde{\sigma}}$	rej., 12σ tail	1.000e3	1.003e3
$\mathbf{b}_1$	constant 1	fixed	—	—
$\mathbf{b}_{j\geq 2}$	$\mathcal{D}_{\sigma_b}$	$\mathsf{SAMP}(H_{\text{non}})$	2.000	2.005
$\vec{\boldsymbol{\alpha}}, \beta$	unif. $\mathcal{C}_1$	XOF trits	—	0.8166
ρ	unif. 32 bytes	system RNG	—	—

Table 8: Sampling methods and measured standard deviations over 64 draws. The 0.8166 for $\vec{\boldsymbol{\alpha}}, \beta$ matches $\sqrt{2/3}$, the standard deviation of a uniform trit.

G Benchmarking: Additional Details

G.1 Sampling

All randomness is derived from SHAKE256; every call absorbs a distinct domain-separation tag followed by length-prefixed inputs.

The discrete Gaussian sampler is itself rejection-based: it proposes uniformly on $[-12\sigma, 12\sigma]$ and accepts with probability $\exp(-x^2/2\sigma^2)$, an acceptance rate of $\sqrt{2\pi}/24 \approx 0.104$ and hence about 9.6 proposals per accepted coefficient, measured at 201 ns per *accepted* coefficient, proposals included. This is distinct from, and far more frequently invoked than, the protocol's own RejSamp.

Sampling is nonetheless not where round 1 goes. The $\nu(m+\kappa)n = 12\,288$ coefficients drawn per signer cost 2.0 ms, against 20.1 ms for the ν products $\bar{\mathbf{A}}\vec{\mathbf{r}}_{i,j}$ — each $\kappa m = 36$ schoolbook $O(n^2)$ polynomial multiplications — for a total of 22.1 ms, matching the 21.7 ms of Table 5. Round 1 is 91% arithmetic, so an NTT, were the modulus to permit one, would be the single largest available speed-up.

The encryption scheme uses $k_e = 4$, $d = 29$, $\eta_e = 2$ and $\Delta = \lceil q/2 \rceil$, giving 33 792 B ciphertexts. Worst-case decryption noise is 8194 against a threshold of 2.68×10^8. AKSY instantiate essentially the same scheme with plaintext modulus $p = 49\,126$ rather than Δ, carrying ≈ 15.6 bits per coefficient against our 1; the same technique here would allow $p \leq 16\,380$ at a $4\times$ margin, cutting d from 29 to 3 and the ciphertext to 7168 B. As ct is preprocessing material this is not on the critical path.

G.2 Rejection sampling and failure rates

G.3 Scaling in the ring degree

Parameters are re-derived at each n from the Table 1 chain, so every point carries the same per-signer restart rate $K = 1.06$: $t = \sqrt{n/((\pi-1)\log_2 e)}$,

$\|\vec{\mathbf{v}}\|_2$	α **measured**	$1/K$ (worst)
7.84×10^1	19 156	0.9370	0.9434
6.42×10^3	234	0.9447	0.9434
6.40×10^4	23	0.9437	0.5989
1.92×10^5	8	0.9167	0.2137
6.40×10^5	2	0.8070	0.0055
1.92×10^6	0.8	0.5067	0.0000

Table 9: RejSamp acceptance as $\|\vec{\mathbf{v}}\|_2$ grows, 3000 trials per row (± 0.4 points of sampling error). The right column is $1/K$ with K the worst-case supremum of the density ratio, so it is a lower bound on the average and the two agree at the top of the table to within that error. What the table shows is that acceptance *collapses* once α falls, so the sampler is live rather than accepting unconditionally. The scheme operates at $\|\vec{\mathbf{v}}\|_2 \approx 4.7 \times 10^3$.

N **per-signer success**	**failure**	**attempts**
1	4.50%	95.50%	4.50%	1.05
2	7.00%	86.50%	13.50%	1.16
3	5.67%	84.00%	16.00%	1.19
5	4.80%	77.00%	23.00%	1.30
8	6.38%	59.50%	40.50%	1.68

Table 10: Session completion over 200 sessions per row. Per-signer failure is counted directly — every signer runs even after one has aborted — rather than back-derived from the session rate, so it is an independent check on the $K_N = K^N$ model rather than a restatement of it. It is flat in N, as it must be, and averages 5.67% against the predicted $1 - 1/K = 5.66\%$; individual rows carry 1–2 points of sampling error.

$T = \kappa_1^2 \eta \sqrt{n(m+\kappa)}$, α the root of $t/\alpha + 1/(2\alpha^2) = \ln K$, and $\sigma_1 = \alpha T + 1$, with q the least prime $\equiv 5 \pmod 8$ exceeding $2B_N$.

Doubling n multiplies the signature by ≈ 2.07 and verification by up to 3.96, approaching the schoolbook $\Theta(n^2)$. The choice $n = 256$ is not forced by the challenge space — $|\mathcal{C}| \approx 2^{184}$ already at $n = 128$ — but by the hardness of the underlying lattice instances, whose dimension is $n(m+\kappa)$.

Within this family larger n is monotonically better for security, since the row count grows faster than B_N: $\delta = 1.00200$ at $n = 512$ and 1.00106 at $n = 1024$, against 1.00330 at $n = 256$.

G.4 Experimental setup

The CPU and hardware details are given in Table 12.

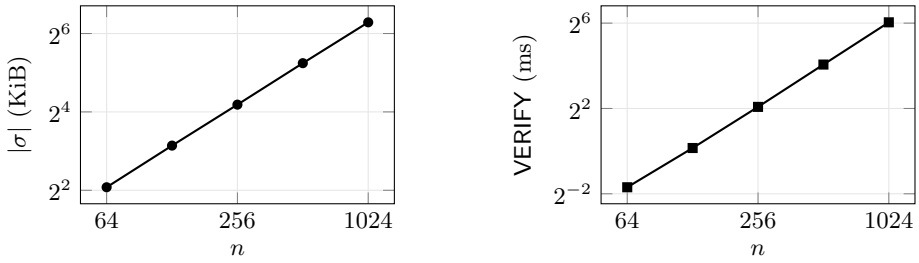

Fig. 9: Signature size and verification time against ring degree, at fixed $\kappa = m = 6$ and $K = 1.06$. Fitted $\log_2$–$\log_2$ slopes are 1.05 and 1.93. Data in Table 11.

n	t	α	σ_1	w	$\lvert\sigma\rvert$	VERIFY
64	4.55	78	1.561e7	32	4320	0.31 ms
128	6.44	111	3.119e7	34	9024	1.11 ms
256	9.10	156	6.236e7	35	18 624	4.21 ms
512	12.87	221	1.247e8	37	38 784	16.65 ms
1024	18.21	312	2.494e8	38	79 872	65.65 ms

Table 11: Data behind Figure 9, at $\kappa = m = 6$, $\kappa_1 = 60$, $\eta = 2$, $N = 16$, $K = 1.06$. Sizes in bytes. Verification is timed with a fixed 30-bit modulus; a larger q needs 128-bit intermediates, a roughly constant factor that does not change the shape of the curve.

Component	**Detail**
CPU	Intel Xeon 6973P-C, `x86_64`
	1 physical core, 2 hardware threads
Frequency	0.8 GHz to 2.6 GHz (scaling enabled)
Cache	48 KiB L1d, 2 MiB L2
Memory	7.8 GiB
Platform	virtualised (Microsoft Azure)
OS	Ubuntu 24.04.4 LTS, kernel 6.17.0
Toolchain	`rustc 1.97.1`, release profile
	`opt-level=3`, LTO, `codegen-units=1`
Execution	single-threaded, no explicit pinning

Table 12: Measurement environment.