Privacy-Preserving Multi-Signatures: Generic Techniques and Constructions Without Pairings

Calvin Abou Haidar[1], Dipayan Das[2], Anja Lehmann[3], Cavit Özbay[3], and Octavio Perez Kempner[1]

[1] NTT Social Informatics Laboratories
[2] Department of Mathematics and Statistics, Florida Atlantic University
[3] Hasso-Plattner-Institute, University of Potsdam

Abstract. Multi-signatures allow a set of parties to produce a single signature for a common message by combining their individual signatures. The result can be verified using the aggregated public key that represents the group of signers. Very recent work by Lehmann and Özbay (PKC '24) studied the use of multi-signatures for ad-hoc *privacy-preserving* group signing, formalizing the notion of multi-signatures with probabilistic yet verifiable key aggregation. Moreover, they proposed new BLS-type multi-signatures, allowing users holding a long-term key pair to engage with different groups, without the aggregated key leaking anything about the corresponding group. This enables key-reuse across different groups in a privacy-preserving way. Unfortunately, their technique cannot be applied to Schnorr-type multi-signatures, preventing state-of-the-art multi-signatures to benefit from those privacy features.
In this work, we revisit the privacy framework from Lehmann and Özbay. Our first contribution is a generic lift that adds privacy to any multi-signature with deterministic key aggregation. As our second contribution, we study two concrete multi-signatures, and give dedicated transforms that take advantage of the underlying structures for improved efficiency. The first one is a slight modification of the popular MuSig2 scheme, achieving the strongest privacy property for free compared to the original scheme. The second is a variant of the lattice-based multi-signature scheme DualMS, making our construction the first post-quantum secure multi-signature for ad-hoc *privacy-preserving* group signing. The light overhead incurred by the modifications in our DualMS variant still allow us to benefit from the competitiveness of the original scheme.

Keywords: Multi-signatures, Schnorr-based signatures, MuSig2, Lattices, Privacy-preserving

1 Introduction

Multi-signatures allow the aggregation of several signatures by different parties on the same messages into one compact signature. If the scheme further supports key aggregation [MPSW19], the signers' individual public keys can also be combined into a single short key. Verification is then carried out using the aggregated key and aggregated signature, and thus completely independent of the number of signers. This makes multi-signatures particularly useful in applications with large groups of signers, like cryptocurrencies.

Since their introduction by Maxwell *et al.* [MPSW19], multi-signatures with key aggregation have attracted significant attention not only in academia [BDN18, DGNW20, NRS21, TZ23, PW23, Che23] but also in real-world applications. Most notably, driven by their relevance in large-scale blockchain deployments, Nick *et al.* [NRS21] introduced MuSig2, a two-round multi-signature scheme that has been standardized for use in the Bitcoin network [Cho24] very recently.

A reason for multi-signatures' popularity is their convenient key management. All parties generate their individual key pairs fully autonomously, allowing them to form signing groups in an ad-hoc manner. Each group is represented through their aggregated public key, and signatures for that group key then require the consensus input from all parties. Signers can also decide to re-use the same signing key across different groups, making it attractive for hardware-protected user keys, *e.g.,* for cryptocurrency wallets [CCK+23] or two-factor authentication [MPs19]. This use of individual and long-term keys is a main advantage over threshold signatures, where each group requires a dedicated key generation.

Lack of Privacy in Multi-Signatures. Apart from their benefits in terms of efficiency and convenience, multi-signatures were also advertised to provide privacy protection, claiming that aggregated keys do

not leak information about the amount or identities of individual signers [MPSW19]. However, in a recent work, Lehmann and Özbay [LÖ24] show that these privacy claims are not correct. This is due to the *deterministic* nature of multi-signatures, or rather their aggregated keys, which enables the tracking of users that re-use their key across groups. More precisely, if users' individual public keys are assumed to be publicly known, all existing multi-signatures trivially reveal sensitive information such as the number of signers and individual signer keys behind an aggregated public key.

Interestingly, this is in stark contrast to the expectations users have in multi-signatures: a recent study [MDM+23] on users' preferences for cryptocurrency wallets reveals that users who prefer *multi-device* wallets, expect the wallet to hide the internal structure and individual signers. Thus, while multi-signatures provide convenient key management, this convenience comes at the cost of privacy.

For existing multi-signature schemes, privacy was shown to be possible only in a rather weak model – called the *all-but-one-public-key* (AbOPK) model – that assumes that there is at least one unknown public key in each signer set. Despite being less realistic, the AbOPK model shows which level of privacy can be provided by multi-signatures with deterministic key aggregation [LÖ24].

Privacy-Preserving Multi-Signatures. To remedy the lack of privacy, Lehmann and Özbay [LÖ24] introduce *multi-signatures with verifiable key aggregation*. These schemes use randomized aggregated keys that can be verified through an aggregation proof π. By using a randomized key aggregation instead of deterministic one, the desired privacy properties can be formulated in a realistic *known-public-key* (KPK) model, where the adversary is assumed to know all signers' public keys but not π. The necessary security and privacy properties are captured by presenting a comprehensive framework that includes group unforgeability (signatures cannot be reused in other context than the group for which they were issued) as well as privacy notions, ranging from hiding the number of signers to even hiding the fact that a signature was produced via key aggregation.

They also present two schemes based on BLS multi-signatures: randBLS-1 and randBLS-2. The first one satisfies the strongest privacy property in the KPK model (FullPriv) but falls short to achieve group unforgeability. The second one provides both properties, but at the expense of slightly modifying the signature structure, making it different from standard BLS signatures.

Filling the Gap. While BLS multi-signatures are attractive due to their simplicity and non-interactive nature, their privacy-preserving variants trade-off group unforgeability for privacy, offering strong guarantees for one but not both. Additionally, they require pairing-friendly curves, which lack standardization, and therefore, hinder practical adoption. Thus, pairing-free signatures would have great advantages for immediate real-world deployment. Further, as soon as efficient quantum computers exist, BLS-based constructions will no longer provide security – making them a non-ideal candidate for long-term strategies either. Thus, while it was shown that privacy-preserving multi-signatures are possible, there is still no scheme that is deployable in the targeted applications today or provides post-quantum guarantees. Our aim is to fill this gap.

1.1 Our Contributions

Our work advances the study of privacy-preserving multi-signatures on multiple fronts. To begin with, in Section 4, we present a generic construction that lifts privacy from the AbOPK model to the KPK model for any scheme that meets certain (minimal) conditions. Subsequently, sections 5 and 6 are dedicated to introducing privacy-preserving variants of state-of-the-art constructions. The first is based on MuSig2 [NRS21], which is secure in the discrete logarithm setting. The second one is based on DualMS [Che23] whose security is reduced to a hard problem in the lattice setting, widely considered to offer post-quantum security [NIS24].

Along the way, we revisit the multi-signature framework from [LÖ24], extending it to account for multiple signing rounds and the optional use of the aggregation proof in the signing algorithm to cover all scenarios. Additionally, we introduce a new privacy notion, wSetPriv, which is particularly useful for lattice-based constructions. Below, we elaborate on each contribution in more detail.

Generic Techniques. In the AbOPK model, group privacy hinges on the presence of a party whose public key is unknown to the adversary. While hiding a public key seems an unrealistic assumption for real-world use cases, it provides insights into designing privacy-preserving schemes in the more realistic KPK model. Indeed, consider the following idea: every group generates an additional (virtual)

dummy user so that its signature share can be computed by any party that knows it's dummy secret key. If this scheme is private in the AbOPK model, then privacy in the KPK model seems linked to the former. With this in mind, we present a generic transformation that lifts privacy of a scheme in the AbOPK model to the KPK model while preserving unforgeability. Our generic construction only requires the use of a pseudorandom function (PRF) but presents one drawback: it requires signers to use the key aggregation proof in signing protocol. Fortunately, for some schemes we are able to remove those limitations as discussed in the technical overview.

Pairing-Free Constructions. MuSig2 produces standard Schnorr signatures while ensuring concurrent security and key aggregation, all with minimal overhead compared to the original scheme. We present PP-MuSig2, a privacy-preserving variant of MuSig2 that provides the strongest guarantees for privacy-preserving group signing with no overhead compared to the original MuSig2. These modifications allow to get the privacy property for free, favoring its adoption.

We also introduce PP-DualMS, the first privacy-preserving multi-signature to offer post-quantum security. To that end, we extend DualMS [Che23], a two-round lattice-based multi-signature that follows the Fiat-Shamir with aborts paradigm. Our choice is based on a simpler alternative to previous lattice-based multi-signature schemes like DOTT [DOTT21] and MuSig-L [BTT22], as the security proof for DualMS is done using straight-line simulation. Unfortunately, unlike PP-MuSig2 (which we prove to offer full-privacy), we are only able to prove a weaker notion of set-privacy, which seems the best possible hope for lattice-based (SIS) constructions as discussed in Section 6.2. The parameters of PP-DualMS are similar to those of DualMS, making the overhead light to achieve weak set privacy.

1.2 Related Work

Multi-Signatures with Verifiable Key Aggregation. The privacy framework of [LÖ24] discusses group unforgeability and three privacy properties. Group unforgeability states that it is infeasible to create a signature for a message m and group key apk when not all signers in the group provided a signature on m for apk. Regarding privacy, full privacy (FullPriv) ensures that the aggregated keys and signatures of a multi-signature scheme are indistinguishable from those generated by the standard counterpart. In other words, an aggregated public key does not leak anything about a group nor its members. In contrast, a weaker property called set privacy (SetPriv) guarantees that aggregated public keys and signatures of different signing groups are indistinguishable. Finally, membership privacy (MemPriv), the most limited privacy property, focus on the indistinguishability for two signer groups that differ by one member only.

It is worth noting that in the KPK model, all three privacy properties of Lehmann and Özbay [LÖ24] guarantee the unlinkability of aggregated keys of a single group. In other words, multiple runs of key aggregation for the same signer group must generate aggregated keys that are not linkable to each other by outsiders. The definitions differ based on their privacy requirements for the aggregated keys across distinct signer groups. Our work builds upon their privacy framework, which we revisit in Section 3.

Signatures with Re-randomizable Keys. Fleischhacker *et al.* [FKM+16] introduced signatures with re-randomizable keys that allow randomizing key pairs of a signature scheme. Randomized key pairs must be indistinguishable from freshly generated key pairs, which gives a similar property to that achieved by privacy-preserving multi-signatures. Subsequent works defined various forms of key malleability. Backes *et al.* proposed signatures with flexible public keys, allowing keys to be re-randomized within equivalence classes [BHKS18]. Mercurial signatures were proposed by Crites and Lysyanskaya in [CL19], further allowing the re-randomization of both messages and public keys. However, none of those works specifically targets multi-signatures. For a more detailed literature review on signatures with randomizable keys, we lead the reader to [CGH+25].

Threshold Signatures with Re-randomizable Keys. Recently, there have been two works that addressed key re-randomizability in threshold signatures. Abe *et al.* [ANPT25] built upon the concept of mercurial signatures to create a new scheme for the threshold setting. In a similar vein, Gouvea and Komlo [GK24] extended the notion of signatures with re-randomizable keys to threshold signatures. Both works have potential applications in deriving an unlinkable group public key from an existing one. However, despite their benefits, these schemes still inherit a similar key management overhead as

traditional threshold signatures: signers must store a separate secret key share for each signing group they are in. Thus, they do not offer the ad-hoc key aggregation properties of multi-signatures.

Multi-Signatures from Key-Homomorphic Signatures. A key-homomorphic signature is a scheme that includes an additional algorithm called Adapt. This algorithm, given a signature σ, a message m, a public key pk, and a linear shift function Δ, produces an adapted signature σ' for the same message m that verifies under an adapted public key pk'. This concept was studied and formalized by Derler and Slamanig, who proposed a definitional framework in [DS19].

At a minimum, the secret and public key elements should belong to groups $(\mathbb{H}, +)$ and $(\mathbb{E}, \cdot)$, where group operations, inversions, membership testing, and uniform sampling can be done efficiently. Additionally, adapted signatures must verify under the corresponding public key and should be indistinguishable from freshly computed signatures. This property is captured in their definition of signature adaptability (see Appendix C.1 for formal definitions).

For some key-homomorphic schemes is possible to define a $\mathsf{Combine}$ algorithm that takes a set of signatures for the same message and produces a signature under the combined public key, without requiring knowledge of the original secret keys. As noted in [DS19], this naturally leads to a black-box construction of multi-signatures from *publicly key-homomorphic* signatures (see Appendix C.2), with BLS signatures being perhaps the most straightforward example.

Unfortunately, the above approach does not easily translate to privacy-preserving multi-signatures. As previously discussed, one of the root issues is that compatibility with standard signature verification is essential to ensure privacy. However, any straightforward construction of multi-signatures from a *publicly key-homomorphic* signature preserves adaptability, now at odds with the strongest unforgeability requirement for group signing. This trade-off seems inherent in multi-signatures based on *publicly key-homomorphic* signatures as evidenced with the constructions from [LÖ24].

1.3 Technical Overview

Lifting Privacy Generically. As mentioned before, to get privacy in the KPK model, we consider a virtual dummy user whose public key is the unknown public key in the AbOPK model. Naturally, our initial idea is to use such virtual user to produce a signature share that finalizes the multi-signature. More in detail, since its secret key is clearly unknown to outsiders of the group, we can think of it as the aggregation proof π. Consequently, the randomized aggregated key can be computed from the deterministic aggregated key that represents the group and the dummy secret key, which acts as the secret key-randomizer. If signers know π, they can play the role of the dummy user during signature generation to produce consistent shares. Whilst simple and seemingly straightforward, this idea requires careful considerations to *generically* apply it.

First of all, we set up the dummy key in a certain way. Instead of sampling a random key pair, the dummy key pair $(\mathsf{dpk}, \mathsf{dsk})$ is created by explicitly fixing the random coins of the key generation algorithm with the hash value $\mathsf{H}(\mathsf{seed}, PK)$ for a fresh seed and list of public keys PK. This extra step in the dummy key pair generation will help us to ensure the unforgeability of our generic construction. The secret value seed is kept internal to the group, by making it part of the aggregation proof. Further, during generation of the aggregated key, we also sample a fresh PRF key, obtaining an aggregation proof given by $(\mathsf{seed}, \mathsf{k}, \pi)$. The PRF key will be used to simulate the partial signatures of dsk.

The remaining challenge is to create valid signatures for the aggregated public key apk, which contains the contributions under dpk. All signers of apk already know seed, so they can locally compute dsk and run the signing protocol for dsk on their own. However, to ensure the correctness of the signing protocol, all signers must compute the identical dummy signature shares. We achieve this by letting all signers derive the dummy signature in a *deterministic* manner. To do so, signers use the PRF key k to derive the random coins of the signing algorithms for dsk as further described in Section 4.

A Generic Template for Key-Homomorphic Signatures. Our generic construction does not rely on any specific property of the underlying signature scheme. However, previous works presented how to get multi-signatures from publicly key-homomorphic schemes in a black-box way [DS19] or from identification schemes [BN06]. Bellare and Neven [BN06] discussed how to obtain multi-signatures from identification schemes, provided that "some homomorphism" is allowed by the underlying scheme. Recently, Tessaro and Zhu [TZ23] constructed multi-signatures by using *linear hash functions* which are identification schemes with such homomorphic properties. We also consider key-homomorphic

signatures from canonical identification schemes (see Appendix C.1 for related background) to derive a generic template for privacy-preserving multi-signatures, which we apply to Schnorr-based multi-signatures.

More in detail, let us consider the structure of two-round Schnorr-based multi-signatures as used in this work. In the first round, known as the commitment phase, all parties generate commitments, which are then aggregated into a single commitment. This aggregated commitment is used to compute a common challenge. In the second round, the signing phase, each party generates a partial signature based on the agreed-upon challenge. Once all shares are gathered, they are combined to produce the final signature. With this in mind, let us further assume that only the party that generated the aggregated public key for the group has access to the dummy secret key. This party might not actually take part in the commitment phase. However, as it receives all the shares to compute the aggregated signature, it can add its contribution at the end. We exploit this fact to produce the final signature and the aggregated public key consistently. As a result, users can execute the signing protocol without knowledge of the aggregated proof, overcoming the limitation of our generic lift.

The intuition behind why the above works is exactly what gives us a template. Given a valid signature $\sigma = (s, R)$ for a message m under public key pk, anyone can publicly adapt it sampling a secret key Δ and computing $\sigma' \leftarrow (s', R)$ with $c \leftarrow \mathsf{H}_1(R, m)$ and $s' \leftarrow s + c \cdot \Delta \bmod p$. As a result, σ' verifies under $\mathsf{pk}' \leftarrow g^{\mathsf{sk}} \cdot g^{\Delta}$ This property is known as *signature adaptability*, not to be confused with the *public adaptability* notion previously discussed for BLS signatures. We also note that key-prefixing for identification schemes is done when computing the challenge c, allowing us to get group unforgeability for free. Our privacy-preserving version of MuSig2 exploits these features treating Δ as the dummy secret key. In Appendix E we also outline how the same template leads to a privacy-preserving version for Guillou-Quisquater multi-signatures.

2 Preliminaries

We present the notation and cryptographic background used in this work. We recap the algebraic one-more discrete logarithm assumption in Appendix A.

Notation. We consider cyclic groups $\mathbb{G}$ of primer order p as well as a group generator GGen that outputs $(\mathbb{G}, g, p)$ where g is used as the base point. GGen takes λ (the security parameter) as input. $\mathbb{Z}_p$ denotes the ring of integers modulo p. Square brackets are used for optional parameters. For any probabilistic algorithm $\mathsf{Alg}(in) \to out$, we define the deterministic variant of the algorithm as $\mathsf{Alg}(in; \rho) \to out$ where ρ is the random tape that Alg is run on.

Pseudorandom Function. A pseudorandom function $\mathsf{PRF} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a function such that $\Pr\left[\mathsf{k} \leftarrow\$ \mathcal{K}; \mathcal{A}^{\mathsf{PRF}(\mathsf{k},\cdot)}(\lambda) = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{RO}(\cdot)}(\lambda) = 1\right] = \mathsf{negl}(\lambda)$ where $\mathsf{RO} : \mathcal{X} \to \mathcal{Y}$ is a random oracle.

Lattice Assumptions. Let $R = \mathbb{Z}[x]/(\mathbf{f}(\mathbf{x}))$ be the quotient polynomial ring defined by the polynomial $\mathbf{f}(\mathbf{x})$. In this paper, we consider $\mathbf{f}(\mathbf{x}) = \mathbf{x}^N + 1$, where N is a power of 2. For any q, we write R_q to denote the ring $\mathbb{Z}_q[x]/(\mathbf{f}(\mathbf{x}))$. We consider an element of R (similarly for R_q) as an element of $\mathbb{Z}^n$ using the usual coefficient embedding. The norm of an element in R is the norm of the representation in $\mathbb{Z}^n$. For any element $\mathbf{v}$ in R, we write $\|\mathbf{v}\|_1, \|\mathbf{v}\|, \|\mathbf{v}\|_\infty$ to represent the $\ell_1, \ell_2, \ell_\infty$ norm of the element $\mathbf{v}$. When computing the norm in R_q, we consider the representations of $\mathbb{Z}_q$ in the interval $[-(q-1)/2, (q-1)/2]$.

For any $\mathbf{x} \in R^m$, we define the discrete Gaussian function with the parameter $(\mathbf{v}, s) \in R^m \times \mathbb{R}$ as $\rho_{\mathbf{v},s}(\mathbf{x}) = \exp(-\pi\|\mathbf{x}-\mathbf{v}\|^2/s^2)$. The discrete Gaussian distribution is $D^m_{\mathbf{v},s}(\mathbf{x}) = \frac{\rho_{\mathbf{v},s}(\mathbf{x})}{\rho_{\mathbf{v},s}(R^m)}$. We omit the subscript for the case $\mathbf{v} = 0$. For any ϵ, we define the *smoothing parameter* $\eta_\epsilon(\Lambda)$ of any lattice Λ as the smallest positive s such that $\rho_{1/s}(\Lambda^* \setminus \{0\}) \leq \epsilon$, Λ^* being the dual lattice of Λ. Following [MR04], we have $\eta_\epsilon(R^m) \leq \sqrt{Nm}$, where $\epsilon = 2^{-Nm}$. Every s used in this paper, the value will be more than that of $\eta_\epsilon(R^m)$. We use the following results on the discrete Gaussian distribution.

Lemma 1 (Gaussian convolution, [MP13], adapted from Theorem 3.3). *Let $s \geq \sqrt{2}\eta_\epsilon(R^m)$. Let $\mathbf{x}_i$ be sampled independently from $D^m_{s_i}$. Then $\sum_{i=1}^n \mathbf{x}_i$ is statistically close to $D^m_{s_i}$, where $s = \sqrt{\sum_{i=1}^n s_i^2}$.*

Lemma 2 (Gaussian tail bound, [Lyu12], Lemma 4.4). *For any $\gamma > 1$,*

$$\Pr\left[\|\mathbf{z}\| > \gamma(s/\sqrt{2\pi}\sqrt{mN} : \mathbf{z} \leftarrow\$ D_s^m\right] < \gamma^{mN} e^{mN(1-\gamma^2)/2}$$

Lemma 3 (Rejection sampling bound, [Lyu12], Lemma 4.5). *Let t such that $t = \omega(\sqrt{\log(mN)})$ and $t = o(\log(mN)$. For any $\mathbf{v} \in R^m$, if $s \geq \sqrt{2\pi}\alpha\|\boldsymbol{v}\|$, then*

$$\Pr\left[M \cdot D_{\mathbf{v},s}^m \geq D_s^m : \mathbf{z} \leftarrow\$ D_s^m\right] \geq (1-\epsilon)$$

where α is any positive integer, $M = e^{t/\alpha + 1/(2\alpha^2)}$, and $\epsilon = 2e^{-t^2/2}$.

Lemma 4 (Theorem 4.6, [Lyu12]). *Let $m, B > 0$ and $\sigma = \omega(B\sqrt{\log m})$. Then there exist some constant M such that for all $\mathbf{v} \in \mathbb{Z}^m$ such that $\|\mathbf{v}\| < B$, the distribution of the following algorithm are within statistical distance $\varepsilon_{rej} = 2^{-\omega(logm)}/M$:*

- *Algorithm $\mathcal{A}$: $\mathbf{z} \leftarrow\$ \mathcal{D}_{\sigma,\mathbf{v}}^m$; Output $(\mathbf{z}, \mathbf{v})$ with probability* $\mathsf{RejSamp}(\sigma, M, \mathbf{z}, \mathbf{v})$
- *Algorithm $\mathcal{B}$: $\mathbf{z} \leftarrow\$ \mathcal{D}_{\sigma}^m$; Output $(\mathbf{z}, \mathbf{v})$ with probability $1/M$*

where $\mathsf{RejSamp}(\sigma, M, \mathbf{z}, \mathbf{v}) = \min(1, \mathcal{D}_{\sigma}^m(\mathbf{z})/M\mathcal{D}_{\sigma,\mathbf{v}}^m(\mathbf{z})))$.

Definition 1 ($\mathsf{MSIS}_{q,k,l,\beta}$ Problem). *The advantage of any algorithm $\mathcal{A}$ against the $\mathsf{MSIS}_{q,k,l,\beta}$ problem is defined as*

$$\boldsymbol{Adv}_{q,k,l,\beta}^{\mathsf{MSIS}} = Pr\left[[\mathbf{A}|\mathbf{I}]\mathbf{x} = \mathbf{0}, 0 < \|\mathbf{x}\| \leq \beta : \mathbf{A} \leftarrow\$ R_q^{k\times l}, \mathbf{x} \leftarrow \mathcal{A}(\mathbf{A}) \in R_q^{l+k}\right]$$

Definition 2 ($\mathsf{MLWE}_{q,k,l,\eta}$ Problem). *The advantage of any algorithm $\mathcal{A}$ against the $\mathsf{MLWE}_{q,k,l,\eta}$ problem is defined as*

$$\begin{aligned}\boldsymbol{Adv}_{q,k,l,\eta}^{\mathsf{MLWE}} =& |Pr[\mathcal{A}(\mathbf{A}, t) = 1 : \mathbf{A} \leftarrow\$ R_q^{k\times l}; \mathbf{s} \leftarrow\$ S_\eta^{l+k}; \mathbf{t} = [\mathbf{A}|\mathbf{I}]\mathbf{s}] \\ & - Pr[\mathcal{A}(\mathbf{A}, t) = 1 : \mathbf{A} \leftarrow\$ R_q^{k\times l}; \mathbf{t} \leftarrow\$ R_q^k]|\end{aligned}$$

where $S_\eta \subset R$ is defined as $S_\eta = \{\mathbf{x} \in R : \|\mathbf{x}\|_\infty \leq \eta\}$.

3 Multi-Signatures with Verifiable Key Aggregation

In this section, we revisit the notion of *multi-signatures with verifiable key aggregation* (MSvKA) from [LÖ24] and propose two extensions. First, we extend its syntax to cover interactive signing protocols and allow the proof of aggregation π to be an optional input to the signing phase, and adapt the security definitions accordingly. Second, we propose a new privacy notion (weak set privacy), that lies in between membership and set privacy, and captures the weakened version of set privacy as achieved by our lattice-based construction from Section 6.

Definition 3 (MS with verifiable key aggregation). *A multi-signature* (MS) *is a tuple of algorithms* $(\mathsf{Pg}, \mathsf{Kg}, \mathsf{KAg}, \mathsf{VfKAg}, \mathsf{MulSign}, \mathsf{Combine}, \mathsf{Vf})$ *such that:*

$\mathsf{Pg}(1^\lambda) \to \mathsf{pp}$**:** *On input security parameter 1^λ, it outputs public parameters* pp*. We only make* pp *explicit in key generation and assume it to be an implicit input to all other algorithms.*

$\mathsf{Kg}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{pk})$**:** *Probabilistic key generation, outputs a key pair* $(\mathsf{sk}, \mathsf{pk})$*.*

$\mathsf{KAg}(PK) \to (\mathsf{apk}, \pi)$**:** *(Possibly probabilistic) key aggregation, that on input a set of public keys $PK = \{\mathsf{pk}_i\}$, outputs an aggregated public key* apk *and a proof of aggregation π.*

$\mathsf{VfKAg}(PK, \mathsf{apk}, \pi) \to b$**:** *Checks if π is a valid proof of aggregation for PK and* apk *and outputs the boolean result for it.*

$\mathsf{MulSign}(\mathsf{sk}_i, PK, \mathsf{apk}, m, [\pi]) \to \mathsf{ps}_i$**:** *On input the secret key sk_i, message m, a set of public keys $PK = \{\mathsf{pk}_i\}$, and aggregated key* apk*, outputs a signature share ps_i. If* MulSign *is an interactive protocol of $\ell > 1$ rounds, we notate sub-algorithms $\mathsf{MulSign}_j$ of the signing protocol as follows.*

$$\begin{aligned}(\mathsf{st}_i^{(1)}, \mathsf{ps}_i^{(1)}) &\leftarrow \mathsf{MulSign}_1(\mathsf{sk}_i, PK, \mathsf{apk}, m, [\pi]) \\ (\mathsf{st}_i^{(j)}, \mathsf{ps}_i^{(j)}) &\leftarrow \mathsf{MulSign}_j(\mathsf{st}_i^{(j-1)}, \mathsf{in}^{(j-1)}) \\ \mathsf{ps}_i^{(\ell)} &\leftarrow \mathsf{MulSign}_\ell(\mathsf{st}_i^{(\ell-1)}, \mathsf{in}^{(\ell-1)})\end{aligned}$$

where $\mathsf{st}_i^{(j)}$ *corresponds to the internal state of the signer* i *at the end of round* j*,* $\mathsf{ps}_i^{(j)}$ *is the partial signature of the signer* i *in round* j*, and* $\mathsf{in}^{(j)}$ *is the set of partial signatures for round* j *for all signers in* PK*. We will omit* i *when this is clear from the context. We assume that* ℓ *is constant for each construction.*

$\mathsf{Combine}(PK, \{\mathsf{ps}_i\}_{\mathsf{pk}_i \in PK}, [m], \pi) \to \sigma$**:** *On input a set of public keys* $PK = \{\mathsf{pk}_i\}$ *and set of shares* $\{\mathsf{ps}_i\}_{\mathsf{pk}_i \in PK}$ *outputs a combined signature* σ *for* PK*.*

$\mathsf{Vf}(\mathsf{apk}, \sigma, m) \to b$**:** *Verifies if* σ *is a valid signature on* m *for* apk*.*

Security Properties. Below we define correctness, unforgeability and privacy.

Definition 4 (MS-Correctness). *A* MS *scheme* Π *is correct if for all* λ*,* m*,* n*, for all* $\mathsf{pp} \leftarrow \mathsf{Pg}(1^\lambda)$*,* $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{Kg}(\mathsf{pp})$ *for* $i \in [n]$*, and for all* $(\mathsf{apk}, \pi) \leftarrow \mathsf{KAg}\{\mathsf{pk}_i\}_{i\in[n]}$*,* $\mathsf{ps}_i \leftarrow \mathsf{MulSign}(\mathsf{sk}_i, \{\mathsf{pk}_i\}_{i\in[n]}, \mathsf{apk}, m)$ *for* $i \in [n]$*,*

$$\mathsf{VfKAg}(\{\mathsf{pk}_i\}_{i\in[n]}, \mathsf{apk}, \pi) = 1 \ \wedge \ \mathsf{Vf}(\mathsf{apk}, \mathsf{Combine}(\{\mathsf{pk}_i\}_{i\in[n]}, \pi, \{\mathsf{ps}_i\}_{i\in[n]}), m) = 1$$

Unforgeability. The unforgeability of multi-signatures with verifiable key aggregation [LÖ24] is defined in a similar game structure to the unforgeability of multi-signatures with deterministic key aggregation where the game simulates an honest signer. The adversary is allowed to interact with a signing oracle that takes a signer group PK, aggregated public key apk, proof of aggregation π, and a message m. The oracle checks if the honest signer is a member of the signer group, and the proof of aggregation holds. If both hold, the oracle runs signing protocol $\mathsf{MulSign}$ with the adversary. At the end of the game, the adversary needs to return a non-trivial/fresh forgery, which is checked by the predicate fresh.

Unforgeability of multi-signatures with verifiable key aggregation is defined in three levels where each level applies a different freshness check on the adversary's forgery. The weakest level, UNF-1, only ensures that it is hard to forge signatures on messages that were not signed. A stronger level (UNF-2) ensures that signatures are bound to the signer group. Finally, the strongest (UNF-3) provides guarantees for both the signer group and the aggregated key. We present the formal definition considering signing rounds explicitly. In Appendix C.4 we recall the original (non-interactive) definition from [LÖ24].

Definition 5 (MS Unforgeability). *A* MS *scheme* Π *is* UNF-X *for* $\mathsf{X} \in \{1, 2, 3\}$ *if for all PPT adversaries* $\mathcal{A}$ *in the experiment from Figure 1 it holds that:* $\Pr\left[\boldsymbol{Exp}_{\Pi,\mathcal{A}}^{\mathsf{MS\text{-}UNF\text{-}X}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$.

Privacy. We recap the privacy definitions from [LÖ24], which considers two models: the *all-but-one-public-key* (AbOPK) model that assumes there is a fresh public key that is unknown to the outsiders in each signing group, and the *known-public-key* (KPK) model that allows revealing all public keys in the signer group to the outsiders. The definitions below cover both models by expressing some additional restrictions for the AbOPK model.

Membership/Set Privacy. Set privacy ensures that an aggregated key does not reveal the underlying signer group to outsiders who do not know the key aggregation proof π. This is captured by giving a challenge aggregated key over one of two adversarially chosen signer sets to the adversary. Membership privacy is a weaker version of set privacy that only concerns on hiding an individual signer behind an aggregated public key. Similar to set privacy, this notion is defined by an indistinguishability game for a challenge aggregated public key. However, to ensure that other differences of the signer sets do not give adversaries trivial wins, the challenge sets can only differ by a single individual public key, which means they must have equal size and identical members except one.

Both privacy notions require the indistinguishability of aggregated public keys and corresponding signature. However, as discussed in Section 6, our (and possibly *any*) SIS lattice-based construction, leaks the size of the signer set with each signature. As a result, neither set privacy nor full privacy (as defined in the next paragraph) can be achieved by such constructions. That said, as our construction exceeds the privacy guarantees of membership privacy, we introduce a new variant: **weak set privacy** (wSetPriv), which lies in between the previous two notions. wSetPriv is similar to set privacy

$\boldsymbol{Exp}^{\mathsf{MS\text{-}UNF\text{-}X}}_{\Pi,\mathcal{A}}$

$\mathsf{pp} \leftarrow \mathsf{Pg}(1^\lambda); (\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{Kg}(\mathsf{pp}); S_1, \ldots, S_\ell, Q \leftarrow \emptyset$

$(\sigma, m, \mathsf{apk}, \pi, PK) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathtt{MulSign}_1, \ldots, \mathtt{MulSign}_\ell}}(\mathsf{pp}, \mathsf{pk}^*)$

return $\mathsf{Vf}(\mathsf{apk}, \sigma, m) \wedge \mathsf{VfKAg}(PK, \mathsf{apk}, \pi) \wedge \mathsf{pk}^* \in PK \wedge \mathsf{fresh}(m, PK, \mathsf{apk}, Q)$

$\mathcal{O}^{\mathtt{MulSign}_1}(\mathsf{sid}_k, PK_k, \mathsf{apk}_k, \pi_k, m_k)$

if $\mathsf{sid}_k \in S_1 \vee pk^* \notin PK_k \vee \mathsf{VfKAg}(PK_k, \mathsf{apk}_k, \pi_k) \neq 1$: **return** $\perp$

$S_1 \leftarrow S_1 \cup \{(\mathsf{sid}_k, m_k, PK_k, \mathsf{apk}_k)\}; (\mathsf{st}^{(k,1)}, \mathsf{ps}^{(k,1)}) \leftarrow \mathsf{MulSign}_1(sk^*, PK_k, \mathsf{apk}_k, \pi_k, m_k)$

return $\mathsf{ps}^{(k,1)}$

$\mathcal{O}^{\mathtt{MulSign}_j}(\mathsf{sid}_k, \mathsf{in}^{(k,j-1)}) // j \in \{2, ..., \ell\}$

if $\mathsf{sid}_k \in S_j \vee \mathsf{sid}_k \notin S_1, ..., S_{j-1}$: **return** $\perp$

$(\boxed{\mathsf{st}^{(k,j)}}, \mathsf{ps}^{(k,j)}) \leftarrow \mathsf{MulSign}_j(\mathsf{st}^{(k,j-1)}, \mathsf{in}^{(k,j-1)})$

Get $(\mathsf{sid}_k, m_k, PK_k, \mathsf{apk}_k) \in S_1$, set $Q \leftarrow Q \cup \{(m_k, PK_k, \mathsf{apk}_k)\}$;

$S_j \leftarrow S_j \cup \{\mathsf{sid}_k\}$; **return** $\mathsf{ps}^{(k,j)}$

UNF-X	UNF-1	UNF-2	UNF-3
$\mathsf{fresh}(m, PK, \mathsf{apk}, Q) = 1$ **if**	$(m, \cdot, \cdot) \notin Q$	$(m, PK, \cdot) \notin Q$	$(m, PK, \mathsf{apk}) \notin Q$

Fig. 1. Unforgeability experiment for multiple signing rounds. Code in the dashed box is only used for $j \in \{2, ..., \ell - 1\}$. Code in the gray box is only used for $j = \ell$.

$\boldsymbol{Exp}^{\mathsf{X\text{-}Y}}_{\Pi,\mathcal{A}}(\lambda)$

$b \leftarrow \{0,1\}, \mathsf{pp} \leftarrow \mathsf{Pg}(1^\lambda), Q \leftarrow \emptyset, n \leftarrow \mathcal{A}(\mathsf{pp})$, **abort if** $n \not> 0$

$(SK, PK) := (\{\mathsf{sk}_i\}_{i\in[n]}, \{\mathsf{pk}_i\}_{i\in[n]}) \leftarrow (\mathsf{Kg}(\mathsf{pp}))_{i\in[n]}$

$(S_0, S_1) \leftarrow \mathcal{A}(SK \setminus \{\mathsf{sk}_1\}, PK \setminus \{\mathsf{pk}_1\})$ **abort if** $1 \notin S_j$ **for** $j \in \{0,1\}$

$\boxed{\textbf{abort if } |S_0| \neq |S_1|}$ $\boxed{\textbf{abort if } |S_j \setminus S_{1-j}| \neq 1 \textbf{ for } j \in \{0,1\}}$

$(\mathsf{apk}_b, \pi_b) \leftarrow \mathsf{KAg}(PK_{S_b})$

$b^* \leftarrow \mathcal{A}^{\mathcal{O}^{\mathtt{Chl}}(\cdot)}(\mathsf{apk}_b)$, **return** 1 **if** $b = b^*$

$\mathcal{O}^{\mathtt{Chl}}(m)$

$\Sigma \leftarrow \{\mathsf{MulSign}(\mathsf{sk}_i, PK_{S_b}, \mathsf{apk}_b, m)\}_{\mathsf{sk}_i \in SK_{S_b}}$; **return** $\sigma \leftarrow \mathsf{Combine}(PK_{S_b}, \pi_b, \Sigma)$

Fig. 2. This is the game for our Set and Membership Privacy definitions ($\mathsf{X} \in \{\mathsf{MemPriv}, \mathsf{wSetPriv}, \mathsf{SetPriv}\}$) for models $\mathsf{Y} \in \{\mathsf{AbOPK}, \mathsf{KPK}\}$. Additional parts for $\mathsf{X} = \mathsf{wSetPriv}$ and $\mathsf{X} = \mathsf{MemPriv}$ are shown in boxed and dashed boxes, respectively. Grayed parts correspond to additions for $\mathsf{Y} = \mathsf{AbOPK}$.

but with the additional restriction that the adversary must pick sets of equal size. However, unlike membership privacy, no further restrictions are placed on the set members. In practice, this means that an adversary who interacts with the scheme can tell apart privacy-preserving groups of different size but nothing else.

Definition 6 (Set/Membership Privacy). *A* MS *scheme* Π *has property* $\mathsf{X} \in \{\mathsf{MemPriv}, \mathsf{wSetPriv}, \mathsf{SetPriv}\}$ *in model* $\mathsf{Y} \in \{\mathsf{AbOPK}, \mathsf{KPK}\}$*, if for all PPT adversaries* $\mathcal{A}$ *in* $\boldsymbol{Exp}^{\mathsf{X\text{-}Y}}_{\Pi,\mathcal{A}}$ *from Figure 2*: $|\Pr\left[\boldsymbol{Exp}^{\mathsf{X}}_{\Pi,\mathcal{A}}(\lambda) = 1\right] - 1/2| \leq \mathsf{negl}(\lambda)$.

Full Privacy. The strongest privacy notion that [LÖ24] defined is full privacy. This notion ensures that an aggregated public key does not even reveal to the outsiders if it is an aggregated key or an individual public key. Similar to the previous privacy properties, this is modeled as an indistinguishability game between an individual public key and an aggregated public key over an adversarially chosen signer group. One further requirement to define this property is setting an individual signing algorithm Sign. This algorithm should be outputting valid signatures corresponding to the individual public keys so that the game can simulate signatures for the challenge public key.

$\boldsymbol{Exp}^{\mathsf{FullPriv\text{-}KPK}}_{\Pi,\mathcal{A}}(\lambda)$

$b \leftarrow \{0,1\}, \mathsf{pp} \leftarrow \mathsf{Pg}(1^\lambda), Q \leftarrow \emptyset, n \leftarrow \mathcal{A}(\mathsf{pp})$, **abort if** $n \not> 0$

$(SK, PK) \leftarrow (\{\mathsf{sk}_i\}_{i\in[n]}, \{\mathsf{pk}_i\}_{i\in[n]}) \leftarrow (\mathsf{Kg}(\mathsf{pp}))_{i\in[n]}$

$S^* \leftarrow \mathcal{A}(SK \setminus \{\mathsf{sk}_1\}, PK \setminus \{\mathsf{pk}_1\})$ **abort if** $1 \notin S^*$

if $b = 0$ **then**

$(\mathsf{apk}^*, \pi^*) \leftarrow \mathsf{KAg}(PK_{S^*}), \mathsf{pk}^* := \mathsf{apk}^*$

if $b = 1$: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Kg}(\mathsf{pp}), \mathsf{pk}^* := \mathsf{pk}$

$b^* \leftarrow \mathcal{A}^{\mathcal{O}^{\mathtt{Chl}}(\cdot)}(\mathsf{pk}^*)$, **return** 1 **if** $b = b^*$

$\mathcal{O}^{\mathtt{Chl}}(m)$

if $b = 1$: **return** $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$

$\Sigma \leftarrow \{\mathsf{MulSign}(\mathsf{sk}_i, PK_{S^*}, \mathsf{apk}^*, m)\}_{\mathsf{sk}_i \in SK_{S^*}}$

return $\sigma \leftarrow \mathsf{Combine}(PK_{S^*}, \pi^*, \Sigma)$

Fig. 3. Our FullPriv game capturing that aggregate signatures and keys are indistinguishable from standard ones. Grayed parts correspond to additions for AbOPK model.

Definition 7 (Full Privacy). *A* MS *scheme Π is fully private for a signing algorithm* Sign *in the* KPK *model if for all PPT adversaries $\mathcal{A}$ in $\boldsymbol{Exp}^{\mathsf{FullPriv\text{-}KPK}}_{\Pi,\mathcal{A}}$ defined in Figure 3 it holds that* $\left|\Pr[\boldsymbol{Exp}^{\mathsf{FullPriv\text{-}KPK}}_{\Pi,\mathcal{A}}(\lambda) = 1] - 1/2\right| \leq \mathsf{negl}(\lambda)$.

4 Lifting Privacy from AbOPK to KPK Generically

In this section, we present a generic technique that converts any AbOPK-secure scheme, where privacy relies on the secrecy of at least one public key, into a KPK-secure scheme. As all existing multi-signatures, apart from [LÖ24], only achieve AbOPK privacy due to their deterministic key aggregation, this gives a direct transformation to lift their weak privacy into the stronger KPK setting. Our construction yields strong privacy guarantees, and also preserves the unforgeability of the underlying scheme. The only drawback of our generic lift is that it requires signers to use the key aggregation proof π already in the signing protocol, instead of needing it solely in the final aggregation.

We start by describing our generic construction, and then prove that it preserves UNF-1 security and boosts privacy from the AbOPK to KPK setting.

4.1 Generic Construction Π_{KPK}

The main idea behind our construction is to sample a fresh *dummy* key pair in every generation of a new aggregate key, and keeping the full dummy key (*i.e.,* also public key) "secret" within the group of signers. The underlying scheme gets then boosted to better privacy, by additionally including a signature under the fresh dummy key. The full description of our lifted construction Π_{KPK} is shown in Figure 4. We recall the intuition outlined in the technical overview below.

Verifiable Key Aggregation. The privacy model AbOPK assumes that there is an unknown public key in the signer group, which we mimic in our construction: When $\Pi_{\mathsf{KPK}}.\mathsf{KAg}$ generates an aggregated public key for a signer group PK, it samples an additional fresh dummy key pair $(\mathsf{dsk}, \mathsf{dpk})$, and includes the dummy public key dpk into the Π_{AbOPK} derivation of apk.

The dummy key pair is created by explicitly fixing the random coins of the $\Pi_{\mathsf{AbOPK}}.\mathsf{Kg}$ algorithm with the hash value $\mathsf{H}_{dm}(\mathsf{seed}, PK)$ for a fresh seed. This extra step in the dummy key pair generation will help us to ensure the unforgeability of Π_{KPK}. The secret value seed is kept internal to the group, by making it part of the aggregation proof π_{KPK}. Further, $\Pi_{\mathsf{KPK}}.\mathsf{KAg}$ also samples a fresh PRF key k and extends the proof of Π_{AbOPK} as $\pi_{\mathsf{KPK}} \leftarrow (\mathsf{seed}, \mathsf{k}, \pi_{\mathsf{AbOPK}})$. The PRF key will be used to simulate the partial signatures of dsk. Verification of an aggregated key apk first re-computes dpk using seed, and then runs the key verification of Π_{AbOPK} using π_{AbOPK} on the extended signer set, including dpk.

Signature Generation. The remaining challenge is to create valid signatures for the aggregated public key, which contains the contributions under the dummy public key dpk. All signers of apk already know seed, so they can locally compute dsk and run the signing protocol for dsk on their own. However, to ensure the correctness of the signing protocol, all signers must compute the identical dummy signature shares. We achieve this by letting all signers derive the dummy signature in a *deterministic* manner. To do so, Π_{KPK} signers use the PRF key k to derive the random coins of the signing algorithms for

dsk. To ensure the freshness of these values, the signers run an extra round to sample nonces nonce_i for each signer i in the signing session. Subsequently, they set the random tape of the dummy signer $\rho_1, \ldots, \rho_\ell \leftarrow F_{\mathsf{k}}(\mathsf{nonce}_1 \| \ldots \| \mathsf{nonce}_{|PK|})$.

It might be tempting to simply hardcode the dummy signer's random coins into the construction. However, this would not allow for a generic boost, as we need to internally "simulate" a proper AbOPK setting, where all signers – including the dummy one – run a normal signing protocol.

4.2 Unforgeability of Π_{KPK}

We show that Π_{KPK} satisfies the same unforgeability guarantees as Π_{AbOPK}. In particular if Π_{AbOPK} is $\mathsf{UNF\text{-}X}$ secure, Π_{KPK} is also $\mathsf{UNF\text{-}X}$ secure.

Let us first sketch the UNF-1 security of Π_{KPK}. Intuitively, as dsk is always treated as a normal secret key input towards the Π_{AbOPK} algorithms, any Π_{KPK} forgery for the signer set PK and $\pi_{\mathsf{KPK}} \leftarrow (\mathsf{seed}, \mathsf{k}, \pi_{\mathsf{AbOPK}})$ can be converted into a Π_{AbOPK} forgery for $PK \cup \{\mathsf{dpk}\}$ and π_{AbOPK}. Further, in the unforgeability game, the adversary is in full control over all secret keys except the challenge key anyway, and thus "leaking" the secret key dsk to the group of signers as part of π_{KPK} does not give the adversary an advantage in Π_{KPK}.

UNF-2, 3 *Security of* Π_{KPK}. The unforgeability of Π_{KPK} for $\mathsf{X} \in \{2, 3\}$ requires further argumentation. This is achieved by the specific way we generate the dsk in $\Pi_{\mathsf{KPK}}.\mathsf{KAg}$. We first show that the naive way of sampling the dummy key pair by running $\Pi_{\mathsf{AbOPK}}.\mathsf{Kg}$ without relying on H_{dm} cannot satisfy UNF-2, 3 security.

If $\Pi_{\mathsf{KPK}}.\mathsf{KAg}$ sets the dummy key pair as $(\mathsf{dsk}, \mathsf{dpk}) \leftarrow \Pi_{\mathsf{AbOPK}}.\mathsf{Kg}(\mathsf{pp})$, then there is no way of differentiating dpk from the real public keys in the key aggregation and signing protocol. This can be exploited in the stronger unforgeability games that also guarantee the immutability of the designated signer set.

Assume that the adversary $\mathcal{A}$, playing against the Π_{KPK} scheme, computes a key pair $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{Kg}(\mathsf{pp})$, setting a dummy key pair $(\mathsf{dsk}, \mathsf{dpk}) \leftarrow \mathsf{Kg}(\mathsf{pp})$ and a PRF key k. Subsequently, $\mathcal{A}$ computes $(\mathsf{apk}, \pi_{\mathsf{AbOPK}}) \leftarrow \Pi_{\mathsf{AbOPK}}.\mathsf{KAg}(PK \cup \{\mathsf{dpk}\})$ and sets $\pi_{\mathsf{KPK}} \leftarrow (\mathsf{dsk}, \mathsf{k}, \pi_{\mathsf{AbOPK}})$, $PK \leftarrow \{\mathsf{pk}^*, \mathsf{pk}_1\}$. It runs the signing protocol honestly with the Π_{KPK} challenger on an arbitrary message m, and for PK, π_{KPK}, obtaining a signature σ. Now, the adversary $\mathcal{A}$ can swap the meaning of pk_1 and dpk and claim a fresh forgery for the signer set $PK' \leftarrow \{\mathsf{pk}^*, \mathsf{dpk}\}$ and $\pi'_{\mathsf{KPK}} \leftarrow (\mathsf{sk}_1, \mathsf{k}, \pi_{\mathsf{AbOPK}})$. This is a non-trivial forgery against Π_{KPK} in the UNF-2, 3 games, which require (at least) the tuple (m, PK) to be fresh. However, any reduction against the UNF-2, 3 security of Π_{AbOPK} must make a signing query for the full signer set $\{\mathsf{pk}^*, \mathsf{pk}_1, \mathsf{dpk}\}$, and cannot use $\mathcal{A}$'s "forgery".

Our lifted scheme Π_{KPK} prevents such attacks. This is achieved by committing to the set of real signers for a dummy key pair by setting the random coins to $\mathsf{H}_{dm}(\mathsf{seed}, PK)$. When H_{dm} is modeled as a random oracle, we can ensure that finding Π_{AbOPK} signer set collisions by exploiting dpk is unlikely. Note that our construction only fixes the random coins of the $\Pi_{\mathsf{AbOPK}}.\mathsf{Kg}$ algorithm, but it does not make any assumptions on the output distribution of $\Pi_{\mathsf{AbOPK}}.\mathsf{Kg}$. This allows our lift to cover the multi-signature schemes that have non-uniform distribution of key pairs such as lattice-based multi-signature schemes.

Theorem 1. *If* Π_{AbOPK} *is a* $\mathsf{UNF\text{-}X}$ *secure* MS *scheme and* H_{dm} *is a random oracle, then* Π_{KPK} *in Figure 4 is* $\mathsf{UNF\text{-}X}$ *secure.*

Proof. Assume that $\mathcal{A}$ is an efficient $\mathsf{UNF\text{-}X}$ adversary against Π_{KPK}. We then build an efficient adversary $\mathcal{B}$ that breaks the $\mathsf{UNF\text{-}X}$ security of Π_{AbOPK} as follows. $\mathcal{B}$ gets the public parameters pp and the challenge public key pk^* from $\boldsymbol{Exp}^{\mathsf{MS\text{-}UNF\text{-}X}}_{\Pi_{\mathsf{AbOPK}}}$ and forwards them to $\mathcal{A}$. For the simulation of the signing queries in the Π_{KPK} game, $\mathcal{B}$ behaves as follows, depending on the signing round:

Queries to $\mathcal{O}^{\mathtt{H_{dm}}}(\mathsf{seed}, PK)$**:** $\mathcal{B}$ chooses $t \leftarrow\!\!\$\ \{0,1\}^l$ and computes $(\mathsf{dsk}, \mathsf{dpk}) \leftarrow \Pi_{\mathsf{AbOPK}}.\mathsf{Kg}(\mathsf{pp}; t)$. If $\{\mathsf{dpk}\} \cup PK \in S_{dm}$ then the event **SetColl** occurs and $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ sets $S_{dm} \leftarrow S_{dm} \cup \{\{\mathsf{dpk}\} \cup PK\}$ and returns t.

1st round $\Pi_1.\mathcal{O}^{\mathtt{MulSign_1}}(\mathsf{sid}_k, PK_k, \mathsf{apk}_k, \pi_{k,1}, m_k)$**:** $\mathcal{A}_0$ verifies that the input is well-formed, in particular that apk_k is a valid aggregated key for $PK_k \cup \{\mathsf{dpk}\}$ and π_{AbOPK}, taken from $\pi_{\mathsf{KPK}} \leftarrow (\mathsf{seed}, \mathsf{k}, \pi_{\mathsf{AbOPK}})$ and $(\mathsf{dsk}_k, \mathsf{dpk}_k) \leftarrow \Pi_{\mathsf{AbOPK}}.\mathsf{Kg}(\mathsf{pp}, \mathsf{H}_{dm}(\mathsf{seed}, PK_k))$. If all checks pass, it runs $\Pi_{\mathsf{KPK}}.\mathsf{MulSign}_1$ honestly, which simply outputs a nonce and is independent of the secret key. It stores $(PK_k, \mathsf{apk}_k, \pi_{k,\mathsf{KPK}}, m_k, \mathsf{dsk}_k, \mathsf{dpk}_k)$ for the later rounds.

$\underline{\Pi_{\mathsf{KPK}}.\mathsf{KAg}(PK)}$

$\mathsf{seed} \leftarrow\!\!\$\ \{0,1\}^\lambda,\ \mathsf{k} \leftarrow\!\!\$\ \mathcal{K}$

$(\cdot, \mathsf{dpk}) \leftarrow \Pi_{\mathsf{AbOPK}}.\mathsf{Kg}(\mathsf{pp}; \mathsf{H}_{dm}(\mathsf{seed}, PK))$

$(\mathsf{apk}, \pi_{\mathsf{AbOPK}}) \leftarrow \Pi_{\mathsf{AbOPK}}.\mathsf{KAg}(PK \cup \{\mathsf{dpk}\})$

return $(\mathsf{apk}, \pi_{\mathsf{KPK}} \leftarrow (\mathsf{seed}, \mathsf{k}, \pi_{\mathsf{AbOPK}}))$

$\underline{\Pi_{\mathsf{KPK}}.\mathsf{VfKAg}(PK, \mathsf{apk}, \pi_{\mathsf{KPK}})}$

Parse π_{KPK} **as** $(\mathsf{seed}, \mathsf{k}, \pi_{\mathsf{AbOPK}})$

$(\cdot, \mathsf{dpk}) \leftarrow \Pi_{\mathsf{AbOPK}}.\mathsf{Kg}(\mathsf{pp}; \mathsf{H}_{dm}(\mathsf{seed}, PK))$

$PK' \leftarrow PK \cup \{\mathsf{dpk}\}$

return $\Pi_{\mathsf{AbOPK}}.\mathsf{VfKAg}(PK', \mathsf{apk}, \pi_{\mathsf{AbOPK}})$

$\underline{\Pi_{\mathsf{KPK}}.\mathsf{MulSign}_1(\mathsf{sk}_i, PK, \mathsf{apk}, m, [\pi_{\mathsf{KPK}}])}$

$\mathsf{nonce}_i \leftarrow\!\!\$\ \{0,1\}^\lambda$, **return** $(\mathsf{st}^{(1)} \leftarrow (\mathsf{sk}_i, PK, \mathsf{apk}, m, \pi_{\mathsf{KPK}}), \mathsf{ps}^{(1)} \leftarrow \mathsf{nonce}_i)$

$\underline{\Pi_{\mathsf{KPK}}.\mathsf{MulSign}_2(\mathsf{st}^{(1)}, \mathsf{in}^{(1)}_{\mathsf{KPK}})}$

Parse $\mathsf{st}^{(1)}$ **as** $(\mathsf{sk}_i, PK, \mathsf{apk}, m, \pi_{\mathsf{KPK}}),\ \mathsf{in}^{(1)}_{\mathsf{KPK}}$ **as** $\{\mathsf{nonce}_j\}_{j\in|PK|}$

Parse π_{KPK} **as** $(\mathsf{seed}, \mathsf{k}, \pi_{\mathsf{AbOPK}})$

$(\mathsf{dsk}, \mathsf{dpk}) \leftarrow \Pi_{\mathsf{AbOPK}}.\mathsf{Kg}(\mathsf{pp}; \mathsf{H}_{dm}(\mathsf{seed}, PK)),\ \rho_1, ..., \rho_\ell \leftarrow F_{\mathsf{k}}(\mathsf{nonce}_1 \| \ldots \| \mathsf{nonce}_{|PK|})$

$(\mathsf{st}^{(1)}_{rl}, \mathsf{ps}^{(1)}_{rl}) \leftarrow \Pi_{\mathsf{AbOPK}}.\mathsf{MulSign}_1(\mathsf{sk}_i, PK \cup \{\mathsf{dpk}\}, \mathsf{apk}, m, \pi_{\mathsf{AbOPK}})$

$(\mathsf{st}^{(1)}_{dm}, \mathsf{ps}^{(1)}_{dm}) \leftarrow \Pi_{\mathsf{AbOPK}}.\mathsf{MulSign}_1(\mathsf{dsk}, PK \cup \{\mathsf{dpk}\}, \mathsf{apk}, m, \pi_{\mathsf{AbOPK}}; \rho_1)$

return $(\mathsf{st}^{(2)} \leftarrow (\mathsf{st}^{(1)}_{rl}, \mathsf{st}^{(1)}_{dm}, \mathsf{ps}^{(1)}_{dm}, (\rho_2, \ldots, \rho_\ell)), \mathsf{ps}^{(2)} \leftarrow \mathsf{ps}^{(1)}_{rl})$

$\underline{\Pi_{\mathsf{KPK}}.\mathsf{MulSign}_j(\mathsf{st}^{(j-1)}, \mathsf{in}^{(j-1)}_{\mathsf{KPK}})}$

Parse $\mathsf{st}^{(j-1)}$ **as** $(\mathsf{st}^{(j-2)}_{rl}, \mathsf{st}^{(j-2)}_{dm}, \mathsf{ps}^{(j-2)}_{dm}, (\rho_j, \ldots, \rho_\ell))$

$\mathsf{in}^{(j-2)}_{\mathsf{AbOPK}} \leftarrow \mathsf{in}^{(j-1)}_{\mathsf{KPK}} \cup \{\mathsf{ps}^{(j-2)}_{dm}\}$

$(\mathsf{st}^{(j-1)}_{rl}, \mathsf{ps}^{(j-1)}_{rl}) \leftarrow \Pi_{\mathsf{AbOPK}}.\mathsf{MulSign}_{j-1}(\mathsf{st}^{(j-2)}_{rl}, \mathsf{in}^{(j-2)}_{\mathsf{AbOPK}})$

$(\mathsf{st}^{(j-1)}_{dm}, \mathsf{ps}^{(j-1)}_{dm}) \leftarrow \Pi_{\mathsf{AbOPK}}.\mathsf{MulSign}_{j-1}(\mathsf{st}^{(j-2)}_{dm}, \mathsf{in}^{(j-2)}_{\mathsf{AbOPK}}; \rho_{j-1})$

return $(\mathsf{st}^{(j)} \leftarrow (\mathsf{st}^{(j-1)}_{rl}, \mathsf{st}^{(j-1)}_{dm}, \mathsf{ps}^{(j-1)}_{dm}, (\rho_j, \ldots, \rho_\ell)), \mathsf{ps}^{(j)} \leftarrow \mathsf{ps}^{(j-1)}_{rl})$

$\underline{\Pi_{\mathsf{KPK}}.\mathsf{MulSign}_{\ell+1}(\mathsf{st}^{(\ell)}, \mathsf{in}^{(\ell)}_{\mathsf{KPK}})}$

Parse $\mathsf{st}^{(\ell)}$ **as** $(\mathsf{st}^{(\ell-1)}_{rl}, \mathsf{st}^{(\ell-1)}_{dm}, \mathsf{ps}^{(\ell-1)}_{dm}, \rho_\ell)$

$\mathsf{in}^{(\ell-1)}_{\mathsf{AbOPK}} \leftarrow \mathsf{in}^{(\ell)}_{\mathsf{KPK}} \cup \{\mathsf{ps}^{(\ell-1)}_{dm}\}$

$\mathsf{ps}^{(\ell)}_{rl} \leftarrow \Pi_{\mathsf{AbOPK}}.\mathsf{MulSign}_\ell(\mathsf{st}^{(\ell-1)}_{rl}, \mathsf{in}^{(\ell-1)}_{\mathsf{AbOPK}})$

$\mathsf{ps}^{(\ell)}_{dm} \leftarrow \Pi_{\mathsf{AbOPK}}.\mathsf{MulSign}_\ell(\mathsf{st}^{(\ell-1)}_{dm}, \mathsf{in}^{(\ell-1)}_{\mathsf{AbOPK}}; \rho_\ell)$

return $\mathsf{ps}^{(\ell+1)} \leftarrow (\mathsf{ps}^{(\ell)}_{rl}, \mathsf{ps}^{(\ell)}_{dm})$

$\underline{\Pi_{\mathsf{KPK}}.\mathsf{Combine}(PK, \{\mathsf{ps}^{(\ell+1)}_i\}_{\mathsf{pk}_i\in PK}, [m], \pi_{\mathsf{KPK}})}$

Parse π_{KPK} **as** $(\mathsf{seed}, \cdot, \pi_{\mathsf{AbOPK}}),\ \mathsf{ps}^{(\ell+1)}_i$ **as** $(\mathsf{ps}^{(\ell)}_{rl,i}, \mathsf{ps}^{(\ell)}_{dm})$ **for** $\mathsf{pk}_i \in PK$

// Wlog, $\mathsf{ps}^{(\ell)}_{dm}$ values are equal, we don't separately index them.

$(\mathsf{dsk}, \mathsf{dpk}) \leftarrow \Pi_{\mathsf{AbOPK}}.\mathsf{Kg}(\mathsf{pp}; \mathsf{H}_{dm}(\mathsf{seed}, PK)),\ PK' \leftarrow PK \cup \{\mathsf{dpk}\}$

return $\Pi_{\mathsf{AbOPK}}.\mathsf{Combine}(PK', \{\mathsf{ps}^{(\ell)}_{dm}\} \cup \{\mathsf{ps}^{(\ell)}_{rl,i}\}_{\mathsf{pk}_i\in PK}, [m], \pi_{\mathsf{AbOPK}})$

Fig. 4. Lift from FullPriv-AbOPK to FullPriv-KPK. F is a PRF with the key space $\mathcal{K}$ and the output space which is big enough to assign for $\rho_1, ..., \rho_\ell$. The hash function H_{dm} is defined over $\mathsf{H}_{dm} : \{0,1\}^* \to \{0,1\}^l$ where l is the length of the random tape that $\Pi_{\mathsf{AbOPK}}.\mathsf{Kg}$ needs. $\Pi_1.\mathsf{Pg}$, $\Pi_1.\mathsf{Kg}$, and $\Pi_1.\mathsf{Vf}$ are identical to Π_0.

2nd round $\Pi_1.\mathcal{O}^{\mathtt{MulSign_2}}(\mathsf{sid}_k, \mathsf{in}_{\mathsf{KPK}}^{(k,1)})$**:** If $\mathsf{sid}_k \in S_2$ or $\mathsf{sid}_k \notin S_1$, then $\mathcal{B}$ returns $\perp$ to $\mathcal{A}_1$. Otherwise, $\mathcal{B}$ parses $\pi_{k,\mathsf{KPK}}$ as $(\cdot, \mathsf{k}_k, \pi_{k,\mathsf{AbOPK}})$ and $\mathsf{in}_{\mathsf{KPK}}^{(k,1)}$ as $\{\mathsf{nonce}_j\}_{j \in PK_k}$. It computes $\rho_{k,1}, \ldots, \rho_{k,\ell} \leftarrow F_{\mathsf{k}_k}(\mathsf{nonce}_1 \| \ldots \| \mathsf{nonce}_{|PK|})$ and sets $PK'_k \leftarrow PK_k \cup \{\mathsf{dpk}_k\}$.

$\mathcal{B}$ then queries its own round-1 signing oracle and obtains the partial signature $\mathsf{ps}_{rl}^{k,1} \leftarrow \Pi_{\mathsf{AbOPK}}.\mathcal{O}^{\mathtt{MulSign_1}}(\mathsf{sid}_k, PK'_k, \mathsf{apk}_k, \pi_{k,0}, m_k)$. Observe that if $\pi_{k,\mathsf{KPK}}$ is a valid Π_{KPK} proof for apk_k and PK_k, then $\pi_{k,\mathsf{AbOPK}}$ is a valid Π_{AbOPK} proof for apk_k and PK'_k. $\mathcal{B}$ computes $\mathsf{ps}_{dm}^{k,1}$ using dsk_k and $\rho_{k,1}$ and stores it to be used in the future oracle calls. Finally, $\mathcal{B}$ returns $\mathsf{ps}_{rl}^{k,1}$ to $\mathcal{A}$.

jth round $\Pi_1.\mathcal{O}^{\mathtt{MulSign_j}}(\mathsf{sid}_k, \mathsf{in}_{\mathsf{KPK}}^{(k,j-1)})$**,** $2 < j$**:** $\mathcal{B}$ first checks if $\mathsf{sid}_k \in S_j$ or $\mathsf{sid}_k \notin S_1, ..., S_{j-1}$, and returns $\perp$ to $\mathcal{A}$ if they hold. Otherwise, $\mathcal{B}$ looks up $\mathsf{ps}_{dm}^{k,j-2}$ that it computed in the oracle call of the previous round and sets $\mathsf{in}_{\mathsf{AbOPK}}^{(k,j-2)} \leftarrow \mathsf{in}_{\mathsf{KPK}}^{(k,j-1)} \cup \{\mathsf{ps}_{dm}^{j-2,1}\}$. Then $\mathcal{B}$ makes a query to its own signing oracle, obtaining $\mathsf{ps}_{rl}^{k,j-1} \leftarrow \Pi_{\mathsf{AbOPK}}.\mathcal{O}^{\mathtt{MulSign_{j-1}}}(\mathsf{sid}_k, \mathsf{in}_0^{(k,j-2)})$. $\mathcal{B}$ also computes $\mathsf{ps}_{dm}^{k,j-1}$ using $\rho_{k,j-1}$. If $j < \ell + 1$, then $\mathcal{B}$ returns $\mathsf{ps}_{rl}^{k,j-1}$ to $\mathcal{A}$. If $j = \ell + 1$, then $\mathcal{B}$ returns $(\mathsf{ps}_{rl}^{k,\ell}, \mathsf{ps}_{dm}^{k,\ell})$ to $\mathcal{A}$.

When $\mathcal{A}$ outputs its Π_{KPK} forgery $(\sigma, m, \mathsf{apk}, \pi_{\mathsf{KPK}}, PK_{\mathsf{KPK}})$, $\mathcal{B}$ parses the key aggregation proof π_{KPK} as $(\mathsf{seed}, k, \pi_{\mathsf{AbOPK}})$ and computes the Π_{AbOPK} forgery as $(\sigma, m, \mathsf{apk}, \pi_{\mathsf{AbOPK}}, PK_{\mathsf{AbOPK}})$ for $(\cdot, \mathsf{dpk}) \leftarrow \Pi_{\mathsf{AbOPK}}.\mathsf{Kg}(\mathsf{pp}, \mathsf{H}_{dm}(\mathsf{seed}, PK_{\mathsf{KPK}}))$ and $PK_{\mathsf{AbOPK}} \leftarrow PK_{\mathsf{KPK}} \cup \{\mathsf{dpk}\}$. $\mathcal{B}$'s simulation of $\mathcal{A}$'s view is perfect as long as $\mathcal{B}$ does not abort. Further, we argue that $\mathcal{B}$ aborts only with a negligible probability. The event **SetColl** can only occur when a freshly computed dpk already exists in one of the previously bookkept sets in S_{dm}. More formally, for a freshly computed dpk, there must be a $PK'' \in S_{dm}$ such that $\mathsf{dpk} \in PK''$. As t is sampled uniformly in $\mathcal{O}^{\mathtt{H_{dm}}}$, dpk that $\mathcal{O}^{\mathtt{H_{dm}}}$ computes is a fresh run of $\Pi_{\mathsf{AbOPK}}.\mathsf{Kg}$. Thus, we will rely on the probability that $\Pi_{\mathsf{AbOPK}}.\mathsf{Kg}$ creates colluding public keys to bound the probability that **SetColl** occurs. Let μ be the maximum probability that a certain pk is output by $\Pi_{\mathsf{AbOPK}}.\mathsf{Kg}$, in essence $\mu = \max_{\mathsf{pk}}(\Pr[(\cdot, \mathsf{pk}') \leftarrow \Pi_{\mathsf{AbOPK}}.\mathsf{Kg}(\mathsf{pp}) : \mathsf{pk} = \mathsf{pk}'])$. μ is a negligible probability as Π_{AbOPK} is an unforgeable multi-signature scheme. Let further q_H be the number of $\mathcal{O}^{\mathtt{H_{dm}}}$ queries that $\mathcal{A}$ makes and n_{max} be the size of the largest PK that $\mathcal{A}$ queries $\mathcal{O}^{\mathtt{H_{dm}}}$ with. Then, for each $PK'' \in S_{dm}$, freshly computed dpk can be a member of PK'' only with probability $(n_{max}+1)\cdot\mu$. As there are at most q_H values in S_{dm}, we conclude that $\Pr[\mathbf{SetColl}] \leq q_H^2 \cdot (n_{max}+1)\cdot\mu$.

Now we argue that if $\mathcal{A}$ wins $\mathsf{UNF\text{-}X}$ against Π_{KPK} without $\mathcal{B}$ aborting, then $\mathcal{B}$ wins $\boldsymbol{Exp}^{\mathsf{MS\text{-}UNF\text{-}X}}_{\Pi_{\mathsf{AbOPK}}}$. We evaluate the winning condition for each $\mathsf{X} \in \{1, 2, 3\}$. Let Q_{KPK} and Q_{AbOPK} be the bookkeeping sets of the $\mathsf{UNF\text{-}X}$ game for Π_{KPK} and Π_{AbOPK}, respectively.

$\mathsf{UNF\text{-}1}$**:** As $\mathcal{B}$ makes queries for the exact same messages as $\mathcal{A}$ makes, if $(m, \cdot, \cdot) \notin Q_{\mathsf{KPK}}$, then $(m, \cdot, \cdot) \notin Q_{\mathsf{AbOPK}}$.

$\mathsf{UNF\text{-}2}$**:** Assume that $(m, PK_{\mathsf{KPK}}, \cdot) \notin Q_{\mathsf{KPK}}$, but $(m, PK_{\mathsf{AbOPK}}, \cdot) \in Q_{\mathsf{AbOPK}}$. This means that there is a $\mathsf{dpk}' \in PK_{\mathsf{AbOPK}}$ which is created in a $\mathcal{O}^{\mathtt{H_{dm}}}$ query such that $PK_{\mathsf{AbOPK}} \setminus \{\mathsf{dpk}'\} \neq PK_{\mathsf{KPK}}$. This implies that the event **SetColl** occurred with $PK_{\mathsf{AbOPK}} \in S_{dm}$ for some $\mathcal{O}^{\mathtt{H_{dm}}}$ query.

$\mathsf{UNF\text{-}3}$ Assume that $(m, PK_{\mathsf{KPK}}, \mathsf{apk}) \notin Q_{\mathsf{KPK}}$, but $(m, PK_{\mathsf{AbOPK}}, \mathsf{apk}) \in Q_{\mathsf{AbOPK}}$. As in the case of $\mathsf{UNF\text{-}2}$, this case only occurs if the event **SetColl** occurs.

Thus, $\Pr\left[\boldsymbol{Exp}^{\mathsf{MS\text{-}UNF\text{-}X}}_{\Pi_{\mathsf{KPK}},\mathcal{A}}(\lambda) = 1\right] \leq \Pr\left[\boldsymbol{Exp}^{\mathsf{MS\text{-}UNF\text{-}X}}_{\Pi_{\mathsf{AbOPK}},\mathcal{B}}(\lambda) = 1\right] + q_H^2 \cdot (n_{max} + 1) \cdot \mu.$ □

4.3 Privacy Lift of Π_{KPK}

We now show that our generic construction delivers its promise, by proving that Π_{KPK} is $\mathsf{X\text{-}KPK}$-secure based on the $\mathsf{X\text{-}AbOPK}$ security of Π_{AbOPK}. In a nutshell, we leverage the fact that π of the challenge key(s) is unknown to the adversary, and thus, it neither knows the underlying dummy (secret) key, nor the PRF key k. This allows us to replace F_{k} with a truly random function, and simulate the added signatures for dpk through the unknown public key of the $\mathsf{X\text{-}AbOPK}$ game.

Theorem 2. *If Π_{AbOPK} is $\mathsf{X\text{-}AbOPK}$ secure and F is a secure PRF, then Π_{KPK} is $\mathsf{X\text{-}KPK}$ secure.*

Proof. We write down the full proof for $\mathsf{X} = \mathsf{FullPriv}$. For the other properties, the proof can be adapted easily by following the same game-hops for π_b instead of π^* as we do for $\mathsf{X} = \mathsf{FullPriv}$.

We present a small sequence of indistinguishable games and give a reduction to the $\mathsf{FullPriv\text{-}AbOPK}$ property of Π_{AbOPK} for the final game. Intuitively, the first two games make the changes to answer

the FullPriv-KPK challenge signing queries with the FullPriv-AbOPK's challenge signing oracle in our final reduction. Subsequently, the third game makes the necessary change to simulate the dummy public key of FullPriv-KPK with the unknown public key of the FullPriv-AbOPK game in our reduction. For clarity, the FullPriv-KPK adversary against Π_{KPK} is called $\mathcal{A}_{\mathsf{KPK}}$. Game_0 is identical to the FullPriv-KPK game played for Π_{KPK}. Furthermore, let W_i be the event that $\mathcal{A}_{\mathsf{KPK}}$ wins Game_i, so $\Pr\left[\boldsymbol{Exp}^{\mathsf{FullPriv\text{-}KPK}}_{\Pi_{\mathsf{KPK}},\mathcal{A}_{\mathsf{KPK}}}(\lambda)=1\right] = \Pr[W_0]$. Throughout out the proof, let the challenge key aggregation proof of FullPriv-KPK game be $\pi^*_{\mathsf{KPK}} \leftarrow (\mathsf{seed}^*, \mathsf{k}^*, \pi^*_{\mathsf{AbOPK}})$.

Game_1: We replace F_{k^*} with a truly random function $f(\cdot)$. This change is indistinguishable by the pseudorandomness of F, *i.e.,* $\Pr[W_1] \leq \Pr[W_0] + \mu_{\mathsf{PRF}}(\lambda)$ where $\mu_{\mathsf{PRF}}(\lambda)$ is the advantage of the pseudorandomness adversary against F.

Game_2: This game is identical to Game_1, but has an additional abort condition. Let $\mathsf{nonce}_{1,i}, \ldots, \mathsf{nonce}_{q_{Chl},i}$ be the nonces that the challenge oracle computes as the signer i's first round Π_{KPK} partial signatures where q_{Chl} is the number of challenge oracle queries that $\mathcal{A}_{\mathsf{KPK}}$ makes. Game_2 aborts if there exists a collision among these nonce values for a signer $i \in S^*$. Otherwise, it runs Game_1 identically. Game_2 aborts with probability $\leq q^2_{Chl}/2^\lambda$ for each signer, *i.e.,* $\Pr[W_2] \leq \Pr[W_1] + n \cdot q^2_{Chl}/2^\lambda$.

Game_3: This game samples seed^* at the beginning of the game, and it aborts if $\mathcal{A}_{\mathsf{KPK}}$ makes any H_{dm} query with seed^*. As seed^* is sampled uniformly, for q_H being the number of H_{dm} queries made by $\mathcal{A}_{\mathsf{KPK}}$, $\Pr[W_3] \leq \Pr[W_2] + q_H/2^\lambda$.

Now we build a FullPriv-AbOPK adversary $\mathcal{A}_{\mathsf{AbOPK}}$ against Π_{AbOPK} using the Game_3-winning Π_{KPK} adversary $\mathcal{A}_{\mathsf{KPK}}$. $\mathcal{A}_{\mathsf{AbOPK}}$ receives pp from the AbOPK challenger $\boldsymbol{Exp}^{\mathsf{FullPriv\text{-}AbOPK}}_{\Pi_{\mathsf{AbOPK}}}(\lambda)$, forwards it to $\mathcal{A}_{\mathsf{KPK}}$, and receives the number of signers n from $\mathcal{A}_{\mathsf{KPK}}$. $\mathcal{A}_{\mathsf{AbOPK}}$ sets the number of signers for FullPriv-AbOPK game as $n+1$. $\mathcal{A}_{\mathsf{AbOPK}}$ receives n pairs of secret and public keys, and forwards them to $\mathcal{A}_{\mathsf{KPK}}$. What remains to be shown is how $\mathcal{A}_{\mathsf{AbOPK}}$ responds to the challenge queries from $\mathcal{A}_{\mathsf{KPK}}$. This is done by first incrementing the signer indexes in $\mathcal{A}_{\mathsf{KPK}}$ query, and then forwarding the adjusted query to $\boldsymbol{Exp}^{\mathsf{FullPriv\text{-}AbOPK}}_{\Pi_{\mathsf{AbOPK}}}$. This simulates the signers of the KPK game using the known public keys the AbOPK game. For the challenge sets, additional to incrementing the signer indexes, $\mathcal{A}_{\mathsf{AbOPK}}$ also adds the index 1 to the challenge signer sets to build valid responses for the AbOPK game. Finally, $\mathcal{A}_{\mathsf{AbOPK}}$ gets the guess of $\mathcal{A}_{\mathsf{KPK}}$, b_{KPK} and forwards that to $\boldsymbol{Exp}^{\mathsf{FullPriv\text{-}AbOPK}}_{\Pi_{\mathsf{AbOPK}}}(\lambda)$.

It is straightforward to see that $\mathcal{A}_{\mathsf{AbOPK}}$ wins when $\mathcal{A}_{\mathsf{KPK}}$ wins. We also argue that $\mathcal{A}_{\mathsf{AbOPK}}$'s simulation of $\mathcal{A}_{\mathsf{KPK}}$'s view for Game_3 is perfect. For all queries except the challenge public key and challenge signatures, the forwarded values from $\boldsymbol{Exp}^{\mathsf{FullPriv\text{-}AbOPK}}_{\Pi_{\mathsf{AbOPK}}}$ provide the same behavior as Game_3. For the challenge public key, the behavior of the unknown public key from the FullPriv-AbOPK game simulates the behavior of dpk^*. For the challenge oracle queries, while Game_3 runs a truly random $f(\cdot)$ to set the random coins of the dummy signer, $\boldsymbol{Exp}^{\mathsf{FullPriv\text{-}AbOPK}}_{\Pi_{\mathsf{AbOPK}}}$ uses random coins directly while running its unknown public key's signing protocols. These two behaviors are *distinguishable* when $f(\cdot)$ is evaluated with the same input as $f(\cdot)$ would output the same value twice. However, Game_3 – following from Game_2 – ensures that $f(\cdot)$ is not run on the same input twice with its abort condition. Thus, we conclude that

$$\Pr\left[\boldsymbol{Exp}^{\mathsf{FullPriv\text{-}KPK}}_{\Pi_{\mathsf{KPK}},\mathcal{A}_{\mathsf{KPK}}}(\lambda)=1\right] \leq \Pr\left[\boldsymbol{Exp}^{\mathsf{FullPriv\text{-}AbOPK}}_{\Pi_{\mathsf{AbOPK}},\mathcal{A}_{\mathsf{AbOPK}}}(\lambda)=1\right] + \mu_{\mathsf{PRF}}(\lambda) + \frac{n \cdot q^2_{Chl} + q_H}{2^\lambda}$$

□

5 PP-MuSig2: A Privacy-Preserving Version of MuSig2

MuSig2 is a two-round multi-signature scheme that produces standard Schnorr signatures while ensuring concurrent security and key aggregation [NRS21] and has just been standardized for use in the Bitcoin network [Cho24]. Unlike the first two-round multi-signature with key aggregation scheme from [MPSW18] (which has been proven to be insecure [NKDM03, DEF+19]), MuSig2 uses multiple nonces to compute a signature share and a linear combination of them to output the final signature. This modification is crucial to prove the scheme secure. Our goal is to present a variant that retains all of its advantages while enabling its use for privacy-preserving group signing. Towards that end,

for simplicity, we consider the most efficient variant that uses two nonces ($\nu = 2$) and is secure in the random oracle (ROM) and algebraic group model (AGM). Our modifications also apply to the other variants defined in [NRS21]. We start by describing our privacy-preserving protocol PP-MuSig2, and then we prove that it satisfies the strongest unforgeability and privacy notions for multi-signatures, UNF-3 and FullPriv-KPK.

5.1 PP-MuSig2 Construction

The first step is to translate the syntax of original (deterministic) MuSig2 into that of a verifiable key-aggregated one. We do this implicitly by applying the MSdKA to MSvKA transformation from [LÖ24]. Since we are only adjusting the syntax, unforgeability is preserved trivially. Moreover, the use of key-prefixing in MuSig2 immediately ensures that we have the strongest level of UNF-3 security in the MSvKA framework. We present both MuSig2 and our privacy-preserving scheme PP-MuSig2 in Figure 5. The corollary below follows from the unforgeability proof of the original MuSig2 by Nick et al. [NRS21] and the transformation of [LÖ24]. We refer the reader to the original papers for the corresponding proofs and discussions on the underlying assumptions/models.

Corollary 1. *The* MS *scheme* MuSig2 *in Figure 5 is* UNF-3*-secure under the algebraic one-more discrete-logarithm assumption in the ROM and AGM.*

The main task is to add privacy. We roughly follow the idea from our generic lift, and introduce a dummy key pair $(\mathsf{dsk}, \mathsf{dpk})$ that is used as "randomizer" in the aggregate keys and signatures. However, we perform several MuSig2-specific optimizations compared to the approach in Section 4. As a result of our fine-tuning, unlike the generic lift presented in Section 4, the resulting scheme PP-MuSig2 enjoys two-round signing protocol without the burden of an additional round. Furthermore, the signing protocol does not require signers to store the key aggregation proofs, which is only necessary to combine the partial signatures.

Verifiable Key Aggregation. The key aggregation algorithm of PP-MuSig2 extends the original MuSig2 aggregated key with a dummy secret key $\mathsf{dsk} \leftarrow \mathsf{H}_{dm}(\pi, PK)$, as our generic lift. As the MuSig2 scheme already uniformly samples its secret keys, PP-MuSig2 does not have to set the random coins for MuSig2.Kg, and instead derives dsk directly using the random oracle H_{dm}. Note that dsk does not impact the hash exponents of the real public keys, so the signers can compute their partial signatures without needing dsk or π. Finally, the aggregated keys can be verified by re-computing dsk using the key aggregation proof π and performing the canonical check over apk.

Signature Generation. In contrast to the generic lift from Sec. 4, PP-MuSig2 does not require signers to simulate the signing protocol for the dummy secret key dsk. In fact, the signing protocol of PP-MuSig2 is identical to MuSig2 except for the final signature combination. The signature combination performs the identical steps to MuSig2, and adds operations to re-compute apk via the contribution of dsk, and adapt the final signature to include the contribution of dsk with the additional term $\mathsf{dsk} \cdot c$ while computing s.

All in all, deployments of PP-MuSig2 do not require any changes to the signing protocol of MuSig2, and all algorithms that need the signer's secret key are exactly the same – enabling a smooth path to upgrade any MuSig2 deployment into its privacy-preserving variant. This holds with one minor caveat though: we assume that MuSig2 gets apk as input, which is not made explicit in the deterministic variant, where the aggregated key can be re-computed from the individual public keys from scratch. However, we believe that giving apk as a direct input is a more sensible choice anyway, and still does not require any changes into the operations that depend on the secret key.

5.2 Unforgeability of PP-MuSig2

In the following, we present a proof overview for the unforgeability of PP-MuSig2, which preserves the UNF-3 security of the underlying MuSig2 construction. Recall, that the latter can be trivially conjectured from [NRS21]. We refer the reader to Appendix B for the full proof.

Theorem 3. *If* MuSig2 *is* UNF-3 *secure, then* PP-MuSig2 *is* UNF-3 *secure for* H_{sig}, H_{agg}, H_{b}, *and* H_{dm} *as random oracles.*

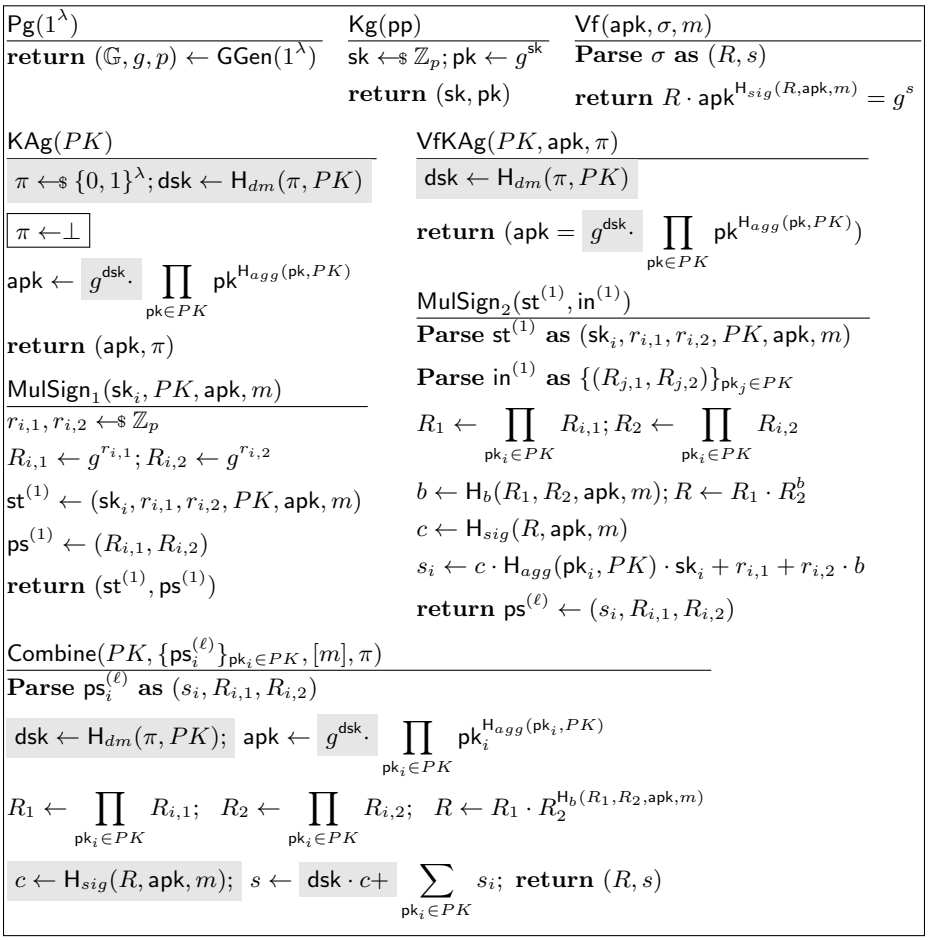

Fig. 5. Description of MuSig2 and our multi-signature scheme PP-MuSig2 for hash functions $\mathsf{H}_{sig}, \mathsf{H}_{agg}, \mathsf{H}_b, \mathsf{H}_{dm} : \{0,1\}^* \to \mathbb{Z}_p$. Codes in gray boxes only occur for PP-MuSig2. The code in black box only occurs for MuSig2.

Proof Overview. Given an efficient UNF-3 adversary $\mathcal{A}$ for PP-MuSig2, we build an efficient UNF-3 adversary $\mathcal{B}$ for the original MuSig2 scheme. Our proof strategy is as follows. $\mathcal{B}$ simulates the challenge public key pk^* as it is to $\mathcal{A}$ and also echoes H_{agg} queries between the MuSig2 challenger and $\mathcal{A}$. By doing so, $\mathcal{B}$ ensures that the deterministic part of the PP-MuSig2 keys are identical to the MuSig2 keys. Eventually, $\mathcal{A}$ returns a non-trivial forgery of the form, $R^* \cdot (\mathsf{apk}^*)^{c^*} = g^{s^*}$ for the randomized aggregated key $\mathsf{apk}^* \leftarrow \mathsf{apk}' \cdot g^{\mathsf{dsk}^*}$ and apk' being the deterministic part of the aggregated key. Then, $\mathcal{B}$ can compute a forgery for the underlying apk' as $(R^*, s^* - \mathsf{dsk}^* \cdot c^*)$.

While the above blueprint will be enough to derive the final forgery, we need to overcome a few challenges in the simulation of $\mathcal{A}$'s view. Note that $\mathcal{A}$'s queries to the random oracles ($\mathcal{O}^{\mathtt{H}_{\mathtt{sig}}}_{\mathsf{PP\text{-}MuSig2}}$ and $\mathcal{O}^{\mathtt{H}_{\mathtt{b}}}_{\mathsf{PP\text{-}MuSig2}}$), as well as its signing queries depend on the randomized aggregated keys. Thus, we cannot simulate these queries by simply forwarding them to the MuSig2 challenger. As we must cancel the dummy key dsk from the final forgery, we must also cancel the dummy key contribution of the aggregated keys from these queries. Performing such cancellation in the $\mathcal{O}^{\mathtt{H}_{\mathtt{sig}}}_{\mathsf{PP\text{-}MuSig2}}$ and $\mathcal{O}^{\mathtt{H}_{\mathtt{b}}}_{\mathsf{PP\text{-}MuSig2}}$ oracles is not immediately possible as we do not know the opening apk' and dsk values for the input apk. However, by performing careful bookkeeping of the queries to $\mathcal{O}^{\mathtt{H}_{\mathtt{dm}}}_{\mathsf{PP\text{-}MuSig2}}$, we can ensure that we know the opening of an input apk if it belongs to a signer set with pk^*.

The final obstacle for simulating these queries is how dummy keys are cancelled out in the random oracles $\mathcal{O}^{\mathtt{H}_{\mathtt{sig}}}_{\mathsf{PP\text{-}MuSig2}}$ and $\mathcal{O}^{\mathtt{H}_{\mathtt{b}}}_{\mathsf{PP\text{-}MuSig2}}$. If we cancel dummy keys naively, *e.g.,* by forwarding a $\mathcal{O}^{\mathtt{H}_{\mathtt{sig}}}_{\mathsf{PP\text{-}MuSig2}}(R, \mathsf{apk} \leftarrow \mathsf{apk}' \cdot g^{\mathsf{dsk}}, m)$ query to MuSig2's challenger as $\mathcal{O}^{\mathtt{H}_{\mathtt{sig}}}_{\mathsf{MuSig2}}(R, \mathsf{apk}', m)$, then we end up in a distinguishable simulation as queries with two distinct aggregated keys for the same signer set would become the same query for the MuSig2 challenger's oracles. Thus, we must find a way to simulate such queries by still ensuring that $\mathcal{B}$ can build a final forgery. This is done by manipulat-

ing the message values while forwarding them to MuSig2's challenger. In particular let $\mathsf{tobin}(\mathsf{apk})$ be the unique binary representation of apk. For $\mathcal{O}^{\mathsf{H}_{\mathsf{sig}}}_{\text{PP-MuSig2}}$, $\mathcal{O}^{\mathsf{H}_b}_{\text{PP-MuSig2}}$, and signing queries of $\mathcal{A}$, we make $\mathcal{B}$ forward the queries with the message m to the MuSig2 challenger by setting the message to $\mathsf{tobin}(\mathsf{apk})\|m$. This prefixing serves as a domain separation for the queries with distinct aggregated keys for the same signer set so that $\mathcal{B}$ is able to simulate $\mathcal{A}$'s view indistinguishably. In the end, adversary $\mathcal{B}$ builds a MuSig2 forgery on the message $\mathsf{tobin}(\mathsf{apk})\|m$ by using $\mathcal{A}$'s PP-MuSig2 forgery on the m for the (deterministic) aggregated key apk'.

5.3 Privacy Analysis of PP-MuSig2

We show that PP-MuSig2 satisfies FullPriv-KPK, guaranteeing that PP-MuSig2 produces signatures and aggregated keys that are indistinguishable from standard Schnorr signatures and keys. Thus, we first recall stand-alone Schnorr signatures and then prove their indistinguishability.

Schnorr Signature. Key generation outputs $\mathsf{sk} \leftarrow\!\!\$\ \mathbb{Z}_p$ and $pk \leftarrow g^{\mathsf{sk}}$. Standard signing SchnorrSign works as follows: For $r \leftarrow\!\!\$\ \mathbb{Z}_p$, $R \leftarrow g^r$, and $c \leftarrow \mathsf{H}_1(R, pk, m)$, output $\sigma \leftarrow (R, s)$ where $s = r + c \cdot \mathsf{sk} \bmod p$. Verification is done by checking $g^s \stackrel{?}{=} R \cdot \mathsf{pk}^c$.

Intuitively, including g^{dsk} in apk for a freshly chosen dsk (or rather fresh π) serves as a random mask in the aggregated key, and produces random looking group elements as apk – just as in standard Schnorr signatures. Similarly, the values R and s that are derived in Combine include hash values that are computed over inputs that contain the re-derived dsk (via π). For each apk, dsk is a fresh, high-entropy value unknown to the outsiders and masks both the R and s values – while still yielding values that can be verified with the standard SchnorrSign verification algorithm and apk.

Theorem 4. PP-MuSig2 *is* FullPriv-KPK *if* H_{dm} *and* H_b *are random oracles and for* $\mathsf{Sign} = \mathsf{SchnorrSign}$.

Our proof presents a sequences of indistinguishable games to show that PP-MuSig2 is FullPriv-KPK-secure. Let W_i be the event that $\mathcal{A}$ wins Game_i and Game_0 is identical to $\boldsymbol{Exp}^{\text{FullPriv-KPK}}_{\text{PP-MuSig2},\mathcal{A}}(\lambda)$. In a nutshell, games from Game_1 to Game_3 aim to change the behavior of FullPriv-KPK game when the challenge bit $b = 0$ such that $\mathsf{apk}^* = g^{\mathsf{ask}^*}$ and $\mathcal{O}^{\mathtt{Chl}}(m)$ queries are answered as $\sigma \leftarrow \mathsf{SchnorrSign}(\mathsf{ask}^*, m)$ for some ask^*. Games Game_4 and Game_5 aim to show that this ask^* value is indistinguishable from an honestly sampled secret key. This concludes our proof as Game_5 behaves identically when $b = 0$ and $b = 1$.

Proof. Game_1: This game is identical to the original FullPriv-KPK game except how we compute the challenge aggregated key apk^*. We first compute an *aggregated secret key* $\mathsf{ask}^* \leftarrow \mathsf{dsk}^* + \sum_{i \in S^*}(\mathsf{sk}_i \cdot \mathsf{H}_{agg}(\mathsf{pk}_i, PK_{S^*}))$ where $\mathsf{dsk}^* \leftarrow \mathsf{H}_{dm}(\pi^*, PK_{S^*})$. This is just an internal change and does not impact $\mathcal{A}$'s view, so $\Pr[W_1] = \Pr[W_0]$.

Game_2: This game introduces an abort condition to $\mathcal{O}^{\mathtt{Chl}}$. Let R_1, R_2 be the aggregated random commitments while $\mathcal{O}^{\mathtt{Chl}}$ is running the Combine algorithm. If there is a previous H_b query with R_1, R_2, then Game_2 aborts. Let q_{H_b} and q_{Chl} be the number of H_b and $\mathcal{O}^{\mathtt{Chl}}$ queries that $\mathcal{A}$ makes. In an honest run of the signing protocol, both R_1 and R_2 are uniformly random, so $\Pr[W_2] \leq \Pr[W_1] + q_{Chl} \cdot q_{\mathsf{H}_b}/p$.

Game_3: This game changes the behavior of $\mathcal{O}^{\mathtt{Chl}}$ to use SchnorrSign instead of multi-signature signing protocol even when the challenge bit $b = 0$. In particular $\mathcal{O}^{\mathtt{Chl}}(m)$ runs $\sigma \leftarrow \mathsf{SchnorrSign}(\mathsf{ask}^*, m)$ when the challenge bit $b = 0$. A Schnorr signature σ can be parsed as (s, R). First, we argue that random commitment $R \leftarrow g^r$ is sampled uniformly random in both cases. PP-MuSig2 sets $r \leftarrow \sum_{i \in [n]}(r_{i,1} + r_{i,2} \cdot b')$ for $b' \leftarrow \mathsf{H}_b(R_{i,1}, R_{i,2}, \mathsf{apk}^*, m)$ and random $r_{i,1}, r_{i,2}$ values. By Game_2, there is no H_b query made by $\mathcal{A}$ for R_1 and R_2, so b' is uniformly random to $\mathcal{A}$. It easily follows that sampling r randomly or computing it by using the random $r_{i,1}, r_{i,2}$ values are identical. Now we show that the s value is consistent with this r value,

$$
\begin{aligned}
s &= \mathsf{dsk}^* \cdot c + \sum_{\mathsf{pk}_i \in PK_{S^*}} s_i = \mathsf{dsk}^* \cdot c + \sum_{\mathsf{pk}_i \in PK_{S^*}} (r_{i,1} + r_{i,2} \cdot b' + sk_i \cdot c \cdot \mathsf{H}_{agg}(\mathsf{pk}_i, PK_{S^*})) \\
&= \mathsf{dsk}^* \cdot c + \sum_{\mathsf{pk}_i \in PK_{S^*}} (r_{i,1} + r_{i,2} \cdot b') + c \cdot \sum_{\mathsf{pk}_i \in PK_{S^*}} (\mathsf{sk}_i \cdot \mathsf{H}_{agg}(\mathsf{pk}_i, PK_{S^*})) \\
&= (r + c \cdot \mathsf{ask}^*)
\end{aligned}
$$

where $R := g^r$ and $c := \mathsf{H}_{sig}(R, \mathsf{apk}^*, m)$. We conclude that $\Pr[W_3] = \Pr[W_2]$.

Game_4: In this game, we sample the challenge key aggregation proof π^* at the beginning of the game and abort if $\mathcal{A}$ makes a H_{dm} query with π^*. The game samples π^* uniformly. Furthermore, the only π^* related output of the game is $\mathsf{H}_{dm}(\pi^*, PK_{S^*})$ which is sampled independently by the random oracle. Thus, for $q_{\mathsf{H}_{dm}}$ being the number of H_{dm} queries made by $\mathcal{A}$, $\Pr[W_4] \leq \Pr[W_3] + q_{\mathsf{H}_{dm}}/2^\lambda$.

Game_5: Finally, in this game, we sample $(\mathsf{ask}^*, \mathsf{apk}^*)$ as a fresh key pair, so $(\mathsf{ask}^*, \mathsf{apk}^*) \leftarrow \mathsf{Kg}(\mathsf{pp})$. By Game_4, the $\mathcal{A}$ does not make any H_{dm} query for π^*, so $\mathsf{dsk}^* \leftarrow \mathsf{H}_{dm}(\pi^*, PK_{S^*})$ is uniformly random to $\mathcal{A}$. It follows that the change in Game_5 does not impact $\mathcal{A}$'s view and $\Pr[W_5] = \Pr[W_4]$.

In Game_5, the challenge public key pk^* is generated as a fresh individual key and $\mathcal{O}^{\mathtt{Chl}}$ queries are always answered with $\mathsf{SchnorrSign}$, so Game_5 reveals no information about the challenge bit b and $\Pr[W_5] = 1/2$. As a result, we conclude that $\left|\Pr\left[\boldsymbol{Exp}^{\mathsf{FullPriv\text{-}KPK}}_{\mathsf{PP\text{-}MuSig2},\mathcal{A}}(\lambda) = 1\right] - 1/2\right| \leq \frac{q_{Chl} \cdot q_{\mathsf{H}_b}}{p} + \frac{q_{\mathsf{H}_{dm}}}{2^\lambda}$ □

6 PP-DualMS: A Privacy-Preserving Version of DualMS

We introduce a variant of DualMS [Che23], which we present in Figure 6 together with the modifications to the original scheme. Compared to previous lattice-based multi-signatures schemes like DOTT [DOTT21] and MuSig-L [BTT22], DualMS provides both a simpler security proof and better parameters.

The deterministic key aggregation technique used in DualMS prevents it from meeting any of the necessary privacy requirements for privacy-preserving group signing. We address this issue by replacing the *deterministic* key aggregation in DualMS with one that is both *probabilistic* and verifiable. Unfortunately, adding a dummy key doesn't seem to suffice in the lattice case. Simply adding a contribution of the dummy key at the combine step skews the distribution with the dummy key and allows for statistical attacks to recover it. An additional rejection sampling step could be performed at the aggregation step in order to prevent this. It would imply that the multi-signature might fail at the aggregation step, which doesn't fit the syntax of multi-signatures.

To avoid this, we can execute MulSign with π as an additional input and add the dummy secret key $\pi = \mathsf{dsk}$ along with the challenge c in the response step. Consequently, this requires that every member of the group knows π. Then the signer needs larger standard-deviation parameter to ensure the rejection sampling step succeeds, which is a key component of lattice-based constructions.

The verification of the aggregated key in the VfKAg algorithm is simple with the knowledge π. In order to generate a share of any message, each signer needs the knowledge of π, as MulSign requires the corresponding hash share generated in the key aggregation algorithm KAg.

DualMS uses three hash functions $\mathsf{H}_{\mathrm{agg}}, \mathsf{H}_{\mathrm{sig}} : \{0,1\}^* \to C$, $\mathsf{H}_{\mathrm{com}} : \{0,1\}^* \to R_q^{k \times l'}$. The scheme is secure based on the hardness of the MLWE and MSIS assumptions in the ROM. We highlight the differences between our new scheme PP-DualMS and the original DualMS in Figure 6. The MS-UNF unforgeability notion verified by DualMS is recalled in the Appendix (Definition 14). We defer the presentation of constants and parameters used in the scheme to Appendix D.

Repetitions and Correctness. By Lemma 3, each signer in the second round of MulSign passes the rejection sampling step with probability $1/M$. As a result, every signer passes the rejection sampling step together with probability $1/M_n$, where $M_n = M^n$. To see the correctness part, let us recall that the commitment of the i-th signer in the MulSign algorithm for the message m is

$$\begin{aligned}\mathbf{w}_i = \bar{\mathbf{A}}\mathbf{y}_i + \bar{\mathbf{B}}\mathbf{r}_i &= \bar{\mathbf{A}}(\mathbf{z}_i - a_i c \mathbf{s}_i - c\tilde{a}\mathbf{s}_{\mathsf{dsk}}) + \bar{\mathbf{B}}\mathbf{r}_i \\ &= \mathbf{A}\mathbf{z}'_i + \mathbf{z}''_i - a_i c \mathbf{t}_i - c\tilde{a}\mathbf{t}_{\mathsf{dsk}} + \mathbf{B}\mathbf{r}'_i + \mathbf{r}''_i\end{aligned}$$

Then, for the multi-signature, we have

$$\begin{aligned}\tilde{\mathbf{w}} &= \sum_{i=1}^{n} \mathbf{w}_i = (\mathbf{A}\sum_{i=1}^{n} \mathbf{z}'_i) + (\sum_{i=1}^{n} \mathbf{z}''_i + \sum_{i=1}^{n} \mathbf{r}''_i) + (\sum_{i=1}^{n} \mathbf{B}\mathbf{r}'_i) - c(\sum_{i=1}^{n} a_i \mathbf{t}_i + \sum_{i=1}^{n} \tilde{a}\mathbf{t}_{\mathsf{dsk}}) \\ &= \mathbf{A}\tilde{\mathbf{z}} + \tilde{\mathbf{e}} + \mathbf{B}\tilde{\mathbf{r}} - c\tilde{\mathbf{t}}\end{aligned}$$

$\underline{\mathsf{Pg}(1^\lambda)}$

$\mathbf{A} \leftarrow\!\$\ R_q^{k\times l}$; $\bar{\mathbf{A}} \leftarrow [\mathbf{A} \mid \mathbf{I}]$;

return $\bar{\mathbf{A}}$

$\underline{\mathsf{Kg}(\mathsf{pp})}$

$\mathsf{sk} = \mathbf{s} \leftarrow\!\$\ S_\eta^{l+k}$; $\mathsf{pk} = \mathbf{t} \leftarrow \bar{\mathbf{A}}\mathbf{s}$

return $(\mathsf{sk}, \mathsf{pk})$

$\underline{\mathsf{KAg}(PK)}$

Parse PK **as** $(\mathbf{t}_1, \ldots, \mathbf{t}_n)$

$(\mathsf{dsk}, \mathsf{dpk}) \leftarrow \mathsf{Kg}(\mathsf{pp})$

$a_i \leftarrow \mathsf{H}_{\mathrm{agg}}(PK, \mathbf{t}_i, \mathsf{dsk})$

$\tilde{a} \leftarrow \mathsf{H}_{\mathrm{agg}}(PK, \mathsf{dsk})$

$\tilde{\mathbf{t}} \leftarrow n \cdot \tilde{a} \cdot \mathsf{dpk} + \sum_{i=1}^{n} a_i \mathbf{t}_i$;

$\pi \leftarrow \mathsf{dsk}$

return $(\tilde{\mathbf{t}}, \pi)$

$\underline{\mathsf{VfKAg}(PK, \mathsf{apk}, \pi)}$

Parse PK **as** $(\mathbf{t}_1, \ldots, \mathbf{t}_n)$;

$\mathsf{dpk} \leftarrow \bar{\mathbf{A}}\pi$

$a_i \leftarrow \mathsf{H}_{\mathrm{agg}}(PK, \mathbf{t}_i, \pi)$

$\tilde{a} \leftarrow \mathsf{H}_{\mathrm{agg}}(PK, \pi)$

return $\mathsf{apk} = n\tilde{a} \cdot \mathsf{dpk} + \sum_{i=1}^{n} a_i \mathbf{t}_i$

$\underline{\mathsf{Combine}(PK, \{\mathsf{ps}_i^{(\ell)}\}_{\mathsf{pk}_i \in PK}, [m], \pi)}$

Parse $\mathsf{ps}_i^{(\ell)}$ **as** $(c_i, \mathbf{z}_1, \mathbf{r}_1)$

if $c_i \neq c_1$ **return** $\perp$

Parse $\mathbf{z}_i$ **as** $[\mathbf{z}_i', \mathbf{z}_i'']$

Parse $\mathbf{r}_i$ **as** $[\mathbf{r}_i', \mathbf{r}_i'']$

$\tilde{\mathbf{z}} \leftarrow \sum_{i=1}^{n} \mathbf{z}_i' \in R^l$; $\tilde{\mathbf{r}} \leftarrow \sum_{i=1}^{n} \mathbf{r}_i' \in R^{l'}$

$\tilde{\mathbf{e}} \leftarrow \sum_{i=1}^{n} (\mathbf{z}_i'' + \mathbf{r}_i'') \in R^k$

return $(c_1, \tilde{\mathbf{z}}, \tilde{\mathbf{r}}, \tilde{\mathbf{e}})$

$\underline{\mathsf{MulSign}_1(\mathsf{sk}_i, PK, \mathsf{apk}, \pi, m)}$

$a_1 \leftarrow \mathsf{H}_{\mathrm{agg}}(PK, \mathbf{t}_1, \pi)$

$\mathbf{B} \leftarrow \mathsf{H}_{\mathrm{com}}(\mathsf{apk}, m) \in R_q^{k\times l'}$

$\bar{\mathbf{B}} \leftarrow [\mathbf{B} \mid \mathbf{I}] \in R_q^{k\times(l'+k)}$

$\mathbf{y}_1 \leftarrow\!\$\ D_s^{l+k}$; $\mathbf{r}_1 \leftarrow\!\$\ D_{s'}^{l'+k}$

$\mathbf{w}_1 \leftarrow \bar{\mathbf{A}}\mathbf{y}_1 + \bar{\mathbf{B}}\mathbf{r}_1 \in R_q^k$

$\mathsf{st}^{(1)} \leftarrow (\mathbf{y}_1, \mathbf{r}_1, \mathbf{w}_1)$; $\mathsf{ps}^{(1)} \leftarrow \mathbf{w}_1$

return $(\mathsf{st}^{(1)}, \mathsf{ps}^{(1)})$

$\underline{\mathsf{MulSign}_2(\mathsf{st}^{(1)}, \mathsf{in}^{(1)}, \pi)}$

Parse $\mathsf{st}^{(1)}$ **as** $(\mathbf{y}_1, \mathbf{r}_1, \mathbf{w}_1)$

Parse $\mathsf{in}^{(1)}$ **as** $\{\mathbf{w}_j\}_{j\in PK}$

Parse π **as** $\mathbf{s}_{\mathsf{dsk}}$;

$\tilde{\mathbf{w}} \leftarrow \sum_{i=1}^{n} \mathbf{w}_i$

$c \leftarrow \mathsf{H}_{\mathrm{sig}}(\mathsf{apk}, m, \tilde{\mathbf{w}})$

$\mathbf{z}_1 \leftarrow \mathbf{y}_1 + a_1 c \mathbf{s}_1 + c\tilde{a}\mathbf{s}_{\mathsf{dsk}}$

$s_1 \leftarrow (c, \mathbf{z}_1, \mathbf{r}_1)$ with probability:

$$\min\left(1, \frac{D_s^{l+k}.(\mathbf{z}_1)}{M \cdot D^{l+k}_{a_1 c\mathbf{s}_1 + c\tilde{a}\mathbf{s}_{\mathsf{dsk}}, s}(\mathbf{z}_1)}\right)$$

return $\mathsf{ps}^{(\ell)} \leftarrow s_1$

$\underline{\mathsf{Vf}(\mathsf{apk}, \sigma, m)}$

Parse σ **as** $(c, \tilde{\mathbf{z}}, \tilde{\mathbf{r}}, \tilde{\mathbf{e}})$

$\mathbf{B} \leftarrow \mathsf{H}_{\mathrm{com}}(\mathsf{apk}, m) \in R_q^{k\times l'}$

$\tilde{\mathbf{w}} \leftarrow \mathbf{A}\tilde{\mathbf{z}} + \mathbf{B}\tilde{\mathbf{r}} + \tilde{\mathbf{e}} - c\tilde{\mathbf{t}}$

return $c = \mathsf{H}_{\mathrm{sig}}(\mathsf{apk}, m, \tilde{\mathbf{w}})$
$\wedge\ \|\tilde{\mathbf{z}}\| \le B_n, \|\tilde{\mathbf{r}}\| \le B_n', \|\tilde{\mathbf{e}}\| \le B_n''$

Fig. 6. Description of DualMS and our multi-signature scheme PP-DualMS for hash functions $\mathsf{H}_{\mathrm{sig}}, \mathsf{H}_{\mathrm{agg}}, \mathsf{H}_{\mathrm{com}}$. Codes in gray boxes only occur for PP-DualMS.

Hence, the verifier correctly recovers the aggregated commitment $\tilde{\mathbf{w}}$, and so the condition $c = \mathsf{H}_{\mathrm{sig}}(\mathsf{apk}, m, \tilde{\mathbf{w}})$ holds true. It remains to show that, with overwhelming probability, $\|\tilde{\mathbf{z}}\| \le B_n, \|\tilde{\mathbf{r}}\| \le B_n', \|\tilde{\mathbf{e}}\| \le B_n''$. First note that, by Lemma 3, each $\mathbf{z}_i$ is statistically close to the discrete Gaussian distribution D_s^{l+k}. Then by the independence of discrete Gaussian distributions, each $\mathbf{z}_i'$ follows the distribution D_s^l. As a result, by Lemma 1, $\tilde{\mathbf{z}} = \sum_{i=1}^{n} \mathbf{z}'_i$ follows the distribution $D_{s_{\tilde{z}}}^l$, where $s_{\tilde{z}} = \sqrt{\sum_{i=1}^{n} s^2} = s\sqrt{n}$. Hence, by Lemma 2, we have $\|\tilde{\mathbf{z}}\| \le B_n = \sqrt{n}\sqrt{Nl}\frac{s\gamma}{\sqrt{2\pi}}$, except with very low probability. Similarly, $\tilde{\mathbf{r}}, \tilde{\mathbf{e}}$ is statistically close to the distribution $D_{s_{\tilde{r}}}^{l'}, D_{s_{\tilde{e}}}^k$, respectively, where $s_{\tilde{r}} = s'\sqrt{n}$, $s_{\tilde{e}} = \sqrt{n(s^2 + s'^2)}$. This gives the norm bound B_n', B_n'' for $\tilde{\mathbf{r}}, \tilde{\mathbf{e}}$, respectively.

6.1 Unforgeability of PP-DualMS

Theorem 5. *If* DualMS *is* UF-CMA*, then* PP-DualMS *is* UNF-3 *with* $\mathsf{H}_{agg}, \mathsf{H}_{sig}$ *and* H_{com} *are modeled as random oracles.*

Proof. We build an adversary $\mathcal{B}$ for the UF-CMA game using an adversary $\mathcal{A}$ for the UNF-3 game.

Adversary $\mathcal{B}$ receives a public key $\mathsf{pk}^* = \mathbf{A}\mathbf{s}^*$ from its challenger and forwards it to $\mathcal{A}$. Notice that $\mathcal{B}$ has the ability to detect all the aggregated keys created by $\mathcal{A}$ and their associated secret. Indeed without loss of generality, each time $\mathcal{A}$ wants to generate one, it needs to query oracle H_{agg} with inputs (L, π) where L is a list of public keys and π is the corresponding secret. In particular, it can also identify potential challenge aggregated keys, as if $\mathsf{pk}^* \in L$, the corresponding group could be used as the one for which the final forgery will be done. Assume for now that $\mathcal{B}$ knows the group L^* and randomness π^* corresponding to the aggregated key apk^* for which $\mathcal{A}$ will output the forgery. When adversary $\mathcal{A}$ computes the coefficients (a_i) for the aggregated key apk^* by querying $\mathsf{H}_{\text{agg}}(L^*, \mathsf{pk}_i, \pi^*)$, for $\mathsf{pk}_i \in L^*$, adversary $\mathcal{B}$ queries it's challenger oracle $\tilde{\mathsf{H}}_{\text{agg}}(L^*, \mathsf{pk}_i)$ and forwards the answer. This makes it so that $\mathsf{DualMS.KeyAgg}(L^*) = \mathsf{apk}^* - na^*\mathbf{A}\pi^*$, in order to make the forgery valid. For each new group uniquely identified by (L, π) such that $L \neq L^*$ or $\pi \neq \pi^*$, adversary $\mathcal{B}$ generates a new key $\mathsf{pk}_{L,\pi}$ and internally sets a corresponding group $S_{L,\pi} = L \cup \{\mathsf{pk}_{L,\pi}\}$. For each query to H_{agg} involving (L, π), adversary $\mathcal{B}$ replaces L by $S_{L,\pi}$ removes π and returns the answer by its challenger's random oracle $\tilde{\mathsf{H}}_{\text{agg}}$. This allows to randomize key aggregation queries transparently and make it so that $\mathsf{DualMS.KeyAgg}(S_{L,\pi}) = \mathsf{KeyAgg}(L, \pi) - n\tilde{a}\mathbf{A}\pi$.

The main issue for $\mathcal{B}$ is to handle signature queries that involve the same group but with a different aggregated public key, since the original key aggregation in DualMS is deterministic. Adversary $\mathcal{B}$ handles this by keeping track of the corresponding aggregated key $\mathsf{apk}_{L,r}$ for every pair (L, π). For every $\mathsf{MulSign}_1$ query using $\mathsf{apk}_{L,r} \neq \mathsf{apk}^*$, $\mathcal{B}$ runs its own $\mathsf{MulSign}_1$ oracle for the corresponding group $S_{L,\pi}$ with the same message. When $\mathcal{A}$ resumes this signing session by calling $\mathsf{MulSign}_2$, it gives the commitments $(\mathbf{w_i})_{i\in[2,k]}$ of all parties but pk^*, where k is the size of L and pk^* is assumed to be user 1 in the group. Since $\mathcal{B}$ added user $\mathsf{pk}_{L,r}$ to the session in its challenger's view, the challenger is expecting an additional share to resume. $\mathcal{B}$ simply sets $\mathbf{w}_{k+1} = \mathbf{0}$ and calls $\mathsf{MulSign}_2$ with $(\mathbf{w_i})_{i\in[2,k+1]}$. This results in an aggregated commitment $\mathbf{w} = \sum_{i=1}^{k+1} \mathbf{w}_i = \sum_{i=1}^{k} \mathbf{w}_i$, as if the last user was not participating. To handle the queries with apk^*, $\mathcal{B}$ does not add a fresh key to the group and simply translates any query with apk^* as queries with the corresponding group L^* to its challenger.

However, the final share $(c, \mathbf{z}, \mathbf{r})$ returned to $\mathcal{B}$ by the $\mathsf{MulSign}_2$ oracle (if it does not abort) will contain a vector $\mathbf{z}$ verifying $\mathbf{z} = \mathbf{y} + c \cdot a\mathbf{s}^*$ distributed as a sample from $\mathcal{D}_s^{l+k}$ due to the rejection sampling. Adversary $\mathcal{A}$ is expecting some vector with the same distribution but verifying the equation $\mathbf{z} = \mathbf{y} + c \cdot (a\mathbf{s}^* + \tilde{a}\pi)$, where $\tilde{a} = \mathsf{H}_{\text{agg}}(L, \pi)$. To solve this problem, adversary $\mathcal{B}$ simply computes $\mathbf{z}' = \mathbf{z} + c \cdot \tilde{a}\pi$ and performs rejection sampling on $\mathbf{z}'$. Adversary $\mathcal{B}$ now just forwards $(c, \mathbf{z}', \mathbf{r})$ to $\mathcal{A}$ with probability $\mathsf{RejSamp}(s, M', \mathbf{z}')$ as the share for pk^*, with M' is a rejection parameter of DualMS fixing the probability that it outputs a signature. As aggregated commitments will match, the aggregated signature will be valid. Lemma 4 ensures that if $\mathcal{B}$ outputs an answer, the distribution of $\mathbf{z}'$ will be at statistical distance at most ϵ_{rej} from $\mathcal{D}_s^{l+k}$. However, the probability to not output is increased compared to the original DualMS. By lemma 4, the challenger outputs a share to $\mathcal{B}$ with probability $1/M'$. With the additional rejection sampling, adversary $\mathcal{B}$ outputs a share to $\mathcal{A}$ with probability $1/M'^2$. By setting the rejection parameter $M = M'^2$ for PP-DualMS, we ensure that the reduction follows the right distribution.

When $\mathcal{A}$ outputs a forgery $(c, \mathbf{z}, \mathbf{r})$, adversary $\mathcal{B}$ forwards $(c, \mathbf{z} - na^*\pi^*, \mathbf{r})$ with probability $\mathsf{RejSamp}(s, M, \mathbf{z} - na^*\pi^*)$ to its challenger. This time it performs rejection sampling towards the challenger. With probability $1/M$, adversary $\mathcal{B}$ outputs a signature with the right distribution. Given that we set the reduction so that $\mathsf{DualMS.KeyAgg}(L^*) = \mathsf{apk}^* - na^*\mathbf{A}\pi^*$, if $(c, \mathbf{z}, \mathbf{r})$ is a valid forgery for apk^*, then $\mathcal{A}$ wins the game, provided that it respects the conditions of MS-UNF game. By lemma 4, the rejection sampling step incurs a loss of a factor $1/M$. To see why the forgery outputted by $\mathcal{B}$ respects the conditions of UNF, observe that the forgery is invalid if and only if another signing query for the same message m^* and the same group L^* was performed by $\mathcal{B}$. In terms of behavior of $\mathcal{A}$, this translates into $\mathcal{A}$ already making a query with apk^*, L^*, m^*, which is forbidden by the UNF-3 conditions.

Until now, we assumed that $\mathcal{B}$ was able to identify the right apk^* chosen by $\mathcal{A}$ for the forgery. To do so, the reduction can simply make a guess on every query $\mathsf{H}_{\text{agg}}(L, \pi)$ where $\mathsf{pk}^* \in L$. It succeeds

$\boldsymbol{Exp}^{\mathsf{wSetPriv\text{-}KPK}}_{\mathsf{PP\text{-}DualMS},\mathcal{A}}(\lambda)$ Game_0

$b \leftarrow\!\$\ \{0,1\}; \mathsf{pp} \leftarrow \mathsf{Pg}(1^\lambda); Q := \emptyset, n \leftarrow \mathcal{A}(\mathsf{pp});$ **abort if** $n \not> 0$

$(SK, PK) := (\{\mathsf{sk}_i\}_{i\in[n]}, \{\mathsf{pk}_i\}_{i\in[n]}) \leftarrow (\mathsf{Kg}(\mathsf{pp}))_{i\in[n]}$

$(S_0, S_1) \leftarrow \mathcal{A}(SK, PK)$

abort if $|S_0| \neq |S_1|$

$(\mathsf{apk}_b, \pi_b) \leftarrow \mathsf{KAg}(PK_{S_b}); b^* \leftarrow \mathcal{A}^{\mathcal{O}^{\mathtt{Chl}}(\cdot)}(\mathsf{apk}_b);$ **return** 1 **if** $b = b^*$

$\mathcal{O}^{\mathtt{Chl}}(m)$

$\Sigma \leftarrow \{\mathsf{MulSign}(\mathsf{sk}_i, PK_{S_b}, \mathsf{apk}_b, m, \pi_b)\}_{\mathsf{sk}_i \in SK_{S_b}};$ **return** $\sigma \leftarrow \mathsf{Combine}(PK_{S_b}, \pi_b, \Sigma, \pi_b)$

Fig. 7. Game_0 of wSetPriv-KPK proof of PP-DualMS.

with probability at least $1/q_{agg}$, where q_{agg} is the number of queries that $\mathcal{A}$ makes to the random oracle $\mathsf{H}_{\mathrm{agg}}$. Finally, if we denote q_{sig} the number of signature queries made by $\mathcal{A}$, this gives

$$\boldsymbol{Exp}^{\mathsf{UNF\text{-}3}}_{\mathsf{PP\text{-}DualMS},\mathcal{A}}(\lambda) \leq \frac{1}{\sqrt{M q_{agg}}} \boldsymbol{Exp}^{\mathsf{MS\text{-}UNF}}_{\mathsf{DualMS},\mathcal{B}}(\lambda) + q_{sig}\epsilon_{err}$$

□

6.2 Privacy Analysis of PP-DualMS

First, note that the PP-DualMS doesn't satisfy the notion of the full privacy (FullPriv) or the set privacy (SetPriv). This is because, with non-negligible probability, one can correctly guess the group size only from the aggregated signatures. To see this, let n be the group size to generate an aggregated signature $\sigma = (c, \tilde{\mathbf{z}}, \tilde{\mathbf{r}}, \tilde{\mathbf{e}})$ on some message m. Recall, the component $\tilde{\mathbf{z}}$ follows the distribution $D^l_{s_{\tilde{z}}}$, where $s_{\tilde{z}} = s\sqrt{n}$, which depends on n. Similarly, the distribution of the component $\tilde{\mathbf{r}}, \tilde{\mathbf{e}}$ also depends on n. As a result, the aggregated signatures leak informations on n.

However, PP-DualMS does not reveal the underlying signer group to the outsiders who do not know the key aggregation proof π. As a result, we can still aim for privacy where leaking group size is acceptable. We now show that PP-DualMS satisfies weak set privacy (wSetPriv). Then it also satisfies membership privacy (MemPriv). To prove wSetPriv, we follow a game-based approach. We end up in a game where the adversary only gets a random aggregated key, which is independent of b, and the aggregated signatures thereof. Since the game is independent of the value b, the winning probability here can't be better than a random guess.

Theorem 6. *The* PP-DualMS *is* wSetPriv-KPK *if* H_{sig} *is modeled as a RO.*

Proof. Our main proof strategy is changing wSetPriv-KPK game's behaviour by replacing the real apk_b with a random apk^*, i.e., independent of b. We recall Game_0 in Figure 7 and present Game_1 in Figure 8. To answer the multi-signature queries made by $\mathcal{A}$ to $\mathcal{O}^{\mathtt{Chl}}$, challenger $\mathcal{C}$ uses the simulator with apk^*. It first samples the local shares $(\mathbf{z}_i, \mathbf{r}_i)$ and then computes the matching aggregated commitment $\mathbf{w}$ to make the signature valid. Note that, since the distribution of the final signature depends only on the number of signers, thanks to the equal group size for both $b = 0$ and $b = 1$, the adversary will not be able to distinguish both cases. As a result, challenger $\mathcal{O}^{\mathtt{Chl}}$ answers the signing queries regardless of the bit b. For the hash query of $\mathsf{H}_{\mathrm{sig}}$ made by $\mathcal{A}$, challenger $\mathcal{C}$ checks if the query already made. If so, it returns the previous value of the query. Otherwise, it samples a fresh $c \leftarrow C$ and return it. We argue that $\mathcal{A}$ can distinguish between Game_0 and Game_1 with only negligible probability.

– In Game_1, challenger $\mathcal{C}$ replaces apk_b by a random apk^*. Since dpk is hidden from $\mathcal{A}$'s knowledge, this is only replacing

$$\mathsf{dpk} := \mathbf{t}_{\mathsf{dpk}} = (\mathsf{apk}_b - \sum_{i=1}^{n} a_i \mathbf{t}_i)/n\tilde{a}$$

by a random one, which is $\mathbf{t}^* = (\mathsf{apk}^* - \sum_{i=1}^n a_i \mathbf{t}_i)/n\tilde{a}$. This modification is indistinguishable under the MLWE assumption.[4]

[4] Since $\mathcal{A}$ knows both the secret keys and the corresponding public keys $(\mathbf{s}_i, \mathbf{t}_i)$ of the users in the wSetPriv model, this does not make the underlying assumption stronger. Indeed, one can simply sample any $\mathbf{s}$ from the distribution S_η^{l+k} to set $\mathbf{t} = \bar{\mathbf{A}}\mathbf{s}$.

$\boldsymbol{Exp}^{\mathsf{wSetPriv\text{-}KPK}}_{\mathsf{PP\text{-}DualMS},\mathcal{A}}(\lambda)$ Game_1

$b \leftarrow\$ \{0,1\}; \mathsf{pp} \leftarrow \mathsf{Pg}(1^\lambda); Q := \emptyset, n \leftarrow \mathcal{A}(\mathsf{pp});$ **abort if** $n \not> 0$

$(SK, PK) := (\{\mathsf{sk}_i\}_{i\in[n]}, \{\mathsf{pk}_i\}_{i\in[n]}) \leftarrow (\mathsf{Kg}(\mathsf{pp}))_{i\in[n]}$

$(S_0, S_1) \leftarrow \mathcal{A}(SK, PK);$ **abort if** $|S_0| \neq |S_1|$

$\mathsf{apk}^* := \tilde{\mathbf{t}} \leftarrow\$ R_q^k; b^* \leftarrow \mathcal{A}^{\mathcal{O}^{\mathtt{Chl}}(\cdot)}(\mathsf{apk}^*);$ **return** 1 **if** $b = b^*$

$\mathcal{O}^{\mathtt{Chl}}(m)$

$\mathbf{z}_i \leftarrow\$ D_s^{l+k}; \mathbf{r}_i \leftarrow\$ D_{s'}^{l'+k}; c \leftarrow\$ C;$ **Parse** $\mathbf{z}_i$ **as** $[\mathbf{z}'_i, \mathbf{z}''_i]$ and $\mathbf{r}_i$ **as** $[\mathbf{r}'_i, \mathbf{r}''_i]$

$\tilde{\mathbf{z}} \leftarrow \sum_{i=1}^{n} \mathbf{z}'_i; \tilde{\mathbf{r}} \leftarrow \sum_{i=1}^{n} \mathbf{r}'_i; \tilde{\mathbf{e}} \leftarrow \sum_{i=1}^{n} \mathbf{z}''_i + \mathbf{r}''_i$

With prob. $1/M^n$: set $\mathsf{H}_{\mathrm{sig}}(\mathsf{apk}^*, m, \mathbf{A}\tilde{\mathbf{z}} + \mathbf{B}\tilde{\mathbf{r}} + \tilde{\mathbf{e}} - c\tilde{\mathbf{t}}) = c$

return $\sigma = (c, \tilde{\mathbf{z}}, \tilde{\mathbf{r}}, \tilde{\mathbf{e}})$

Fig. 8. Game_1 of wSetPriv-KPK proof of PP-DualMS.

– Note that since the reduction simulates the signing queries and sets the aggregated commitment $\mathbf{w}$ after the challenge, adversary $\mathcal{A}$ might be able to distinguish the modified game if $(\mathsf{apk}^*, m, \mathbf{w})$ was already queried to the oracle $\mathsf{H}_{\mathrm{sig}}$ by $\mathcal{A}$ for some message m for which a signing query was issued. This collision on the aggregated commitment happens with probability at most $q_{\mathsf{H}}/2^\lambda$, where q_{H} is the total number of random oracle queries made by adversary $\mathcal{A}$.
– Finally, for replying the MulSign queries, $\mathcal{C}$ samples each $\mathbf{z}_i \leftarrow\$ D_s^{l+k}$, without the knowledge of the secret $(\mathbf{s}_i, \mathbf{s}_{\mathsf{dsk}})$. As a consequence of the Rejection sampling (Lemma 4), the statistical distance between the actual and the simulated distribution of each $\mathbf{z}_i$ is at most ϵ_{rej}, where $\epsilon_{rej} = 2e^{-t^2/2}/M$. Hence, by the triangle inequality of the statistical distance, we have the actual and the simulated distribution of all $\mathbf{z}_i$ is at most $n\epsilon_{rej}$ for each signature query.

Since apk^* is independent of b, if we denote q_{sig} the number of signature queries made by the adversary $\mathcal{A}$, we have $|\boldsymbol{Exp}^{\mathsf{wSetPriv\text{-}KPK}}_{\mathsf{PP\text{-}DualMS},\mathcal{A}}(\lambda) - 1/2| \leq \mathbf{Adv}^{\mathsf{MLWE}}_{q,k,l,\eta} + q_{\mathsf{H}}/2^\lambda + q_{sig} n\epsilon_{rej}$. □

7 Conclusion

In this work, we advanced the state-of-the-art of privacy-preserving multi- signatures. Besides presenting a generic transformation to lift privacy of deterministic schemes secure in the AbOPK model to the KPK model, we also presented two Schnorr-based constructions. The first one seemingly integrates to existing deployments such as those in the Bitcoin network while the second one paves the way for post-quantum secure privacy-preserving multi-signatures. Notably, none of our constructions incur in any overhead compared to their plain (non privacy-preserving) counterparts.

Future Work. We consider the study of code and isogeny-based signature schemes to build privacy-preserving multi-signatures as an interesting line of work. In particular, considering the existing canonical identification schemes with signature adaptation from the literature (*e.g.,* [Ste94, SSH11], and CSI-Fish [ESS21]).

Another open problem is to consider alternative methods for creating key aggregation proofs. Current privacy models assume that the key aggregation is performed non-interactively by a single party, requiring a secure channel between this party and all signers to share the key aggregation proof confidentially. Addition of a DKG or a pre-processing phase to ensure the privacy without relying on such a secure channel may be an interesting future work.

Acknowledgments. This research was partially funded by the HPI Research School on Systems Design. It was also supported by the German Federal Ministry of Education and Research (BMBF) through funding of the ATLAS project under reference number 16KISA037. The work of Dipayan Das was done while affiliated with NTT Social Informatics Laboratories.

References

AABN02. M. Abdalla, J. H. An, M. Bellare, and C. Namprempre. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In *EUROCRYPT 2002*, *LNCS* 2332, pages 418–433. Springer, Heidelberg, April / May 2002.

ANPT25. M. Abe, M. Nanri, O. Perez Kempner, and M. Tibouchi. Interactive threshold mercurial signatures and applications. In *Advances in Cryptology – ASIACRYPT 2024*, pages 69–103, Singapore, 2025. Springer Nature Singapore.

BDN18. D. Boneh, M. Drijvers, and G. Neven. Compact multi-signatures for smaller blockchains. In *ASIACRYPT 2018, Part II*, *LNCS* 11273, pages 435–464. Springer, Heidelberg, December 2018.

BHKS18. M. Backes, L. Hanzlik, K. Kluczniak, and J. Schneider. Signatures with flexible public key: Introducing equivalence classes for public keys. In *ASIACRYPT 2018, Part II*, *LNCS* 11273, pages 405–434. Springer, Heidelberg, December 2018.

BN06. M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *ACM CCS 2006*, pages 390–399. ACM Press, October / November 2006.

BN07. M. Bellare and G. Neven. Identity-based multi-signatures from RSA. In *CT-RSA 2007*, *LNCS* 4377, pages 145–162. Springer, Heidelberg, February 2007.

BTT22. C. Boschini, A. Takahashi, and M. Tibouchi. MuSig-L: Lattice-based multi-signature with single-round online phase. In *CRYPTO 2022, Part II*, *LNCS* 13508, pages 276–305. Springer, Heidelberg, August 2022.

CCK+23. P. Chatzigiannis, K. Chalkias, A. Kate, E. V. Mangipudi, M. Minaei, and M. Mondal. SoK: Web3 recovery mechanisms. Cryptology ePrint Archive, Paper 2023/1575, 2023.

CGH+25. S. Celi, S. Griffy, L. Hanzlik, O. Perez-Kempner, and D. Slamanig. Sok: Signatures with randomizable keys. In *Financial Cryptography and Data Security*, pages 160–187, Cham, 2025. Springer Nature Switzerland.

Che23. Y. Chen. DualMS: Efficient lattice-based two-round multi-signature with trapdoor-free simulation. In *CRYPTO 2023, Part V*, *LNCS* 14085, pages 716–747. Springer, Heidelberg, August 2023.

Cho24. A. Chow. Musig2 psbt fields. Online, 2024.

CJKT06. C. Castelluccia, S. Jarecki, J. Kim, and G. Tsudik. Secure acknowledgment aggregation and multisignatures with limited robustness. *Computer Networks*, 50(10):1639–1652, 2006. I. Web Dynamics II. Algorithms for Distributed Systems.

CL19. E. C. Crites and A. Lysyanskaya. Delegatable anonymous credentials from mercurial signatures. In *CT-RSA 2019*, *LNCS* 11405, pages 535–555. Springer, Heidelberg, March 2019.

DEF+19. M. Drijvers, K. Edalatnejad, B. Ford, E. Kiltz, J. Loss, G. Neven, and I. Stepanovs. On the security of two-round multi-signatures. In *2019 IEEE Symposium on Security and Privacy*, pages 1084–1101. IEEE Computer Society Press, May 2019.

DGNW20. M. Drijvers, S. Gorbunov, G. Neven, and H. Wee. Pixel: Multi-signatures for consensus. In *USENIX Security 2020*, pages 2093–2110. USENIX Association, August 2020.

DMRT21. C. Delpech de Saint Guilhem, E. Makri, D. Rotaru, and T. Tanguy. The return of eratosthenes: Secure generation of RSA moduli using distributed sieving. In *ACM CCS 2021*, pages 594–609. ACM Press, November 2021.

DOTT21. I. Damgård, C. Orlandi, A. Takahashi, and M. Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. In *PKC 2021, Part I*, *LNCS* 12710, pages 99–130. Springer, Heidelberg, May 2021.

DS19. D. Derler and D. Slamanig. Key-homomorphic signatures: definitions and applications to multi-party signatures and non-interactive zero-knowledge. *DCC*, 87(6):1373–1413, 2019.

ESS21. E. Eaton, D. Stebila, and R. Stracovsky. Post-quantum key-blinding for authentication in anonymity networks. In *LATINCRYPT 2021*, *LNCS* 12912, pages 67–87. Springer, Heidelberg, October 2021.

FKM+16. N. Fleischhacker, J. Krupp, G. Malavolta, J. Schneider, D. Schröder, and M. Simkin. Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys. In *PKC 2016, Part I*, *LNCS* 9614, pages 301–330. Springer, Heidelberg, March 2016.

GK24. C. P. L. Gouvea and C. Komlo. Re-randomized FROST. Cryptology ePrint Archive, Paper 2024/436, 2024.

GQ88. L. C. Guillou and J.-J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both trasmission and memory. In *EUROCRYPT'88*, *LNCS* 330, pages 123–128. Springer, Heidelberg, May 1988.

KMP16. E. Kiltz, D. Masny, and J. Pan. Optimal security proofs for signatures from identification schemes. In *CRYPTO 2016, Part II*, *LNCS* 9815, pages 33–61. Springer, Heidelberg, August 2016.

LÖ24. A. Lehmann and C. Özbay. Multi-signatures for ad-hoc and privacy-preserving group signing. In *PKC 2024, Part I*, *LNCS* 14601, pages 196–228. Springer, Heidelberg, April 2024.

Lyu12. V. Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT 2012*, *LNCS* 7237, pages 738–755. Springer, Heidelberg, April 2012.

MDM+23. E. V. Mangipudi, U. Desai, M. Minaei, M. Mondal, and A. Kate. Uncovering impact of mental models towards adoption of multi-device crypto-wallets. In *ACM CCS 2023*, pages 3153–3167. ACM Press, November 2023.
MP13. D. Micciancio and C. Peikert. Hardness of SIS and LWE with small parameters. In *CRYPTO 2013, Part I*, *LNCS* 8042, pages 21–39. Springer, Heidelberg, August 2013.
MPs19. A. Marcedone, R. Pass, and a. shelat. Minimizing trust in hardware wallets with two factor signatures. In *FC 2019*, *LNCS* 11598, pages 407–425. Springer, Heidelberg, February 2019.
MPSW18. G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille. Simple schnorr multi-signatures with applications to bitcoin. Cryptology ePrint Archive, Report 2018/068, Version 20180118:124757, 2018. https://eprint.iacr.org/archive/2018/068/20180118:124757.
MPSW19. G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille. Simple schnorr multi-signatures with applications to bitcoin. *DCC*, 87(9):2139–2164, 2019.
MR04. D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th FOCS*, pages 372–381. IEEE Computer Society Press, October 2004.
NIS24. NIST. Nist releases first 3 finalized post-quantum encryption standards. Online, 2024.
NKDM03. A. Nicolosi, M. N. Krohn, Y. Dodis, and D. Mazières. Proactive two-party signatures for user authentication. In *NDSS 2003*. The Internet Society, February 2003.
NRS21. J. Nick, T. Ruffing, and Y. Seurin. MuSig2: Simple two-round Schnorr multi-signatures. In *CRYPTO 2021, Part I*, *LNCS* 12825, pages 189–221, Virtual Event, August 2021. Springer, Heidelberg.
PW23. J. Pan and B. Wagner. Chopsticks: Fork-free two-round multi-signatures from non-interactive assumptions. In *EUROCRYPT 2023, Part V*, *LNCS* 14008, pages 597–627. Springer, Heidelberg, April 2023.
RY07. T. Ristenpart and S. Yilek. The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks. In *EUROCRYPT 2007*, *LNCS* 4515, pages 228–245. Springer, Heidelberg, May 2007.
SSH11. K. Sakumoto, T. Shirai, and H. Hiwatari. Public-key identification schemes based on multivariate quadratic polynomials. In *CRYPTO 2011*, *LNCS* 6841, pages 706–723. Springer, Heidelberg, August 2011.
Ste94. J. Stern. A new identification scheme based on syndrome decoding. In *CRYPTO'93*, *LNCS* 773, pages 13–21. Springer, Heidelberg, August 1994.
TZ23. S. Tessaro and C. Zhu. Threshold and multi-signature schemes from linear hash functions. In *EUROCRYPT 2023, Part V*, *LNCS* 14008, pages 628–658. Springer, Heidelberg, April 2023.

Appendix

A Algebraic One-More Discrete Logarithm (AOMDL) Assumption

Definition 8 (AOMDL Problem [NRS21]). *Let* GGen *be a group generation algorithm, and let* $\boldsymbol{Exp}^{\mathsf{AOMDL}}_{\mathsf{GGen},\mathcal{A}}$ *be the experiment defined below.*

$\underline{\boldsymbol{Exp}^{\mathsf{AOMDL}}_{\Pi,\mathcal{A}}}$
$(\mathbb{G}, p, g) \leftarrow \mathsf{GGen}(\lambda)$
$c := 0; q := 0$
$\vec{y} \leftarrow \mathcal{A}^{\mathcal{CH},\mathcal{DLog}_g}(\mathbb{G}, p, g)$
$\vec{x} := (x_1, \ldots, x_c)$
return $(\vec{y} = \vec{x} \wedge q < c)$

$\underline{\mathcal{O}^{\mathtt{CH}}()}$
$c := c + 1$
$x_c \leftarrow\!\$\ \mathbb{Z}_p; X := g^{x_c}$
return X

$\underline{\mathcal{O}^{\mathcal{DLog}_g}(X, (\alpha, (\beta_i)_{1 \le i \le c}))}$
$//X = g^{\alpha} \prod_{i=1}^{c} X_i^{\beta_i}$ *for* $X_i = g^{x_i}$
$q := q + 1$
return $\alpha + \sum_{i=1}^{c} \beta_i x_i$

The Algebraic One-More Discrete Logarithm (AOMDL) problem is hard for GGen *if for any* PPT *adversary* $\mathcal{A}$, $\boldsymbol{Adv}^{\mathsf{AOMDL}}_{G,\mathcal{A}} := \Pr\left[\boldsymbol{Exp}^{\mathsf{AOMDL}}_{\mathsf{GGen},\mathcal{A}} = 1\right] = \mathsf{negl}(\lambda)$.

B Proof of Theorem 3

Proof. Our goal is to build an efficient UNF-3 adversary $\mathcal{B}$ for $\mathsf{MuSig2}$ making use of an efficient UNF-3 adversary $\mathcal{A}$ for PP-$\mathsf{MuSig2}$. We proceed as follows. $\mathcal{B}$ initializes the empty tables T_{dm}, T_b, T_{sig}, and T_{apk}. Then $\mathcal{B}$ gets public parameters and the challenge public key $(\mathsf{pp}, \mathsf{pk}^*)$ from $\boldsymbol{Exp}^{\mathsf{MS\text{-}UNF\text{-}3}}_{\mathsf{MuSig2}}$ and forwards them to $\mathcal{A}$. $\mathcal{B}$ simulates $\mathcal{A}$'s queries as follows. We only describe how $\mathcal{B}$ answers fresh random oracle queries for simplicity.

Queries to $\mathcal{O}^{\mathsf{H}_{\mathsf{agg}}}_{\mathsf{PP\text{-}MuSig2}}(\mathsf{pk}_i, PK)$: Query $t \leftarrow \mathcal{O}^{\mathsf{H}_{\mathsf{agg}}}_{\mathsf{MuSig2}}(\mathsf{pk}_i, PK)$ and return t to $\mathcal{A}$.

Queries to $\mathcal{O}^{\mathsf{H}_{\mathsf{dm}}}_{\mathsf{PP\text{-}MuSig2}}(\pi, PK)$: Choose $\mathsf{dsk} \leftarrow \mathbb{Z}_p$ and compute the deterministic part of the aggregated key $\mathsf{apk}' \leftarrow \prod_{\mathsf{pk}_i} \mathsf{pk}_i^{a_i}$ and the aggregated key $\mathsf{apk} \leftarrow g^{\mathsf{dsk}} \cdot \mathsf{apk}'$ for $a_i \leftarrow \mathsf{H}_{agg}(\mathsf{pk}_i, PK)$ by simulating $\mathcal{O}^{\mathsf{H}_{\mathsf{agg}}}_{\mathsf{PP\text{-}MuSig2}}$ queries. Check if the following conditions occur:

- If $T_{apk}[\mathsf{apk}] \neq \perp$, then the event **KeyColl** occurs so abort.
- If $\mathsf{pk}^* \in PK$, then check additionally if there exists any non-empty $T_{sig}[\cdot, \mathsf{apk}, \cdot]$ or $T_b[\cdot, \cdot, \mathsf{apk}, \cdot]$ record, and abort if there exists such a record. This abort event is called **BadAgg**.

If the abort conditions do not occur, set $T_{apk}[\mathsf{apk}] \leftarrow (\pi, PK, \mathsf{apk}')$, set $T_{dm}[\pi, PK] \leftarrow \mathsf{dsk}$ and return $T[\pi, PK]$ to $\mathcal{A}$.

Queries to $\mathcal{O}^{\mathsf{H}_{\mathsf{sig}}}_{\mathsf{PP\text{-}MuSig2}}(R, \mathsf{apk}, m)$: If $T_{apk}[\mathsf{apk}] = \perp$ or $T_{apk}[\mathsf{apk}] = (\cdot, PK, \cdot)$ for $\mathsf{pk}^* \notin PK$, then choose $T_{sig}[R, \mathsf{apk}, m] \leftarrow \mathbb{Z}_p$ and return $T_{sig}[R, \mathsf{apk}, m]$. Otherwise, lookup $T_{apk}[\mathsf{apk}] = (\pi, PK, \mathsf{apk}')$. Make a $T_{sig}[R, \mathsf{apk}, m] \leftarrow \mathcal{O}^{\mathsf{H}_{\mathsf{sig}}}_{\mathsf{MuSig2}}(R, \mathsf{apk}', \mathsf{tobin}(\mathsf{apk}) \| m)$ query and return $T_{sig}[R, \mathsf{apk}, m]$ to $\mathcal{A}$.

Queries to $\mathcal{O}^{\mathsf{H}_{\mathsf{b}}}_{\mathsf{PP\text{-}MuSig2}}(R_1, R_2, \mathsf{apk}, m)$: If $T_{apk}[\mathsf{apk}] = \perp$ or $T_{apk}[\mathsf{apk}] = (\cdot, PK, \cdot)$ for $\mathsf{pk}^* \notin PK$, choose $T_b[R_1, R_2, \mathsf{apk}, m] \leftarrow \mathbb{Z}_p$, and return $T_b[R_1, R_2, \mathsf{apk}, m]$. Otherwise, lookup $T_{apk}[\mathsf{apk}] = (\pi, PK, \mathsf{apk}')$. Make a $T_b[R_1, R_2, \mathsf{apk}, m] \leftarrow \mathcal{O}^{\mathsf{H}_{\mathsf{b}}}_{\mathsf{MuSig2}}(R_1, R_2, \mathsf{apk}', \mathsf{tobin}(\mathsf{apk}) \| m)$ query and return $T_b[R_1, R_2, \mathsf{apk}, m]$ to $\mathcal{A}$.

1st round Signing Queries ($\mathcal{O}^{\mathtt{MulSign}_1}_{\mathsf{PP\text{-}MuSig2}}(\mathsf{sid}_i, PK_i, \mathsf{apk}_i, \pi_i, m_i)$): If $\mathsf{sid}_i \in S_1$ or $\mathsf{pk}^* \notin PK_i$, or $\mathsf{VfKAg}(PK_i, \mathsf{apk}_i, \pi_i) \neq 1$, then return $\perp$ to $\mathcal{A}$. Otherwise, lookup $T_{apk}[apk_i] \leftarrow (\pi, PK, \mathsf{apk}')$. Make a $\mathsf{ps}^{(i,1)} \leftarrow \mathcal{O}^{\mathtt{MulSign}_1}_{\mathsf{MuSig2}}(\mathsf{sid}_i, PK, \mathsf{apk}', \perp, \mathsf{tobin}(\mathsf{apk}_i) \| m_i)$ query and return $\mathsf{ps}^{(i,1)}$ to $\mathcal{A}$.

2nd round Signing Query ($\mathcal{O}^{\mathtt{MulSign}_2}(\mathsf{sid}_i, \mathsf{in}^{(i,1)})$): If $\mathsf{sid}_i \in S_2$ or $\mathsf{sid}_i \notin S_1$, then return $\perp$ to $\mathcal{A}$. Otherwise, make a $\mathsf{ps}^{(i,2)} \leftarrow \mathcal{O}^{\mathtt{MulSign}_2}_{\mathsf{MuSig2}}(\mathsf{sid}_i, \mathsf{in}^{(i,1)})$ query and return $\mathsf{ps}^{(i,2)}$ to $\mathcal{A}$.

We first argue that $\mathcal{B}$'s simulation of $\mathcal{A}$'s view is perfect unless $\mathcal{B}$ aborts. For the random oracle queries, observe that $\mathcal{B}$ returns a uniformly random value from $\mathbb{Z}_p$ unless it aborts for each fresh query. For a first round signing query, $\mathcal{B}$ forwards the request to $\boldsymbol{Exp}^{\mathsf{MS\text{-}UNF\text{-}3}}_{\mathsf{MuSig2}}$ for the PK it looks up from the table T_{agg} instead of apk_i and the message $\mathsf{tobin}(\mathsf{apk}_i) \| m_i$ instead of m_i. First, if **KeyColl** event does not occur and VfKAg call for the oracle holds, then PK_i is identical to PK that $\mathcal{B}$ looks up from T_{agg}. Second, if the events **KeyColl** and **BadAgg** do not occur, the random oracle queries for H_{sig} and H_b programmed in a way to ensure that $\mathcal{B}$ returns consistent signature shares.

Next we argue that $\mathcal{B}$ aborts with a negligible probability. Observe that $T_{dm}[\pi, PK]$ is sampled randomly. Thus, for each $\mathcal{O}^{\mathsf{H}_{\mathsf{dm}}}$ query, the event **KeyColl** can occur with probability less than $(q_{\mathsf{H}_{dm}} + q_S)/p$ for $q_{\mathsf{H}_{dm}}$ and q_S being the number of $\mathcal{O}^{\mathsf{H}_{\mathsf{dm}}}$ and the first round signing queries that $\mathcal{A}$ makes. We conclude that $\Pr[\mathbf{KeyColl}] \leq (q_{\mathsf{H}_{dm}} + q_S)^2/p$. Similarly, as the uniform $T_{dm}[\pi, PK]$ provides uniform aggregated keys, the event **BadAgg** occurs with probabiltity less than $(q_{\mathsf{H}_{dm}} + q_S)/p$ for each $\mathcal{O}^{\mathsf{H}_{\mathsf{sig}}}$ and $\mathcal{O}^{\mathsf{H}_{\mathsf{b}}}$ query. Thus, for $q_{H_{sig}}$ and q_{H_b} being the number of queries to these oracles, we conclude that $\Pr[\mathbf{BaddAgg}] \leq (q_{H_{sig}} + q_{H_b})(q_{\mathsf{H}_{dm}} + q_S)/p$.

Now we argue that if $\mathcal{A}$ wins $\boldsymbol{Exp}^{\mathsf{MS\text{-}UNF\text{-}3}}_{\mathsf{PP\text{-}MuSig2}}$ game against $\mathcal{B}$ and $\mathcal{B}$ does not abort, then $\mathcal{B}$ can win $\boldsymbol{Exp}^{\mathsf{MS\text{-}UNF\text{-}3}}_{\mathsf{MuSig2}}$. Let $(\sigma^* \leftarrow (R^*, s^*), m^*, \mathsf{apk}^*, \pi^*, PK^*)$ be $\mathcal{A}$'s forgery. $\mathcal{B}$ looks up $\mathsf{dsk}^* \leftarrow T_{dm}[\pi^*, PK^*]$, $(\cdot, \cdot, \mathsf{apk}') \leftarrow T_{apk}[\mathsf{apk}^*]$, and $c^* \leftarrow T_{sig}[R^*, \mathsf{apk}^*, m^*]$. Finally, $\mathcal{B}$ returns the following forgery to $\boldsymbol{Exp}^{\mathsf{MS\text{-}UNF\text{-}3}}_{\mathsf{MuSig2}}$:

$$(\sigma \leftarrow (R^*, s^* - \mathsf{dsk}^* \cdot c^*), \mathsf{tobin}(\mathsf{apk}^*) \| m^*, \mathsf{apk}', \perp, PK^*)$$

It is straightforward to observe that the forgery is a valid signature, and we argue that it is a non-trivial $\mathsf{UNF\text{-}3}$ forgery. We investigate two cases to show that the forgery which is built by $\mathcal{B}$ is non-trivial.

Case 1: $(m^*, \cdot, \cdot) \notin Q_{\mathsf{PP\text{-}MuSig2}}$. In this case, regardless of what we prefix the messages with, we know that $(\mathsf{tobin}(\mathsf{apk}^*) \| m^*, \cdot, \cdot) \notin Q_{\mathsf{MuSig2}}$.

Case 2: $(m^*, PK, \mathsf{apk}) \in Q_{\mathsf{PP\text{-}MuSig2}}$, for $(PK, \mathsf{apk}) \neq (PK^*, \mathsf{apk}^*)$. The signing query of $\mathcal{B}$ to $\boldsymbol{Exp}^{\mathsf{MS\text{-}UNF\text{-}3}}_{\mathsf{MuSig2}}$ for (m^*, PK, apk) is on the message $\mathsf{tobin}(\mathsf{apk}) \| m^*$. The message $\mathsf{tobin}(\mathsf{apk}) \| m^*$ is equal to $\mathsf{tobin}(\mathsf{apk}^*) \| m^*$ only when the event **KeyColl** occurs, so if $\mathcal{A}$ wins without $\mathcal{B}$ aborting, then $(\mathsf{tobin}(\mathsf{apk}^*) \| m^*, PK^*, \mathsf{apk}) \notin Q_{\mathsf{MuSig2}}$.

Thus we conclude that,

$$\Pr\left[\boldsymbol{Exp}^{\text{MS-UNF-3}}_{\text{PP-MuSig2},\mathcal{A}}(\lambda)=1\right] \leq \Pr\left[\boldsymbol{Exp}^{\text{MS-UNF-3}}_{\text{MuSig2},\mathcal{B}}(\lambda)=1\right] + \Pr[\textbf{KeyColl}] + \Pr[\textbf{BadAgg}]$$

$$\text{therefore, } \Pr\left[\boldsymbol{Exp}^{\text{MS-UNF-3}}_{\text{PP-MuSig2},\mathcal{A}}(\lambda)=1\right] \leq \Pr\left[\boldsymbol{Exp}^{\text{MS-UNF-3}}_{\text{MuSig2},\mathcal{B}}(\lambda)=1\right] + \frac{(q_{\mathsf{H}_{dm}}+q_S)(q_{H_{sig}}+q_{H_b}+q_{\mathsf{H}_{dm}}+q_S)}{p}$$

□

C Additional Cryptographic Background and Definitions

C.1 Key-Homomorphic Signatures

We recall that for key-homomorphic scheme one requires that secret and public key elements belong to groups $(\mathbb{H},+)$ and $(\mathbb{E},\cdot)$, where group operations, inversions, membership testing, and sampling from the uniform distribution can all be performed efficiently.

Definition 9 (Secret Key to Public Key Homomorphism [DS19]). *A signature scheme Σ provides a secret key to public key homomorphism, if there exists an efficiently computable map $\mu : \mathbb{H} \to \mathbb{E}$ such that for all $sk, sk' \in \mathbb{H}$ it holds that $\mu(\mathsf{sk}+\mathsf{sk}') = \mu(\mathsf{sk})\cdot\mu(\mathsf{sk}')$, and for all $(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{Kg}$, it holds that $\mathsf{pk} = \mu(\mathsf{sk})$.*

Definition 10 (Key-Homomorphic Signatures [DS19]). *A signature scheme is called key-homomorphic, if it provides a secret key to public key homomorphism and an additional* PPT *algorithm* Adapt*, defined as:*

$\mathsf{Adapt}(\mathsf{pk}, m, \sigma, \Delta)$*: Takes a public key* pk*, a message m, a signature σ, and a function Δ as input, and outputs a public key pk' and a signature σ', such that for all $\Delta \in \mathbb{H}$ and all $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Kg}(1^\kappa)$, all messages $m \in \mathcal{M}$, and all σ with $\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1$ and $(\mathsf{pk}', \sigma') \leftarrow \mathsf{Adapt}(\mathsf{pk}, m, \sigma, \Delta)$, it holds that*

$$\Pr[\mathsf{Verify}(\mathsf{pk}', m, \sigma') = 1] = 1 \quad \wedge \quad \mathsf{pk}' = \mu(\Delta)\cdot\mathsf{pk}.$$

Importantly, adapted signatures and related key pair should look like freshly computed ones. This is captured by the adaptability notion shown below.

Definition 11 (Adaptability of Signatures [DS19]). *A key-homomorphic signature scheme provides adaptability of signatures, if for every $\kappa \in \mathbb{N}$ and every message $m \in \mathcal{M}$, it holds that*

$$[(\mathsf{sk},\mathsf{pk}), \mathsf{Adapt}(\mathsf{pk}, m, \mathsf{Sign}(\mathsf{sk}, m), \Delta)],$$

where $(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{Kg}(1^\kappa)$, $\Delta \leftarrow\!\!\$\ \mathbb{H}$, and

$$[(\mathsf{sk}, \mu(\mathsf{sk})), (\mu(\mathsf{sk})\cdot\mu(\Delta), \mathsf{Sign}(\mathsf{sk}+\Delta, m))],$$

where $\mathsf{sk} \leftarrow\!\!\$\ \mathbb{H}$, $\Delta \leftarrow\!\!\$\ \mathbb{H}$, are identically distributed.

A stronger notion known as *perfect adaption* (see Definition 17 from [DS19]) states that adapted signatures should also be indistinguishable from freshly computed ones even when the initial signature used in Adapt is known.

Some key-homomorphic schemes also support another property: signatures for the same message m can be *publicly combined* to compute a signature for m under the *combined* original public keys. We recall the formal definition next.

Definition 12 (Publicly Key-Homomorphic Signature [DS19]). *A signature scheme is called publicly key-homomorphic, if it provides a secret key to public key homomorphism and an additional* PPT *algorithm* Combine*, defined as:*

$\mathsf{Combine}((\mathsf{pk}_i)_{i=1}^n, m, (\sigma_i)_{i=1}^n)$*: This algorithm takes as input public keys $(\mathsf{pk}_i)_{i\in[n]}$, a message m, and signatures $(\sigma_i)_{i\in[n]}$, and outputs a public key pk' and a signature σ', such that for all $n > 1$, for all $((\mathsf{sk}_i,\mathsf{pk}_i) \leftarrow \mathsf{Kg}(1^\kappa))_{i=1}^n$, all messages $m \in \mathcal{M}$, and all $(\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_i, m))_{i=1}^n$ and $(\mathsf{pk}', \sigma') \leftarrow \mathsf{Combine}((\mathsf{pk}'_i)_{i=1}^n, m, (\sigma_i)_{i=1}^n)$ it holds that*

$\mathsf{Pg}(1^\kappa)$: Run $\mathsf{pp} \leftarrow \Sigma.\mathsf{Pg}(1^\kappa)$ and return pp.
$\mathsf{Kg}(\mathsf{pp})$: Run $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma.\mathsf{Kg}(\mathsf{pp})$ and return $(\mathsf{sk}, \mathsf{pk})$.
$\mathsf{Sign}(\mathsf{pp}, \mathsf{pk}, m, \mathsf{sk})$: Let $i \in [n]$. Every participating S_i with $\mathsf{pk}_i \in \mathrm{PK}$ proceeds as follows:
- Compute $\sigma_i \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}_i, m)$ and broadcast σ_i.
- Receive all signatures σ_j for $j \neq i$.
- Compute $(\mathsf{pk}, \sigma) \leftarrow \mathsf{Combine}(\mathrm{PK}, m, (\sigma_\ell)_{\ell \in [n]})$ and output σ.

Verify$(\mathsf{pp}, \mathrm{PK}, m, \sigma)$: Return 1 iff $\Sigma.\mathsf{Verify}\Big(\prod_{\mathsf{pk} \in \mathrm{PK}} \mathsf{pk}, m, \sigma\Big) = 1$.

Fig. 9. Black-Box construction of multi-signatures [DS19].

Prover: sk		Verifier: pk
$(R, \mathsf{st}) \leftarrow P_1(\mathsf{sk})$	$\xrightarrow{R}$	
	$\xleftarrow{c}$	$c \leftarrow \mathsf{ChSet}$
$s \leftarrow P_2(\mathsf{sk}, R, c, \mathsf{st})$	$\xrightarrow{s}$	$d = \mathsf{V}(\mathsf{pk}, R, c, s)$

$\underline{\mathsf{Sign}(\mathsf{sk}, m)}$:	$\underline{\mathsf{Verify}(\mathsf{pk}, m, \sigma)}$:
$(R, \mathsf{st}) \leftarrow P_1(\mathsf{sk})$	**Parse** σ **as** (R, s)
$c \leftarrow \mathsf{H}(R, \mathsf{pk}, m)$	$c \leftarrow \mathsf{H}(R, \mathsf{pk}, m)$
$s \leftarrow P_2(\mathsf{sk}, R, c, \mathsf{st})$	**return** $\mathsf{V}(\mathsf{pk}, R, c, s)$
return $\sigma = (R, s)$	

Fig. 10. Canonical identification scheme (left side) and signature scheme (right side).

$$\mathsf{pk}' = \textstyle\prod_{i=1}^{n} \mathsf{pk}_i \quad \textit{and} \quad \Pr[\mathsf{Verify}(\mathsf{pk}', m, \sigma') = 1] = 1.$$

In [DS19], it was shown that Schnorr signatures are adaptable according to Definition 11 when considering the Adapt algorithm as follows:

$\mathsf{Adapt}(\mathsf{pk}, m, \sigma, \Delta)$: Let $\Delta \in \mathbb{Z}_p$, $\mathsf{pk} = g^{\mathsf{sk}}$ and $c \leftarrow \mathsf{H}_1(R, m)$. Return (pk', σ'), where $\mathsf{pk}' \leftarrow g^{\mathsf{sk}} \cdot g^{\Delta}$ and $\sigma' \leftarrow (s', R)$ with $s' \leftarrow s + c \cdot \Delta \bmod p$.

C.2 Black-Box Construction of Multi-signatures From Key-Homomorphic Signatures

Derler and Slamanig showed that multi-signatures can be built in a black-box way from publicly key-homomorphic signatures [DS19] (Theorem 3). Their formalization follows the model by Ristenpart and Yilek [RY07] for key-registration but other models with extractable secret keys are also suitable. We recall their black-box construction in Figure 9, referring the reader to [DS19] for further details.

C.3 Signatures from identification schemes.

A *canonical* identification scheme, formalized by Abdalla *et al.* [AABN02], is a three-move public-key authentication protocol as depicted in Fig.C.3. In the following, we recall the formal definitions from [KMP16].

Definition 13 (Canonical Identification Scheme). *A canonical identification scheme* ID *is defined as a tuple of algorithms* $\mathsf{ID} := (\mathsf{IGen}, \mathsf{P}, \mathsf{ChSet}, \mathsf{V})$.

$\mathsf{Pg}(1^\lambda) \to \mathsf{pp}$**:** *On input security parameter* 1^λ*, it outputs public parameters* pp*, which are implicitly passed to other algorithms.*
$\mathsf{IGen}() \to (\mathsf{pk}, \mathsf{sk})$**:** *On input the security parameter* ppar*, outputs public and secret keys* $(\mathsf{pk}, \mathsf{sk})$*. We assume that* pk *defines* ChSet*, the set of challenges.*
$\mathsf{P}(\mathsf{sk}) \to (R, s)$**:** *The prover algorithm* $\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2)$ *is split into two algorithms.* P_1 *takes as input the secret key* sk *and returns a commitment* R *and a state* st*;* P_2 *takes as input the secret key* sk*, a commitment* R*, a challenge* c*, and a state* st *and returns a response* s*.*
$\mathsf{V}(\mathsf{pk}) \to 0/1$**:** *The verifier algorithm* V *takes the public key* pk *and the conversation transcript* (R, c, s) *as input and outputs 1 iff the transcript is valid.*

We require that for all $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{IGen}(\mathsf{pp})$*, all* $(\mathsf{R}, \mathsf{st}) \in \mathsf{P}_1(\mathsf{st})$*, all* $c \in \mathsf{ChSet}$ *and all* $s \in \mathsf{P}_2(\mathsf{sk}, \mathsf{R}, c, \mathsf{St})$*, we have* $\mathsf{V}(\mathsf{pk}, \mathsf{R}, c, s) = 1$.

We say that $\Sigma := (\mathsf{Pg}, \mathsf{Kg}, \mathsf{Sign}, \mathsf{Verify})$ is a signature from an identification scheme $\mathsf{ID} = (\mathsf{IGen}, \mathsf{P}, \mathsf{ChSet}, \mathsf{V})$ when Σ employs a hash function $\mathsf{H} : \{0,1\} \to \mathsf{ChSet}$ and is defined in terms of ID as shown in Fig.C.3 (Pg and Kg remain unchanged). As shown, we consider the *strong* Fiat-Shamir transform.

$\boldsymbol{Exp}^{\mathsf{MS\text{-}UNF}}_{\Pi,\mathcal{A}}$

$\mathsf{pp} \leftarrow \mathsf{Pg}(1^\lambda); (\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{Kg}(\mathsf{pp}); Q \leftarrow \emptyset; S \leftarrow \emptyset; ctr \leftarrow 0$

$(\sigma, m^*, L^*) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{MulSign}}}(\mathsf{pp}, \mathsf{pk}^*)$

return $\mathsf{Vf}(\mathsf{KeyAgg}(L^*), \sigma, m^*) \wedge (L^*, m^*) \notin Q \wedge \mathsf{pk}^* \in L^*$

$\mathcal{O}^{\mathsf{MulSign}_1}(L, m)$

if $\mathsf{pk}^* \notin L$ **return** $\perp$

$ctr \leftarrow ctr + 1; S \leftarrow S \cup \{ctr\}; Q := Q \cup \{(m, L)\}$

$(msg_1, \mathsf{st}_{ctr}) \leftarrow \mathsf{MulSign}_1(\mathsf{sk}^*, L, m)$

return msg_1

$\mathcal{O}^{\mathsf{MulSign}_2}(id, \{msg_2, \ldots, msg_n\})$

if $id \notin S$ **return** $\perp$

$\mathsf{ps}_1 \leftarrow \mathsf{MulSign}_2(\mathsf{st}_{id}, \{msg_2, \ldots\})$

$S \leftarrow S \setminus \{id\}$

return ps_1

Fig. 11. Unforgeability for MS schemes with deterministic key aggregation.

C.4 Multi-signature Unforgeability

Definition 14 (MS Unforgeability). *A multi-signature scheme Π is* MS-UNF *if for all PPT adversaries $\mathcal{A}$ in the experiment from Figure 11 it holds that:*

$$\Pr\left[\boldsymbol{Exp}^{\mathsf{MS\text{-}UNF}}_{\Pi,\mathcal{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

D Parameters for DualMS

We give constants and parameters used for both DualMS and PP-DualMS in Table 1 and refer the readers to [Che23, Table 1] for more details.

Table 1. Parameter descriptions for PP-DualMS. Differences with DualMS are marked in blue. We refer the reader to [Che23, Table 1] for a full description of each parameter.

Parameters	Description
n	Number of parties in the multi-signature
C	Challenge set where $C = \{c \in R : \|c\|_\infty = 1, \|c\|_1 = \kappa\}$
$\mathcal{S}_\eta$	Key set where $\mathcal{S}_\eta = \{s \in R_q : \|s\|_\infty \leq \eta\}$
T	$2\kappa^2\eta\sqrt{Nm}$
T'	$2\kappa^2\eta'\sqrt{Nm'}$
s	Deviation parameter for $\mathbf{y}_1$, which is $\sqrt{2\pi\alpha T}$
s'	Deviation parameter for $\mathbf{r}_1$, which is $\sqrt{2\pi\alpha T'}$
t	Parameter for defining M, which is $\omega(\sqrt{\log(mN)})$ and $o(\log(mN)$
M	Expected number of repetitions per signer, which is $\exp(t/\alpha + 1/(2\alpha^2))^2$
γ	Tail bound parameter for the discrete Gaussian distribution
B_n	Accepted norm of $\tilde{\mathbf{z}}$, which is $\sqrt{n}\sqrt{Nl}\frac{s\gamma}{\sqrt{2\pi}}$
B'_n	Accepted norm of $\tilde{\mathbf{r}}$, which is $\sqrt{n}\sqrt{Nl'}\frac{s\gamma}{\sqrt{2\pi}}$
B''_n	Accepted norm of $\tilde{\mathbf{e}}$, which is $\sqrt{n(s^2+s'^2)}\sqrt{Nk}\frac{\gamma}{\sqrt{2\pi}}$

E Privacy-Preserving Guillou-Quisquater Multi-signature

In [GQ88], Guillou and Quisquater presented a identification protocol based on RSA that can be turned into a signature scheme (henceforth referred to as GQ) applying the Fiat-Shamir transform. Unlike Schnorr-based signatures, GQ requires a trusted setup to generate the system parameter $N = pq$ for p and q primes. Alternatively, some distributed protocols to compute RSA moduli exist in the literature with different trade-offs (see, for instance, [DMRT21]). In Figure E, we present privacy-preserving variant of the GQ multi-signature from [CJKT06, BN07]. Security of the modified scheme can be proven similarly to that of our PP-MuSig2 but based on the generalized forking lemma from [BN06].

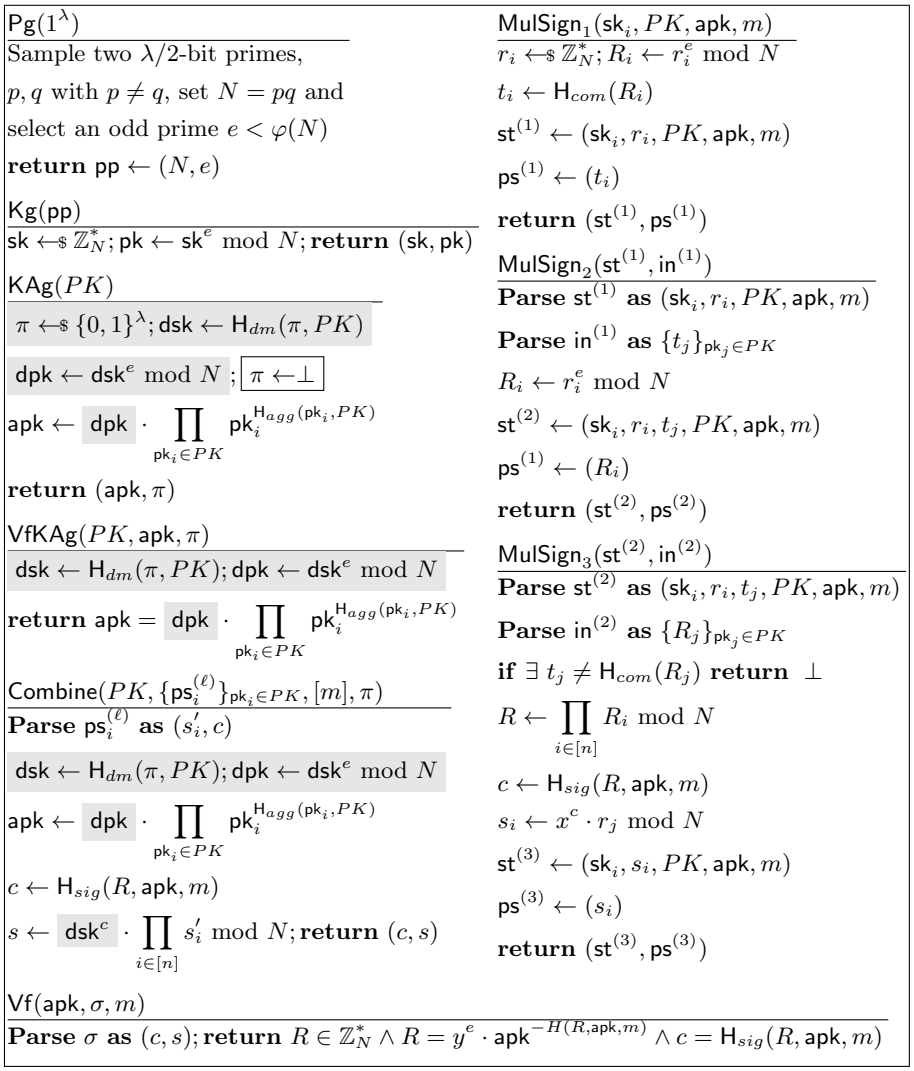

Fig. 12. Description of GQ and our multi-signature scheme PP-GQ for hash functions $\mathsf{H}_{sig}, \mathsf{H}_{agg} : \{0,1\}^* \to \mathbb{Z}_e$ and $\mathsf{H}_{dm}, \mathsf{H}_{com} : \{0,1\}^* \to \mathbb{Z}_N^*$. Codes in gray boxes only occur for PP-GQ. The code in black box only occurs for GQ.