T-Spoon: Tightly Secure Two-Round Multi-Signatures with Key Aggregation

Renas Bacho [1,2] Benedikt Wagner [3]

May 12, 2025

[1] CISPA Helmholtz Center for Information Security
renas.bacho@cispa.de
[2] Saarland University

[3] Ethereum Foundation
benedikt.wagner@ethereum.org

Abstract

Multi-signatures over pairing-free cyclic groups have seen significant advancements in recent years, including achieving two-round protocols and supporting key aggregation. Key aggregation enables the combination of multiple public keys into a single succinct aggregate key for verification and has essentially evolved from an optional feature to a requirement.

To enhance the concrete security of two-round schemes, Pan and Wagner (Eurocrypt 2023, 2024) introduced the first tightly secure constructions in this setting. However, their schemes do not support key aggregation, and their approach inherently precludes a single short aggregate public key. This leaves the open problem of achieving tight security and key aggregation simultaneously.

In this work, we solve this open problem by presenting the first tightly secure two-round multi-signature scheme in pairing-free groups supporting key aggregation. As for Pan and Wagner's schemes, our construction is based on the DDH assumption. In contrast to theirs, it also has truly compact signatures, with signature size asymptotically independent of the number of signers.

Contents

1 Introduction **3**
- 1.1 Our Contribution . . . 4
- 1.2 Related Work . . . 4

2 Technical Overview **5**
- 2.1 Background: Multi-Signatures from Committed Lossy Identification . . . 5
- 2.2 Tight Security and the Problem with Aggregation . . . 7
- 2.3 Overview of Our Solution . . . 8
- 2.4 Proving Security . . . 9

3 Preliminaries **10**
- 3.1 Pairing-Free Groups and Assumptions . . . 10
- 3.2 Multi-Signatures . . . 11

4 Our Construction **13**
- 4.1 Building Block: Linear Function Families . . . 13
- 4.2 Building Block: Commitments . . . 15
- 4.3 Multi-Signature Construction . . . 17

5 Instantiation and Efficiency **22**
- 5.1 Linear Function Family . . . 22
- 5.2 Commitment . . . 22
- 5.3 Efficiency . . . 24

A Postponed Proofs **31**

1 Introduction

A multi-signature scheme [IN83] enables a group of N signers, each holding a key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$, to sign a message m in a potentially interactive signing protocol. The resulting signature σ can be verified against m and the list of public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_N$, providing the guarantee that *all* N signers have participated in the signing process. Non-trivial multi-signatures should additionally satisfy *compactness*, meaning that the size of the signature is sublinear (ideally constant) in N. In this work, we study two-round multi-signatures over pairing-free cyclic groups. Our model is the widely established *plain public key model* [BN06], where honest signers generate their keys independently, and malicious signers may let their keys depend on other keys.

Key Aggregation. A particularly valuable feature for improving efficiency in multi-signature schemes is *key aggregation*, introduced by Maxwell et al. [MPSW18]. A scheme supports key aggregation, if the N public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_N$ can be aggregated into a short aggregate public key $\tilde{\mathsf{pk}}$ that represents the group of N signers. Verification of the signature is then performed with respect to $\tilde{\mathsf{pk}}$ rather than the full list of public keys. Since its introduction, key aggregation has been a focal point of research, and significant advancements have been made to design schemes with key aggregation [BDN18, MPSW19, FH20b], and in particular *two-round* schemes with key aggregation [AB21, BD21, CKM21, NRS21, TSS+23, TZ23]. In fact, key aggregation has evolved from an optional feature to an essential requirement for modern multi-signature schemes.

Concrete Security. Security proofs of early two-round constructions [BD21, NRS21, TZ23] in the random oracle model[1] employed nested rewinding techniques, which resulted in extremely loose security bounds. Specifically, the bounds are of the form[2]

$$\epsilon^4 \leq Q_H^3 \epsilon',$$

where ϵ represents the adversary's advantage against the scheme, ϵ' is the reduction's advantage against the underlying assumption, and Q_H denotes the number of random oracle queries. To illustrate how loose this bound is, suppose $\epsilon' \leq 2^{-128}$ and $Q_H = 2^{20}$. Under these parameters, the bound implies $\epsilon \leq 2^{-17}$, which is far too weak to provide meaningful security guarantees.

This unsatisfactory state of affairs was independently identified by Takemure et al. [TSS+23] and by Pan and Wagner [PW23]. Both works addressed the issue by proposing new two-round schemes that avoid rewinding. Namely:

- Takemure et al.'s scheme (TSSHO) and the first scheme by Pan and Wagner (Chopsticks I) have a security bound of the form
$$\epsilon \leq \Theta(Q_S)\epsilon',$$
where Q_S denotes the number of signing queries.
- Pan and Wagner also introduced the first *tightly secure* scheme (Chopsticks II), i.e., a scheme with a security bound of the form
$$\epsilon \leq \Theta(1)\epsilon'.$$

In a follow-up work [PW24], Pan and Wagner further improved the efficiency of their constructions, resulting in the schemes Toothpicks I and Toothpicks II.

The Open Problem. Despite these advances, a significant challenge remains: among the rewinding-free schemes, only the non-tight schemes TSSHO, Chopsticks I, and Toothpicks I support key aggregation. Even more problematic, the techniques used by Pan and Wagner to achieve tight security in Chopsticks II and Toothpicks II appear fundamentally incompatible with key aggregation. Specifically, these schemes require each signer to maintain two keys, resulting in 2^N possible aggregate public keys for N signers. This leaves the open problem of constructing a tightly secure two-round scheme with key aggregation. Such a construction would combine the strong concrete security guarantees of tight security with the efficiency and practicality of using a single joint public key.

[1] Some schemes achieve tighter bounds in the algebraic group model [FKL18], but we focus on proofs that do not rely on such additional idealizations.

[2] For simplicity, we omit additional additive terms here.

Scheme	Key Agg	Assumption	Loss	Pub Key	Communication	Signature
Musig-DN [NRSW20]	✓	DDH, DLOG	$\Theta(Q_H^3/\epsilon^3)$	$1\langle\mathbb{G}\rangle$	$2\langle\mathbb{G}\rangle + 1\langle\mathbb{Z}_p\rangle + \lvert\pi\rvert$	$1\langle\mathbb{G}\rangle + 1\langle\mathbb{Z}_p\rangle$
Musig2 [NRS21]	✓	AOMDL	$\Theta(Q_H^3/\epsilon^3)$	$1\langle\mathbb{G}\rangle$	$4\langle\mathbb{G}\rangle + 1\langle\mathbb{Z}_p\rangle$	$1\langle\mathbb{G}\rangle + 1\langle\mathbb{Z}_p\rangle$
HBMS [BD21]	✓	DLOG	$\Theta(Q_S^4 Q_H^3/\epsilon^3)$	$1\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle + 2\langle\mathbb{Z}_p\rangle$	$1\langle\mathbb{G}\rangle + 2\langle\mathbb{Z}_p\rangle$
TZ [TZ23]	✓	DLOG	$\Theta(Q_H^3/\epsilon^3)$	$1\langle\mathbb{G}\rangle$	$4\langle\mathbb{G}\rangle + 2\langle\mathbb{Z}_p\rangle$	$1\langle\mathbb{G}\rangle + 2\langle\mathbb{Z}_p\rangle$
TSSHO [TSS+23]	✓	DDH	$\Theta(Q_S)$	$2\langle\mathbb{G}\rangle$	$2\langle\mathbb{G}\rangle + 2\langle\mathbb{Z}_p\rangle$	$3\langle\mathbb{Z}_p\rangle$
Chopsticks I [PW23]	✓	DDH	$\Theta(Q_S)$	$2\langle\mathbb{G}\rangle$	$3\langle\mathbb{G}\rangle + 1\langle\mathbb{Z}_p\rangle + \lambda$	$3\langle\mathbb{G}\rangle + 4\langle\mathbb{Z}_p\rangle$
Chopsticks II [PW23]	✗	DDH	$\Theta(1)$	$4\langle\mathbb{G}\rangle$	$6\langle\mathbb{G}\rangle + 2\langle\mathbb{Z}_p\rangle + \lambda + 1$	$6\langle\mathbb{G}\rangle + 8\langle\mathbb{Z}_p\rangle + N$
Toothpicks I [PW24]	✓	DDH	$\Theta(Q_S)$	$2\langle\mathbb{G}\rangle$	$2\langle\mathbb{G}\rangle + 1\langle\mathbb{Z}_p\rangle + \lambda$	$3\langle\mathbb{Z}_p\rangle + 2\lambda$
Toothpicks II [PW24]	✗	DDH	$\Theta(1)$	$4\langle\mathbb{G}\rangle$	$2\langle\mathbb{G}\rangle + 1\langle\mathbb{Z}_p\rangle + \lambda + 1$	$3\langle\mathbb{Z}_p\rangle + 2\lambda + N$
T-Spoon (Ours)	✓	DDH	$\Theta(1)$	$2\langle\mathbb{G}\rangle$	$3\langle\mathbb{G}\rangle + 1\langle\mathbb{Z}_p\rangle + \lambda + 1$	$2\langle\mathbb{G}\rangle + 9\langle\mathbb{Z}_p\rangle$

Table 1: Comparison of two-round multi-signature schemes in the discrete logarithm setting without pairings in the plain public key model. We compare whether the schemes support key aggregation, the assumption the schemes rely on, and the security loss, where Q_H, Q_S denote the number of random oracle and signing queries, respectively, and ϵ denotes the advantage of an adversary against the scheme. For the security loss, we do not consider proofs in the algebraic group model [FKL18]. We also compare the size of public keys, the communication complexity per signer, and the signature size. We denote the size of a group element by $\langle\mathbb{G}\rangle$ and the size of a field element by $\langle\mathbb{Z}_p\rangle$. Here, λ is the security parameter and N is the number of signers. In Musig-DN, $|\pi|$ denotes the size of a zk-SNARK and DDH holds in a different cyclic group $\mathbb{E}$, which is used for proving correct computation of the deterministic nonces.

1.1 Our Contribution

We address this open problem by introducing T-Spoon, a novel tightly secure two-round multi-signature scheme. In the setting of pairing-free cyclic groups, T-Spoon is the *first* two-round scheme to achieve the following key properties:

- *Tight Security and Key Aggregation.* Our scheme is the first that is tightly secure and simultaneously supports key aggregation.
- *Tight Security and Compactness.* T-Spoon is also the first tightly secure scheme to offer asymptotically compact signatures, with signature sizes of $\Theta(\lambda)$ bits, where λ is the security parameter. In contrast, signatures in Toothpicks II [PW24] grow linearly in the number of signers, requiring $\Omega(\lambda + N)$ bits for N signers.

A comparison with existing two-round schemes is provided in Table 1. While the compactness property is primarily of theoretical interest and an asymptotic improvement in nature, the concrete signature size of T-Spoon improves over the state-of-the-art Toothpicks II [PW24] as soon as[3] $N \geq 7 \cdot 256$. In addition, T-Spoon reduces the public key size by a factor of 2 compared to Toothpicks II.

As our technical starting point, we use the committed lossy identification techniques from [TSS+23, PW23, PW24]. To achieve our result, we then introduce a novel *signer partitioning technique*. This technique fully eliminates the need for two public keys per signer and allows us to do key aggregation.

1.2 Related Work

Here, we discuss more related work on multi-signatures.

Modeling Multi-Signatures. Following the introduction of multi-signatures by Itakura and Nakamura [IN83], several schemes have been proposed [OO93, Har94, LHL95, OO98]. Most of them [OO93, HMP95, OO98, OO99] require a trusted third party for key generation, while others [Har94, LHL95] have been shown to be insecure due to rogue-key attacks [HMP95, Lan96, MH96]. In such an attack, the adversary can choose its public key as a function of honest signers' keys, allowing it to forge signatures easily. In order to prevent rogue-key attacks without relying on a trusted third party for key generation, subsequent works have opted for interactive key generation [MOR01] or the knowledge of secret key (KOSK) assumption [Bol03, LOS+06], which requires the adversary to reveal its secret keys at public key registration directly. Notably, some works [BJ08, DEF+19, CKM21] have implemented the KOSK

[3] This assumes $\langle\mathbb{G}\rangle = \langle\mathbb{Z}_p\rangle = 256$.

assumption by requiring each party to include a proof of possession of its secret key at public key registration. A more realistic model, which we also follow in this work, is the plain public key model, first formalized by Bellare and Neven [BN06]. This model is now widely accepted as the most robust and realistic one, and we refer to [BN06, RY07] for more discussion on these models.

Constructions over Pairing-Free Cyclic Groups. Many multi-signatures with several rounds of communication for signing have been proposed. Three-round schemes have been constructed in [MOR01, BN06, BJ08, BCJ08, BDN18, MPSW19, FH20b], with all of them being based on standard assumptions (concretely, DDH and DLOG). Among them, the works [BDN18, MPSW19, FH20b] support key aggregation. Further, two-round schemes have been constructed in [DEF+19, NRSW20, AB21, BD21, CKM21, NRS21, LK23, TSS+23, TZ23, PW23, PW24], with some of them being partially non-interactive (i.e., the first signing round is message-independent) [NRS21, TZ23]. Among them, the works [AB21, LK23] provide security proofs only in the algebraic group model, the works [DEF+19, CKM21] use proofs of possession for key generation, and all works support key aggregation (the first ones of two schemes in [PW23, PW24]). Recently, a non-interactive scheme has been proposed [RSY25], but it is only one-time secure. Notably, some of the above works [MPSW19, NRSW20, AB21, CKM21, NRS21] construct Schnorr multi-signatures.

Tight Security without the AGM. Several (pairing-free) multi-signatures have tight security proofs in the algebraic group model [AB21, BD21, NRS21, LK23][4], but we focus in this work on proofs without the algebraic group model. There are three-round schemes [BN06, BJ08, LBG09, FH20b] with tight security proofs. Notably, these constructions are based on the DDH assumption, while the works [BJ08, LBG09] also each provide a scheme based on the CDH assumption, and only [FH20b] supports key aggregation. However, the work [BJ08] relies on proofs of possession for key generation. There are also two-round schemes [PW23, PW24] with tight security proofs, both being based on the DDH assumption. However, none of these constructions support key aggregation. Further, in the pairing-based setting, there are non-interactive schemes [BNN07, QLH12, BW24] with tight security proofs (without the algebraic group model), with all of them being based on the CDH assumption.

Constructions from Other Assumptions. Recently, many lattice-based multi-signatures have been proposed [FH19, FH20a, DOTT21, LWZ+23], with some of them being two-round [DOTT21, BTT22, Che23, LLL+24]. Two recent works [TS23, AAB+24] analyze the non-interactive multi-signature obtained by aggregating Falcon signatures [PFH+20] using LaBRADOR [BS23]. Further, two other recent works [FSZ22, FHSZ23] propose non-interactive multi-signatures in the synchronized setting, where messages are signed with respect to time steps and only signatures for the same time step can be aggregated. There is also an extensive amount of works on pairing-based multi-signatures [Bol03, LOS+06, QX10, QLH12, DGNW20]. Notably, some of these works [Bol03, RY07, BDN18, BCG+23a, BW24] focus on BLS multi-signatures, and some [BCG+23b, RL24] have tight security proofs in the algebraic group model. There are RSA-based multi-signatures [IN83, OO93] and group action-based multi-signatures [DFMS24]. Finally, there is a recent work [DKKW25] on (synchronized) hash-based multi-signatures which follows the folklore approach of aggregating signatures using succinct arguments of knowledge [ACL+22, WW22, DGKV22].

2 Technical Overview

In this section, we present an informal overview of our work. We begin by reviewing the construction of two-round multi-signatures in previous works [TSS+23, PW23, PW24]. Then, we discuss why earlier tightly secure schemes fail to support key aggregation. Finally, we outline our ideas and highlight some technical subtleties when proving security.

2.1 Background: Multi-Signatures from Committed Lossy Identification

To eliminate the need for rewinding in their security proofs, Takemure et al. [TSS+23] and Pan and Wagner [PW23, PW24] leverage lossy identification [AFLT12] in conjunction with specially designed dual-mode commitment schemes. Understanding this concept of *committed lossy identification* is essential to grasp why Pan and Wagner's tightly secure constructions do not support key aggregation and how we can address this limitation.

[4]The works [BD21, NRS21] also provide (non-tight) security proofs without the AGM, which are given in Table 1.

Linear Lossy Identification. The first building block that we recall is linear lossy identification [CP93, AFLT12]. Specifically, let $\mathsf{F}\colon \mathcal{D} \to \mathcal{R}$ be a homomorphism over $\mathcal{S}$-vector spaces $\mathcal{D}$ and $\mathcal{R}$. Also, assume that random images $X = \mathsf{F}(x)$ are computationally indistinguishable[5] from random elements $X \xleftarrow{\$} \mathcal{R}$. Using this function, we define a natural identification protocol in which a prover holding x convinces a verifier holding $X = \mathsf{F}(x)$ of its knowledge of x: the prover sends $R = \mathsf{F}(r)$ for a random $r \xleftarrow{\$} \mathcal{D}$; the verifier sends a random challenge $c \xleftarrow{\$} \mathcal{S}$; the prover responds with $s = cx + r$; the verifier checks if $\mathsf{F}(s) = cX + R$. This identification scheme satisfies the following properties:

- *Completeness and Honest-Verifier Zero-Knowledge.* If indeed $X = \mathsf{F}(x)$, an honest prover always convinces the verifier. Moreover, if c were known before choosing R, one could simulate valid transcripts without knowledge of x.
- *Lossy Soundness.* If X is instead drawn uniformly from $\mathcal{R}$, then even an unbounded prover cannot convince the verifier. In this case, we call X a *lossy key* and note that, with overwhelming probability, $X \notin \mathsf{F}(\mathcal{D})$.

An Insecure Prototype. This identification scheme can be adapted into a (insecure!) multi-signature prototype: Each signer i has a public key $X_i = \mathsf{F}(x_i)$ and secret key x_i. To sign a message m, signer i samples r_i and computes $R_i = \mathsf{F}(r_i)$, and sends R_i to the other signers. The signers compute $R = \sum_i R_i$ and derive a challenge c via a random oracle applied to $(X_i)_i, R, \mathsf{m}$. Each signer computes its response $s_i = cx_i + r_i$, and they aggregate responses as $s = \sum_i s_i$. The signature is the transcript (R, c, s), which can be verified using the public keys leveraging the linearity F. This scheme is *insecure* because malicious signers can craft their first-round messages R_i based on those of honest signers. To prevent such attacks, one must either introduce an additional round or use the special commitment schemes discussed next.

Special Commitments. The key idea to transform the above prototype into a secure scheme without adding rounds is to commit to the first-round messages $R_i \in \mathcal{R}$ in a homomorphic manner [DOTT21, TSS+23, PW23, PW24], meaning that adding commitments corresponds to adding the first-round messages. To avoid rewinding, the commitment scheme should also support a *dual-mode* property [TSS+23, PW23, PW24], with two indistinguishable modes of setting up commitment keys ck:

- *Hiding Mode.* Commitments can be equivocated using a trapdoor.
- *Binding Mode.* It is (inefficiently) possible to extract R from the commitment.

Notably, in the concrete commitment schemes proposed in prior work [TSS+23, PW23, PW24], indistinguishability for Q keys[6] can be argued *tightly* from the DDH assumption.

Putting it together. We now describe a (non-tightly) secure version of the above prototype, following [TSS+23, PW23, PW24]. Interestingly, this scheme also supports key aggregation: given signers with public keys X_i, their aggregate public key is defined as $\tilde{X} = \sum_i a_i X_i$ for random coefficients $a_i \in \mathcal{S}$. Signing a message m involves the following steps:

1. All signers derive a commitment key $\mathsf{ck} = \mathsf{H}(\tilde{X}, \mathsf{m})$ via a random oracle H.
2. All signers compute a first round message $R_i = \mathsf{F}(r_i)$ as in the insecure prototype. However, instead of sending R_i directly, signer i commits to it using randomness φ_i as $\mathsf{com}_i = \mathsf{Com}(\mathsf{ck}, R_i; \varphi_i)$ and only sends com_i.
3. The signers aggregate the commitments to obtain com, which intuitively commits to $R = \sum_i R_i$. They then derive the challenge c via a random oracle applied to $\tilde{X}, \mathsf{com}, \mathsf{m}$.
4. Each signer computes its response as $s_i = a_i c x_i + r_i$ and sends both s_i and the opening φ_i to the other signers. Using the homomorphic properties of the commitment, they can all compute φ from the φ_i, which serves as a valid opening for (R, com).
5. The final signature consists of[7] (R, φ, c, s). Notably, (R, c, s) forms a valid transcript with respect to the aggregate public key $\tilde{X}$. Verification involves verifying this transcript and checking that c is derived correctly from $\tilde{X}, \mathsf{com}, \mathsf{m}$ for $\mathsf{com} = \mathsf{Com}(\mathsf{ck}, R; \varphi)$.

[5] For example, we often consider the homomorphism $x \mapsto (xG, xH)$ for group generators $G, H \in \mathbb{G}$, which satisfies this property under the DDH assumption.

[6] By that, we mean Q keys in hiding mode are indistinguishable from Q keys in binding mode.

[7] Of course, including R in the signature is not really necessary as it is determined by X, c and s.

Security Proof without Rewinding. The scheme described above supports key aggregation and, as we will now demonstrate, can be proven secure without relying on rewinding. However, it is important to note that the scheme is not tightly secure due to a guessing argument in the proof. Before delving into the proof, let us recall the security model: there is a single honest signer (say, signer $i = 1$), and the reduction must simulate this signer for the adversary. The adversary's goal is to produce a forgery, i.e., a fresh message along with a valid signature for it. We also summarize the following key observations from earlier works [TSS+23, PW23, PW24]:

- *Simulating the Signer.* The reduction has two options to simulate the honest signer: (1) it knows the secret key x_1, or (2) the commitment key ck is in hiding mode and the reduction knows the corresponding trapdoor. Especially, (2) works using equivocation in combination with honest-verifier zero-knowledge.
- *Statistically Hard Case.* If the aggregated public key $\tilde{X}$ is lossy *and* ck is in binding mode, then generating a valid signature becomes a statistically hard problem. This follows from combining extraction with lossy soundness.
- *Aggregation.* If X_1 (the honest signer's public key) is lossy, then $\tilde{X}$ is lossy, with overwhelming probability.

These observations suggest the following proof strategy: First, we guess the random oracle query to H that is used to define the commitment key ck for the forgery. We embed a binding key in the guessed query and hiding keys (with trapdoors) in all other queries. Due to the indistinguishability of the two modes (hiding and binding), the adversary cannot detect this setup. Using the trapdoors associated with the hiding keys, the reduction can simulate all signing queries without needing the honest signer's secret key x_1. Since the reduction no longer requires x_1, we can switch X_1 to a lossy key. By the aggregation property, this ensures that $\tilde{X}$ is also lossy. If our guess was correct, the forgery corresponds to the statistically hard case (where $\tilde{X}$ is lossy and ck is in binding mode). In this case, generating a valid signature is infeasible, completing the proof.

2.2 Tight Security and the Problem with Aggregation

Now that we understand the basic mechanics underlying multi-signatures from committed lossy identification, let us focus on the tightly secure constructions by Pan and Wagner [PW23, PW24]. These avoid the guessing argument sketched above, but at the cost of loosing the key aggregation feature.

Katz-Wang and Its Limitations. A well-known approach to avoid the guessing argument is the Katz-Wang pseudorandom bit technique [GJKW07]. Applying this technique to our setting, each pair $(\tilde{X}, \mathsf{m})$ would be associated with two commitment keys,

$$\mathsf{ck}_0 = \mathsf{H}(0, \tilde{X}, \mathsf{m}),\ \mathsf{ck}_1 = \mathsf{H}(1, \tilde{X}, \mathsf{m}).$$

During signing, signers select one of these keys, $\mathsf{ck}_{b_{\tilde{X},\mathsf{m}}}$, based on a bit $b_{\tilde{X},\mathsf{m}}$ pseudorandomly derived from $(\tilde{X}, \mathsf{m})$, where only the signer knows the pseudorandom function's key. This allows a tight security proof: the reduction can set $\mathsf{ck}_{b_{\tilde{X},\mathsf{m}}}$ to hiding mode and $\mathsf{ck}_{1-b_{\tilde{X},\mathsf{m}}}$ to binding mode. Consequently, the reduction can simulate the signer without the secret key using the trapdoor for $\mathsf{ck}_{b_{\tilde{X},\mathsf{m}}}$. As it no longer needs x_1, the reduction can switch X_1 to lossy. Since $b_{\tilde{X},\mathsf{m}}$ remains pseudorandom for the forgery, the adversary forges with respect to the binding key $\mathsf{ck}_{1-b_{\tilde{X},\mathsf{m}}}$ with probability $1/2$. This corresponds to the statistically hard case.

Unfortunately, as Pan and Wagner observe, this approach fails in the multi-signature setting. If all signers independently pseudorandomly select their commitment keys, they will end up with different keys. This breaks the homomorphic properties of the commitment scheme, making the scheme incorrect.

The Two-Key Solution. Pan and Wagner conclude that correctness requires all signers to use the *same* commitment key. To make this work and achieve a tight proof, Pan and Wagner propose a two-key approach. Each signer generates two public keys, $X_{i,0}$ and $X_{i,1}$, and holds the corresponding secret keys, $x_{i,0}$ and $x_{i,1}$. During signing, each signer i pseudorandomly selects one secret key, x_{i,b_i}, based on a bit b_i, while all signers use the same commitment key $\mathsf{ck} = \mathsf{H}(\tilde{X}, \mathsf{m})$. This approach enables a tight reduction and ensures correctness. The reduction randomly selects one public key to make lossy, retaining the

secret key for the other. The commitment key is made hiding if and only if the lossy key is used. This allows the signer to be simulated using either the secret key of the non-lossy key or the trapdoor of the hiding commitment key. For the forgery, they can argue that with probability $1/4$, the lossy public key is used and the commitment key is binding.

The Problem with Key Aggregation. While the two-key approach yields tight security, it is fundamentally incompatible with key aggregation. For N signers, there are 2^N possible combinations of public keys, one of which is selected pseudorandomly during signing. To verify the signature, the verifier must know which combination was used. In Pan and Wagner's scheme, each signer includes their pseudorandom choice bit b_i in the signature, resulting in a signature size of $\Omega(\lambda + N)$ bits. These bits b_i allow the verifier to reconstruct the specific combination of public keys during verification. However, this precludes a single short aggregate public key, making the scheme unsuitable for key aggregation.

2.3 Overview of Our Solution

Having an understanding of committed lossy identification and the limitations of existing tightly secure schemes in supporting key aggregation, we now present our solution. Here, we provide an intuitive overview, whereas the challenges involved in the security proof and how we address them are deferred to Section 2.4.

Signer Partitioning. To achieve a single aggregate public key, each signer should hold only one public key X_i. However, for tight security, relying on a single commitment key in addition to a single public key fails. Revisiting the naive application of the Katz-Wang technique sketched in the beginning of Section 2.2, which did not yield a correct scheme, our first insight is: a correct and tightly secure scheme can be obtained, even if signers use *different* commitment keys. Signers do not need to use the same commitment key!

Namely, our approach is as follows: in the first round, each signer i pseudorandomly decides which commitment key ck_{b_i} to use. Subsequently, signers exchange their commitments and choice bits b_i. Next, we *partition* the N signers into two sets:

$$\mathcal{I}_0 := \{i \in [N] \mid b_i = 0\} \quad \text{and} \quad \mathcal{I}_1 := \{i \in [N] \mid b_i = 1\}.$$

Crucially, we aggregate commitments only *within* each set, resulting in two aggregated commitments, $\bar{\mathsf{com}}_0$ and $\bar{\mathsf{com}}_1$, which are then used to derive the challenge c. Based on c, each signer computes their response s_i as before, and these responses are aggregated only within the subsets:

$$\bar{s}_0 := \sum_{i \in \mathcal{I}_0} s_i, \quad \bar{s}_1 := \sum_{i \in \mathcal{I}_1} s_i.$$

Similarly, the openings φ_i are aggregated into openings $\bar{\varphi}_0$ and $\bar{\varphi}_1$. The final signature includes the choice bits b_i, the challenge c, and the two sub-signatures $\sigma_0 = (\bar{s}_0, \bar{\varphi}_0)$ and $\sigma_1 = (\bar{s}_1, \bar{\varphi}_1)$. Verification is performed by checking (σ_b, c) against the *sub-aggregate key*

$$\tilde{X}_b := \sum_{i \in \mathcal{I}_b} a_i \cdot X_i \text{ for both } b \in \{0, 1\}.$$

Adding Key Aggregation. At first glance, partitioning the set of signers into two subsets may seem contradictory, especially since the goal is to use a *single* aggregate key. Indeed, up to this point, there is no apparent benefit: the signature still includes the choice bits b_i, and the verifier needs to reconstruct the partition and compute the sub-aggregate keys during verification. Thus, verification still uses the individual public keys X_i. This is where our second idea comes into play: instead of including the choice bits b_i, we include the sub-aggregate keys[8] $\tilde{X}_0$ and $\tilde{X}_1$ in the signature. The verifier can then use these sub-aggregate keys to verify the two sub-signatures. Additionally, the verifier must check that they satisfy the relation:

$$\tilde{X}_0 + \tilde{X}_1 = \tilde{X}. \tag{1}$$

This can be done only using the aggregate key $\tilde{X}$.

[8] It suffices to include only one sub-aggregate key, as the other can be derived from the aggregate public key.

Security Concerns. Naturally, one may wonder whether this scheme remains secure. After all, an adversary attempting to forge a signature could *maliciously* select the sub-aggregate keys $\tilde{X}_0$ and $\tilde{X}_1$, without ever involving the honest signer's public key X_1. Since the verifier no longer explicitly uses X_1, it is not immediately clear whether this introduces any potential attack vectors. The only guarantee we have in this case is Equation (1).

Security Intuition. As we will see in Section 2.4, not having the choice bits explicitly introduces several novel challenges in the security proof. However, despite these challenges and concerns, there is reason to believe that the scheme remains secure. Specifically, one can argue that the aggregate key $\tilde{X}$ is lossy if X_1 is lossy. Furthermore, due to Equation (1), it can be argued that at least one of the maliciously chosen sub-aggregate keys, $\tilde{X}_0$ or $\tilde{X}_1$, must also be lossy. This is because non-lossy keys form a linear subspace. If this lossy sub-aggregate key is matched with the binding commitment key, security should hold. A key challenge in the proof will be to argue that this happens with sufficiently high probability.

2.4 Proving Security

In this final part of our technical overview, we outline the subtleties that arise when proving the security of our construction.

The Common Approach. We begin by recalling the approach commonly employed in prior works to prove the security of constructions based on pseudorandom bits in the style of Katz-Wang, e.g., [PW23, PW24]. The proof proceeds via a sequence of game hops, starting from the real security experiment. The goal is to transition to a final game in which forging signatures becomes a statistically hard problem. In the context of committed lossy identification, this final game ensures that a lossy public key is paired with a binding commitment key in the forgery. Conceptually, the common approach involves introducing the following changes in order: (1) *branch decision*, (2) *commitment key programming*, and (3) *lossy key*.

Step (1) (*branch decision*) is construction-specific and ensures that the forgery will be on the correct "branch". For instance, we might switch to a game where the adversary only succeeds if the forgery matches $\mathsf{ck}_{1-b_{\tilde{X},\mathsf{m}}}$ with the honest user's public key. Since $b_{\tilde{X},\mathsf{m}}$ is pseudorandom, the pair $(\tilde{X},\mathsf{m})$ in the forgery is fresh, and all commitment keys have the same distribution, we can argue that the success probability is changed only by a factor of $1/2$.

Step (2) (*commitment key programming*) involves changing every $\mathsf{ck}_{b_{\tilde{X},\mathsf{m}}}$ to hiding and every $\mathsf{ck}_{1-b_{\tilde{X},\mathsf{m}}}$ to binding. As the two modes are indistinguishable, this change remains unnoticed by the adversary.

Step (3) (*lossy key*) switches the honest signer's key X_1 to lossy. Crucially, this step relies on having done Step (2) before: the game can now be efficiently simulated *without* the secret key x_1. For constructions involving key aggregation, we can further argue that the aggregate key $\tilde{X}$ is lossy as well.

If the bits b_i were explicitly included in the signature, this approach would directly work for our scheme. However, recall that to support key aggregation, we removed the b_i bits from the signature and replaced them with the sub-aggregate keys $\tilde{X}_0$ or $\tilde{X}_1$. To adapt the proof template for this scheme, we define the *branch decision* as follows: we are on the correct branch if $\tilde{X}_{1-b_{\tilde{X},\mathsf{m}}}$ (which is matched with the binding commitment key) is lossy. Unfortunately, this introduces a problem: our branch decision is only well-defined if X_1 is already lossy, allowing us to argue that one of the $\tilde{X}_b$ keys is lossy. Consequently, Step (3) would have to precede Step (1). However, as previously noted, Step (2) must always be performed before Step (3). Thus, Step (2) (*commitment key programming*) must also precede Step (1). The problem is that once we program the commitment keys in Step (2), the value of $b_{\tilde{X},\mathsf{m}}$ becomes statistically leaked through the distribution of the keys. As a result, we can no longer argue that we are on the correct branch with probability $1/2$.

Delayed Branch Decision. To address this issue, our idea is to *delay* the branch decision until after X_1 has been switched to a lossy key. However, as outlined above, we must ensure that all commitment keys have identical distributions when introducing the branch decision. To achieve this, we deviate from the common proof structure: we first set *all* commitment keys to the hiding mode. Next, we switch X_1 to lossy, and only then perform the branch decision step. Crucially, since all commitment keys have the same distribution when we do the branch decision, the bit $b_{\tilde{X},\mathsf{m}}$ is not leaked during this process.

Efficient Branch Decision. Building on this idea, we now structure our proof as follows:

1. We make all commitment keys hiding. This allows us to efficiently simulate the honest signer without requiring knowledge of x_1.

2. We switch X_1 to a lossy key and argue that the aggregate key $\tilde{X}$ is lossy as well. Consequently, for the forgery containing sub-aggregate keys $\tilde{X}_0$ or $\tilde{X}_1$, we know that at least one of them is also lossy.

3. We perform the *branch decision*: the adversary now succeeds only if $\tilde{X}_{1-b_{\tilde{X},\mathsf{m}}}$ (in the forgery) is lossy. Since all commitment keys have identical distribution (all are hiding), $b_{\tilde{X},\mathsf{m}}$ remains hidden. Thus, the success probability is reduced by at most a factor of $1/2$.

4. We switch every $\mathsf{ck}_{1-b_{\tilde{X},\mathsf{m}}}$ to the binding mode. Since $\mathsf{ck}_{b_{\tilde{X},\mathsf{m}}}$ is still in hiding mode, simulating signatures for the honest signer is done via equivocation of this commitment.

5. Finally, we observe that in the forgery, $\tilde{X}_{1-b_{\tilde{X},\mathsf{m}}}$ is lossy and paired with the binding commitment key $\mathsf{ck}_{1-b_{\tilde{X},\mathsf{m}}}$.

It is important to note that when switching all $\mathsf{ck}_{1-b_{\tilde{X},\mathsf{m}}}$ to binding mode, we rely on commitment key indistinguishability via an efficient reduction. However, this introduces a subtle challenge: the reduction must *efficiently* verify the branch decision to determine whether the game outputs 1. Specifically, it needs to efficiently check whether $\tilde{X}_{1-b_{\tilde{X},\mathsf{m}}}$ is lossy. This is problematic, as distinguishing lossy keys from normal keys should be computationally hard.

Solution. To understand our solution to this final problem, note that the linear function under consideration maps a field element x to a pair of group elements (xG, xH) for two generators G and H. If the discrete logarithm δ is known such that $H = \delta G$, then it becomes possible to efficiently distinguish lossy keys from normal ones. The key question, therefore, is whether we are allowed to know δ at the point where the commitment keys are switched to binding mode. For the most efficient commitment schemes proposed in [TSS+23, PW24], commitment key indistinguishability holds only when δ remains unknown. Fortunately, we prove that for the commitment scheme from [PW23], commitment key indistinguishability holds even if δ is known. With this property, we can then finish our proof.

3 Preliminaries

We denote the security parameter by $\lambda \in \mathbb{N}$ and assume that all algorithms get 1^λ implicitly as input. We use standard cryptographic notions such as negligible and probabilistic polynomial-time (PPT). For a given finite set S, we assume a canonical ordering (e.g., lexicographically) and write $\langle S \rangle$ to denote a unique encoding of this set. We write $x \xleftarrow{\$} S$ to denote that x is sampled uniformly at random from S, and we write $x \leftarrow \mathcal{D}$ to denote that x is sampled according to a distribution $\mathcal{D}$. For a probabilistic algorithm $\mathcal{A}$, we write $y := \mathcal{A}(x; \rho)$ to denote that $\mathcal{A}$ outputs y on input x with random coins ρ, and we write $y \leftarrow \mathcal{A}(x)$ when ρ is sampled uniformly at random. Further, we write $y \in \mathcal{A}(x)$ to denote that there are coins ρ such that $\mathcal{A}$ outputs y on input x with these coins ρ. If $\mathcal{A}$ is deterministic, we instead write $y := \mathcal{A}(x)$. We write $\mathbf{T}(\mathcal{A})$ to denote the running time of $\mathcal{A}$. All our algorithms are probabilistic, unless stated otherwise. For a positive integer $L \in \mathbb{N}$, we define $[L] := \{1, \ldots, L\}$ and $[\![L]\!] := \{0, \ldots, L\}$. Finally, for a security game $\mathsf{Game}_{\mathcal{A}}$ parameterized by an adversary $\mathcal{A}$, we denote the advantage of $\mathcal{A}$ in $\mathsf{Game}_{\mathcal{A}}$ as usual by $\mathsf{Adv}^{\mathsf{Game}}_{\mathcal{A}}(\lambda) := \Pr[\mathsf{Game}_{\mathcal{A}}(\lambda) \Rightarrow 1]$. Numerical variables in games (such as counters) are implicitly initialized to 0, and sets, maps, and lists as empty.

3.1 Pairing-Free Groups and Assumptions

Our construction will work over a cyclic group $\mathbb{G}$ of prime order p with a generator $G \in \mathbb{G}$, written additively. We let GGen be an algorithm that takes as input 1^λ and outputs $\mathfrak{G} := (\mathbb{G}, p, G)$, and refer to GGen as a *group generation algorithm*.

Computational Assumptions. We will base security of our construction on the DDH assumption. For convenience, however, we will also make use of a slightly modified version of it, namely the uDDH3 assumption. Importantly, it has been shown [EHK+13, PW23] that it is tightly equivalent to DDH. We formally define both next.

Definition 1 (DDH Assumption)**.** Let GGen be a group generation algorithm. We say that the DDH assumption holds relative to GGen, if for all PPT algorithms $\mathcal{A}$, the following advantage is negligible:

$$\mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{A},\mathsf{GGen}}(\lambda) := \Big| \Pr\left[\mathcal{A}(\mathfrak{G}, H, aG, aH) = 1 \;\middle|\; \mathfrak{G} \leftarrow \mathsf{GGen}(1^\lambda),\ H \xleftarrow{\$} \mathbb{G},\ a \xleftarrow{\$} \mathbb{Z}_p\right] - \Pr\left[\mathcal{A}(\mathfrak{G}, H, aG, bH) = 1 \;\middle|\; \mathfrak{G} \leftarrow \mathsf{GGen}(1^\lambda),\ H \xleftarrow{\$} \mathbb{G},\ a, b \xleftarrow{\$} \mathbb{Z}_p\right] \Big|.$$

Definition 2 (uDDH3 Assumption)**.** Let GGen be a group generation algorithm. We say that the uDDH3 assumption holds relative to GGen, if for all PPT algorithms $\mathcal{A}$, the following advantage is negligible:

$$\mathsf{Adv}^{\mathsf{uDDH3}}_{\mathcal{A},\mathsf{GGen}}(\lambda) := \Big| \Pr\left[\mathcal{A}(\mathfrak{G}, (H_{i,j})_{i,j\in[3]}) = 1 \;\middle|\; \begin{array}{l} \mathfrak{G} \leftarrow \mathsf{GGen}(1^\lambda), \\ a, b \xleftarrow{\$} \mathbb{Z}_p,\ H_{1,1}, H_{2,1}, H_{3,1} \xleftarrow{\$} \mathbb{G} \\ H_{1,2} := aH_{1,1}, H_{1,3} := bH_{1,1} \\ H_{2,2} := aH_{2,1}, H_{2,3} := bH_{2,1} \\ H_{3,2} := aH_{3,1}, H_{3,3} := bH_{3,1} \end{array}\right] - \Pr\left[\mathcal{A}(\mathfrak{G}, (H_{i,j})_{i,j\in[3]}) = 1 \;\middle|\; \begin{array}{l} \mathfrak{G} \leftarrow \mathsf{GGen}(1^\lambda), \\ \forall (i,j) \in [3]\times[3] : H_{i,j} \xleftarrow{\$} \mathbb{G} \end{array}\right] \Big|.$$

3.2 Multi-Signatures

We use the definition of multi-signatures and their security from [PW23, PW24]. For this, we assume that the public keys of signers are given by a set $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ with a unique encoding denoted as $\langle\mathcal{P}\rangle$. For security, we assume that there is a single honest signer with public key pk_1 and that the adversary controls all other signers. Further, it can choose public keys of corrupt signers arbitrarily and even as a function of the honest signer's public key pk_1. These are standard assumptions for multi-signatures in the plain public key model [BN06, CKM21, DOTT21].

Alg $\mathsf{MS.Exec}(\mathcal{P}, \mathcal{S}, \mathsf{m})$
01 **parse** $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\} := \mathcal{P},\ \{\mathsf{sk}_1, \ldots, \mathsf{sk}_N\} := \mathcal{S}$
02 **for** $i \in [N] : (\mathsf{pm}_{1,i}, St_{1,i}) \leftarrow \mathsf{Sig}_0(\mathcal{P}, \mathsf{sk}, \mathsf{m})$
03 $\mathcal{M}_1 := (\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N})$
04 **for** $i \in [N] : (\mathsf{pm}_{2,i}, St_{2,i}) \leftarrow \mathsf{Sig}_1(St_{1,i}, \mathcal{M}_1)$
05 $\mathcal{M}_2 := (\mathsf{pm}_{2,1}, \ldots, \mathsf{pm}_{2,N})$
06 **for** $i \in [N] : \sigma_i \leftarrow \mathsf{Sig}_2(St_{2,i}, \mathcal{M}_2)$
07 **if** $\exists i \neq j \in [N]$ s.t. $\sigma_i \neq \sigma_j$: **return** $\perp$
08 **return** $\sigma := \sigma_1$

Figure 1: Algorithm $\mathsf{MS.Exec}$ for a multi-signature scheme $\mathsf{MS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$. The algorithm specifies an honest execution of the signing protocol Sig among N signers with public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_N$ and secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$ for a message m.

Definition 3 (Multi-Signature Scheme)**.** A (2-round) multi-signature scheme is a tuple of PPT algorithms $\mathsf{MS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{par}$ takes as input the security parameter 1^λ and outputs global system parameters par. We assume that par implicitly defines sets of public keys, secret keys, messages and signatures, respectively. All algorithms related to MS take par at least implicitly as input.
- $\mathsf{Gen}(\mathsf{par}) \to (\mathsf{pk}, \mathsf{sk})$ takes as input system parameters par, and outputs a public key pk and a secret key sk.
- $\mathsf{Sig} = (\mathsf{Sig}_0, \mathsf{Sig}_1, \mathsf{Sig}_2)$ is split into three algorithms:
  - $\mathsf{Sig}_0(\mathcal{P}, \mathsf{sk}, \mathsf{m}) \to (\mathsf{pm}_1, St_1)$ takes as input a set $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ of public keys, a secret key sk, and a message m, and outputs a protocol message pm_1 and a state St_1.

- $\mathsf{Sig}_1(St_1, \mathcal{M}_1) \to (\mathsf{pm}_2, St_2)$ takes as input a state St_1 and a tuple $\mathcal{M}_1 = (\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N})$ of protocol messages, and outputs a protocol message pm_2 and a state St_2.
- $\mathsf{Sig}_2(St_2, \mathcal{M}_2) \to \sigma_i$ takes as input a state St_2 and a tuple $\mathcal{M}_2 = (\mathsf{pm}_{2,1}, \ldots, \mathsf{pm}_{2,N})$ of protocol messages, and outputs a signature σ.

- $\mathsf{Ver}(\mathcal{P}, \mathsf{m}, \sigma) \to b$ is deterministic, takes as input a set $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ of public keys, a message m, and a signature σ, and outputs a bit $b \in \{0, 1\}$.

We require that MS is complete in the following sense. For all $\mathsf{par} \in \mathsf{Setup}(1^\lambda)$, all $N = \mathsf{poly}(\lambda)$, all $(\mathsf{pk}_i, \mathsf{sk}_i) \in \mathsf{Gen}(\mathsf{par})$ for $i \in [N]$, and all messages m, we have

$$\Pr\left[\mathsf{Ver}(\mathcal{P}, \mathsf{m}, \sigma) = 1 \;\middle|\; \begin{array}{l} \mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}, \mathcal{S} = \{\mathsf{sk}_1, \ldots, \mathsf{sk}_N\}, \\ \sigma \leftarrow \mathsf{MS.Exec}(\mathcal{P}, \mathcal{S}, \mathsf{m}) \end{array}\right] = 1,$$

where algorithm MS.Exec is defined in Figure 1.

Definition 4 (MS-EUF-CMA Security)**.** Let $\mathsf{MS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ be a multi-signature scheme and consider the game **MS-EUF-CMA** defined in Figure 2. We say that MS is MS-EUF-CMA secure, if for all PPT adversaries $\mathcal{A}$, the following advantage is negligible:

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

Game MS-EUF-CMA$^{\mathcal{A}}_{\mathsf{MS}}(\lambda)$
01 $\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda)$
02 $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{par})$
03 $\mathsf{SIG} := (\mathsf{SIG}_0, \mathsf{SIG}_1, \mathsf{SIG}_2)$
04 $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{SIG}}(\mathsf{par}, \mathsf{pk})$
05 **if** $\mathsf{pk} \notin \mathcal{P}^*$: **return** 0
06 **if** $(\mathcal{P}^*, \mathsf{m}^*) \in \mathsf{Queried}$: **return** 0
07 **return** $\mathsf{Ver}(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$

Oracle $\mathrm{SIG}_0(\mathcal{P}, \mathsf{m})$
08 **parse** $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\} := \mathcal{P}$
09 **if** $\mathsf{pk}_1 \neq \mathsf{pk}$: **return** $\perp$
10 $\mathsf{Queried} := \mathsf{Queried} \cup \{(\mathcal{P}, \mathsf{m})\}$
11 $ctr := ctr + 1,\ sid := ctr$
12 $\mathsf{round}[sid] := 1$
13 $(\mathsf{pm}_1, St_1) \leftarrow \mathsf{Sig}_0(\mathcal{P}, \mathsf{sk}, \mathsf{m})$
14 $(\mathsf{pm}_1[sid], St_1[sid]) := (\mathsf{pm}_1, St_1)$
15 **return** $(\mathsf{pm}_1[sid], sid)$

Oracle $\mathrm{SIG}_1(sid, \mathcal{M}_1)$
16 **if** $\mathsf{round}[sid] \neq 1$: **return** $\perp$
17 **parse** $(\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N}) := \mathcal{M}_1$
18 **if** $\mathsf{pm}_1[sid] \neq \mathsf{pm}_{1,1}$: **return** $\perp$
19 $\mathsf{round}[sid] := \mathsf{round}[sid] + 1$
20 $(\mathsf{pm}_2, St_2) \leftarrow \mathsf{Sig}_1(St_1[sid], \mathcal{M}_1)$
21 $(\mathsf{pm}_2[sid], St_2[sid]) := (\mathsf{pm}_2, St_2)$
22 **return** $\mathsf{pm}_2[sid]$

Oracle $\mathrm{SIG}_2(sid, \mathcal{M}_2)$
23 **if** $\mathsf{round}[sid] \neq 2$: **return** $\perp$
24 **parse** $(\mathsf{pm}_{2,1}, \ldots, \mathsf{pm}_{2,N}) := \mathcal{M}_2$
25 **if** $\mathsf{pm}_2[sid] \neq \mathsf{pm}_{2,1}$: **return** $\perp$
26 $\mathsf{round}[sid] := \mathsf{round}[sid] + 1$
27 $\sigma \leftarrow \mathsf{Sig}_2(St_2[sid], \mathcal{M}_2)$
28 **return** σ

Figure 2: The game **MS-EUF-CMA** for a (two-round) multi-signature scheme MS and an adversary $\mathcal{A}$. To simplify the presentation, we assume that the canonical ordering of sets is chosen such that pk is always at the first position if it is included.

Definition 5 (Key Aggregation)**.** A multi-signature scheme $\mathsf{MS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ is said to support key aggregation, if the algorithm Ver can be split into two deterministic polynomial time algorithms Agg, VerAgg with the following syntax:

- $\mathsf{Agg}(\mathcal{P}) \to \tilde{\mathsf{pk}}$ takes as input a set $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ of public keys and outputs an aggregated key $\tilde{\mathsf{pk}}$.
- $\mathsf{VerAgg}(\tilde{\mathsf{pk}}, \mathsf{m}, \sigma) \to b$ is deterministic, takes as input an aggregated key $\tilde{\mathsf{pk}}$, a message m, and a signature σ, and outputs a bit $b \in \{0, 1\}$.

That is, algorithm $\mathsf{Ver}(\mathcal{P}, \mathsf{m}, \sigma)$ can be written as $\mathsf{VerAgg}(\mathsf{Agg}(\mathcal{P}), \mathsf{m}, \sigma)$.

4 Our Construction

Here, we present our multi-signature construction. Conceptually, it relies on a linear function in combination with a special commitment scheme. We introduce these building blocks first, following the definitions of [PW23], but with an important change in the definition of the commitment scheme. Then, we present our two-round multi-signature construction, which combines these two building blocks in a novel way. In this way, we obtain the first two-round scheme over pairing-free groups that achieves tightness and key aggregation at the same time.

4.1 Building Block: Linear Function Families

Following prior works [PW23, PW24], we phrase our construction abstractly building on a linear function family. We recall the definition of the syntax of linear function families from [PW23, PW24].

Definition 6 (Linear Function Family)**.** A linear function family (LFF) is a tuple of PPT algorithms $\mathsf{LF} = (\mathsf{LF.Gen}, \mathsf{F})$ with the following syntax:

- $\mathsf{LF.Gen}(1^\lambda) \to \mathsf{par}$ takes as input the security parameter 1^λ and outputs parameters par. We assume that par implicitly defines the following sets:
  - A set of scalars $\mathcal{S}_{\mathsf{par}}$, which forms a field.
  - A domain $\mathcal{D}_{\mathsf{par}}$, which forms a vector space over $\mathcal{S}_{\mathsf{par}}$.
  - A range $\mathcal{R}_{\mathsf{par}}$, which forms a vector space over $\mathcal{S}_{\mathsf{par}}$.

  We omit the subscript par if it is clear from the context, and naturally denote the operations of these fields and vector spaces by $+$ and $\cdot$. We assume that these operations can be evaluated efficiently.
- $\mathsf{F}(\mathsf{par}, x) \to X$ is deterministic, takes as input parameters par, an element $x \in \mathcal{D}$, and outputs an element $X \in \mathcal{R}$. For all parameters par, $\mathsf{F}(\mathsf{par}, \cdot)$ realizes a homomorphism, i.e.

$$\forall s \in \mathcal{S}, x, y \in \mathcal{D} : \ \mathsf{F}(\mathsf{par}, s \cdot x + y) = s \cdot \mathsf{F}(\mathsf{par}, x) + \mathsf{F}(\mathsf{par}, y).$$

  We omit the input par if it is clear from the context.

Note that such a linear function family induces a linear identification scheme. As this identification scheme is implicitly a part of our signing protocol, it is instructive to have an intuitive understanding of it. Informally, we consider a prover holding a random secret key $x \in \mathcal{D}$ and a verifier holding the corresponding public key $X = \mathsf{F}(x)$. The prover and the verifier engage in the following natural protocol:

1. The prover samples $r \xleftarrow{\$} \mathcal{D}$ and sets $R = \mathsf{F}(r)$ to the verifier.
2. The verifier samples a random challenge $c \xleftarrow{\$} \mathcal{S}$ and sends it to the prover.
3. The prover responds with $s := cx + r$ and the verifier accepts if and only if $\mathsf{F}(s) - c \cdot X = R$.

Intuitively, soundness and zero-knowledge of this identification scheme are crucial for the security of the multi-signature construction. Focusing on soundness, a technique used in earlier works [AFLT12, PW23, PW24] is to first switch the public key to a uniform element in the range, which is sometimes called a *lossy key*. For that, we require that key indistinguishability holds. We recall the definition from [PW23, PW24].

Definition 7 (Key Indistinguishability)**.** Let $\mathsf{LF} = (\mathsf{LF.Gen}, \mathsf{F})$ be a linear function family. We say that LF satisfies key indistinguishability, if for any PPT algorithm $\mathcal{A}$, the following advantage is negligible:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{keydist}}_{\mathcal{A},\mathsf{LF}}(\lambda) := |\Pr\left[\mathcal{A}(\mathsf{par}, X) = 1 \middle|\ \mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda),\ x \xleftarrow{\$} \mathcal{D},\ X := \mathsf{F}(x)\ \right] \\ - \Pr\left[\mathcal{A}(\mathsf{par}, X) = 1 \middle|\ \mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda),\ X \xleftarrow{\$} \mathcal{R}\ \right]|.\end{aligned}$$

Having switched the key to lossy (i.e., uniform in the range) one first argues that with high probability it is not in the image. Then, one argues that even an unbounded prover cannot make the verifier accept. For that, earlier works [PW23, PW24] have defined *lossy soundness* for linear function families. Then, they have shown that lossy soundness holds for their specific linear function families. We instead noticed that lossy soundness holds generically, and prove this in the following lemma. A similar observation (using a different abstraction) has been made in [BLT+24].

Lemma 1 (Lossy Soundness)**.** *Let* $\mathsf{LF} = (\mathsf{LF.Gen}, \mathsf{F})$ *be a linear function family. For any (potentially unbounded) algorithm* $\mathcal{A}$*, we have*

$$\Pr\left[X \notin \mathsf{F}(\mathcal{D}) \wedge \mathsf{F}(s) - c \cdot X = R \;\middle|\; \begin{array}{l} \mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda), \\ (St, X, R) \leftarrow \mathcal{A}(\mathsf{par}), \\ c \xleftarrow{\$} \mathcal{S},\ s \leftarrow \mathcal{A}(St, c) \end{array}\right] \leq \frac{1}{|\mathcal{S}|}.$$

Proof. First of all, we condition on fixed par, a fixed[9] $X \notin \mathsf{F}(\mathcal{D})$, and fixed St, R. We argue that the probability, taken over c, that there *exists* a suitable s with $\mathsf{F}(s) - c \cdot X = R$ is at most $1/|\mathcal{S}|$. To see this, assume there are two distinct $c \neq c' \in \mathcal{S}$ for which there exists such an s, s', respectively. Then, we have

$$\mathsf{F}(s) - c \cdot X = R = \mathsf{F}(s') - c' \cdot X.$$

Rearranging, this implies

$$X = \frac{1}{c - c'}\left(\mathsf{F}(s) - \mathsf{F}(s')\right) = \mathsf{F}\left(\frac{s - s'}{c - c'}\right),$$

which contradicts $X \notin \mathsf{F}(\mathcal{D})$. □

We will not apply lossy soundness to the key of the honest signer, but instead to (a sub-key of) the aggregated public key. To do so, we will need the following lemma, which intuitively shows that the aggregate key will not be in the image of the function, given that the key of the honest signer is sampled uniformly. This property was also implicit in the proofs for instantiations in [PW23, PW24], but we again observe that it holds generically.

Lemma 2 (Aggregation Preserves Lossiness)**.** *Let* $\mathsf{LF} = (\mathsf{LF.Gen}, \mathsf{F})$ *be a linear function family. For any (potentially unbounded) algorithm* $\mathcal{A}$*, we have*

$$\Pr\left[\sum_{i=1}^{N} a_i X_i \in \mathsf{F}(\mathcal{D}) \;\middle|\; \begin{array}{c} \mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda),\ X_1 \xleftarrow{\$} \mathcal{R}, \\ (X_i, a_i)_{i=2}^{N} \leftarrow \mathcal{A}(\mathsf{par}, X_1),\ a_1 \xleftarrow{\$} \mathcal{S} \end{array}\right] \leq \frac{|\mathsf{F}(\mathcal{D})|}{|\mathcal{R}|} + \frac{1}{|\mathcal{S}|}.$$

Proof. First of all, if F is surjective, i.e., if $\mathsf{F}(\mathcal{D}) = \mathcal{R}$, the claim holds trivially, as the right hand side is at least 1. So, consider the case in which F is not surjective. With probability at most $|\mathsf{F}(\mathcal{D})|/|\mathcal{R}|$, we have $X_1 \in \mathsf{F}(\mathcal{D})$. So, condition on $X_1 \notin \mathsf{F}(\mathcal{D})$. Note that $\mathsf{F}(\mathcal{D})$ is a proper subspace of $\mathcal{R}$ and therefore there exists a subspace $\mathcal{L}$ such that $\mathcal{R} = \mathsf{F}(\mathcal{D}) \oplus \mathcal{L}$. Fix such an $\mathcal{L}$ and let $L_1, \ldots, L_k \in \mathcal{L}$ be any basis of $\mathcal{L}$. Let $U_1, \ldots, U_t$ be any basis of $\mathsf{F}(\mathcal{D})$. We can thus write each X_i in a unique way as $X_i = \sum_j \alpha_{i,j} U_j + \sum_j \beta_{i,j} L_j$, with coefficients $\alpha_{i,j}, \beta_{i,j} \in \mathcal{S}$. Now, because $X_1 \notin \mathsf{F}(\mathcal{D})$, we know that there exists some $j^* \in [k]$ such that $\beta_{1,j^*} \neq 0$. With this, we get

$$\Pr\left[\sum_{i=1}^{N} a_i X_i \in \mathsf{F}(\mathcal{D})\right] \leq \Pr\left[\sum_{i=1}^{N} a_i \beta_{i,j^*} = 0\right] \leq \Pr\left[a_1 = -\frac{1}{\beta_{1,j^*}} \sum_{i=2}^{N} a_i \beta_{i,j^*}\right],$$

where we have used $\beta_{1,j^*} \neq 0$. The probability of this event is $1/|\mathcal{S}|$, as a_1 is sampled independently of the left hand side. □

Remark 1. Note that for earlier works [PW23, PW24], lossy soundness was analyzed with respect to either (1) a lossy public key that was uniformly sampled from the range, or (2) an honest aggregation of public keys where one of the public keys was lossy and uniform. In the context of our construction, however, this will not be enough. Namely, the key we will be considering is chosen by the adversary as part of the forgery signature, and all that we know is that it is one of two keys which sum to the aggregate key. In this way, the only thing we can guarantee is that it is not in the range of the function. This is why we need a slightly stronger form of lossy soundness in which the adversary outputs the key.

Finally, we show a lemma that intuitively says that an aggregated public key represents the set of signers in a unique way. The only assumption that we make is that there is one random public key and its aggregation coefficient is sampled honestly.

[9] Clearly, if $X \in \mathsf{F}(\mathcal{D})$, we are done.

Lemma 3 (Collision Resistance of Aggregation)**.** *Let* $\mathsf{LF} = (\mathsf{LF.Gen}, \mathsf{F})$ *be a linear function family. For any (potentially unbounded) algorithm* $\mathcal{A}$*, we have*

$$\Pr\left[\sum_{i=1}^{N} a_i X_i = \sum_{i=1}^{N'} a'_i X'_i \;\middle|\; \begin{array}{c} \mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda),\ X_1 \xleftarrow{\$} \mathcal{R} \setminus \{0\}, \\ \left(a_1, (X_i, a_i)_{i=2}^{N}, (X'_i, a'_i)_{i=2}^{N'}\right) \leftarrow \mathcal{A}(\mathsf{par}, X_1), \\ a'_1 \xleftarrow{\$} \mathcal{S},\ X'_1 := X_1 \end{array}\right] \leq \frac{1}{|\mathcal{S}|}.$$

Proof. We condition on a fixed $X_1 \neq 0$. For this fixed X_1, note that there exists a homomorphism $\vartheta\colon \mathcal{R} \to \mathcal{S}$ such that $\vartheta(X_1) \neq 0$. Concretely, we can represent X_1 in any fixed basis of the vector space $\mathcal{R}$ and pick the projection to the non-zero coordinate of X_1 for ϑ. With this, condition on fixed a_1, $(X_i, a_i)_{i=2}^{N}$, and $(X'_i, a'_i)_{i=2}^{N'}$ and consider the experiment of sampling $a'_1 \xleftarrow{\$} \mathcal{S}$. Then, with $X'_1 := X_1$, we get

$$\begin{aligned} \Pr\left[\sum_{i=1}^{N} a_i X_i = \sum_{i=1}^{N'} a'_i X'_i\right] &\leq \Pr\left[a'_1 X_1 = \sum_{i=1}^{N} a_i X_i - \sum_{i=2}^{N'} a'_i X'_i\right] \\ &\leq \Pr\left[a'_1 \vartheta(X_1) = \vartheta\left(\sum_{i=1}^{N} a_i X_i - \sum_{i=2}^{N'} a'_i X'_i\right)\right] \\ &\leq \Pr\left[a'_1 = \vartheta(X_1)^{-1}\vartheta\left(\sum_{i=1}^{N} a_i X_i - \sum_{i=2}^{N'} a'_i X'_i\right)\right] = \frac{1}{|\mathcal{S}|}. \end{aligned}$$

□

4.2 Building Block: Commitments

We combine the linear function family with a special commitment scheme, which we define following [PW23] but with an important change. This commitment scheme allows us to homomorphically commit to the first-round message $R \in \mathcal{R}$ of the identification scheme that is induced by the linear function family. We require that there are two indistinguishable ways of setting up the commitment key: a statistically hiding mode and a statistically binding mode. In the statistically binding mode, we can (inefficiently[10]) extract messages $R \in \mathcal{R}$ from commitments. In the hiding mode, we can first output a commitment, and then, once we learn the challenge c of the induced identification scheme, we can simulate a transcript for that challenge and an opening for R. In other words, we have a special equivocation feature that allows us to open commitments to messages $R \in \mathcal{R}$ coming from the honest verifier zero-knowledge simulator of the identification scheme. So far, what we have discussed is the commitment scheme as introduced in [PW23]. The important change that we introduce is a stronger indistinguishability notion of commitment keys. Namely, we require that indistinguishability holds even if we give the distinguisher access to an oracle that can decide if inputs are in the image of the linear function. Looking ahead, we will show that the commitment scheme by Pan and Wagner [PW23] satisfies this stronger notion. We will require this stronger notion when we switch hiding commitment keys to binding commitment keys in the proof of our multi-signature construction. Another important difference to the notion of Pan and Wagner's is that in their notion, they consider the case where $(\mathsf{par}, x) \in \mathsf{Good}$ for $x \xleftarrow{\$} \mathcal{D}$, the adversary gets x, and in one of the two games, commitment keys are uniform. We instead require a notion, in which the adversary gets $X \xleftarrow{\$} \mathcal{R}$ and commitment keys are either in binding mode or in hiding mode with respect to X.

Definition 8 (Special Commitment Scheme)**.** Let $\mathsf{LF} = (\mathsf{LF.Gen}, \mathsf{F})$ be a linear function family with set of scalars $\mathcal{S}$, domain $\mathcal{D}$, and range $\mathcal{R}$. Further, let $\mathcal{G} = \{\mathcal{G}_{\mathsf{par}}\}, \mathcal{H} = \{\mathcal{H}_{\mathsf{par}}\}$ be families of subsets of abelian groups with efficiently computable group operations $\oplus$ and $\otimes$, respectively, and let $\mathcal{K} = \{\mathcal{K}_{\mathsf{par}}\}$ be a family of sets. An $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$-special commitment scheme for LF with key space $\mathcal{K}$, randomness space $\mathcal{G}$, and commitment space $\mathcal{H}$ is a tuple of PPT algorithms $\mathsf{CMT} = (\mathsf{BGen}, \mathsf{TGen}, \mathsf{Com}, \mathsf{TCom}, \mathsf{TCol})$ with the following syntax:

[10] Inefficient extraction is enough for our use case as the reduction that uses this extraction solves a statistically hard problem.

Game Q-$\mathbf{OKEYDIST}^{\mathcal{A}}_{0,\mathsf{CMT}}(\lambda)$	**Game** Q-$\mathbf{OKEYDIST}^{\mathcal{A}}_{1,\mathsf{CMT}}(\lambda)$
01 $\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda),\ X \xleftarrow{\$} \mathcal{R}$	05 $\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda),\ X \xleftarrow{\$} \mathcal{R}$
02 **for** $i \in [Q] : \mathsf{ck}_i \leftarrow \mathsf{BGen}(\mathsf{par})$	06 **for** $i \in [Q] : (\mathsf{ck}_i, \mathsf{td}_i) \leftarrow \mathsf{TGen}(\mathsf{par}, X)$
03 $\beta \leftarrow \mathcal{A}^{\mathrm{O}}(\mathsf{par}, X, (\mathsf{ck}_i)_{i\in[Q]})$	07 $\beta \leftarrow \mathcal{A}^{\mathrm{O}}(\mathsf{par}, X, (\mathsf{ck}_i)_{i\in[Q]})$
04 **return** β	08 **return** β
	Oracle $\mathrm{O}(Y)$ for $Y \in \mathcal{R}$
	09 **if** $Y \in \mathsf{F}(\mathcal{D}) :$ **return** 1, **else return** 0

Figure 3: The games $\mathbf{OKEYDIST}_0, \mathbf{OKEYDIST}_1$ for a special commitment scheme CMT and an adversary $\mathcal{A}$. The the difference to the definition in [PW23, PW24] is the oracle O and that $\mathcal{A}$ gets $X \in \mathcal{R}$ instead of $x \in \mathcal{D}$ with $(\mathsf{par}, x) \in \mathsf{Good}$.

- $\mathsf{BGen}(\mathsf{par}) \to \mathsf{ck}$ takes as input parameters par, and outputs a key $\mathsf{ck} \in \mathcal{K}_{\mathsf{par}}$.
- $\mathsf{TGen}(\mathsf{par}, X) \to (\mathsf{ck}, \mathsf{td})$ takes as input parameters par, and an element $X \in \mathcal{R}$, and outputs a key $\mathsf{ck} \in \mathcal{K}_{\mathsf{par}}$ and a trapdoor td.
- $\mathsf{Com}(\mathsf{ck}, R; \varphi) \to \mathsf{com}$ takes as input a key ck, an element $R \in \mathcal{R}$, and a randomness $\varphi \in \mathcal{G}_{\mathsf{par}}$, and outputs a commitment $\mathsf{com} \in \mathcal{H}_{\mathsf{par}}$.
- $\mathsf{TCom}(\mathsf{ck}, \mathsf{td}) \to (\mathsf{com}, St)$ takes as input a key ck and a trapdoor td, and outputs a commitment $\mathsf{com} \in \mathcal{H}_{\mathsf{par}}$ and a state St.
- $\mathsf{TCol}(St, c) \to (\varphi, R, s)$ takes as input a state St, and an element $c \in \mathcal{S}$, and outputs randomness $\varphi \in \mathcal{G}_{\mathsf{par}}$, and elements $R \in \mathcal{R}, s \in \mathcal{D}$.

We omit the subscript par if it is clear from the context.

Further, the algorithms are required to satisfy the following properties:

- **Homomorphism.** For all $\mathsf{par} \in \mathsf{LF.Gen}(1^\lambda), \mathsf{ck} \in \mathcal{K}_{\mathsf{par}}, R_0, R_1 \in \mathcal{R}$ and $\varphi_0, \varphi_1 \in \mathcal{G}$, the following holds:
$$\mathsf{Com}(\mathsf{ck}, R_0; \varphi_0) \otimes \mathsf{Com}(\mathsf{ck}, R_1; \varphi_1) = \mathsf{Com}(\mathsf{ck}, R_0 + R_1; \varphi_0 \oplus \varphi_1).$$
- **Good Parameters.** There is a set Good, such that membership to Good can be decided in polynomial time, and
$$\Pr\left[(\mathsf{par}, x) \notin \mathsf{Good} \mid \mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda),\ x \xleftarrow{\$} \mathcal{D}\right] \leq \varepsilon_{\mathsf{g}},$$
- **Uniform Keys.** For all $(\mathsf{par}, x) \in \mathsf{Good}$, the following distributions are identical:
$$\{(\mathsf{par}, x, \mathsf{ck}) \mid \mathsf{ck} \xleftarrow{\$} \mathcal{K}_{\mathsf{par}}\} \text{ and } \{(\mathsf{par}, x, \mathsf{ck}) \mid (\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, \mathsf{F}(x))\}.$$
- **Special Trapdoor Property.** For all $(\mathsf{par}, x) \in \mathsf{Good}$, and all $c \in \mathcal{S}$, the following distributions $\mathcal{T}_0$ and $\mathcal{T}_1$ have statistical distance at most ε_{t}:
$$\mathcal{T}_0 := \left\{ (\mathsf{par}, \mathsf{ck}, \mathsf{td}, x, c, \mathsf{com}, \mathsf{tr}) \;\middle|\; \begin{array}{l} (\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, \mathsf{F}(x)) \\ (\mathsf{com}, St) \leftarrow \mathsf{TCom}(\mathsf{ck}, \mathsf{td}), \\ \mathsf{tr} \leftarrow \mathsf{TCol}(St, c) \end{array} \right\}$$
$$\mathcal{T}_1 := \left\{ (\mathsf{par}, \mathsf{ck}, \mathsf{td}, x, c, \mathsf{com}, \mathsf{tr}) \;\middle|\; \begin{array}{l} (\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, \mathsf{F}(x)) \\ r \xleftarrow{\$} \mathcal{D},\ R := \mathsf{F}(r),\ \varphi \xleftarrow{\$} \mathcal{G}, \\ \mathsf{com} := \mathsf{Com}(\mathsf{ck}, R; \varphi), \\ s := c \cdot x + r,\ \mathsf{tr} := (\varphi, R, s) \end{array} \right\}$$
- **Oracle-Aided Multi-Key Indistinguishability.** For every $Q = \mathsf{poly}(\lambda)$ and any PPT algorithm $\mathcal{A}$, the following advantage is negligible:
$$\mathsf{Adv}^{Q\text{-keydist}}_{\mathcal{A},\mathsf{CMT}}(\lambda) := \Big| \Pr\left[Q\text{-}\mathbf{OKEYDIST}^{\mathcal{A}}_{0,\mathsf{CMT}}(\lambda) \Rightarrow 1\right] - \Pr\left[Q\text{-}\mathbf{OKEYDIST}^{\mathcal{A}}_{1,\mathsf{CMT}}(\lambda) \Rightarrow 1\right] \Big|,$$

where games $\mathbf{OKEYDIST}_0, \mathbf{OKEYDIST}_1$ are defined in Figure 3.

- **Statistically Binding.** There exists some (unbounded) algorithm Ext, such that for every (unbounded) algorithm $\mathcal{A}$ the following probability is at most ε_{b}:

$$\Pr\left[\mathsf{Com}(\mathsf{ck}, R'; \varphi') = \mathsf{com} \wedge R \neq R' \;\middle|\; \begin{array}{l} \mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda), \\ \mathsf{ck} \leftarrow \mathsf{BGen}(\mathsf{par}),\ (\mathsf{com}, St) \leftarrow \mathcal{A}(\mathsf{ck}), \\ R \leftarrow \mathsf{Ext}(\mathsf{ck}, \mathsf{com}),\ (R', \varphi') \leftarrow \mathcal{A}(St) \end{array}\right].$$

4.3 Multi-Signature Construction

We abstractly present our tightly secure two-round multi-signature scheme with key aggregation. Let $\mathsf{LF} = (\mathsf{LF.Gen}, \mathsf{F})$ be a linear function family with set of scalars $\mathcal{S}$, domain $\mathcal{D}$, and range $\mathcal{R}$. Further, let $\mathsf{CMT} = (\mathsf{BGen}, \mathsf{TGen}, \mathsf{Com}, \mathsf{TCom}, \mathsf{TCol})$ be a special commitment scheme for LF with key space $\mathcal{K}$, randomness space $\mathcal{G}$, and commitment space $\mathcal{H}$. We make use of random oracles $\mathsf{H}\colon \{0,1\}^* \to \mathcal{K}$, $\mathsf{H}_a\colon \{0,1\}^* \to \mathcal{S}$, $\mathsf{H}_b\colon \{0,1\}^* \to \{0,1\}$, and $\mathsf{H}_c\colon \{0,1\}^* \to \mathcal{S}$. We give a verbal description of our scheme next and present it formally as pseudocode in Figure 4.

Setup and Key Generation. The public parameters of the scheme are $\mathsf{par} := \mathsf{LF.Gen}(1^\lambda)$, which defines the linear function $\mathsf{F}(\cdot) = \mathsf{F}(\mathsf{par}, \cdot)$. To generate a key, a signer samples $x \xleftarrow{\$} \mathcal{D}$ and $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$. Then, it sets

$$\mathsf{sk} := (x, \mathsf{seed}), \quad \mathsf{pk} := \mathsf{F}(x) \in \mathcal{R}.$$

Key Aggregation. For N signers, let $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ be the set of their public keys. For all $i \in [N]$, the i-th key aggregation coefficient is defined as $a_i := \mathsf{H}_a(\langle \mathcal{P} \rangle, \mathsf{pk}_i)$. Then, the aggregated public key $\tilde{\mathsf{pk}}$ is computed as

$$\tilde{\mathsf{pk}} := \sum_{i=1}^{N} a_i \cdot \mathsf{pk}_i \in \mathcal{R}.$$

Signing Protocol. Suppose N signers with public keys $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ want to sign a message $\mathsf{m} \in \{0,1\}^*$. We describe the signing protocol $\mathsf{Sig} = (\mathsf{Sig}_0, \mathsf{Sig}_1, \mathsf{Sig}_2)$ from the perspective of the first signer with secret key $\mathsf{sk}_1 = (x_1, \mathsf{seed}_1)$ and public key pk_1. Let $\tilde{\mathsf{pk}}$ be the aggregated public key as defined above.

1. *Commitment Phase.* The signer derives commitment keys

$$\mathsf{ck}_0 = \mathsf{H}(0, \tilde{\mathsf{pk}}, \mathsf{m}) \ \text{ and } \ \mathsf{ck}_1 = \mathsf{H}(1, \tilde{\mathsf{pk}}, \mathsf{m}).$$

   Further, it computes a bit $b_1 := \mathsf{H}_b(\mathsf{seed}_1, \tilde{\mathsf{pk}}, \mathsf{m})$, samples an element $r_1 \xleftarrow{\$} \mathcal{D}$, and computes $R_1 := \mathsf{F}(r_1)$. Then, the signer commits to R_1 via the commitment key ck_{b_1} by

$$\mathsf{com}_1 := \mathsf{Com}(\mathsf{ck}_{b_1}, R_1; \varphi_1) \in \mathcal{H} \quad \text{for randomness } \varphi_1 \xleftarrow{\$} \mathcal{G}.$$

   It then sends $\mathsf{pm}_{1,1} := (b_1, \mathsf{com}_1)$ to the other signers.

2. *Response Phase.* Let $\mathcal{M}_1 = (\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N})$ be the list of messages output in the commitment phase, where $\mathsf{pm}_{1,i} = (b_i, \mathsf{com}_i)$ is sent by signer $i \in [N]$. For the sets $\mathcal{I}_0 := \{i \in [N] \mid b_i = 0\}$ and $\mathcal{I}_1 := \{i \in [N] \mid b_i = 1\}$, the signer aggregates the commitments as

$$\bar{\mathsf{com}}_0 := \bigotimes_{i \in \mathcal{I}_0} \mathsf{com}_i, \quad \bar{\mathsf{com}}_1 := \bigotimes_{i \in \mathcal{I}_1} \mathsf{com}_i,$$

   and computes $\tilde{\mathsf{pk}}_0 := \sum_{i \in \mathcal{I}_0} a_i \cdot \mathsf{pk}_i$, $\tilde{\mathsf{pk}}_1 := \tilde{\mathsf{pk}} - \tilde{\mathsf{pk}}_0$. Note that $\tilde{\mathsf{pk}}_1 = \sum_{i \in \mathcal{I}_1} a_i \cdot \mathsf{pk}_i$. Then, it derives the challenge $c := \mathsf{H}_c(\tilde{\mathsf{pk}}, \tilde{\mathsf{pk}}_0, \bar{\mathsf{com}}_0, \bar{\mathsf{com}}_1, \mathsf{m})$ and computes its response $s_1 := c \cdot a_1 x_1 + r_1$. Finally, it sends $\mathsf{pm}_{2,1} := (s_1, \varphi_1)$ to the other signers.

3. *Aggregation Phase.* Let $\mathcal{M}_2 = (\mathsf{pm}_{2,1}, \ldots, \mathsf{pm}_{2,N})$ be the list of messages output in the response phase, where $\mathsf{pm}_{2,i} = (s_i, \varphi_i)$ is sent by signer $i \in [N]$. For the final signature, the signer aggregates the responses and commitment randomness as

$$\bar{s}_0 := \sum_{i \in \mathcal{I}_0} s_i, \quad \bar{s}_1 := \sum_{i \in \mathcal{I}_1} s_i, \quad \bar{\varphi}_0 := \bigoplus_{i \in \mathcal{I}_0} \varphi_i, \quad \bar{\varphi}_1 := \bigoplus_{i \in \mathcal{I}_1} \varphi_i.$$

Further, it defines $\sigma_0 := (\bar{s}_0, \bar{\varphi}_0)$, $\sigma_1 := (\bar{s}_1, \bar{\varphi}_1)$. The final signature is then $\sigma := (\tilde{\mathsf{pk}}_0, \sigma_0, \sigma_1, c)$.

Signature Verification. Let $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ be a set of public keys, let $\mathsf{m} \in \{0,1\}^*$ be a message, and let $\sigma := (\tilde{\mathsf{pk}}_0, \sigma_0, \sigma_1, c)$ be a signature. First, parse $(\bar{s}_0, \bar{\varphi}_0) := \sigma_0$ and $(\bar{s}_1, \bar{\varphi}_1) := \sigma_1$. To verify the signature σ, one first computes the aggregated public key $\tilde{\mathsf{pk}}$ as above. Then, one sets $\tilde{\mathsf{pk}}_1 := \tilde{\mathsf{pk}} - \tilde{\mathsf{pk}}_0$ and reconstructs the commitments

$$\bar{\mathsf{com}}_0 := \mathsf{Com}(\mathsf{ck}_0, \mathsf{F}(\bar{s}_0) - c \cdot \tilde{\mathsf{pk}}_0; \bar{\varphi}_0), \quad \bar{\mathsf{com}}_1 := \mathsf{Com}(\mathsf{ck}_1, \mathsf{F}(\bar{s}_1) - c \cdot \tilde{\mathsf{pk}}_1; \bar{\varphi}_1),$$

where commitment keys are derived as $\mathsf{ck}_0 = \mathsf{H}(0, \tilde{\mathsf{pk}}, \mathsf{m})$ and $\mathsf{ck}_1 = \mathsf{H}(1, \tilde{\mathsf{pk}}, \mathsf{m})$. Then, one accepts the signature if and only if $c = \mathsf{H}_c(\tilde{\mathsf{pk}}, \tilde{\mathsf{pk}}_0, \bar{\mathsf{com}}_0, \bar{\mathsf{com}}_1, \mathsf{m})$. Correctness of the construction follows by inspection. It is also clear that the scheme satisfies Definition 5. We now prove tight security of our construction, based on the security properties of the linear function family and the commitment scheme. Note that our security bound contains an additive term $|\mathsf{F}(\mathcal{D})|/|\mathcal{R}|$. Therefore, it is only meaningful if F is not surjective. For the concrete linear function that we will use, this term is negligible.

Theorem 1. *Let* $\mathsf{LF} = (\mathsf{LF.Gen}, \mathsf{F})$ *be a linear function family, and let* CMT *be an* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-special commitment scheme for* LF*. Further, let* $\mathsf{H}\colon \{0,1\}^* \to \mathcal{K}$*,* $\mathsf{H}_a\colon \{0,1\}^* \to \mathcal{S}$*,* $\mathsf{H}_b\colon \{0,1\}^* \to \{0,1\}$*, and* $\mathsf{H}_c\colon \{0,1\}^* \to \mathcal{S}$ *be random oracles. Consider the multi-signature scheme* $\mathsf{MS} := \mathsf{T\text{-}Spoon}[\mathsf{LF}, \mathsf{CMT}]$ *defined in Figure 4. Let* $\mathcal{A}$ *be any PPT algorithm making* $Q_{\mathsf{H}}, Q_{\mathsf{H}_a}, Q_{\mathsf{H}_b}, Q_{\mathsf{H}_c}, Q_S$ *queries to oracles* $\mathsf{H}, \mathsf{H}_a, \mathsf{H}_b, \mathsf{H}_c, \textsc{Sig}_0$*, respectively. Then, there are PPT algorithms* $\mathcal{B}_i$ *with* $\mathbf{T}(\mathcal{B}_i) \approx \mathbf{T}(\mathcal{A})$ *for every* $i \in \{1,2\}$ *such that*

$$\mathsf{Adv}^{\mathsf{MS\text{-}EUF\text{-}CMA}}_{\mathcal{A},\mathsf{MS}}(\lambda) \le 2\varepsilon_{\mathsf{g}} + Q_S\varepsilon_{\mathsf{t}} + 2Q_{\mathsf{H}}Q_{\mathsf{H}_c}\varepsilon_{\mathsf{b}} + \frac{Q_{\mathsf{H}_a}|\mathsf{F}(\mathcal{D})| + 1}{|\mathcal{R}|} + \frac{Q_{\mathsf{H}_a} + Q_{\mathsf{H}_a}^2 + 2Q_{\mathsf{H}_c}}{|\mathcal{S}|} + \frac{Q_{\mathsf{H}_b}}{2^\lambda} + \mathsf{Adv}^{\mathsf{keydist}}_{\mathcal{B}_1,\mathsf{LF}}(\lambda) + 2 \cdot \mathsf{Adv}^{Q_{\mathsf{H}}\text{-}\mathsf{keydist}}_{\mathcal{B}_2,\mathsf{CMT}}(\lambda).$$

Proof. The proof is structured as a sequence of hybrid games, with any two subsequent games being indistinguishable. In the final game, we will argue that winning the game is a statistically hard problem. For completeness, we present all games as pseudocode in Figures 5 and 6.

Game $\mathbf{G}_0$: This game is the real security game $\mathbf{MS\text{-}EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{MS}}$ for multi-signatures. In the beginning, the game samples parameters $\mathsf{par} \leftarrow \mathsf{LF.Gen}(\lambda)$. It also samples $x \xleftarrow{\$} \mathcal{D}$ and $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$ for a secret key $\mathsf{sk} := (x, \mathsf{seed})$ and public key $\mathsf{pk} := X := \mathsf{F}(x)$. Then, the game runs $\mathcal{A}$ on input $(\mathsf{par}, \mathsf{pk})$ with access to signing oracles and random oracles as follows.

- *Signing Oracles* $\textsc{Sig}_0, \textsc{Sig}_1, \textsc{Sig}_2$: These oracles simulate algorithms $\mathsf{Sig}_0, \mathsf{Sig}_1, \mathsf{Sig}_2$ on secret key sk, respectively. Whenever $\mathcal{A}$ calls the oracle $\textsc{Sig}_0(\mathcal{P}, \mathsf{m})$ to start a new session in which message m is signed for public keys $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$, the oracle adds $(\mathcal{P}, \mathsf{m})$ to a list $\mathsf{Queried}$. Here, we always assume that $\mathsf{pk} = \mathsf{pk}_1$. Further, we ignore the oracle $\textsc{Sig}_2$, which would compute the combined signature from individual signatures. This is without loss of generality, since algorithm Sig_2 does not make use of any secret state and can be publicly run using the messages output in Sig_0 and Sig_1. Thus, one could always build a wrapper adversary that internally runs $\mathcal{A}$ and simulates the oracle $\textsc{Sig}_2$ for itself. Clearly, the wrapper adversary has the same running time and advantage as $\mathcal{A}$.
- *Random Oracles* $\mathsf{H}, \mathsf{H}_a, \mathsf{H}_b, \mathsf{H}_c$: The random oracles $\mathsf{H}, \mathsf{H}_a, \mathsf{H}_c$ are simulated honestly via standard lazy sampling. For this purpose, the game holds internal maps h, h_a, h_c that assign the inputs of the

respective random oracles to their outputs. On the other hand, the random oracle H_b is simulated via an indirection:

$$\mathsf{H}_b(\mathsf{seed}', \tilde{\mathsf{pk}}, \mathsf{m}) = \begin{cases} \hat{\mathsf{H}}_b(\mathsf{seed}', \tilde{\mathsf{pk}}, \mathsf{m}), & \text{if } \mathsf{seed}' \neq \mathsf{seed} \\ \Gamma(\tilde{\mathsf{pk}}, \mathsf{m}), & \text{if } \mathsf{seed}' = \mathsf{seed} \end{cases},$$

where $\hat{\mathsf{H}}_b$ and Γ are internal oracles, also implemented lazily and only known to the game. In this way, the game will be able to distinguish queries made by $\mathcal{A}$ from queries it made itself. When the game would call $\mathsf{H}_b(\mathsf{seed}, \cdot, \cdot)$ in the signing oracle, it instead calls Γ directly.

At the end of the game, the adversary outputs a forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$ and the game outputs 1 if and only if: $\mathsf{pk} \in \mathcal{P}^*$, $(\mathcal{P}^*, \mathsf{m}^*) \notin \mathsf{Queried}$, and $\mathsf{Ver}(\mathcal{P}^*, \mathsf{m}^*, \sigma^*) = 1$. Henceforth, we write $\tilde{\mathsf{pk}}^* := \mathsf{Agg}(\mathcal{P}^*)$ and

$$(\tilde{\mathsf{pk}}_0^*, \sigma_0^*, \sigma_1^*, c^*) := \sigma^*, \;\; (\bar{s}_0^*, \bar{\varphi}_0^*) := \sigma_0^*, \;\; (\bar{s}_1^*, \bar{\varphi}_1^*) := \sigma_1^*, \;\; \tilde{\mathsf{pk}}_1^* := \tilde{\mathsf{pk}}^* - \tilde{\mathsf{pk}}_0^*.$$

Clearly, we have

$$\mathsf{Adv}^{\mathsf{MS\text{-}EUF\text{-}CMA}}_{\mathcal{A},\mathsf{MS}}(\lambda) = \Pr[\mathbf{G}_0 \Rightarrow 1].$$

Game $\mathbf{G}_1$: In this game, we add two abort conditions. First, the game aborts if $(\mathsf{par}, x) \notin \mathsf{Good}$, where Good is the set of good parameters as specified in the definition of the special commitment scheme (cf. Definition 8). By definition, the probability of abort here is upper bounded by ε_{g}. Second, the game aborts if the adversary makes a random oracle query $\mathsf{H}_b(\mathsf{seed}, \cdot, \cdot)$. Note that this does not include the queries that are made by the game itself, as the game directly calls oracle Γ instead. As $\mathcal{A}$ only gets information about seed through the values $\mathsf{H}_b(\mathsf{seed}, \cdot, \cdot)$, and seed is sampled uniformly at random from the set $\{0,1\}^\lambda$, the probability of abort here is upper bounded by $Q_{\mathsf{H}_b}/2^\lambda$ (via a union bound over all random oracle queries to H_b). Overall, we therefore get

$$|\Pr[\mathbf{G}_0 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]| \leq \frac{Q_{\mathsf{H}_b}}{2^\lambda} + \varepsilon_{\mathsf{g}}.$$

Game $\mathbf{G}_2$: In this game, we change how the random oracle H for the commitment keys is simulated. Recall that until now, when H is queried on an input $(b, \tilde{\mathsf{pk}}, \mathsf{m})$ for which the hash value has not been defined yet, the game samples a commitment key $\mathsf{ck}_b \xleftarrow{\$} \mathcal{K}$ uniformly at random and returns it. From now on, the game samples ck_b in hiding mode: that is, $(\mathsf{ck}_b, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, X)$ where the trapdoor td is stored in a map tr as $tr[b, \tilde{\mathsf{pk}}, \mathsf{m}] := \mathsf{td}$. By the uniform keys property of CMT and $(\mathsf{par}, x) \in \mathsf{Good}$ due to the changes in $\mathbf{G}_1$, we have

$$\Pr[\mathbf{G}_1 \Rightarrow 1] = \Pr[\mathbf{G}_2 \Rightarrow 1].$$

Game $\mathbf{G}_3$: In this game, we change the signing oracle. This will allow us to simulate the signing oracle without the secret key x, but instead with the trapdoors for commitment keys. Recall that until now, when $\textsc{Sig}_0$ is queried on an input $(\mathcal{P}, \mathsf{m})$, the game runs $\mathsf{Sig}_0(\mathcal{P}, \mathsf{sk}, \mathsf{m})$: i.e., it computes a bit $b := \mathsf{H}_b(\mathsf{seed}, \tilde{\mathsf{pk}}, \mathsf{m})$, samples an element $r \xleftarrow{\$} \mathcal{D}$, and commits to $R := \mathsf{F}(r)$ via the commitment key $\mathsf{ck}_b := \mathsf{H}(b, \tilde{\mathsf{pk}}, \mathsf{m})$ by sampling randomness $\varphi \xleftarrow{\$} \mathcal{G}$ and returning $\mathsf{com} := \mathsf{Com}(\mathsf{ck}_b, R; \varphi)$. From now on, we change this as follows, keeping in mind that the commitment key ck_b is in hiding mode due to the changes in $\mathbf{G}_2$. Instead of computing com as above, it computes it as $(\mathsf{com}, St) \leftarrow \mathsf{TCom}(\mathsf{ck}_b, \mathsf{td})$ where $\mathsf{td} = tr[b, \tilde{\mathsf{pk}}, \mathsf{m}]$. With that, we also change the oracle $\textsc{Sig}_1$ as follows. Recall that until now, the game runs Sig_1, which first computes a challenges c from the first-round messages using the random oracle H_c, and then returns φ and $s := c \cdot a_1 x + r$. From now on, the game instead derives $(\varphi, R, s) \leftarrow \mathsf{TCol}(St, ca_1)$ and returns (s, φ). By applying the special trapdoor property of CMT to each signing query, we get

$$|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_3 \Rightarrow 1]| \leq Q_S \varepsilon_{\mathsf{t}}.$$

Game $\mathbf{G}_4$: In this game, we no longer require that $(\mathsf{par}, x) \in \mathsf{Good}$, which was introduced in $\mathbf{G}_1$. It is straightforward to see that

$$|\Pr[\mathbf{G}_3 \Rightarrow 1] - \Pr[\mathbf{G}_4 \Rightarrow 1]| \leq \varepsilon_{\mathsf{g}}.$$

Note that after these changes, we no longer need the secret key x.

Game $\mathbf{G}_5$: In this game, we change how the public key $\mathsf{pk} = X$ is generated. Recall that until now, it is generated by sampling $x \xleftarrow{\$} \mathcal{D}$ and then computing $X := \mathsf{F}(x)$. From now on, we sample it in lossy mode:

i.e., the game samples $\mathsf{pk} := X \xleftarrow{\$} \mathcal{R}$. We can bound the distinguishing advantage between games $\mathbf{G}_4$ and $\mathbf{G}_5$ via a direct reduction $\mathcal{B}_1$ against the key indistinguishability of the linear function family LF. To this end, $\mathcal{B}_1$ gets (par, X) as input, simulates $\mathbf{G}_5$ for $\mathcal{A}$, and outputs whatever the game outputs. Thus, we get

$$|\Pr[\mathbf{G}_4 \Rightarrow 1] - \Pr[\mathbf{G}_5 \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{keydist}}_{\mathcal{B}_1,\mathsf{LF}}(\lambda).$$

Game $\mathbf{G}_6$: In this game, we add an abort condition. Concretely, the game aborts if $\tilde{\mathsf{pk}}^* \in \mathsf{F}(\mathcal{D})$, i.e., the aggregated public key for the forgery is not lossy. We can bound the probability of this event using Lemma 2. To this end, we build a reduction $\mathcal{I}$ that runs in the game specified in Lemma 2 and wins (if some index guessing of it was correct). We now describe the reduction $\mathcal{I}$. It gets as input (par, X). Then, it samples an index $i^* \xleftarrow{\$} [Q_{\mathsf{H}_a}]$ and simulates $\mathbf{G}_5$ for $\mathcal{A}$ with the following changes:

- Consider the i^*-th query to random oracle H_a, and denote it by $\mathsf{H}_a(\langle\mathcal{P}\rangle, \mathsf{pk}_1)$. If the hash value is already defined or $\mathsf{pk}_1 \neq \mathsf{pk}$, the reduction aborts its execution. Otherwise, let $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ (where $\mathsf{pk} = \mathsf{pk}_1$), and the reduction queries $a_i := \mathsf{H}_a(\langle\mathcal{P}\rangle, \mathsf{pk})$ for all $2 \leq i \leq N$. Then, it outputs $(\mathsf{pk}_2, a_2), \ldots, (\mathsf{pk}_N, a_N)$ to the game specified in Lemma 2, and obtains $a_1 \in \mathcal{S}$. The reduction then programs $\mathsf{H}_a(\langle\mathcal{P}\rangle, \mathsf{pk}_1) := a_1$ and continues the simulation.
- Later, when $\mathcal{A}$ outputs the forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$, the reduction checks all the verification steps as before. Additionally, it checks if the value $\mathsf{H}_a(\mathcal{P}^*, \mathsf{pk})$ was defined during the i^*-th query to random oracle H_a. If this is not the case, the reduction aborts.

It is clear that the reduction $\mathcal{I}$ is successful against the game specified in Lemma 2 if $\mathbf{G}_5$ outputs 1 and the guess i^* was correct. As the view of $\mathcal{A}$ in its interaction with the reduction is independent of i^*, we can bound the probability of this using Lemma 2 and get

$$|\Pr[\mathbf{G}_5 \Rightarrow 1] - \Pr[\mathbf{G}_6 \Rightarrow 1]| \leq Q_{\mathsf{H}_a}\left(\frac{|\mathsf{F}(\mathcal{D})|}{|\mathcal{R}|} + \frac{1}{|\mathcal{S}|}\right).$$

Note that after these changes, we have the guarantee that the aggregated public key $\tilde{\mathsf{pk}}^*$ is always lossy (i.e., not in the range $\mathsf{F}(\mathcal{D})$ of the linear function $\mathsf{F}(\cdot)$). In particular, this means that one of the two sub-aggregate keys $\tilde{\mathsf{pk}}^*_0, \tilde{\mathsf{pk}}^*_1$ is also lossy. This follows directly from the relation $\tilde{\mathsf{pk}}^* = \tilde{\mathsf{pk}}^*_0 + \tilde{\mathsf{pk}}^*_1$ and the fact that the image of a linear function is a vector space (and thus closed under addition).

Game $\mathbf{G}_7$: In this game, we add another abort condition. Concretely, the game aborts if $\tilde{\mathsf{pk}}_{1-b^*} \in \mathsf{F}(\mathcal{D})$ for $b^* := \Gamma(\tilde{\mathsf{pk}}^*, \mathsf{m}^*)$ in the forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$ output by $\mathcal{A}$. Assuming that $\mathbf{G}_6$ outputs 1, we claim that b^* is independent of $\mathcal{A}$'s view, except with probability $1/|\mathcal{R}| + Q^2_{\mathsf{H}_a}/|\mathcal{S}|$. For this, first consider the following event, conditioned on $X \neq 0$: There exists a pair of public key lists $\mathcal{P} \neq \mathcal{P}'$ that have been submitted to H_a such that $X \in \mathcal{P}, \mathcal{P}'$ and $\mathsf{Agg}(\mathcal{P}) = \mathsf{Agg}(\mathcal{P}')$, i.e., their aggregated public keys collide. By Lemma 3, the probability of having a collision for one such pair is bounded by $1/|\mathcal{S}|$. Thus, by a union bound over all such pairs, we find that the probability of the event occurring is bounded by $Q^2_{\mathsf{H}_a}/|\mathcal{S}|$. Further, due to the changes in $\mathbf{G}_5$ (sampling $X \xleftarrow{\$} \mathcal{R}$), the probability that $X = 0$ is $1/|\mathcal{R}|$. From that and due to the changes in $\mathbf{G}_1$, the claim follows. Finally, due to the changes in $\mathbf{G}_6$, we know that one of the two sub-aggregate keys $\tilde{\mathsf{pk}}^*_0, \tilde{\mathsf{pk}}^*_1$ is lossy. Given that b^* is independent of $\mathcal{A}$'s view, it is clear that $\tilde{\mathsf{pk}}_{1-b^*}$ is lossy with probability $1/2$. Therefore, we get

$$\Pr[\mathbf{G}_7 \Rightarrow 1] \geq \frac{1}{2}\left(\Pr[\mathbf{G}_6 \Rightarrow 1] - \frac{1}{|\mathcal{R}|} - \frac{Q^2_{\mathsf{H}_a}}{|\mathcal{S}|}\right).$$

Game $\mathbf{G}_8$: In this game, we change how the random oracle H for the commitment keys is simulated. Recall that until now, when H is queried on an input $(b, \tilde{\mathsf{pk}}, \mathsf{m})$ for which the hash value has not been defined yet, the game samples ck_b in hiding mode, i.e., $(\mathsf{ck}_b, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, X)$. From now on, we change this as follows, distinguishing two cases depending on the bit $\Gamma(\tilde{\mathsf{pk}}, \mathsf{m})$. If $b = \Gamma(\tilde{\mathsf{pk}}, \mathsf{m})$, then the game samples ck_b as before in hiding mode, i.e., $(\mathsf{ck}_b, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, X)$ where the trapdoor td is stored in a map tr as $tr[b, \tilde{\mathsf{pk}}, \mathsf{m}] := \mathsf{td}$. On the other hand, if $b = 1 - \Gamma(\tilde{\mathsf{pk}}, \mathsf{m})$, then the game samples ck_b in binding mode, i.e., $\mathsf{ck}_b \leftarrow \mathsf{BGen}(\mathsf{par})$. We can bound the distinguishing advantage between games $\mathbf{G}_7$ and $\mathbf{G}_8$ via a direct reduction $\mathcal{B}_2$ against the oracle-aided multi-key indistinguishability of the special

commitment scheme CMT. To this end, $\mathcal{B}_2$ gets par, X, and commitment keys $\mathsf{ck}_1, \ldots, \mathsf{ck}_{Q_\mathsf{H}}$ as input. Then, it simulates $\mathbf{G}_8$ for $\mathcal{A}$, while embedding the commitment keys in random oracle responses for queries $\mathsf{H}(b, \tilde{\mathsf{pk}}, \mathsf{m})$ where $b = 1 - \Gamma(\tilde{\mathsf{pk}}, \mathsf{m})$. Importantly, the reduction uses the oracle $\mathrm{O}(\cdot)$ from the oracle-aided multi-key indistinguishability to decide the key lossiness checks that have been introduced in $\mathbf{G}_6$ and $\mathbf{G}_7$. In the end, it outputs whatever the game outputs. Clearly, $\mathcal{B}_2$ perfectly simulates $\mathbf{G}_8$ if the keys $\mathsf{ck}_1, \ldots, \mathsf{ck}_{Q_\mathsf{H}}$ are generated via algorithm BGen. Otherwise, if the keys are generated via algorithm TGen, then it perfectly simulates $\mathbf{G}_7$ for $\mathcal{A}$. Thus, we get

$$|\Pr[\mathbf{G}_7 \Rightarrow 1] - \Pr[\mathbf{G}_8 \Rightarrow 1]| \leq \mathsf{Adv}^{Q_\mathsf{H}\text{-keydist}}_{\mathcal{B}_2,\mathsf{CMT}}(\lambda).$$

Game $\mathbf{G}_9$: In this game, we change how the random oracle H_c for the challenges is simulated and add an abort condition. Recall that until now, when H_c is queried on an input $(\tilde{\mathsf{pk}}, \tilde{\mathsf{pk}}_0, \bar{\mathsf{com}}_0, \bar{\mathsf{com}}_1, \mathsf{m})$ for which the hash value has not been defined yet, the game answers it via lazy sampling. From now on, we change this as follows, depending on the bit $\Gamma(\tilde{\mathsf{pk}}, \mathsf{m})$. Concretely, set $b := 1 - \Gamma(\tilde{\mathsf{pk}}, \mathsf{m})$. Then we know that the commitment key $\mathsf{ck}_b := \mathsf{H}(b, \tilde{\mathsf{pk}}, \mathsf{m})$ is in binding mode due to the changes in $\mathbf{G}_8$. For these queries, the game now runs $R \leftarrow \mathsf{Ext}(\mathsf{ck}_b, \bar{\mathsf{com}}_b)$ and stores the value in a map r as $r[\tilde{\mathsf{pk}}, \mathsf{m}, \bar{\mathsf{com}}_b] := R$. Here, Ext is the (unbounded) extractor for the statistical binding property of the special commitment scheme CMT. Further, we add an abort condition. For this, consider the forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$ and let $\tilde{\mathsf{pk}}^* := \mathsf{Agg}(\mathcal{P}^*)$ and

$$(\tilde{\mathsf{pk}}^*_0, \sigma^*_0, \sigma^*_1, c^*) := \sigma^*, \;\; (\bar{s}^*_0, \bar{\varphi}^*_0) := \sigma^*_0, \;\; (\bar{s}^*_1, \bar{\varphi}^*_1) := \sigma^*_1, \;\; \tilde{\mathsf{pk}}^*_1 := \tilde{\mathsf{pk}}^* - \tilde{\mathsf{pk}}^*_0.$$

Assuming $\mathbf{G}_8$ does not return 0, recall that the game sets $b^* := \Gamma(\tilde{\mathsf{pk}}^*, \mathsf{m}^*)$ (see $\mathbf{G}_7$). Further, note that $r[\tilde{\mathsf{pk}}^*, \mathsf{m}^*, \bar{\mathsf{com}}_{b^*}]$ is defined once the forgery signature is verified by the game. The game then computes the value $R^*_{1-b^*} := \mathsf{F}(\bar{s}^*_{1-b^*}) - c^* \cdot \tilde{\mathsf{pk}}^*_{1-b^*}$ as the verification algorithm does, and aborts if $R^*_{1-b^*} \neq r[\tilde{\mathsf{pk}}^*, \mathsf{m}^*, \bar{\mathsf{com}}_{b^*}]$. We can bound the distinguishing advantage between games $\mathbf{G}_8$ and $\mathbf{G}_9$ via an (unbounded) reduction $\mathcal{B}_3$ from the statistical binding property of the special commitment scheme CMT. To this end, $\mathcal{B}_3$ gets par and a commitment key ck^* as input. Then, it samples indices $i_\mathsf{H} \xleftarrow{\$} [Q_\mathsf{H}]$ and $i_{\mathsf{H}_c} \xleftarrow{\$} [Q_{\mathsf{H}_c}]$, and simulates $\mathbf{G}_8$ for $\mathcal{A}$ with the following changes. For the i_H-th query to H, if it had to sample a binding key, it instead responds with ck^*. And for the i_{H_c}-th query to H_c, if it had to run Ext, it outputs $\bar{\mathsf{com}}_b$ where $b = 1 - \Gamma(\tilde{\mathsf{pk}}, \mathsf{m})$ and its state to the binding experiment. Finally, when $\mathcal{A}$ outputs its forgery, these guesses were correct for the forgery (i.e., query i_H was used to derive ck_{1-b^*} and query i_{H_c} was used to derive c^*), and $R^*_{1-b^*} \neq r[\tilde{\mathsf{pk}}^*, \mathsf{m}^*, \bar{\mathsf{com}}_{b^*}]$, the reduction outputs $(R^*_{1-b^*}, \bar{\varphi}^*_{1-b^*})$. Otherwise, it aborts the execution. It follows that if the reduction guesses the correct queries, then it breaks the statistical binding property. As the view of $\mathcal{A}$ is as in $\mathbf{G}_9$ and independent of the indices $i_\mathsf{H}, i_{\mathsf{H}_c}$, we get

$$|\Pr[\mathbf{G}_8 \Rightarrow 1] - \Pr[\mathbf{G}_9 \Rightarrow 1]| \leq Q_\mathsf{H} Q_{\mathsf{H}_c} \varepsilon_\mathsf{b}.$$

Finally, we bound the probability that $\mathbf{G}_9$ outputs 1 using Lemma 1. For this, we build an unbounded reduction $\mathcal{I}$ from the lossy soundness experiment defined in Lemma 1:

- The reduction gets as input parameters par for the linear function. Then, it samples an index $\hat{i} \xleftarrow{\$} [Q_{\mathsf{H}_c}]$ and simulates $\mathbf{G}_9$ for $\mathcal{A}$ until $\mathcal{A}$ outputs its forgery, except for query $\hat{i}$ to oracle H_c. Note that this includes sampling $\mathsf{pk} = X \xleftarrow{\$} \mathcal{R}$.
- Let $\mathsf{H}_c(\tilde{\mathsf{pk}}, \tilde{\mathsf{pk}}_0, \bar{\mathsf{com}}_0, \bar{\mathsf{com}}_1, \mathsf{m})$ be this $\hat{i}$-th query to H_c. If the hash value for this query is already defined, the reduction proceeds as $\mathbf{G}_9$ would do. Otherwise, it runs $R \leftarrow \mathsf{Ext}(\mathsf{ck}_b, \bar{\mathsf{com}}_b)$ where $b = 1 - \Gamma(\tilde{\mathsf{pk}}, \mathsf{m})$ and stores the value in a map r as $r[\tilde{\mathsf{pk}}, \mathsf{m}, \bar{\mathsf{com}}_b] := R$. Then, the reduction outputs its state, the key $\tilde{\mathsf{pk}}_b$, and the extracted R to the lossy soundness game and obtains a value $c \in \mathcal{S}$. It sets $h_c[\tilde{\mathsf{pk}}, \tilde{\mathsf{pk}}_0, \bar{\mathsf{com}}_0, \bar{\mathsf{com}}_1, \mathsf{m}] := c$ and continues the simulation of $\mathbf{G}_9$.
- Later, when $\mathcal{A}$ outputs the forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$, the reduction checks all the verification steps in $\mathbf{G}_9$. Additionally, it checks if the value $\mathsf{H}_c(\tilde{\mathsf{pk}}^*, \tilde{\mathsf{pk}}^*_0, \bar{\mathsf{com}}^*_0, \bar{\mathsf{com}}^*_1, \mathsf{m}^*)$ was defined during the $\hat{i}$-th query to random oracle H_c. If this is not the case, the reduction aborts its execution. Otherwise, it returns $\bar{s}^*_{1-b^*}$ for $b^* = \Gamma(\tilde{\mathsf{pk}}^*, \mathsf{m}^*)$ to the lossy soundness experiment.

Unless the reduction aborts due to wrong guessing of $\hat{i}$, the view of $\mathcal{A}$ is exactly as in $\mathbf{G}_9$. Assuming the reduction does not abort, note that the key $\tilde{\mathsf{pk}}_{1-b^*}$ is not in the image of F (see $\mathbf{G}_7$). Also, by construction,

this is the key that the reduction gave to the lossy soundness game. Due to the change in $\mathbf{G}_9$, we also know that

$$R = r[\tilde{\mathsf{pk}}^*, \mathsf{m}^*, \bar{\mathsf{com}}_{b^*}] = R^*_{1-b^*} = \mathsf{F}(\bar{s}^*_{1-b^*}) - c^* \cdot \tilde{\mathsf{pk}}^*_{1-b^*}.$$

Hence, it is clear that the reduction outputs a valid solution to the lossy soundness experiment. As the view of $\mathcal{A}$ is independent of $\hat{i}$ before a potential abort and due to Lemma 1, we get that

$$|\Pr[\mathbf{G}_9 \Rightarrow 1]| \leq \frac{Q_{\mathsf{H}_c}}{|\mathcal{S}|}.$$

□

5 Instantiation and Efficiency

In the previous section, we have constructed our multi-signature scheme abstractly given a linear function family and a special commitment scheme. We will now instantiate these two building blocks based on the DDH assumption. Essentially, we use the instantiation from [PW23], but we need to show our stronger *oracle-aided* multi-key indistinguishability for it.

5.1 Linear Function Family

We use the well-known [KMP16] linear function family $\mathsf{LF}_{\mathsf{DDH}} = (\mathsf{LF.Gen}, \mathsf{F})$ based on the DDH assumption defined as follows. For that, recall the group generation algorithm GGen that on input 1^λ outputs the description $\mathfrak{G} := (\mathbb{G}, p, G)$ of a prime order group $\mathbb{G}$ of order p with generator G. Then, Gen runs GGen, samples $H \xleftarrow{\$} \mathbb{G}$, and outputs parameters $\mathsf{par} := (\mathfrak{G}, H) = (\mathbb{G}, p, G, H)$. This defines the set of scalars, domain, range, and the function $\mathsf{F}(\mathsf{par}, \cdot)$ as follows:

$$\mathcal{S} := \mathbb{Z}_p, \;\; \mathcal{D} := \mathbb{Z}_p, \;\; \mathcal{R} := \mathbb{G}^2, \;\; \mathsf{F}(\mathsf{par}, x) := x(G, H) = (xG, xH).$$

It is easy to see that this defines a linear function family. Further, it has been shown in [PW23] that $\mathsf{LF}_{\mathsf{DDH}}$ satisfies key indistinguishability from the DDH assumption. We restate the statement from [PW23] and refer to their paper for a proof of it.

Lemma 4 ([PW23])**.** *Assume that the* DDH *assumption holds relative to* GGen*. Then, the linear function family* $\mathsf{LF}_{\mathsf{DDH}}$ *satisfies key indistinguishability. Concretely, for any PPT algorithm* $\mathcal{A}$*, there is a PPT algorithm* $\mathcal{B}$ *with* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$ *and*

$$\mathsf{Adv}^{\mathsf{keydist}}_{\mathcal{A},\mathsf{LF}_{\mathsf{DDH}}}(\lambda) \leq \mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{B},\mathsf{GGen}}(\lambda).$$

5.2 Commitment

We use the commitment scheme $\mathsf{CMT}_{\mathsf{DDH}} = (\mathsf{BGen}, \mathsf{TGen}, \mathsf{Com}, \mathsf{TCom}, \mathsf{TCol})$ for the linear function family $\mathsf{LF}_{\mathsf{DDH}}$ from [PW23]. Let parameters of $\mathsf{LF}_{\mathsf{DDH}}$ be given as above, then the commitment scheme has key space $\mathcal{K} := \mathbb{G}^{3\times 3}$, message space $\mathcal{D} := \mathbb{G}^2$, randomness space $\mathcal{G} := \mathbb{Z}_p^3$, and commitment space $\mathcal{H} := \mathbb{G}^3$. The sets $\mathcal{G}$ and $\mathcal{H}$ are naturally augmented with componentwise group operations. We next describe the algorithms of the commitment scheme verbally.

- $\mathsf{BGen}(\mathsf{par}) \to \mathsf{ck}$: Sample $G_1, G_2, G_3 \xleftarrow{\$} \mathbb{G}$ and $a, b \xleftarrow{\$} \mathbb{Z}_p$, and set

$$\mathsf{ck} := \mathbf{A} := \begin{pmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} := \begin{pmatrix} G_1 & aG_1 & bG_1 \\ G_2 & aG_2 & bG_2 \\ G_3 & aG_3 & bG_3 \end{pmatrix} \in \mathbb{G}^{3\times 3}.$$

- $\mathsf{TGen}(\mathsf{par}, X = (X_1, X_2)) \to (\mathsf{ck}, \mathsf{td})$: Sample $d_{i,j} \xleftarrow{\$} \mathbb{Z}_p$ for all $(i, j) \in [3] \times [3]$, and set

$$\mathsf{ck} := \mathbf{A} := \begin{pmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} := \begin{pmatrix} d_{1,1}G & d_{1,2}G & d_{1,3}G \\ d_{2,1}X_1 & d_{2,2}X_1 & d_{2,3}X_1 \\ d_{3,1}X_2 & d_{3,2}X_2 & d_{3,3}X_2 \end{pmatrix} \in \mathbb{G}^{3\times 3}.$$

Further, set $\mathsf{td} := (\mathbf{D}, X_1, X_2)$ with

$$\mathbf{D} := \begin{pmatrix} d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,1} & d_{2,2} & d_{2,3} \\ d_{3,1} & d_{3,2} & d_{3,3} \end{pmatrix} \in \mathbb{Z}_p^{3\times 3}.$$

- $\mathsf{Com}(\mathsf{ck}, R = (R_1, R_2); \varphi) \to \mathsf{com}$: Let $\varphi = (\alpha, \beta, \gamma) \in \mathbb{Z}_p^3$, and compute[11]

$$\mathsf{com} := \begin{pmatrix} C_0 \\ C_1 \\ C_2 \end{pmatrix} := \begin{pmatrix} \alpha A_{1,1} + \beta A_{1,2} + \gamma A_{1,3} \\ \alpha A_{2,1} + \beta A_{2,2} + \gamma A_{2,3} + R_1 \\ \alpha A_{3,1} + \beta A_{3,2} + \gamma A_{3,3} + R_2 \end{pmatrix} = \mathbf{A} \cdot \varphi + \begin{pmatrix} 0 \\ R_1 \\ R_2 \end{pmatrix} \in \mathbb{G}^3.$$

- $\mathsf{TCom}(\mathsf{ck}, \mathsf{td}) \to (\mathsf{com}, St)$: Sample $\tau, \rho_1, \rho_2, s \xleftarrow{\$} \mathbb{Z}_p$. Set $St := (\mathsf{td}, \tau, \rho_1, \rho_2, s)$ and compute

$$\mathsf{com} := \begin{pmatrix} C_0 \\ C_1 \\ C_2 \end{pmatrix} := \begin{pmatrix} \tau G \\ \rho_1 X_1 + sG \\ \rho_2 X_2 + sH \end{pmatrix} = \begin{pmatrix} \tau G \\ \rho_1 X_1 \\ \rho_2 X_2 \end{pmatrix} + s \begin{pmatrix} 0 \\ G \\ H \end{pmatrix} \in \mathbb{G}^3.$$

- $\mathsf{TCol}(St, c) \to (\varphi, R, s)$: Set $R := (R_1, R_2) := (sG - cX_1, sH - cX_2)$. If matrix $\mathbf{D}$ is not invertible, return $\perp$. Otherwise, compute

$$\varphi := \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} := \mathbf{D}^{-1} \cdot \begin{pmatrix} \tau \\ \rho_1 + c \\ \rho_2 + c \end{pmatrix} \in \mathbb{Z}_p^3.$$

It has been shown in [PW23] that $\mathsf{CMT}_{\mathsf{DDH}}$ satisfies the properties of homomorphism, good parameters, uniform keys, special trapdoor, and statistically binding. Additionally, we need to prove that it also has oracle-aided multi-key indistinguishability, which we do next.

Lemma 5. *Assume that the* DDH *assumption holds relative to* GGen. *Then,* $\mathsf{CMT}_{\mathsf{DDH}}$ *is an* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-special commitment scheme for* $\mathsf{LF}_{\mathsf{DDH}}$ *with* $\varepsilon_{\mathsf{b}} \le 1/p$, $\varepsilon_{\mathsf{g}} \le 2/p$, *and* $\varepsilon_{\mathsf{t}} \le 6/p$. *Concretely, for any PPT algorithm* $\mathcal{A}$, *there is a PPT algorithm* $\mathcal{B}$ *with* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$ *and*

$$\mathsf{Adv}^{Q\text{-keydist}}_{\mathcal{A},\mathsf{CMT}_{\mathsf{DDH}}}(\lambda) \le \mathsf{Adv}^{\mathsf{uDDH3}}_{\mathcal{B},\mathsf{GGen}}(\lambda) + 5/p + 1/p^3.$$

Proof. The homomorphism property is straightforward to check. Now, we define the good parameter set Good exactly as in the proof of Theorem 3 in [PW23] as

$$\mathsf{Good} = \{((G, H), x) \in \mathbb{G}^2 \times \mathbb{Z}_p \mid (G, H) \in \mathsf{LF}_{\mathsf{DDH}}.\mathsf{Gen}(1^\lambda) \wedge H \neq 1 \wedge x \neq 0\}.$$

Having done this, we directly refer to the paper [PW23] for proofs of good parameters, uniform keys, special trapdoor, and statistically binding, with their respective bounds as stated. As such, we now focus on oracle-aided multi-key indistinguishability. To this end, let $Q = \mathsf{poly}(\lambda)$, and we have to bound

$$\left| \Pr\left[Q\text{-}\mathbf{OKEYDIST}^{\mathcal{A}}_{0,\mathsf{CMT}_{\mathsf{DDH}}}(\lambda) \Rightarrow 1\right] - \Pr\left[Q\text{-}\mathbf{OKEYDIST}^{\mathcal{A}}_{1,\mathsf{CMT}_{\mathsf{DDH}}}(\lambda) \Rightarrow 1\right] \right|.$$

In the following, we construct a reduction $\mathcal{B}$ that simulates the oracle-aided multi-key indistinguishability for $\mathcal{A}$ and runs in the uDDH3 game. Our reduction $\mathcal{B}$ works as follows:

1. $\mathcal{B}$ gets as input $\mathfrak{G} = (\mathbb{G}, p, G)$ and group elements $(H_{i,j})_{i,j\in[3]}$.
2. $\mathcal{B}$ samples $h \xleftarrow{\$} \mathbb{Z}_p$, and sets $H := hG$, $\mathsf{par} := (\mathbb{G}, p, G, H)$. Further, it samples $X := (X_1, X_2) \xleftarrow{\$} \mathbb{G}^2$.
3. If $H = 0$, $X_1 = 0$ or $X_2 = 0$, then $\mathcal{B}$ returns 0 and terminates.

[11] We regard vectors as column vectors, unless explicitly stated otherwise.

4. Otherwise, $\mathcal{B}$ defines commitment keys $\mathsf{ck}_i := \mathbf{A}_i \in \mathbb{G}^{3\times 3}$ for all $i \in [Q]$ as follows:
   - Let $\mathbf{H} \in \mathbb{Z}_p^{3\times 3}$ be the matrix of scalars of the $(H_{i,j})$ to base G. Further, let $\mathbf{H}_0 \in \mathbb{Z}_p^{3\times 1}$ be its first column and let $\mathbf{H}_0 \in \mathbb{Z}_p^{3\times 2}$ be the remaining submatrix, i.e., $\mathbf{H} = [\mathbf{H}_0|\mathbf{H}_1]$.
   - It samples $\mathbf{T}_i \xleftarrow{\$} \mathbb{Z}_p^{3\times 3}$ and $\mathbf{S}_i \xleftarrow{\$} \mathbb{Z}_p^{1\times 2}$. With that, we let $\mathbf{D}_i := \mathbf{T}_i\mathbf{H} + [\mathbf{0}|\mathbf{T}_i\mathbf{H}_0\mathbf{S}_i] \in \mathbb{Z}_p^{3\times 3}$.
   - Finally, it computes the commitment key as $\mathbf{A}_i := \mathbf{D}_i G \in \mathbb{G}^{3\times 3}$ (componentwise). One can easily check that $\mathcal{B}$ can efficiently compute $\mathbf{A}_i$, given $\mathbf{T}_i$, $\mathbf{S}_i$, and $(H_{i,j})_{i,j\in[3]}$.
5. $\mathcal{B}$ runs $\mathcal{A}$ on input $(\mathsf{par}, X, (\mathsf{ck}_i)_{i\in[Q]})$ with access to oracle O that it simulates as follows:
   - On query input $Y := (Y_1, Y_2) \in \mathbb{G}^2$, reduction $\mathcal{B}$ computes $\tilde{Y}_2 := hY_1$.
   - Then, it checks if $\tilde{Y}_2 = Y_2$. If so, it returns 1 to $\mathcal{A}$. Otherwise, it returns 0.
6. Finally, $\mathcal{B}$ returns whatever $\mathcal{A}$ returns.

We analyze $\mathcal{B}$'s success probability. For this, we claim that if there are $a, b \in \mathbb{Z}_p$ such that $H_{i,2} = aH_{i,1}$ and $H_{i,3} = bH_{i,1}$ for all $i \in [3]$, then $\mathcal{B}$ provides a simulation statistically close to Q-$\mathbf{OKEYDIST}_{0,\mathsf{CMT}_{\mathsf{DDH}}}$. On the other hand, if the $H_{i,j}$ are uniform and independent, then $\mathcal{B}$ provides a simulation statistically close to Q-$\mathbf{OKEYDIST}_{1,\mathsf{CMT}_{\mathsf{DDH}}}$.

For the first claim, observe that $\mathbf{H}_1$ can then be written as $\mathbf{H}_1 = \mathbf{H}_0\mathbf{R}$ where $\mathbf{R} = [a, b] \in \mathbb{Z}_p^{1\times 2}$. Then, we get the identity

$$\mathbf{D}_i = \mathbf{T}_i\mathbf{H} + [\mathbf{0}|\mathbf{T}_i\mathbf{H}_0\mathbf{S}_i] = [\mathbf{T}_i\mathbf{H}_0|\mathbf{T}_i\mathbf{H}_0(\mathbf{R} + \mathbf{S}_i)].$$

Assuming $\mathbf{H}_0 \in \mathbb{Z}_p^{3\times 1}$ has full rank, which happens with probability at least $1 - 1/p^3$, we find that the commitment key $\mathbf{A}_i$ is distributed exactly as a commitment key in Q-$\mathbf{OKEYDIST}_{0,\mathsf{CMT}_{\mathsf{DDH}}}$.

For the second claim, observe that a uniform $\mathbf{H} \in \mathbb{Z}_p^{3\times 3}$ is invertible with probability at least $1 - 3/p$. Then, assuming $\mathbf{H}$ to be invertible, the matrix $\mathbf{T}_i\mathbf{H} \in \mathbb{Z}_p^{3\times 3}$ is also uniform and $\mathbf{D}_i$ thus also. Finally, assuming $X_1 \neq 0$ and $X_2 \neq 0$, which happens with probability at least $1 - 2/p$, we see that all $\mathbf{A}_i$ are uniform and independent and thus distributed exactly as commitment keys in Q-$\mathbf{OKEYDIST}_{0,\mathsf{CMT}_{\mathsf{DDH}}}$. With that, we conclude the proof. □

5.3 Efficiency

We discuss the efficiency of our scheme. For this, we first explain an optimization from [PW24], which also applies to our scheme, to improve communication complexity. Namely, the commitment randomness $\varphi \in \mathcal{G}$ can be generated from a λ-bit seed via a random oracle $\bar{\mathsf{H}}\colon \{0,1\}^* \to \mathcal{G}$. Then, in the response phase, each signer sends its seed instead of φ, and signers can then locally derive all φ's using $\bar{\mathsf{H}}$ and aggregate them as usual. This modification is secure by the unpredictability of the random oracle. Using this optimization, we compute the communication $|\mathsf{Comm}|$ per signer and the final signature size $|\sigma|$ as

$$|\mathsf{Comm}| = 3\langle\mathbb{G}\rangle + 1\langle\mathbb{Z}_p\rangle + \lambda + 1, \quad |\sigma| = 2\langle\mathbb{G}\rangle + 9\langle\mathbb{Z}_p\rangle.$$

Concrete Efficiency. In Table 2, we compare the concrete efficiency and security of two-round multi-signature schemes in the discrete logarithm setting without pairings in the plain public key model. For that, we assume that all schemes are instantiated with the secp256k1 curve and the SHA-256 hash function. Further, we assume that the underlying hardness assumption is 128-bit hard and that the number of hash and signing queries is $Q_H = 2^{30}$ and $Q_S = 2^{20}$, respectively. To compute the security level, we take the security bounds as provided in the papers. For the verification cost per signature, we count the number of multi-scalar-multiplications in $\mathbb{G}$ and ignore hash evaluations.

Scheme	Pub Key	Communication	Signature	Security	Verification
Musig2 [NRS21]	33	164	65	9	1 msm_2
HBMS [BD21]	33	97	97	-11	1 msm_3
TZ [TZ23]	33	196	97	8	1 msm_3
TSSHO [TSS+23]	66	130	96	106	2 msm_3
Chopsticks I [PW23]	66	147	227	106	3 msm_5
Chopsticks II [PW23]	132	278	$454 + \lceil N/8 \rceil$	126	6 msm_{N+4}
Toothpicks I [PW24]	66	114	128	106	2 msm_4
Toothpicks II [PW24]	132	114	$128 + \lceil N/8 \rceil$	125	2 msm_{N+3}
T-Spoon (Ours)	66	147	354	126	3 $\mathsf{msm}_5 + 3$ msm_6

Table 2: Concrete efficiency and security comparison of two-round multi-signature schemes in the discrete logarithm setting without pairings in the plain public key model. We compare the size of public keys, the communication cost per signer, the verification cost per signature, and the security level given by the security reduction in the random oracle model assuming the underlying assumption is 128-bit hard and $Q_H = 2^{30}, Q_S = 2^{20}$ for the number of hash and signing queries, respectively; all sizes are given in bytes. We omit Musig-DN [NRSW20] from our comparison, as it uses expensive zk-SNARKs. Further, msm_k denotes a multi-scalar-multiplication of size k in $\mathbb{G}$, and N denotes the number of signers.

References

[AAB+24] Marius A. Aardal, Diego F. Aranha, Katharina Boudgoust, Sebastian Kolby, and Akira Takahashi. Aggregating falcon signatures with LaBRADOR. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part I*, volume 14920 of *LNCS*, pages 71–106. Springer, Cham, August 2024. (Cited on Page 5.)

[AB21] Handan Kilinç Alper and Jeffrey Burdges. Two-round trip schnorr multi-signatures via delinearized witnesses. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 157–188, Virtual Event, August 2021. Springer, Cham. (Cited on Page 3, 5.)

[ACL+22] Martin R. Albrecht, Valerio Cini, Russell W. F. Lai, Giulio Malavolta, and Sri Aravinda Krishnan Thyagarajan. Lattice-based SNARKs: Publicly verifiable, preprocessing, and recursively composable - (extended abstract). In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 102–132. Springer, Cham, August 2022. (Cited on Page 5.)

[AFLT12] Michel Abdalla, Pierre-Alain Fouque, Vadim Lyubashevsky, and Mehdi Tibouchi. Tightly-secure signatures from lossy identification schemes. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 572–590. Springer, Berlin, Heidelberg, April 2012. (Cited on Page 5, 6, 13.)

[BCG+23a] Foteini Baldimtsi, Konstantinos Kryptos Chalkias, Francois Garillot, Jonas Lindstrom, Ben Riva, Arnab Roy, Mahdi Sedaghat, Alberto Sonnino, Pun Waiwitlikhit, and Joy Wang. Subset-optimized bls multi-signature with key aggregation. Cryptology ePrint Archive, Paper 2023/498, 2023. https://eprint.iacr.org/2023/498. (Cited on Page 5.)

[BCG+23b] Foteini Baldimtsi, Konstantinos Kryptos Chalkias, Francois Garillot, Jonas Lindstrom, Ben Riva, Arnab Roy, Mahdi Sedaghat, Alberto Sonnino, Pun Waiwitlikhit, and Joy Wang. Subset-optimized BLS multi-signature with key aggregation. Cryptology ePrint Archive, Paper 2023/498, 2023. (Cited on Page 5.)

[BCJ08] Ali Bagherzandi, Jung Hee Cheon, and Stanislaw Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008*, pages 449–458. ACM Press, October 2008. (Cited on Page 5.)

[BD21] Mihir Bellare and Wei Dai. Chain reductions for multi-signatures and the HBMS scheme. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 650–678. Springer, Cham, December 2021. (Cited on Page 3, 4, 5, 25.)

[BDN18] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 435–464. Springer, Cham, December 2018. (Cited on Page 3, 5.)

[BJ08] Ali Bagherzandi and Stanislaw Jarecki. Multisignatures using proofs of secret key possession, as secure as the Diffie-Hellman problem. In Rafail Ostrovsky, Roberto De Prisco, and Ivan Visconti, editors, *SCN 08*, volume 5229 of *LNCS*, pages 218–235. Springer, Berlin, Heidelberg, September 2008. (Cited on Page 4, 5.)

[BLT+24] Renas Bacho, Julian Loss, Stefano Tessaro, Benedikt Wagner, and Chenzhi Zhu. Twinkle: Threshold signatures from DDH with full adaptive security. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part I*, volume 14651 of *LNCS*, pages 429–459. Springer, Cham, May 2024. (Cited on Page 13.)

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, October / November 2006. (Cited on Page 3, 5, 11.)

[BNN07] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Unrestricted aggregate signatures. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *ICALP 2007*, volume 4596 of *LNCS*, pages 411–422. Springer, Berlin, Heidelberg, July 2007. (Cited on Page 5.)

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer, Berlin, Heidelberg, January 2003. (Cited on Page 4, 5.)

[BS23] Ward Beullens and Gregor Seiler. LaBRADOR: Compact proofs for R1CS from module-SIS. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 518–548. Springer, Cham, August 2023. (Cited on Page 5.)

[BTT22] Cecilia Boschini, Akira Takahashi, and Mehdi Tibouchi. MuSig-L: Lattice-based multi-signature with single-round online phase. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 276–305. Springer, Cham, August 2022. (Cited on Page 5.)

[BW24] Renas Bacho and Benedikt Wagner. Tightly secure non-interactive BLS multi-signatures. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part II*, volume 15485 of *LNCS*, pages 397–422. Springer, Singapore, December 2024. (Cited on Page 5.)

[Che23] Yanbo Chen. DualMS: Efficient lattice-based two-round multi-signature with trapdoor-free simulation. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 716–747. Springer, Cham, August 2023. (Cited on Page 5.)

[CKM21] Elizabeth Crites, Chelsea Komlo, and Mary Maller. How to prove schnorr assuming schnorr: Security of multi- and threshold signatures. Cryptology ePrint Archive, Report 2021/1375, 2021. (Cited on Page 3, 4, 5, 11.)

[CP93] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer, Berlin, Heidelberg, August 1993. (Cited on Page 6.)

[DEF+19] Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs. On the security of two-round multi-signatures. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1084–1101, 2019. (Cited on Page 4, 5.)

[DFMS24] Giuseppe D'Alconzo, Andrea Flamini, Alessio Meneghetti, and Edoardo Signorini. A framework for group action-based multi-signatures and applications to LESS, MEDS, and ALTEQ. Cryptology ePrint Archive, Paper 2024/1691, 2024. (Cited on Page 5.)

[DGKV22] Lalita Devadas, Rishab Goyal, Yael Kalai, and Vinod Vaikuntanathan. Rate-1 non-interactive arguments for batch-NP and applications. In *63rd FOCS*, pages 1057–1068. IEEE Computer Society Press, October / November 2022. (Cited on Page 5.)

[DGNW20] Manu Drijvers, Sergey Gorbunov, Gregory Neven, and Hoeteck Wee. Pixel: Multi-signatures for consensus. In Srdjan Capkun and Franziska Roesner, editors, *USENIX Security 2020*, pages 2093–2110. USENIX Association, August 2020. (Cited on Page 5.)

[DKKW25] Justin Drake, Dmitry Khovratovich, Mikhail Kudinov, and Benedikt Wagner. Hash-based multi-signatures for post-quantum ethereum. Cryptology ePrint Archive, Paper 2025/055, 2025. (Cited on Page 5.)

[DOTT21] Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 99–130. Springer, Cham, May 2021. (Cited on Page 5, 6, 11.)

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Berlin, Heidelberg, August 2013. (Cited on Page 10.)

[FH19] Masayuki Fukumitsu and Shingo Hasegawa. A tightly-secure lattice-based multisignature. In *Proceedings of the 6th on ASIA Public-Key Cryptography Workshop*, APKC '19, page 3–11, New York, NY, USA, 2019. Association for Computing Machinery. (Cited on Page 5.)

[FH20a] Masayuki Fukumitsu and Shingo Hasegawa. A lattice-based provably secure multisignature scheme in quantum random oracle model. In Khoa Nguyen, Wenling Wu, Kwok Yan Lam, and Huaxiong Wang, editors, *Provable and Practical Security*, pages 45–64, Cham, 2020. Springer International Publishing. (Cited on Page 5.)

[FH20b] Masayuki Fukumitsu and Shingo Hasegawa. A tightly secure ddh-based multisignature with public-key aggregation. In *2020 Eighth International Symposium on Computing and Networking Workshops (CANDARW)*, pages 321–327, 2020. (Cited on Page 3, 5.)

[FHSZ23] Nils Fleischhacker, Gottfried Herold, Mark Simkin, and Zhenfei Zhang. Chipmunk: Better synchronized multi-signatures from lattices. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 386–400. ACM Press, November 2023. (Cited on Page 5.)

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Cham, August 2018. (Cited on Page 3, 4.)

[FSZ22] Nils Fleischhacker, Mark Simkin, and Zhenfei Zhang. Squirrel: Efficient synchronized multi-signatures from lattices. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 1109–1123. ACM Press, November 2022. (Cited on Page 5.)

[GJKW07] Eu-Jin Goh, Stanislaw Jarecki, Jonathan Katz, and Nan Wang. Efficient signature schemes with tight reductions to the Diffie-Hellman problems. *Journal of Cryptology*, 20(4):493–514, October 2007. (Cited on Page 7.)

[Har94] Lein Harn. Group-oriented (mi) threshold signature and digital multisignature. *1EE Proceedings of Computers and Digital Techniques*, 141(5):307–313, 1994. (Cited on Page 4.)

[HMP95] Patrick Horster, Markus Michels, and Holger Petersen. *Meta-Multisignature schemes based on the discrete logarithm problem*, pages 128–142. Springer US, Boston, MA, 1995. (Cited on Page 4.)

[IN83] Kazuharu Itakura and Katsuhiro Nakamura. A public-key cryptosystem suitable for digital multisignatures. *NEC Research & Development*, (71):1–8, 1983. (Cited on Page 3, 4, 5.)

[KMP16] Eike Kiltz, Daniel Masny, and Jiaxin Pan. Optimal security proofs for signatures from identification schemes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 33–61. Springer, Berlin, Heidelberg, August 2016. (Cited on Page 22.)

[Lan96] Susan K. Langford. Weakness in some threshold cryptosystems. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 74–82. Springer, Berlin, Heidelberg, August 1996. (Cited on Page 4.)

[LBG09] Duc-Phong Le, Alexis Bonnecaze, and Alban Gabillon. Multisignatures as secure as the diffie-hellman problem in the plain public-key model. In Hovav Shacham and Brent Waters, editors, *Pairing-Based Cryptography – Pairing 2009*, pages 35–51, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. (Cited on Page 5.)

[LHL95] Chuan-Ming Li, Tzonelih Hwang, and Narn-Yih Lee. Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders. In Alfredo De Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 194–204. Springer, Berlin, Heidelberg, May 1995. (Cited on Page 4.)

[LK23] Kwangsu Lee and Hyoseung Kim. Two-round multi-signatures from okamoto signatures. *Mathematics*, 11:3223, 07 2023. (Cited on Page 5.)

[LLL+24] Qiqi Lai, Feng-Hao Liu, Yang Lu, Haiyang Xue, and Yong Yu. Scalable two-round n-out-of-n and multi-signatures from lattices in the quantum random oracle model. Cryptology ePrint Archive, Paper 2024/1574, 2024. (Cited on Page 5.)

[LOS+06] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 465–485. Springer, Berlin, Heidelberg, May / June 2006. (Cited on Page 4, 5.)

[LWZ+23] Xiao Liang, Xiaohui Wang, Qianyi Zhang, Shuai Yuan, and Zhitao Guan. A lattice-based multisignature scheme for blockchain-enabled systems. In Wei Quan, editor, *Emerging Networking Architecture and Technologies*, pages 336–346, Singapore, 2023. Springer Nature Singapore. (Cited on Page 5.)

[MH96] Markus Michels and Patrick Horster. On the risk of disruption in several multiparty signature schemes. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology — ASIACRYPT '96*, pages 334–345, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. (Cited on Page 4.)

[MOR01] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. Accountable-subgroup multisignatures: Extended abstract. In Michael K. Reiter and Pierangela Samarati, editors, *ACM CCS 2001*, pages 245–254. ACM Press, November 2001. (Cited on Page 4, 5.)

[MPSW18] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple schnorr multi-signatures with applications to bitcoin. Cryptology ePrint Archive, Report 2018/068, 2018. (Cited on Page 3.)

[MPSW19] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple schnorr multi-signatures with applications to bitcoin. *DCC*, 87(9):2139–2164, 2019. (Cited on Page 3, 5.)

[NRS21] Jonas Nick, Tim Ruffing, and Yannick Seurin. MuSig2: Simple two-round Schnorr multi-signatures. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 189–221, Virtual Event, August 2021. Springer, Cham. (Cited on Page 3, 4, 5, 25.)

[NRSW20] Jonas Nick, Tim Ruffing, Yannick Seurin, and Pieter Wuille. MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1717–1731. ACM Press, November 2020. (Cited on Page 4, 5, 25.)

[OO93] Kazuo Ohta and Tatsuaki Okamoto. A digital multisignature scheme based on the Fiat-Shamir scheme. In Hideki Imai, Ronald L. Rivest, and Tsutomu Matsumoto, editors, *ASIACRYPT'91*, volume 739 of *LNCS*, pages 139–148. Springer, Berlin, Heidelberg, November 1993. (Cited on Page 4, 5.)

[OO98] Kazuo Ohta and Tatsuaki Okamoto. On concrete security treatment of signatures derived from identification. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 354–369. Springer, Berlin, Heidelberg, August 1998. (Cited on Page 4.)

[OO99] Kazuo Ohta and Tatsuaki Okamoto. Multi-signature schemes secure against active insider attacks (special section on cryptography and information security). *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 82:21–31, 1999. (Cited on Page 4.)

[PFH+20] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions. (Cited on Page 5.)

[PW23] Jiaxin Pan and Benedikt Wagner. Chopsticks: Fork-free two-round multi-signatures from non-interactive assumptions. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 597–627. Springer, Cham, April 2023. (Cited on Page 3, 4, 5, 6, 7, 9, 10, 11, 13, 14, 15, 16, 22, 23, 25, 31.)

[PW24] Jiaxin Pan and Benedikt Wagner. Toothpicks: More efficient fork-free two-round multi-signatures. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part I*, volume 14651 of *LNCS*, pages 460–489. Springer, Cham, May 2024. (Cited on Page 3, 4, 5, 6, 7, 9, 10, 11, 13, 14, 16, 24, 25.)

[QLH12] Haifeng Qian, Xiangxue Li, and Xinli Huang. Tightly secure non-interactive multisignatures in the plain public key model. *Informatica (Vilnius)*, 3, 01 2012. (Cited on Page 5.)

[QX10] Haifeng Qian and Shouhuai Xu. Non-interactive multisignatures in the plain public-key model with efficient verification. *Information Processing Letters*, 111(2):82–89, 2010. (Cited on Page 5.)

[RL24] Matthieu Rambaud and Christophe Levrat. Practical non-interactive multi-signatures, and a multi-to-aggregate signatures compiler. Cryptology ePrint Archive, Paper 2024/1081, 2024. (Cited on Page 5.)

[RSY25] Lior Rotem, Gil Segev, and Eylon Yogev. From one-time to two-round reusable multi-signatures without nested forking. In Elette Boyle and Mohammad Mahmoody, editors, *Theory of Cryptography*, pages 371–399, Cham, 2025. Springer Nature Switzerland. (Cited on Page 5.)

[RY07] Thomas Ristenpart and Scott Yilek. The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 228–245. Springer, Berlin, Heidelberg, May 2007. (Cited on Page 5.)

[TS23] Toi Tomita and Junji Shikata. Compact aggregate signature from module-lattices. Cryptology ePrint Archive, Report 2023/471, 2023. (Cited on Page 5.)

[TSS+23] Kaoru Takemure, Yusuke Sakai, Bagus Santoso, Goichiro Hanaoka, and Kazuo Ohta. More efficient two-round multi-signature scheme with provably secure parameters. Cryptology ePrint Archive, Report 2023/155, 2023. (Cited on Page 3, 4, 5, 6, 7, 10, 25.)

[TZ23] Stefano Tessaro and Chenzhi Zhu. Threshold and multi-signature schemes from linear hash functions. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 628–658. Springer, Cham, April 2023. (Cited on Page 3, 4, 5, 25.)

[WW22] Brent Waters and David J. Wu. Batch arguments for NP and more from standard bilinear group assumptions. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 433–463. Springer, Cham, August 2022. (Cited on Page 5.)

Supplementary Material

A Postponed Proofs

Proof of Lemma 5. The homomorphism property is straightforward to check. Now, we define the good parameter set Good exactly as in the proof of Theorem 3 in [PW23] as

$$\mathsf{Good} = \{((G,H),x) \in \mathbb{G}^2 \times \mathbb{Z}_p \mid (G,H) \in \mathsf{LF}_{\mathsf{DDH}}.\mathsf{Gen}(1^\lambda) \wedge H \neq 1 \wedge x \neq 0\}.$$

Having done this, we directly refer to the paper [PW23] for proofs of good parameters, uniform keys, special trapdoor, and statistically binding, with their respective bounds as stated. As such, we now focus on oracle-aided multi-key indistinguishability. To this end, let $Q = \mathsf{poly}(\lambda)$, and we have to bound

$$\left| \Pr\left[Q\text{-}\mathbf{OKEYDIST}^{\mathcal{A}}_{0,\mathsf{CMT}_{\mathsf{DDH}}}(\lambda) \Rightarrow 1\right] - \Pr\left[Q\text{-}\mathbf{OKEYDIST}^{\mathcal{A}}_{1,\mathsf{CMT}_{\mathsf{DDH}}}(\lambda) \Rightarrow 1\right] \right|.$$

In the following, we construct a reduction $\mathcal{B}$ that simulates the oracle-aided multi-key indistinguishability for $\mathcal{A}$ and runs in the uDDH3 game. Our reduction $\mathcal{B}$ works as follows:

1. $\mathcal{B}$ gets as input $\mathfrak{G} = (\mathbb{G}, p, G)$ and group elements $(H_{i,j})_{i,j\in[3]}$.
2. $\mathcal{B}$ samples $h \xleftarrow{\$} \mathbb{Z}_p$, and sets $H := hG$, $\mathsf{par} := (\mathbb{G}, p, G, H)$. Further, it samples $X := (X_1, X_2) \xleftarrow{\$} \mathbb{G}^2$.
3. If $H = 0$, $X_1 = 0$ or $X_2 = 0$, then $\mathcal{B}$ returns 0 and terminates.
4. Otherwise, $\mathcal{B}$ defines commitment keys $\mathsf{ck}_i := \mathbf{A}_i \in \mathbb{G}^{3\times 3}$ for all $i \in [Q]$ as follows:
   - Let $\mathbf{H} \in \mathbb{Z}_p^{3\times 3}$ be the matrix of scalars of the $(H_{i,j})$ to base G. Further, let $\mathbf{H}_0 \in \mathbb{Z}_p^{3\times 1}$ be its first column and let $\mathbf{H}_0 \in \mathbb{Z}_p^{3\times 2}$ be the remaining submatrix, i.e., $\mathbf{H} = [\mathbf{H}_0|\mathbf{H}_1]$.
   - It samples $\mathbf{T}_i \xleftarrow{\$} \mathbb{Z}_p^{3\times 3}$ and $\mathbf{S}_i \xleftarrow{\$} \mathbb{Z}_p^{1\times 2}$. With that, we let $\mathbf{D}_i := \mathbf{T}_i\mathbf{H} + [\mathbf{0}|\mathbf{T}_i\mathbf{H}_0\mathbf{S}_i] \in \mathbb{Z}_p^{3\times 3}$.
   - Finally, it computes the commitment key as $\mathbf{A}_i := \mathbf{D}_i G \in \mathbb{G}^{3\times 3}$ (componentwise). One can easily check that $\mathcal{B}$ can efficiently compute $\mathbf{A}_i$, given $\mathbf{T}_i$, $\mathbf{S}_i$, and $(H_{i,j})_{i,j\in[3]}$.
5. $\mathcal{B}$ runs $\mathcal{A}$ on input $(\mathsf{par}, X, (\mathsf{ck}_i)_{i\in[Q]})$ with access to oracle O that it simulates as follows:
   - On query input $Y := (Y_1, Y_2) \in \mathbb{G}^2$, reduction $\mathcal{B}$ computes $\tilde{Y}_2 := hY_1$.
   - Then, it checks if $\tilde{Y}_2 = Y_2$. If so, it returns 1 to $\mathcal{A}$. Otherwise, it returns 0.
6. Finally, $\mathcal{B}$ returns whatever $\mathcal{A}$ returns.

We analyze $\mathcal{B}$'s success probability. For this, we claim that if there are $a, b \in \mathbb{Z}_p$ such that $H_{i,2} = aH_{i,1}$ and $H_{i,3} = bH_{i,1}$ for all $i \in [3]$, then $\mathcal{B}$ provides a simulation statistically close to Q-$\mathbf{OKEYDIST}_{0,\mathsf{CMT}_{\mathsf{DDH}}}$. On the other hand, if the $H_{i,j}$ are uniform and independent, then $\mathcal{B}$ provides a simulation statistically close to Q-$\mathbf{OKEYDIST}_{1,\mathsf{CMT}_{\mathsf{DDH}}}$.

For the first claim, observe that $\mathbf{H}_1$ can then be written as $\mathbf{H}_1 = \mathbf{H}_0\mathbf{R}$ where $\mathbf{R} = [a, b] \in \mathbb{Z}_p^{1\times 2}$. Then, we get the identity

$$\mathbf{D}_i = \mathbf{T}_i\mathbf{H} + [\mathbf{0}|\mathbf{T}_i\mathbf{H}_0\mathbf{S}_i] = [\mathbf{T}_i\mathbf{H}_0|\mathbf{T}_i\mathbf{H}_0(\mathbf{R} + \mathbf{S}_i)].$$

Assuming $\mathbf{H}_0 \in \mathbb{Z}_p^{3\times 1}$ has full rank, which happens with probability at least $1 - 1/p^3$, we find that the commitment key $\mathbf{A}_i$ is distributed exactly as a commitment key in Q-$\mathbf{OKEYDIST}_{0,\mathsf{CMT}_{\mathsf{DDH}}}$.

For the second claim, observe that a uniform $\mathbf{H} \in \mathbb{Z}_p^{3\times 3}$ is invertible with probability at least $1 - 3/p$. Then, assuming $\mathbf{H}$ to be invertible, the matrix $\mathbf{T}_i\mathbf{H} \in \mathbb{Z}_p^{3\times 3}$ is also uniform and $\mathbf{D}_i$ thus also. Finally, assuming $X_1 \neq 0$ and $X_2 \neq 0$, which happens with probability at least $1 - 2/p$, we see that all $\mathbf{A}_i$ are uniform and independent and thus distributed exactly as commitment keys in Q-$\mathbf{OKEYDIST}_{0,\mathsf{CMT}_{\mathsf{DDH}}}$. With that, we conclude the proof. □

Alg $\mathsf{Setup}(1^\lambda)$

$\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda)$ **return** par

Alg $\mathsf{Gen}(\mathsf{par})$

$x \xleftarrow{\$} \mathcal{D},\ \mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$

$\mathsf{sk} := (x, \mathsf{seed}),\ \mathsf{pk} := \mathsf{F}(x)$

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

Alg $\mathsf{Agg}(\mathcal{P})$

parse $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\} := \mathcal{P}$

for $i \in [N] : a_i := \mathsf{H}_a(\langle \mathcal{P} \rangle, \mathsf{pk}_i)$

return $\tilde{\mathsf{pk}} := \sum_{i=1}^{N} a_i \cdot \mathsf{pk}_i$

Alg $\mathsf{Sig}_0(\mathcal{P}, \mathsf{sk}_1, \mathsf{m})$

parse $(x_1, \mathsf{seed}_1) := \mathsf{sk}_1$

$\tilde{\mathsf{pk}} := \mathsf{Agg}(\mathcal{P})$

$\mathsf{ck}_0 := \mathsf{H}(0, \tilde{\mathsf{pk}}, \mathsf{m})$

$\mathsf{ck}_1 := \mathsf{H}(1, \tilde{\mathsf{pk}}, \mathsf{m})$

$b_1 := \mathsf{H}_b(\mathsf{seed}_1, \tilde{\mathsf{pk}}, \mathsf{m})$

$r_1 \xleftarrow{\$} \mathcal{D},\ \varphi_1 \xleftarrow{\$} \mathcal{G}$

$\mathsf{com}_1 := \mathsf{Com}(\mathsf{ck}_{b_1}, R_1; \varphi_1)$

$\mathsf{pm}_{1,1} := (b_1, \mathsf{com}_1)$

$St_1 := (\tilde{\mathsf{pk}}, \mathsf{sk}_1, r_1, \varphi_1, \mathsf{m})$

return $(\mathsf{pm}_{1,1}, St_1)$

Alg $\mathsf{Sig}_1(St_1, \mathcal{M}_1)$

parse $(\tilde{\mathsf{pk}}, \mathsf{sk}_1, r_1, \varphi_1, \mathsf{m}) := St_1$

parse $(x_1, \mathsf{seed}_1) := \mathsf{sk}_1$

parse $((b_i, \mathsf{com}_i))_{i=1}^N := \mathcal{M}_1$

$\mathcal{I}_0 := \{i \in [N] \mid b_i = 0\}$

$\mathcal{I}_1 := \{i \in [N] \mid b_i = 1\}$

$\bar{\mathsf{com}}_0 := \bigotimes_{i \in \mathcal{I}_0} \mathsf{com}_i$

$\bar{\mathsf{com}}_1 := \bigotimes_{i \in \mathcal{I}_1} \mathsf{com}_i$

for $i \in [N] : a_i := \mathsf{H}_a(\langle \mathcal{P} \rangle, \mathsf{pk}_i)$

$\tilde{\mathsf{pk}}_0 := \sum_{i \in \mathcal{I}_0} a_i \cdot \mathsf{pk}_i$

$\tilde{\mathsf{pk}}_1 := \tilde{\mathsf{pk}} - \tilde{\mathsf{pk}}_0$

$c := \mathsf{H}_c(\tilde{\mathsf{pk}}, \tilde{\mathsf{pk}}_0, \bar{\mathsf{com}}_0, \bar{\mathsf{com}}_1, \mathsf{m})$

$a_1 := \mathsf{H}_a(\langle \mathcal{P} \rangle, \mathsf{pk}_1)$

$s_1 := c \cdot a_1 x_1 + r_1$

$\mathsf{pm}_{2,1} := (s_1, \varphi_1)$

$St_2 := (\mathcal{I}_0, \mathcal{I}_1, c, \tilde{\mathsf{pk}}_0)$

return $(\mathsf{pm}_{2,1}, St_2)$

Alg $\mathsf{Sig}_2(St_2, \mathcal{M}_2)$

parse $(\mathcal{I}_0, \mathcal{I}_1, c, \tilde{\mathsf{pk}}_0) := St_2$

parse $((s_i, \varphi_i))_{i=1}^N := \mathcal{M}_2$

$\bar{s}_0 := \sum_{i \in \mathcal{I}_0} s_i$

$\bar{s}_1 := \sum_{i \in \mathcal{I}_1} s_i$

$\bar{\varphi}_0 := \bigoplus_{i \in \mathcal{I}_0} \varphi_i$

$\bar{\varphi}_1 := \bigoplus_{i \in \mathcal{I}_1} \varphi_i$

$\sigma_0 := (\bar{s}_0, \bar{\varphi}_0)$

$\sigma_1 := (\bar{s}_1, \bar{\varphi}_1)$

return $\sigma := (\tilde{\mathsf{pk}}_0, \sigma_0, \sigma_1, c)$

Alg $\mathsf{VerAgg}(\tilde{\mathsf{pk}}, \mathsf{m}, \sigma)$

parse $(\tilde{\mathsf{pk}}_0, \sigma_0, \sigma_1, c) := \sigma$

parse $(\bar{s}_0, \bar{\varphi}_0) := \sigma_0$

parse $(\bar{s}_1, \bar{\varphi}_1) := \sigma_1$

$\mathsf{ck}_0 := \mathsf{H}(0, \tilde{\mathsf{pk}}, \mathsf{m})$

$\mathsf{ck}_1 := \mathsf{H}(1, \tilde{\mathsf{pk}}, \mathsf{m})$

$\bar{\mathsf{com}}_0 := \mathsf{Com}(\mathsf{ck}_0, \mathsf{F}(\bar{s}_0) - c \cdot \tilde{\mathsf{pk}}_0; \bar{\varphi}_0)$

$\bar{\mathsf{com}}_1 := \mathsf{Com}(\mathsf{ck}_1, \mathsf{F}(\bar{s}_1) - c \cdot (\tilde{\mathsf{pk}} - \tilde{\mathsf{pk}}_0); \bar{\varphi}_1)$

if $c = \mathsf{H}_c(\tilde{\mathsf{pk}}, \tilde{\mathsf{pk}}_0, \bar{\mathsf{com}}_0, \bar{\mathsf{com}}_1, \mathsf{m})$:

  return 1

return 0

Alg $\mathsf{Ver}(\mathcal{P}, \mathsf{m}, \sigma)$

$\tilde{\mathsf{pk}} := \mathsf{Agg}(\mathcal{P})$

return $\mathsf{VerAgg}(\tilde{\mathsf{pk}}, \mathsf{m}, \sigma)$

Figure 4: Our two-round multi-signature scheme $\mathsf{T\text{-}Spoon}[\mathsf{LF}, \mathsf{CMT}]$ for a linear function family $\mathsf{LF} = (\mathsf{LF.Gen}, \mathsf{F})$ and a special commitment scheme $\mathsf{CMT} = (\mathsf{BGen}, \mathsf{TGen}, \mathsf{Com}, \mathsf{TCom}, \mathsf{TCol})$. Here, $\mathsf{H}\colon \{0,1\}^* \to \mathcal{K}$, $\mathsf{H}_a\colon \{0,1\}^* \to \mathcal{S}$, $\mathsf{H}_b\colon \{0,1\}^* \to \{0,1\}$, and $\mathsf{H}_c\colon \{0,1\}^* \to \mathcal{S}$ are random oracles.

Game $\mathbf{G}_0, \ldots, \mathbf{G}_9$

$\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda)$, $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$
$x \xleftarrow{\$} \mathcal{D}$ // $\mathbf{G}_0$-$\mathbf{G}_4$
if $(\mathsf{par}, x) \notin \mathsf{Good}$: **return** 0 // $\mathbf{G}_1$-$\mathbf{G}_3$
$\mathsf{pk} := X := \mathsf{F}(x)$ // $\mathbf{G}_0$-$\mathbf{G}_4$
$\mathsf{pk} := X \xleftarrow{\$} \mathcal{R}$ // $\mathbf{G}_5$-
$(\mathcal{P}^*, \mathsf{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathrm{SIG}_0, \mathrm{SIG}_1}(\mathsf{par}, \mathsf{pk})$
if $\mathsf{pk} \notin \mathcal{P}^*$: **return** 0
if $(\mathcal{P}^*, \mathsf{m}^*) \in \mathsf{Queried}$: **return** 0
parse $\{\mathsf{pk}_1^*, \ldots, \mathsf{pk}_N^*\} := \mathcal{P}$
for $i \in [N] : a_i^* := \mathsf{H}_a(\langle \mathcal{P}^* \rangle, \mathsf{pk}_i^*)$
$\tilde{\mathsf{pk}}^* := \sum_{i=1}^N a_i^* \cdot \mathsf{pk}_i^*$
if $h_b[\mathsf{seed}, \tilde{\mathsf{pk}}^*, \mathsf{m}^*] \neq \bot$: **return** 0 // $\mathbf{G}_1$-
parse $(\tilde{\mathsf{pk}}_0^*, \sigma_0^*, \sigma_1^*, c^*) := \sigma^*$
parse $(\bar{s}_0^*, \bar{\varphi}_0^*) := \sigma_0^*$, $(\bar{s}_1^*, \bar{\varphi}_1^*) := \sigma_1^*$
$\tilde{\mathsf{pk}}_1^* := \tilde{\mathsf{pk}}^* - \tilde{\mathsf{pk}}_0^*$
if $\tilde{\mathsf{pk}}^* \in \mathsf{F}(\mathcal{D})$: **return** 0 // $\mathbf{G}_6$-
$b^* := \Gamma(\tilde{\mathsf{pk}}^*, \mathsf{m}^*)$ // $\mathbf{G}_7$-
if $\tilde{\mathsf{pk}}_{1-b^*}^* \in \mathsf{F}(\mathcal{D})$: **return** 0 // $\mathbf{G}_7$-
$\mathsf{ck}_0^* := \mathsf{H}(0, \tilde{\mathsf{pk}}^*, \mathsf{m}^*)$, $\mathsf{ck}_1^* := \mathsf{H}(1, \tilde{\mathsf{pk}}^*, \mathsf{m}^*)$
$\bar{\mathsf{com}}_0^* := \mathsf{Com}(\mathsf{ck}_0^*, \mathsf{F}(\bar{s}_0^*) - c^* \cdot \tilde{\mathsf{pk}}_0^*; \bar{\varphi}_0^*)$
$\bar{\mathsf{com}}_1^* := \mathsf{Com}(\mathsf{ck}_1^*, \mathsf{F}(\bar{s}_1^*) - c^* \cdot \tilde{\mathsf{pk}}_1^*; \bar{\varphi}_1^*)$
if $c^* \neq \mathsf{H}_c(\tilde{\mathsf{pk}}^*, \tilde{\mathsf{pk}}_0^*, \bar{\mathsf{com}}_0^*, \bar{\mathsf{com}}_1^*, \mathsf{m}^*)$: **return** 0
$R_{1-b^*}^* := \mathsf{F}(\bar{s}_{1-b^*}^*) - c^* \cdot \tilde{\mathsf{pk}}_{1-b}^*$ // $\mathbf{G}_9$-
if $R_{1-b^*}^* \neq r[\tilde{\mathsf{pk}}^*, \mathsf{m}^*, \bar{\mathsf{com}}_b^*]$: **return** 0 // $\mathbf{G}_9$-
return 1

Oracle $\mathrm{SIG}_0(\mathcal{P}, \mathsf{m})$

parse $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\} := \mathcal{P}$
if $\mathsf{pk}_1 \neq \mathsf{pk}$: **return** $\bot$
$\mathsf{Queried} := \mathsf{Queried} \cup \{(\mathcal{P}, \mathsf{m})\}$
$ctr := ctr + 1$, $sid := ctr$, $\mathsf{round}[sid] := 1$
parse $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\} := \mathcal{P}$
for $i \in [N] : a_i := \mathsf{H}_a(\langle \mathcal{P} \rangle, \mathsf{pk}_i)$
$\tilde{\mathsf{pk}} := \sum_{i=1}^N a_i \cdot \mathsf{pk}_i$
$\mathsf{ck}_0 := \mathsf{H}(0, \tilde{\mathsf{pk}}, \mathsf{m})$, $\mathsf{ck}_1 := \mathsf{H}(1, \tilde{\mathsf{pk}}, \mathsf{m})$
$b_1 := \Gamma(\tilde{\mathsf{pk}}, \mathsf{m})$
$r_1 \xleftarrow{\$} \mathcal{D}$, $\varphi_1 \xleftarrow{\$} \mathcal{G}$ // $\mathbf{G}_0$-$\mathbf{G}_2$
$\mathsf{com}_1 := \mathsf{Com}(\mathsf{ck}_{b_1}, R_1; \varphi_1)$ // $\mathbf{G}_0$-$\mathbf{G}_2$
$St_1[sid] := (\tilde{\mathsf{pk}}, r_1, \varphi_1, \mathsf{m})$ // $\mathbf{G}_0$-$\mathbf{G}_2$
$\mathsf{td} := tr[b_1, \tilde{\mathsf{pk}}, \mathsf{m}]$ // $\mathbf{G}_3$-
$(\mathsf{com}_1, St) \leftarrow \mathsf{TCom}(\mathsf{ck}, \mathsf{td})$ // $\mathbf{G}_3$-
$St_1[sid] := (\tilde{\mathsf{pk}}, St, \mathsf{m})$ // $\mathbf{G}_3$-
return $(\mathsf{pm}_1[sid] := (b_1, \mathsf{com}_1), sid)$

Oracle $\mathrm{SIG}_1(sid, \mathcal{M}_1)$

if $\mathsf{round}[sid] \neq 1$: **return** $\bot$
parse $(\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N}) := \mathcal{M}_1$
if $\mathsf{pm}_1[sid] \neq \mathsf{pm}_{1,1}$: **return** $\bot$
$\mathsf{round}[sid] := \mathsf{round}[sid] + 1$
parse $(\tilde{\mathsf{pk}}, r_1, \varphi_1, \mathsf{m}) := St_1[sid]$ // $\mathbf{G}_0$-$\mathbf{G}_2$
parse $(\tilde{\mathsf{pk}}, St, \mathsf{m}) := St_1[sid]$ // $\mathbf{G}_3$-
parse $((b_i, \mathsf{com}_i))_{i=1}^N := \mathcal{M}_1$
$\mathcal{I}_0 := \{i \in [N] \mid b_i = 0\}$
$\mathcal{I}_1 := \{i \in [N] \mid b_i = 1\}$
$\bar{\mathsf{com}}_0 := \bigotimes_{i \in \mathcal{I}_0} \mathsf{com}_i$, $\bar{\mathsf{com}}_1 := \bigotimes_{i \in \mathcal{I}_1} \mathsf{com}_i$
for $i \in [N] : a_i := \mathsf{H}_a(\langle \mathcal{P} \rangle, \mathsf{pk}_i)$
$\tilde{\mathsf{pk}}_0 := \sum_{i \in \mathcal{I}_0} a_i \cdot \mathsf{pk}_i$, $\tilde{\mathsf{pk}}_1 := \tilde{\mathsf{pk}} - \tilde{\mathsf{pk}}_0$
$c := \mathsf{H}_c(\tilde{\mathsf{pk}}, \tilde{\mathsf{pk}}_0, \bar{\mathsf{com}}_0, \bar{\mathsf{com}}_1, \mathsf{m})$
$a_1 := \mathsf{H}_a(\langle \mathcal{P} \rangle, \mathsf{pk}_1)$
$s_1 := c \cdot a_1 x_1 + r_1$ // $\mathbf{G}_0$-$\mathbf{G}_2$
$(\varphi_1, R_1, s_1) \leftarrow \mathsf{TCol}(St, c \cdot a_1)$ // $\mathbf{G}_3$-
$St_2 := (\mathcal{I}_0, \mathcal{I}_1, c, \tilde{\mathsf{pk}}_0)$
return $\mathsf{pm}_2[sid] := (s_1, \varphi_1)$

Figure 5: Games $\mathbf{G}_0, \ldots, \mathbf{G}_9$ in the proof of Theorem 1. Lines with highlighted lines are only executed in the respective games, and $\mathbf{G}_i$- denotes that the line is executed in every game from $\mathbf{G}_i$ to the final game $\mathbf{G}_9$. Random oracles are given in Figure 6. Here, $\Gamma\colon \{0,1\}^* \to \{0,1\}$ is a random oracle that the game holds internally and that is not provided to $\mathcal{A}$.

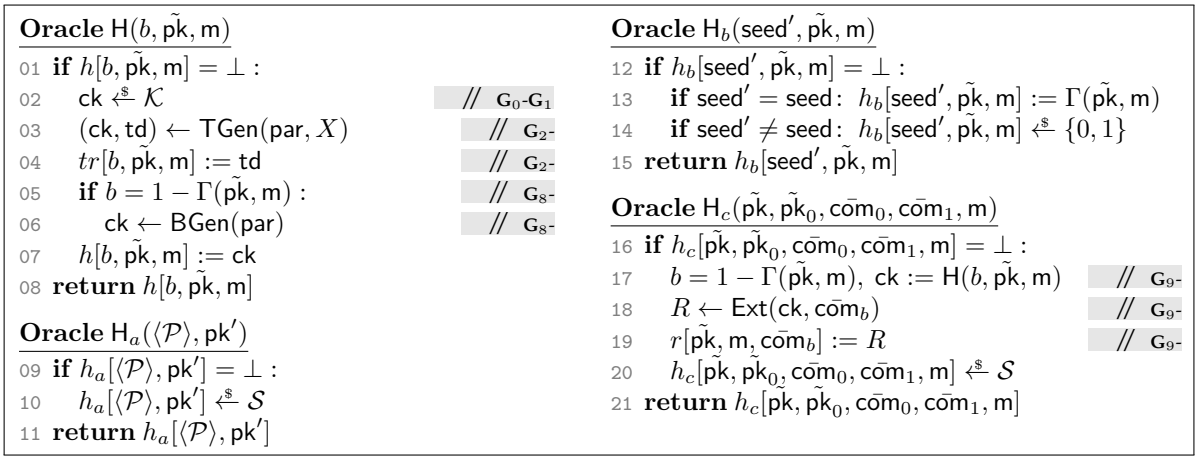

Figure 6: Random oracles in the proof of Theorem 1. The remaining parts of the game are given in Figure 5. Lines with highlighted lines are only executed in the respective games, and $\mathbf{G}_i$- denotes that the line is executed in every game from $\mathbf{G}_i$ to the final game $\mathbf{G}_9$. Here, $\Gamma\colon \{0,1\}^* \to \{0,1\}$ is a random oracle that the game holds internally and that is not provided to $\mathcal{A}$.