Chopsticks: Fork-Free Two-Round Multi-Signatures from Non-Interactive Assumptions

Jiaxin Pan*[1] Benedikt Wagner [2,3]

May 4, 2023

[1] NTNU – Norwegian University of Science and Technology, Trondheim, Norway
jiaxin.pan@ntnu.no
[2] CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
benedikt.wagner@cispa.de
[3] Saarland University, Saarbrücken, Germany

Abstract

Multi-signatures have been drawing lots of attention in recent years, due to their applications in cryptocurrencies. Most early constructions require three-round signing, and recent constructions have managed to reduce the round complexity to two. However, their security proofs are mostly based on non-standard, interactive assumptions (e.g. one-more assumptions) and come with a huge security loss, due to multiple uses of rewinding (aka the Forking Lemma). This renders the quantitative guarantees given by the security proof useless.

In this work, we improve the state of the art by proposing two efficient two-round multi-signature schemes from the (standard, non-interactive) Decisional Diffie-Hellman (DDH) assumption. Both schemes are proven secure in the random oracle model without rewinding. We do not require any pairing either. Our first scheme supports key aggregation but has a security loss linear in the number of signing queries, and our second scheme is the *first* tightly secure construction.

A key ingredient in our constructions is a new homomorphic dual-mode commitment scheme for group elements, that allows to equivocate for messages of a certain structure. The definition and efficient construction of this commitment scheme is of independent interest.

Keywords: Multi-Signatures, Tightness, Forking Lemma, Commitment Scheme, Round Complexity

1 Introduction

A multi-signature scheme [IN83, BN06] allows N parties to jointly sign a message, where each party i holds an independent key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$. Recently, multi-signature schemes have been drawing new attention due to their applications in cryptocurrencies. In this setting, multiple parties share ownership of funds, and can use multi-signatures to sign transactions spending these funds. For details, we refer to [BDN18]. A trivial construction is that each signer i computes a signature σ_i using sk_i, and the final signature is $(\sigma_1, \ldots, \sigma_N)$. Yet, this trivial approach leads to large signature size. Motivated by this, cryptographers are proposing more sophisticated multi-signature schemes with interactive signing protocols to compress the signature size. In this work, we focus on concrete security of two-round multi-signature schemes.

Security Models. There are different models in which multi-signatures have been proposed and analyzed. Namely, schemes may require interactive key generation [MOR01], or require that keys are verified and include a proof of possesion of the secret key [DEF+19, CKM21]. Other schemes require to

*Supported by the Research Council of Norway under Project No. 324235.

use a knowledge of secret key assumption [Bol03, LOS+06]. Besides these models, the widely accepted model for multi-signatures nowadays is the so called *plain public key model*, introduced by Bellare and Neven in their seminal work [BN06]. In this model, each signer generates her key pair independently, and no knowledge assumption or proof of possession is needed. In this paper, we are interested in the plain public key model.

Concrete Security and Tightness. Cryptographic schemes are proven secure using reductions. To prove security of a scheme S, we transform any adversary $\mathcal{A}_{\mathsf{S}}$ against the security of S with success probability ϵ_{S} into a solver $\mathcal{A}_{\Pi}$ for some underlying hard problem Π with success probability ϵ_{Π}. Thereby, we establish a bound $\epsilon_{\mathsf{S}} \leq L \cdot \epsilon_{\Pi}$. We call L the *security loss*. Ideally, we want the underlying hardness assumption to be as standard as possible, since a more standard assumption gives us more confidence on the scheme's security. We also want the security loss as small as possible, since it relates the concrete security of our scheme to the hardness of the underlying computational problem. This is reflected when we use the security proof as a quantitative statement to derive concrete parameters for scheme S based on cryptanalytic results for the well-studied problem Π. Roughly speaking, to get κ bits of security for S, we have to guarantee $\kappa + \log L$ bits of security for Π. If L is large, or depends on choices of the adversary unknown at deployment time, instantiating the scheme in this way leads to prohibitively large parameters, or is not even possible. This motivates striving for a *tight reduction*, i.e. a reduction where L is a small constant. Tightness has been studied for many primitives, including standard digital signatures and related primitives, e.g., [KW03, BKP14, LP20, HJK+21]. Unfortunately, most of existing multi-signature schemes are non-tight. Even worse, existing two-round multi-signature schemes have only non-tight reductions based on strong, non-standard assumptions.

Limitations of Existing Constructions. An overview of existing schemes (based on assumptions in cyclic groups) and their properties and security loss can be found in Table 1. In the plain public key model, Bellare and Neven [BN06] constructed a three-round multi-signature scheme (BN) based on the Discrete Logarithm Assumption (DLOG). Proving the security of this scheme relies on rewinding and uses the (general) Forking Lemma [BN06], which leads to a highly non-tight security bound. To improve this, Bellare and Neven introduced a second three-round construction (BN+) tightly based on the Decisional Diffie-Hellman (DDH) Assumption. Further works focus on key aggregation [MPSW19, BDN18, FH21]. This feature allows to publicly compute a single aggregated key from a given list of public keys, which can later be used for verification. The key aggregation property saves bandwidth and is desirable in many applications. Notably, the three-round scheme Musig [MPSW19, BDN18] can be seen as a variant of BN that supports key aggregation. The scheme is based on DLOG and a double forking technique is introduced for its analysis. This leads to a security bound of the form $\epsilon_{\mathsf{S}}^4 \leq L \cdot \epsilon_{\Pi}$, which is useless in terms of concrete security. Using the Decisional Diffie-Hellman (DDH) assumption, a tightly secure variant Musig+ of Musig has been proposed in [FH21].

To further reduce round complexity, recent works focused on two-round constructions [NRS21, BD21, AB21, CKM21, DOTT21]. However, while achieving certain desirable properties (e.g. deterministic signing [NRSW20]) the proposed schemes have their drawbacks in terms of assumptions and concrete security. The scheme [NRSW20] makes use of heavy cryptographic machinery and is not comparable with others in terms of efficiency. Further, even in the more idealized models such as the algebraic group model, security proofs of most two-round schemes rely on non-standard interactive assumptions [NRS21, CKM21, BD21, AB21]. The only exceptions are [DOTT21, BD21, BTT22]. A second drawback is the apparent need for (double) rewinding in the random oracle model [DEF+19, NRS21, DOTT21, BD21, BTT22]. While such security proofs show the absence of major structural attacks, concrete parameters are not supported by cryptanalytic evidence.

Our Goal. Motivated by the state of the art, we study whether interactive assumptions and rewinding techniques are necessary for two-round multi-signatures. If not, we want to construct a scheme without either of them. Ideally, our scheme comes with additional features such as key aggregation or a fully tight security proof. We summarize our central question as follows, which is of both practical and theoretical interest.

Can we construct two-round multi-signatures
from non-interactive pairing-free assumptions without the use of rewinding?

1.1 Our Contribution

Our work answers the above question in the affirmative. Our contributions are the *first* two multi-signature schemes that are two-round from a non-interactive assumption without using the Forking Lemma. Both of our schemes are proven secure in the random oracle model based on the DDH assumption. Concretely, we construct

1. a two-round multi-signature scheme with a security loss $O(Q_S)$ and key aggregation, where Q_S is the number of signing queries, and
2. the first two-round multi-signature scheme with a fully tight security proof

We compare our schemes with existing schemes in Table 1[1]. For roughly 128 bit security, our second scheme can be instantiated with standardized 128 bit secure curves, in contrast to all previous two-round schemes. For our first scheme, its proof is non-tight, but it does not rely on rewinding and has tighter security based on standard, non-interactive assumptions than other non-tight schemes (such as HBMS and Musig2). Hence, as long as the number of signing queries Q_S is less than $2^{192-128} = 2^{64}$, we can implement our first scheme with a standardized 192-bit secure curve to achieve 128-bit security, while this is not the case for HBMS and Musig2. We note that our schemes do not have some additional beneficial properties (e.g. having Schnorr-compatible signatures or supporting preprocessing) as in Musig2 [NRS21]. We leave achieving these properties without rewinding as an interesting open problem.

Scheme	Assumption	Rounds	Key Aggregation	Loss
BN [BN06]	DLOG	3	✗	$O(Q_H/\epsilon)$
BN+ [BN06]	DDH	3	✗	$O(1)$
Musig [MPSW19, BDN18]	DLOG	3	✓	$O(Q_H^3/\epsilon^3)$
Musig+ [FH21]	DDH	3	✓	$O(1)$
Musig2 [NRS21]	AOMDL	2	✓	$O(Q_H^3/\epsilon^3)$
HBMS [BD21]	DLOG	2	✓	$O(Q_S^4 Q_H^3/\epsilon^3)$
Ours (Section 3.2)	DDH	2	✓	$O(Q_S)$
Ours (Section 3.3)	DDH	2	✗	$O(1)$

Table 1: Comparison of existing multi-signature schemes (top) in the random oracle model with our schemes (bottom). Here, Q_H, Q_S denote the number of random oracle and signing queries, respectively, ϵ denotes the advantage of an adversary against the scheme. The algebraic one-more discrete logarithm (AOMDL) assumption is a (stronger) interactive variant of DLOG.

A crucial building block for our construction is a special kind of DDH-based commitment scheme without pairings. Concretely, our commitment scheme has the following properties.

- It commits to pairs of group elements in a homomorphic way.
- It has a dual-mode property, i.e. indistinguishable keys in statistically hiding and statistically binding mode, with tight multi-key indistinguishability.
- The hiding mode offers a special form of equivocation trapdoor, which allows to open commitments to group elements output by the Honest-Verifier Zero-Knowledge (HVZK) simulator of Schnorr-like identification protocols.

Such a commitment scheme can be useful to construct other interactive signature variants, and we believe that this is of independent interest. In this paper, we construct the first commitment scheme satisfying the above properties simultaneously without using pairings. Our commitment scheme can be seen as an extension of the commitment scheme in [BCJ08][2]. Contrary to our scheme, the commitment scheme in [BCJ08] commits to single group elements and no statistically binding mode is shown, which makes it less desirable for our multi-signature constructions. Other previous commitment schemes either have no

[1] We do not consider proofs in the (idealized) algebraic group model and do not list schemes that are not in the plain public key model.

[2] Drijvers et al. [DEF+19] showed a flaw in the proof of the multi-signature scheme presented in [BCJ08], but it does not affect their commitment scheme.

trapdoor property [GOS06, GS08], or homomorphically commit to ring or field elements [GQ88, Ped92]. To the best of our knowledge, there is only a solution using pairings [Gro09].

1.2 Concurrent Work

In a concurrent work (also at Eurocrypt 2023), Tessaro and Zhu [TZ23] also presented (among other contributions) a new two-round multi-signature scheme. Both our work and theirs focus on avoiding interactive assumptions. However, while we additionally remove the security loss, Tessaro and Zhu concentrate on having a partially non-interactive scheme. That is, the first round of the signing protocol is independent of the message being signed. In a nutshell, they generalize Musig2 to linear function families. Then, under a suitable instantiation, the interactive assumption for Musig2 can be avoided. Similar to Musig2, the resulting scheme is partially non-interactive. Still, their scheme inherits the security loss of Musig2 due to (double) rewinding.

1.3 Technical Overview

We give an intuitive overview of our constructions and the challenges we solve.

Schnorr-Based Multi-Signatures. We start by recalling the basic template for multi-signatures based on the Schnorr identification scheme [Sch91]. Let $\mathbb{G}$ be a group of prime order p with generator g. We explain the template using the vector space homomorphism $\mathsf{F} : x \mapsto g^x$ mapping from $\mathbb{Z}_p$ to $\mathbb{G}$, and write both domain and range additively. In a first approach to get a multi-signature scheme, we let each signer i with secret key sk_i sample a random $r_i \in \mathbb{Z}_p$, and send $R_i := \mathsf{F}(r_i)$ to all other signers. Then, an aggregated R is computed as $R = \sum_i R_i$. From this R, signers derive challenges c_i using a random oracle. Then, each signer computes a response $s_i = c_i\mathsf{sk}_i + r_i$ and sends this response. Finally, the signature contains R and the aggregated response $s = \sum_i s_i$. Verification is very similar to the verification of Schnorr signatures. As each signer in this simple two-round scheme is almost identical to the prover algorithm of the Schnorr identification scheme, one may hope that this scheme is secure. However, early works already noted that it is not [BN06].

While there are concrete attacks against the scheme, for our purposes it is more important to understand where the security proof fails. The proof fails when we try to simulate honest signer without knowing its secret key sk_1. Following Schnorr signatures and identification, this would be done by sampling $R_1 := \mathsf{F}(s_1) - c_1\mathsf{pk}_1$ for random c_1 an s_1, and then programming the random oracle accordingly at position R. The problem in the multi-signature setting is that we first have to output R_1, and then the adversary can output the remaining R_i, such that he has full control over the aggregate R. Thus, the random oracle may already be defined. Previous works [BN06, MPSW19, BDN18] solve this issue by introducing an additional round, in which all signers commit to their R_i using a random oracle. This allows us to extract all R_i from these commitments in the reduction, and therefore R has enough entropy to program the random oracle.

A second problem that we encounter in the above approach is the extraction of a solution from the forgery. Namely, to extract a discrete logarithm of pk_1, we need to rely on rewinding. Some of the well-known schemes [MPSW19, BDN18] even use rewinding multiple times. This leads to security bounds with essentially no useful quantitative guarantee for concrete security.

Towards A Scheme without Rewinding. To avoid rewinding, our first idea is to rely on a different homomorphism F. Namely, we borrow techniques from lossy identification [KW03, AFLT12, KMP16] and use $\mathsf{F} : x \mapsto (g^x, h^x)$ for a second generator $h \in \mathbb{G}$. We can then give a non-rewinding security proof for the three-round schemes in [BN06, MPSW19, BDN18]. Concretely, we first switch pk_1 from the range of F to a random element in $\mathbb{G}^2$, using the DDH assumption. Then, we can argue that a forgery is hard to compute using a statistical argument. We note that this idea is (implicitly) already present in [BN06, FH21]. As we will see, combining it with techniques to avoid the extra round is challenging.

Towards Two-Round Schemes. To go from a three-round scheme as above to a two-round scheme, our goal is to avoid the first round. Recall that this round was needed to simulate R_1 using random oracle programming. Our idea to tackle the simulation problem is a bit different. Namely, going back to the (insecure) two-round scheme, our goal is to send R_1 *after* we learn c_1. If we manage to do that, we can simulate by setting it as $R_1 := \mathsf{F}(s_1) - c_1\mathsf{pk}_1$ for random s_1. Of course, just sending R_1 after learning c_1 should only be possible for the reduction. Following Damgård [Dam00], this high-level strategy can be

implemented using a trapdoor commitment scheme Com, and sending $\mathsf{com}_1 = \mathsf{Com}(\mathsf{ck}, R_1)$ as the first message. The challenges c_i are then derived from an aggregated commitment com using the random oracle. Later, the reduction can open this commitment to $\mathsf{F}(s_1) - c_1\mathsf{pk}_1$ using the trapdoor for commitment key ck. To support aggregation, the commitment scheme should have homomorphic properties. Note that this approach has been used in the lattice setting in a recent work [DOTT21]. However, implementing such a commitment scheme for (pairs of) group elements is highly non-trivial, as we will see. Also, as already pointed out in [DOTT21], it is hard to make this two-round approach work while avoiding rewinding at the same time. The reason is that a trapdoor commitment scheme can not be statistically binding. But if we want to make use of the statistical argument from lossy identification discussed above, we need that R is fixed before the c_i are sampled, which requires statistical binding. With a computationally binding commitment scheme, we end up in a rewinding reduction (to binding) again. Our first technical main contribution is to overcome this issue.

Chopstick One: Our Scheme Without Rewinding. Our idea to overcome the above problem is to demand a dual-mode property from the commitment scheme Com. Namely, there should be an indistinguishable second way to set up the commitment key ck, such that for such a key the scheme is statistically binding. This does not solve the problem yet, because we require ck to be in trapdoor mode for simulation, and in binding mode for the final forgery. The solution is to sample ck in a message-dependent way using another random oracle, which is (for other reasons) already done in earlier works [DEF+19, DOTT21]. In this way, we can embed a binding commitment key in some randomly guessed random oracle queries, and a trapdoor key in others. Note that this requires a tight multi-key indistinguishability of the commitment scheme. Assuming we have such a commitment scheme, we end up with our first construction, which is presented formally in Section 3.2. Of course, this strategy still has a security loss linear in the number of signing queries due to the guessing argument, but it avoids rewinding, leading to an acceptable security bound. In addition, we can implement the approach in a way that supports key aggregation.

Chopstick Two: Our Fully Tight Scheme. The security loss in our first scheme results from partitioning random oracle queries into two classes, namely queries returning binding keys, and queries returning trapdoor keys. To do such a partitioning in a tight way, we may try to use a Katz-Wang random bit approach [GJKW07]. This simple approach can be used in standard digital signatures. However, it turns out that it does not work for our case. To see this, recall that following this approach, we would compute two message-dependent commitment keys

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

Then, for each message, we would embed a binding key in one branch, and a trapdoor key in the other branch, e.g. ck_0 binding and ck_1 with trapdoor. In the signing protocol, we would abort one of the branches pseudorandomly based on the message. Then we could use the trapdoor branch in the signing, and hope that the forgery uses the binding branch. However, this strategy crucially relies on the fact that the aborting happens in a way that is pseudorandom to the adversary. Otherwise the adversary could always choose the trapdoor branch for his forgery. While we can implement this in a signature scheme, in our multi-signature scheme this fails, because all signers must use the *same commitment key* to make aggregation possible. At the same time, the aborted branch must depend on secret data of the simulated signer to remain pseudorandom.

To solve this problem, we observe that the above approach uses a pseudorandom "branch selection" and aborts the other branch. Our solution can be phrased as a pseudorandom "branch-to-key matching". Namely, we give each signer two public keys $(\mathsf{pk}_{i,0}, \mathsf{pk}_{i,1})$. The signing protocol is run in two instances in parallel. One instance uses ck_0, and one uses ck_1 as above. More precisely, we commit to R_0 via ck_0 and to R_1 via ck_1. Then we aggregate and determine the challenges $c_{i,0}$ and $c_{i,1}$. However, before sending the response $s_i = (s_{i,0}, s_{i,1})$, *each signer separately* determines which key to use in which instance, i.e. it computes

$$s_{i,0} = c_{i,0} \cdot x_{i,b_i} + r_{i,0}, \quad s_{i,1} = c_{i,1} \cdot x_{i,1-b_i} + r_{i,1},$$

where b_i is a pseudorandom bit that each signer i computes *independently*, and that will be included in the final signature to make verification possible. This decouples the public key that is used from the commitment key that is used. Now we are ready to discuss the implication of this change. Namely, our reduction chooses $\mathsf{pk}_{1,0}$ honestly and $\mathsf{pk}_{1,1}$ as a lossy key, i.e. random instead of in the range of F. Then,

in each signing interaction, the reduction can match the honest public key with the binding commitment key and the lossy public key with the trapdoor commitment key by setting b_1 accordingly. In this way, we can simulate one branch using the actual secret key, and the other branch using the commitment trapdoor. For the forgery, we hope that the matching is the other way around, such that binding commitment key and lossy public key match, which makes the statistical argument from lossy identification possible. Overall, this approach leads to our fully tight scheme, presented in Section 3.3.

The Challenge of Instantiating the Commitment. One may observe that we shifted a lot of the challenges that we encountered into properties of the underlying commitment scheme. This naturally raises the question if such a commitment scheme can be found. In fact, constructing this commitment scheme can be understood as our second technical main contribution.

Let us first explain why it is non-trivial to construct such a scheme. The main barrier results from the algebraic structure that we demand. Namely, we need to commit to group elements[3] $R \in \mathbb{G}$. A naive idea would be to use any trapdoor commitment scheme, e.g. Pedersen commitments, by first encoding R in the appropriate message space. However, this would destroy all homomorphic properties that we need, and we should not forget that we need a dual-mode property. This brings us to Groth-Sahai commitments [GS08], which can commit to group elements. Indeed, these commitments are homomorphic, and have (indistinguishable from) random keys, such that we can sample them using a random oracle. They are also dual-mode based on DDH, which allows us to use the random self-reducibility of DDH to show tight multi-key indistinguishability. However, the trapdoor property turns out to be the main challenge. To see why this is problematic, note that the opening information of these commitments typically contains elements from $\mathbb{Z}_p$ that are somehow used as exponents. There are exceptions to this rule, like [Gro09], but they use pairings and the DLIN assumption, which we aim to avoid. This means that the trapdoor should allow us to sample exponents, given a group element R to which we want to open the commitment. This naturally corresponds to having a trapdoor for the discrete logarithm problem, which we do not have.

Our Solution: Weakly Equivocable Commitments. Our starting point is the commitment scheme for group elements given in [GS08]. Namely, commitment keys correspond to matrices $\mathbf{A} = (A_{i,j})_{i,j} \in \mathbb{G}^{2\times 2}$, and to commit to a message $R = g^r \in \mathbb{G}$ with randomness $(\alpha, \beta) \in \mathbb{Z}_p$, one computes

$$\mathsf{com} := (C_0, C_1)^t := \left(A_{1,1}^{\alpha} \cdot A_{1,2}^{\beta}, R \cdot A_{2,1}^{\alpha} \cdot A_{2,2}^{\beta}\right)^t .$$

That is, setting $\mathbf{E} = (E_{i,j})_{i,j} \in \mathbb{Z}_p$ such that $g^{E_{i,j}} = A_{i,j}$, we can write the discrete logarithm of com as $(0, r)^t + \mathbf{E} \cdot (\alpha, \beta)^t$. In binding mode, matrix $\mathbf{E}$ is a matrix of rank 1, while $\mathbf{E}$ has full rank in hiding mode. It is easy to see that this commitment scheme to group elements is homomorphic. However, we stress that there is no simple solution to implement a trapdoor for equivocation. To see this, note that if we want to open a commitment com to a message $R' \in \mathbb{G}$, we need to output a suitable tuple (α, β). If we knew the discrete logarithm of com, then we still would need to know the discrete logarithm of R' to find such a tuple. The key insight of our trapdoor construction is that we do not need to be able to open com to any message R'. Instead, it will be sufficient if we can open it to messages of the form $R' = g^s \cdot \mathsf{pk}^c$, where we do not know c when we fix the commitment com, but *we know* pk *when setting up* $\mathbf{A}$. To explain why this helps, assume we want to find a valid opening (α, β) in this case. Then we need to satisfy

$$\mathsf{com} = \begin{pmatrix} C_0 \\ C_1 \end{pmatrix} = \begin{pmatrix} 0 \\ g^s \mathsf{pk}^c \end{pmatrix} \cdot g^{\mathbf{E} \cdot (\alpha, \beta)^t}.$$

It seems like we did not make progress, because even if we know the discrete logarithms of C_0, C_1, the term pk^c is not known in the exponent. Now, our key idea to solve this is to write and generate $\mathbf{A}$ with respect to basis pk in the second row. Namely, we generate $\mathbf{A}$ as

$$\mathbf{A} = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} := \begin{pmatrix} g^{d_{1,1}} & g^{d_{1,2}} \\ \mathsf{pk}^{d_{2,1}} & \mathsf{pk}^{d_{2,2}} \end{pmatrix}.$$

In this way, the equation that we need to satisfy becomes

$$\begin{pmatrix} C_0 \\ C_1 \end{pmatrix} = \begin{pmatrix} g^{d_{1,1}\alpha + d_{1,2}\beta} \\ g^s \mathsf{pk}^{c + d_{2,1}\alpha + d_{2,2}\beta} \end{pmatrix}.$$

[3] In the actual construction, we need to commit to pairs of group elements, but we consider the simpler setting of one group element in this overview.

Next, we get rid of the term g^s by shifting C_1 accordingly. Namely, recall that we can sample s at random long before we learn c. Setting $C_0 = g^\tau$ and $C_1 = g^s \mathsf{pk}^\rho$ for random τ, ρ, we obtain the equation

$$\begin{pmatrix} g^\tau \\ \mathsf{pk}^\rho \end{pmatrix} = \begin{pmatrix} g^{d_{1,1}\alpha + d_{1,2}\beta} \\ \mathsf{pk}^{c + d_{2,1}\alpha + d_{2,2}\beta} \end{pmatrix}.$$

Given the trapdoor $\mathbf{D} = (d_{i,j})_{i,j}$, this can easily be solved for (α, β) by solving $(\tau, \rho - c)^t = \mathbf{D} \cdot (\alpha, \beta)^t$. We are confident that such a weak and structured equivocation property can be used in other applications as well, and formally define this type of commitment scheme in Section 3.1.

2 Preliminaries

We denote the security parameter by $\lambda \in \mathbb{N}$, and all algorithms get 1^λ implicitly as input. We write $x \xleftarrow{\$} S$ if x is sampled uniformly at random from a finite set S, and we write $x \leftarrow \mathcal{D}$ if x is sampled according to a distribution $\mathcal{D}$. We write $y \leftarrow \mathcal{A}(x)$, if y is output from (probabilistic) algorithm $\mathcal{A}$ on input x with uniform coins. To make the coins explicit, we use the notation $y = \mathcal{A}(x; \rho)$. The notation $y \in \mathcal{A}(x)$ indicates that y is a possible output of $\mathcal{A}(x)$. We use standard asymptotic notation, and the notions of negligible functions, and PPT algorithms. If $\mathbf{G}$ is a security game, we write $\mathbf{G} \Rightarrow b$ to state that $\mathbf{G}$ outputs b. In all our games, numerical variables are implicitly initialized with 0, and lists and sets are initialized with $\emptyset$. We define $[K] := \{1, \ldots, K\}$, and denote the Bernoulli distribution with parameter $\gamma \in [0, 1]$ by $\mathcal{B}_\gamma$.

Multi-Signatures. We introduce syntax and security for multi-signatures, following the established security notions in the plain public key model [BN06]. The only minor difference to [BN06] is that we assume the list of public keys participating in the signing protocol is given by a set, and not a multi-set. This is inline with other works, e.g., [CKM21, DOTT21]. We opt for using sets for simplicity of exposition, and as a signer can always refuse to sign when its key would be contained twice. We will assume that there is an canonical ordering of given sets, e.g. lexicographically, that allows us to uniquely encode sets $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$. For this encoding, we write $\langle \mathcal{P} \rangle$ throughout the paper. Further, for simplicity of notation, we assume that the honest public key in our security definition is the entry pk_1 in this set.

Alg $\mathsf{MS.Exec}(\mathcal{P}, \mathcal{S}, \mathsf{m})$
01 **let** $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$, $\mathcal{S} = \{\mathsf{sk}_1, \ldots, \mathsf{sk}_N\}$
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: The algorithm $\mathsf{MS.Exec}$ for a (two-round) multi-signature scheme $\mathsf{MS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$, representing an honest execution of the signing protocol Sig.

Definition 1 (Multi-Signature Scheme)**.** A (two-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 SIG take at least implicitly par 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, i.e. for all $\mathsf{par} \in \mathsf{Setup}(1^\lambda)$, all $N = \mathsf{poly}(\lambda)$, all $(\mathsf{pk}_j, \mathsf{sk}_j) \in \mathsf{Gen}(\mathsf{par})$ for $j \in [N]$, 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 $\mathsf{MS.Exec}$ is defined in Figure 1.

Definition 2 (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 $\mathsf{Agg}, \mathsf{VerAgg}$ 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 $b \in \{0,1\}$.

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

Definition 3 (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 $\mathrm{SIG} := (\mathrm{SIG}_0, \mathrm{SIG}_1, \mathrm{SIG}_2)$
04 $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathrm{SIG}}(\mathsf{par}, \mathsf{pk})$
05 **if** $\mathsf{pk} \notin \mathcal{P}^* \vee (\mathcal{P}^*, \mathsf{m}^*) \in \mathcal{L}$:
06 **return** 0
07 **return** $\mathsf{Ver}(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$

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

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

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

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

Linear Function Families. To present our constructions in a modular way, we make use of the abstraction of linear function families. Our definition is close to previous definitions [HKL19, KLR21, CAHL+22]. As it is not needed for our instantiations, we restrict our setting to vector spaces instead of pseudo modules.

Definition 4 (Linear Function Family)**.** A linear function family (LFF) is a tuple of PPT algorithms $\mathsf{LF} = (\mathsf{Gen}, \mathsf{F})$ with the following syntax:

- $\mathsf{Gen}(1^\lambda) \to \mathsf{par}$ 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 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 $\cdot$.

- $\mathsf{F}(\mathsf{par}, x) \to X$ 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.

We formalize necessary conditions under which a linear function family can be used to construct so called lossy identification [AFLT12]. Our constructions will rely on such linear function families. We also give a similar definition that captures a similar property in the context of key aggregation.

Definition 5 (Lossiness Admitting LFF)**.** We say that a linear function family $\mathsf{LF} = (\mathsf{Gen}, \mathsf{F})$ is ε_{l}-lossiness admitting, if the following properties hold:

- **Key Indistinguishability.** For any PPT algorithm $\mathcal{A}$, the following advantage is negligible:

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

- **Lossy Soundness.** For any unbounded algorithm $\mathcal{A}$, the following probability is at most ε_{l}:

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

Definition 6 (Aggregation Lossy Soundness)**.** We say that a linear function family $\mathsf{LF} = (\mathsf{Gen}, \mathsf{F})$ satisfies $\varepsilon_{\mathsf{al}}$-aggregation lossy soundness, if for any unbounded algorithm $\mathcal{A}$, the following probability is at most $\varepsilon_{\mathsf{al}}$:

$$\Pr\left[\mathsf{F}(s) - c \cdot \sum_{i=1}^{N} a_i X_i = R \;\middle|\; \begin{array}{l}\mathsf{par} \leftarrow \mathsf{Gen}(1^\lambda),\ X_1 \xleftarrow{\$} \mathcal{R}, \\ (St, (X_2, a_2), \ldots, (X_N, a_N)) \leftarrow \mathcal{A}(\mathsf{par}, X_1), \\ a_1 \xleftarrow{\$} \mathcal{S},\ (St', R) \leftarrow \mathcal{A}(St, a_1), \\ c \xleftarrow{\$} \mathcal{S},\ s \leftarrow \mathcal{A}(St', c)\end{array}\right].$$

Assumptions. We recall the computational assumptions that we need.

Definition 7 (DDH Assumption)**.** Let GGen be an algorithm that on input 1^λ outputs the description of a prime order group $\mathbb{G}$ of order p with generator g. We say that the DDH assumption holds relative to GGen, if for all PPT algorithms $\mathcal{A}$, the following advantage is negligible:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{A},\mathsf{GGen}}(\lambda) := |\Pr\left[\mathcal{A}(\mathbb{G}, p, g, h, g^a, h^a) = 1 \middle|\ \begin{array}{l}(\mathbb{G}, g, p) \leftarrow \mathsf{GGen}(1^\lambda), \\ h \xleftarrow{\$} \mathbb{G}, a \xleftarrow{\$} \mathbb{Z}_p\end{array}\right] \\ - \Pr\left[\mathcal{A}(\mathbb{G}, p, g, h, g^a, g^b) = 1 \middle|\ \begin{array}{l}(\mathbb{G}, g, p) \leftarrow \mathsf{GGen}(1^\lambda), \\ h \xleftarrow{\$} \mathbb{G}, a, b \xleftarrow{\$} \mathbb{Z}_p\end{array}\right]|.\end{aligned}$$

In the following, we define an equivalent variant of the DDH assumption, uDDH3. uDDH3 is the 2-fold $\mathcal{U}_{3,1}$-Matrix-DDH (MDDH) assumption (with terminology in [EHK+13]). By its random self-reducibility [EHK+13, Lemma 1], the 2-fold $\mathcal{U}_{3,1}$-Matrix-DDH (MDDH) assumption (namely, the uDDH3 assumption) is tightly equivalent to the $\mathcal{U}_{3,1}$-MDDH assumption. By Lemma 1 in [LP20], $\mathcal{U}_{3,1}$-MDDH is tightly equivalent to $\mathcal{U}_1$-MDDH that is the DDH assumption. Hence, the DDH and uDDH3 assumptions are tightly equivalent.

Definition 8 (uDDH3 Assumption)**.** Let GGen be an algorithm that on input 1^λ outputs the description of a prime order group $\mathbb{G}$ of order p with generator g. We say that the $\mathsf{uDDH3}$ assumption holds relative to GGen, if for all PPT algorithms $\mathcal{A}$, the following advantage is negligible:

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

3 Constructions

In this section, we present our construction of two-round multi-signatures. First, we give a definition of a special commitment scheme that will be used in both constructions. Then, we present the constructions in an abstract way. For the instantiation, we refer to Section 4.

3.1 Preparation: Special Commitments

In this section we define a special kind of commitment scheme. We will make use of such a scheme in our constructions of multi-signatures. Before we give the definition, we explain the desired properties at a high level. First of all, we want to be able to commit to elements $R \in \mathcal{R}$ in the range of a given linear function family. Second, we need the commitment scheme to be homomorphic in both messages and randomness, allowing us to aggregate commitments during the signing protocol. Third, we need a certain dual mode property, ensuring that we can set up keys either in a perfectly hiding or in a perfectly binding mode. This will allow us to make the commitment key for the forgery binding, while associating a equivocation trapdoor to the keys used to answer signing queries. We emphasize that we do not need a full-fledged equivocation feature. This is because we already know parts of the structure of messages to which we want to open the commitment. Looking ahead, this is the reason we can instantiate the commitment in the DDH setting.

Game Q-$\mathbf{KEYDIST}^{\mathcal{A}}_{0,\mathsf{CMT}}(\lambda)$	**Game** Q-$\mathbf{KEYDIST}^{\mathcal{A}}_{1,\mathsf{CMT}}(\lambda)$
01 $\mathsf{par}\leftarrow\mathsf{LF.Gen}(1^\lambda),\ x\xleftarrow{\$}\mathcal{D}$	06 $\mathsf{par}\leftarrow\mathsf{LF.Gen}(1^\lambda),\ x\xleftarrow{\$}\mathcal{D}$
02 **if** $(\mathsf{par},x)\notin\mathsf{Good}$: **return** 0	07 **if** $(\mathsf{par},x)\notin\mathsf{Good}$: **return** 0
03 **for** $i\in[Q]$: $\mathsf{ck}_i\leftarrow\mathsf{BGen}(\mathsf{par})$	08 **for** $i\in[Q]$: $\mathsf{ck}_i\xleftarrow{\$}\mathcal{K}_{\mathsf{par}}$
04 $\beta\leftarrow\mathcal{A}(\mathsf{par},x,(\mathsf{ck}_i)_{i\in[Q]})$	09 $\beta\leftarrow\mathcal{A}(\mathsf{par},x,(\mathsf{ck}_i)_{i\in[Q]})$
05 **return** β	10 **return** β

Figure 3: The games $\mathbf{KEYDIST}_0, \mathbf{KEYDIST}_1$ for a special commitment scheme CMT and an adversary $\mathcal{A}$.

Definition 9 (Special Commitment Scheme)**.** Let $\mathsf{LF} = (\mathsf{LF.Gen}, \mathsf{F})$ be a linear function family and $\mathcal{G} = \{\mathcal{G}_{\mathsf{par}}\}, \mathcal{H} = \{\mathcal{H}_{\mathsf{par}}\}$ be families of subsets of abelian groups with efficiently computable group operations $\oplus$ and $\otimes$, respectively. 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:

- $\mathsf{BGen}(\mathsf{par}) \to \mathsf{ck}$ 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:
$$\{(\mathsf{par}, x, \mathsf{ck}) \mid \mathsf{ck} \xleftarrow{\$} \mathcal{K}_{\mathsf{par}}\} \text{ and } \{(\mathsf{par}, x, \mathsf{ck}) \mid (\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, X)\}.$$
- **Special Trapdoor Property.** For all $(\mathsf{par}, x) \in \mathsf{Good}$, and all $c \xleftarrow{\$} \mathcal{S}$, the following distributions $\mathcal{T}_0$ and $\mathcal{T}_1$ have statistical distance at most ε_{t}:
$$\mathcal{T}_0 := \left\{ (\mathsf{par}, \mathsf{ck}, \mathsf{td}, x, c, \mathsf{com}, \mathsf{tr}) \;\middle|\; \begin{array}{l} (\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, \mathsf{F}(x)) \\ (\mathsf{com}, St) \leftarrow \mathsf{TCom}(\mathsf{ck}, \mathsf{td}), \\ \mathsf{tr} \leftarrow \mathsf{TCol}(St, c) \end{array} \right\}$$
$$\mathcal{T}_1 := \left\{ (\mathsf{par}, \mathsf{ck}, \mathsf{td}, x, c, \mathsf{com}, \mathsf{tr}) \;\middle|\; \begin{array}{l} (\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, \mathsf{F}(x)) \\ r \xleftarrow{\$} \mathcal{D},\ R := \mathsf{F}(r),\ \varphi \xleftarrow{\$} \mathcal{G}, \\ \mathsf{com} := \mathsf{Com}(\mathsf{ck}, R; \varphi), \\ s := c \cdot x + r,\ \mathsf{tr} := (\varphi, R, s) \end{array} \right\}$$
- **Multi-Key Indistinguishability.** For every $Q = \mathsf{poly}(\lambda)$ and any PPT algorithm $\mathcal{A}$, the following advantage is negligible:
$$\begin{aligned} \mathsf{Adv}^{Q\text{-}\mathsf{keydist}}_{\mathcal{A},\mathsf{CMT}}(\lambda) := |\Pr\left[Q\text{-}\mathbf{KEYDIST}^{\mathcal{A}}_{0,\mathsf{CMT}}(\lambda) \Rightarrow 1\right] \\ - \Pr\left[Q\text{-}\mathbf{KEYDIST}^{\mathcal{A}}_{1,\mathsf{CMT}}(\lambda) \Rightarrow 1\right]|, \end{aligned}$$
where games $\mathbf{KEYDIST}_0, \mathbf{KEYDIST}_1$ 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}:
$$\Pr\left[\begin{array}{rl} \mathsf{Com}(\mathsf{ck}, R'; \varphi') & = \mathsf{com} \\ \wedge\ R & \neq R' \end{array} \;\middle|\; \begin{array}{l} \mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda), \\ \mathsf{ck} \leftarrow \mathsf{BGen}(\mathsf{par}),\ (\mathsf{com}, St) \leftarrow \mathcal{A}(\mathsf{ck}), \\ R \leftarrow \mathsf{Ext}(\mathsf{ck}, \mathsf{com}),\ (R', \varphi') \leftarrow \mathcal{A}(St) \end{array}\right].$$

3.2 Our Construction with Key Aggregation

In this section, we construct a two-round multi-signature scheme with key aggregation. Although the scheme will not be tight, the security proof will not use rewinding, leading to an acceptable security loss. For our scheme, we need a lossiness admitting linear function family $\mathsf{LF} = (\mathsf{LF.Gen}, \mathsf{F})$. It should also satisfy aggregation lossy soundness. Further, let $\mathsf{CMT} = (\mathsf{BGen}, \mathsf{TGen}, \mathsf{Com}, \mathsf{TCom}, \mathsf{TCol})$ be 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 $\mathcal{H}$. We make use of random oracles $\mathsf{H}\colon \{0,1\}^* \to \mathcal{K}$, $\mathsf{H}_a\colon \{0,1\}^* \to \mathcal{S}$, and $\mathsf{H}_c\colon \{0,1\}^* \to \mathcal{S}$. We give a verbal description of our scheme $\mathsf{MS}_a[\mathsf{LF}, \mathsf{CMT}]$. Formally, the scheme is presented in Figure 9.

Setup and Key Generation. The public parameters of the scheme are $\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda)$ defining the linear function $\mathsf{F} = \mathsf{F}(\mathsf{par}, \cdot)$. To generate a key (algorithm Gen), a user samples $\mathsf{sk} := x \xleftarrow{\$} \mathcal{D}$. The public key is $\mathsf{pk} := X := \mathsf{F}(x)$.

Key Aggregation. For N users with public keys $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$, the aggregated public key $\tilde{\mathsf{pk}}$ is computed (by algorithm Agg) as
$$\tilde{\mathsf{pk}} := \tilde{X} := \sum_{i=1}^{N} a_i \cdot X_i,$$

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

Signing Protocol. Suppose N users 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 (algorithms $\mathsf{Sig}_0, \mathsf{Sig}_1, \mathsf{Sig}_2$) from the perspective of the first user, which holds a secret key $\mathsf{sk}_1 = x_1$ for public key $\mathsf{pk}_1 = X_1$.

1. *Commitment Phase.* The user derives the aggregated public key $\tilde{\mathsf{pk}}$ as described above. Then, it derives a commitment key $\mathsf{ck} := \mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$ depending on the message. The user samples an element $r_1 \xleftarrow{\$} \mathcal{D}$ and sets $R_1 := \mathsf{F}(r_1)$. Next, it commits to R_1 by sampling $\varphi_1 \xleftarrow{\$} \mathcal{G}$ and setting $\mathsf{com}_1 := \mathsf{Com}(\mathsf{ck}, R_1; \varphi_1)$. Finally, it sends $\mathsf{pm}_{1,1} := \mathsf{com}_1$ to all users.

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. Here, message $\mathsf{pm}_{1,i}$ is sent by user i and has the form $\mathsf{pm}_{1,i} = \mathsf{com}_i$. With this notation, the user aggregates the commitments via $\mathsf{com} := \bigotimes_{i\in[N]} \mathsf{com}_i$. It computes the challenge c and coefficient a_1 via $c := \mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m})$ and $a_1 := \mathsf{H}_a(\langle\mathcal{P}\rangle, \mathsf{pk}_1)$. Then, it computes the response s_1 as $s_1 := c \cdot a_1 \cdot x_1 + r_1$.

   Finally, the user sends $\mathsf{pm}_{2,1} := (s_1, \varphi_1)$ to all users.

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. Here, message $\mathsf{pm}_{2,i}$ is sent by user i and has the form $\mathsf{pm}_{2,i} = (s_i, \varphi_i)$. To compute the final signature, users aggregate the responses and commitment randomness as follows:

$$s := \sum_{i\in[N]} s_i, \;\; \varphi := \bigoplus_{i\in[N]} \varphi_i.$$

   They output the final signature $\sigma := (\mathsf{com}, s, \varphi)$.

Verification. For verification (algorithm Ver), let $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ be a set of public keys, $\mathsf{m} \in \{0,1\}^*$ be a message, and $\sigma = (\mathsf{com}, s, \varphi)$ be a signature. To verify σ, we determine the aggregated public key $\tilde{\mathsf{pk}} = \tilde{X}$ as above. We reconstruct the commitment key $\mathsf{ck} := \mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$, and the challenge $c := \mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m})$. Then, we output 1 if and only if the following equation holds:

$$\mathsf{com} = \mathsf{Com}\left(\mathsf{ck}, \mathsf{F}(s) - c \cdot \tilde{X}; \varphi\right).$$

Completeness easily follows from the homomorphic properties of CMT and F. For a similar calculation, we refer to the proof of Lemma 2.

Lemma 1. *Let* LF *be a linear function family. Let* CMT *be a* $(\varepsilon_\mathsf{b}, \varepsilon_\mathsf{g}, \varepsilon_\mathsf{t})$*-special commitment scheme for* LF*. Then* $\mathsf{MS}_a[\mathsf{LF}, \mathsf{CMT}]$ *is complete.*

Theorem 1. *Let* LF *be a* ε_l*-lossiness admitting linear function family with* ε_al*-aggregation lossy soundness. Let* CMT *be a* $(\varepsilon_\mathsf{b}, \varepsilon_\mathsf{g}, \varepsilon_\mathsf{t})$*-special commitment scheme for* LF*. Further, let* $\mathsf{H}\colon \{0,1\}^* \to \mathcal{K}, \mathsf{H}_a\colon \{0,1\}^* \to \mathcal{S}$*, and* $\mathsf{H}_c\colon \{0,1\}^* \to \mathcal{S}$ *be random oracles. Then* $\mathsf{MS}_a[\mathsf{LF}, \mathsf{CMT}]$ *is* $\mathsf{MS\text{-}EUF\text{-}CMA}$ *secure.*

Concretely, for any PPT algorithm $\mathcal{A}$ *that makes at most* $Q_\mathsf{H}, Q_{\mathsf{H}_a}, Q_{\mathsf{H}_c}, Q_S$ *queries to oracles* $\mathsf{H}, \mathsf{H}_a, \mathsf{H}_c, \textsc{Sig}_0$*, respectively, there are PPT algorithms* $\mathcal{B}, \mathcal{B}'$ *with* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A}), \mathbf{T}(\mathcal{B}') \approx \mathbf{T}(\mathcal{A})$ *and*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{MS\text{-}EUF\text{-}CMA}}_{\mathcal{A},\mathsf{MS}_a[\mathsf{LF},\mathsf{CMT}]}(\lambda) \le\; & \varepsilon_\mathsf{g} + 4Q_S^2\varepsilon_\mathsf{t} + 4Q_S\varepsilon_\mathsf{g} + 4Q_S Q_\mathsf{H} Q_{\mathsf{H}_c}\varepsilon_\mathsf{b} \\ & + \frac{4Q_S}{|\mathcal{R}|} + \frac{4Q_S Q_{\mathsf{H}_a} Q_{\mathsf{H}_c}}{|\mathcal{S}|} + 4Q_S Q_{\mathsf{H}_a} Q_{\mathsf{H}_c}\varepsilon_\mathsf{al} \\ & + 4Q_S\left(\mathsf{Adv}^{Q_\mathsf{H}\text{-}\mathsf{keydist}}_{\mathcal{B},\mathsf{CMT}}(\lambda) + \mathsf{Adv}^{\mathsf{keydist}}_{\mathcal{B}',\mathsf{LF}}(\lambda)\right).\end{aligned}$$

We postpone the proof to Supplementary Material Section A.

3.3 Our Tight Construction

In this section, we present a tightly secure two-round multi-signature scheme $\mathsf{MS}_t[\mathsf{LF}, \mathsf{CMT}] = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$. Let us first describe the building blocks that we need. We make use of a lossiness admitting linear function family $\mathsf{LF} = (\mathsf{LF.Gen}, \mathsf{F})$. Also, let $\mathsf{CMT} = (\mathsf{BGen}, \mathsf{TGen}, \mathsf{Com}, \mathsf{TCom}, \mathsf{TCol})$ be 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 $\mathcal{H}$. We make use of random oracles $\mathsf{H}\colon \{0,1\}^* \to \mathcal{K}$, $\mathsf{H}_b\colon \{0,1\}^* \to \{0,1\}$, and $\mathsf{H}_c\colon \{0,1\}^* \to \mathcal{S}$. We give a verbal description of the scheme. Formally, the scheme is presented in Figure 10.

Setup and Key Generation. The public parameters of the scheme are $\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda)$. They define the linear function $\mathsf{F} = \mathsf{F}(\mathsf{par}, \cdot)$. To generate a key (algorithm Gen), a user samples $x_0, x_1 \xleftarrow{\$} \mathcal{D}$ and a seed $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$. Then, it sets

$$\mathsf{sk} := (x_0, x_1, \mathsf{seed}), \quad \mathsf{pk} := (X_0, X_1) := (\mathsf{F}(x_0), \mathsf{F}(x_1)).$$

Signing Protocol. Suppose N users 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 (algorithms $\mathsf{Sig}_0, \mathsf{Sig}_1, \mathsf{Sig}_2$) from the perspective of the first user, which holds a secret key $\mathsf{sk}_1 = (x_{1,0}, x_{1,1}, \mathsf{seed}_1)$ for public key $\mathsf{pk}_1 = (X_{1,0}, X_{1,1})$.

1. *Commitment Phase.* The user derives commitment keys $\mathsf{ck}_0 := \mathsf{H}(0, \langle\mathcal{P}\rangle, \mathsf{m}), \mathsf{ck}_1 := \mathsf{H}(1, \langle\mathcal{P}\rangle, \mathsf{m})$ depending on the message. Then, the user computes a bit $b_1 := \mathsf{H}_b(\mathsf{seed}_1, \langle\mathcal{P}\rangle, \mathsf{m})$. It samples two elements $r_{1,0}, r_{1,1} \xleftarrow{\$} \mathcal{D}$ and sets

   $$R_{1,0} := \mathsf{F}(r_{1,0}), \quad R_{1,1} := \mathsf{F}(r_{1,1}).$$

   Next, it commits to the resulting elements by sampling $\varphi_{1,0}, \varphi_{1,1} \xleftarrow{\$} \mathcal{G}$ and setting

   $$\mathsf{com}_{1,0} := \mathsf{Com}(\mathsf{ck}_0, R_{1,0}; \varphi_{1,0}), \quad \mathsf{com}_{1,1} := \mathsf{Com}(\mathsf{ck}_1, R_{1,1}; \varphi_{1,1}).$$

   Finally, it sends $\mathsf{pm}_{1,1} := (b_1, \mathsf{com}_{1,0}, \mathsf{com}_{1,1})$ to all users.

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. Here, message $\mathsf{pm}_{1,i}$ is sent by user i and has the form $\mathsf{pm}_{1,i} = (b_i, \mathsf{com}_{i,0}, \mathsf{com}_{i,1})$. With this notation, the user sets $B := b_1 \ldots b_N \in \{0,1\}^N$. Then, it aggregates the commitments via

   $$\mathsf{com}_0 := \bigotimes_{i\in[N]} \mathsf{com}_{i,0}, \quad \mathsf{com}_1 := \bigotimes_{i\in[N]} \mathsf{com}_{i,1}.$$

   It computes user specific challenges via

   $$c_{1,0} := \mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}_0, \mathsf{m}, \langle\mathcal{P}\rangle, B, 0), \quad c_{1,1} := \mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}_1, \mathsf{m}, \langle\mathcal{P}\rangle, B, 1),$$

   and the responses as

   $$s_{1,0} := c_{1,0} \cdot x_{1,b_1} + r_{1,0}, \quad s_{1,1} := c_{1,1} \cdot x_{1,1-b_1} + r_{1,1}.$$

   Observe that the bit b_1 determines the link between the responses, challenges, and public keys. Finally, the user sends $\mathsf{pm}_{2,1} := (s_{1,0}, s_{1,1}, \varphi_{1,0}, \varphi_{1,1})$ to all users.

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. Here, message $\mathsf{pm}_{2,i}$ is sent by user i and has the form $\mathsf{pm}_{2,i} = (s_{i,0}, s_{i,1}, \varphi_{i,0}, \varphi_{i,1})$. To compute the final signature, users aggregate the responses and commitment randomness as follows:

   $$s_0 := \sum_{i\in[N]} s_{i,0}, \; s_1 := \sum_{i\in[N]} s_{i,1}, \; \varphi_0 := \bigoplus_{i\in[N]} \varphi_{i,0}, \; \varphi_1 := \bigoplus_{i\in[N]} \varphi_{i,1}.$$

   They define $\sigma_0 := (\mathsf{com}_0, \varphi_0, s_0), \sigma_1 := (\mathsf{com}_1, \varphi_1, s_1)$ and output the final signature $\sigma := (\sigma_0, \sigma_1, B)$.

Verification. For verification (algorithm Ver), let $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ be a set of public keys, $\mathsf{m} \in \{0,1\}^*$ be a message, and $\sigma = (\sigma_0, \sigma_1, B)$ be a signature. To verify σ, we write $B = b_1 \ldots b_N, \sigma_0 = (\mathsf{com}_0, \varphi_0, s_0)$ and $\sigma_1 = (\mathsf{com}_1, \varphi_1, s_1)$. Further, we write the public keys pk_i as $\mathsf{pk}_i = (X_{i,0}, X_{i,1})$. We reconstruct the commitment keys $\mathsf{ck}_0 := \mathsf{H}(0, \langle\mathcal{P}\rangle, \mathsf{m}), \mathsf{ck}_1 := \mathsf{H}(1, \langle\mathcal{P}\rangle, \mathsf{m})$, and the user specific challenges

$$c_{i,0} := \mathsf{H}_c(\mathsf{pk}_i, \mathsf{com}_0, \mathsf{m}, \langle\mathcal{P}\rangle, B, 0), \quad c_{i,1} := \mathsf{H}_c(\mathsf{pk}_i, \mathsf{com}_1, \mathsf{m}, \langle\mathcal{P}\rangle, B, 1).$$

Then, we output 1 if and only if the following two equations hold:

$$\mathsf{com}_0 = \mathsf{Com}\left(\mathsf{ck}_0, \mathsf{F}(s_0) - \sum_{i=1}^{N} c_{i,0} \cdot X_{i,b_i}; \varphi_0\right)$$
$$\mathsf{com}_1 = \mathsf{Com}\left(\mathsf{ck}_1, \mathsf{F}(s_1) - \sum_{i=1}^{N} c_{i,1} \cdot X_{i,1-b_i}; \varphi_1\right).$$

Lemma 2. *Let* LF *be a linear function family. Let* CMT *be a* $(\varepsilon_\mathsf{b}, \varepsilon_\mathsf{g}, \varepsilon_\mathsf{t})$*-special commitment scheme for* LF*. Then* $\mathsf{MS}_t[\mathsf{LF}, \mathsf{CMT}]$ *is complete.*

The proof is an easy calculation and is given in Supplementary Material Section B.

Theorem 2. *Let* LF *be a* ε_l*-lossiness admitting linear function family. Let* CMT *be a* $(\varepsilon_\mathsf{b}, \varepsilon_\mathsf{g}, \varepsilon_\mathsf{t})$*-special commitment scheme for* LF*. Further, let* $\mathsf{H}\colon \{0,1\}^* \to \mathcal{K}, \mathsf{H}_b\colon \{0,1\}^* \to \{0,1\}, \mathsf{H}_c\colon \{0,1\}^* \to \mathcal{S}$ *be random oracles. Then* $\mathsf{MS}_t[\mathsf{LF}, \mathsf{CMT}]$ *is* $\mathsf{MS\text{-}EUF\text{-}CMA}$ *secure.*

Concretely, for any PPT algorithm $\mathcal{A}$ *that makes at most* $Q_\mathsf{H}, Q_{\mathsf{H}_b}, Q_{\mathsf{H}_c}, Q_S$ *queries to oracles* $\mathsf{H}, \mathsf{H}_b, \mathsf{H}_c, \textsc{Sig}_0$*, respectively, there are PPT algorithms* $\mathcal{B}, \mathcal{B}'$ *with* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A}), \mathbf{T}(\mathcal{B}') \approx \mathbf{T}(\mathcal{A})$ *and*

$$\mathsf{Adv}^{\mathsf{MS\text{-}EUF\text{-}CMA}}_{\mathcal{A},\mathsf{MS}_t[\mathsf{LF},\mathsf{CMT}]}(\lambda) \le \frac{Q_{\mathsf{H}_b}}{2^\lambda} + 4\varepsilon_\mathsf{g} + 2Q_S\varepsilon_\mathsf{t} + 2Q_\mathsf{H}Q_{\mathsf{H}_c}\varepsilon_\mathsf{b} + 2Q_{\mathsf{H}_c}\varepsilon_\mathsf{l} + 2 \cdot \mathsf{Adv}^{Q_\mathsf{H}\text{-}\mathsf{keydist}}_{\mathcal{B},\mathsf{CMT}}(\lambda) + 2 \cdot \mathsf{Adv}^{\mathsf{keydist}}_{\mathcal{B}',\mathsf{LF}}(\lambda).$$

Proof. Set $\mathsf{MS} := \mathsf{MS}_t[\mathsf{LF}, \mathsf{CMT}]$. Let $\mathcal{A}$ be a PPT algorithm as in the statement. We prove the claim via a sequence of games $\mathbf{G}_0$-$\mathbf{G}_8$. The games are formally presented in Figures 6 to 8, and we describe and analyze them verbally. For each game $\mathbf{G}_i, i \in [8]$, we define

$$\mathsf{Adv}_i := \Pr[\mathbf{G}_i \Rightarrow 1].$$

Game $\mathbf{G}_0$: We define $\mathbf{G}_0$ to be exactly as $\mathbf{MS\text{-}EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{MS}}$, with the following modification: The adversary $\mathcal{A}$ does not get access to oracle $\textsc{Sig}_2$. Note that in MS, algorithm Sig_2 does not make any use of the secret key or a secret state and can be publicly run using the messages output in Sig_0 and Sig_1. Therefore, for any adversary in the original game, there is an adversary in game $\mathbf{G}_0$ that simulates oracle $\textsc{Sig}_2$ and has the same advantage.

Before we proceed, let us describe game $\mathbf{G}_0$ in more detail to fix some notation. In the beginning, the game samples parameters $\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda)$. It also samples a public key $\mathsf{pk}^* = (X_{1,0}, X_{1,1}) = (\mathsf{F}(x_{1,0}), \mathsf{F}(x_{1,1}))$ for a secret key $\mathsf{sk}^* = (x_{1,0}, x_{1,1}, \mathsf{seed}_1)$ with $x_{1,0}, x_{1,1} \xleftarrow{\$} \mathcal{D}, \mathsf{seed}_1 \xleftarrow{\$} \{0,1\}^\lambda$. Then, it runs $\mathcal{A}$ on input $\mathsf{par}, \mathsf{pk}^*$ with access to the following oracles:

- Signing oracles $\textsc{Sig}_0, \textsc{Sig}_1$: The oracles simulate algorithms Sig_0 and Sig_1 on secret key sk^*, respectively. Here, $\mathcal{A}$ can submit a query $\textsc{Sig}_0(\mathcal{P}, \mathsf{m})$ to start a new interaction in which message m is signed for public keys $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$. We assume that $\mathsf{pk}^* = \mathsf{pk}_1$, and the oracle adds $(\mathcal{P}, \mathsf{m})$ to a list $\mathcal{L}$.
- Random oracles $\mathsf{H}, \mathsf{H}_b, \mathsf{H}_c$: The random oracles H, H_c are simulated honestly via lazy sampling. To this end, the game holds maps h, h_c that map the inputs of the respective random oracles to their outputs. Random oracle H_b, however, is simulated by forwarding the query to an internal oracle $\bar{\mathsf{H}}_b$ with the same interface. This oracle holds a similar map $\hat{h}_b$, is kept internally by the game, and is not provided to the adversary. Looking ahead, this indirection allows us to distinguish queries to H_b that some of the following games issue from the queries that the adversary issues.

In the end, $\mathcal{A}$ outputs a forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$. The game outputs 1 if and only if $\mathsf{pk}^* \in \mathcal{P}^*$,$(\mathcal{P}^*, \mathsf{m}^*) \notin \mathcal{L}$, and $\mathsf{Ver}(\mathcal{P}^*, \mathsf{m}^*, \sigma^*) = 1$. Without loss of generality, we assume that the public key pk^* is equal to pk_1 for $\mathcal{P}^* = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$. To fix notation, write $\sigma^* = (\sigma_0^*, \sigma_1^*, B^*)$, $B^* = b_1^* \ldots b_N^*$ and $\sigma_0^* = (\mathsf{com}_0^*, \varphi_0^*, s_0^*), \sigma_1^* = (\mathsf{com}_1^*, \varphi_1^*, s_1^*)$. Clearly, we have

$$\mathsf{Adv}_0 = \mathsf{Adv}^{\mathsf{MS\text{-}EUF\text{-}CMA}}_{\mathcal{A},\mathsf{MS}_t[\mathsf{LF},\mathsf{CMT}]}(\lambda).$$

Game $\mathbf{G}_1$: In game $\mathbf{G}_1$, we add an abort. Namely, the game sets $\mathsf{bad} := 1$, and aborts, if the adversary makes a random oracle query $\mathsf{H}_b(\mathsf{seed}_1, \cdot)$. Note that this does not include the queries that are made by the game itself, as these are done using oracle $\bar{\mathsf{H}}_b$ directly. As the only information about seed_1 that $\mathcal{A}$ gets are the values of $\mathsf{H}_b(\mathsf{seed}_1, \cdot)$, and seed_1 is sampled uniformly at random from $\{0,1\}^\lambda$, we can upper bound the probability of bad by $Q_{\mathsf{H}_b}/2^\lambda$. Therefore, we have

$$|\mathsf{Adv}_0 - \mathsf{Adv}_1| \leq \Pr[\mathsf{bad}] \leq \frac{Q_{\mathsf{H}_b}}{2^\lambda}.$$

Game $\mathbf{G}_2$: In game $\mathbf{G}_2$, we restrict the winning condition. Namely, the game outputs 0, if the forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$ output by $\mathcal{A}$ satisfies $b_1^* \neq 1 - \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P}^* \rangle, \mathsf{m}^*)$. Recall that b_1^* is the bit related to $\mathsf{pk}_1 = \mathsf{pk}^*$ that is included in the signature σ^*. Assuming $\mathbf{G}_1$ outputs 1, we know that $(\mathcal{P}^*, \mathsf{m}^*) \notin \mathcal{L}$. Therefore, $\mathcal{A}$ can only get information about the bit $\bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P}^* \rangle, \mathsf{m}^*)$, if it queries the wrapper random oracle H_b at this position. However, in this case $\mathbf{G}_1$ would set $\mathsf{bad} := 1$ and abort. Thus, the view of $\mathcal{A}$ is independent of bit $\bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P}^* \rangle, \mathsf{m}^*)$. We obtain

$$\mathsf{Adv}_2 = \Pr[\mathbf{G}_2 \Rightarrow 1] = \Pr\left[\mathbf{G}_1 \Rightarrow 1 \wedge b_1^* = 1 - \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P}^* \rangle, \mathsf{m}^*)\right] = \frac{1}{2}\mathsf{Adv}_1.$$

Game $\mathbf{G}_3$: In game $\mathbf{G}_3$, the game aborts if $(\mathsf{par}, x_{1,1}) \notin \mathsf{Good}$, where Good is as in the definition of a special commitment scheme. It is clear that

$$|\mathsf{Adv}_2 - \mathsf{Adv}_3| \leq \Pr[(\mathsf{par}, x_{1,1}) \notin \mathsf{Good}] \leq \varepsilon_{\mathsf{g}}.$$

Game $\mathbf{G}_4$: In game $\mathbf{G}_4$, we change the behavior of random oracle H. Recall that before, to answer a query $\mathsf{H}(b, \langle \mathcal{P} \rangle, \mathsf{m})$ for which the hash value has not been defined, a key $\mathsf{ck} \xleftarrow{\$} \mathcal{K}$ was sampled and returned. In this game, the oracle instead distinguishes two cases. In the first case, if $b = 1 - \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P} \rangle, \mathsf{m})$, the game samples $(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, X_{1,1})$. It also stores $tr[\langle \mathcal{P} \rangle, \mathsf{m}] := \mathsf{td}$, where tr is a map. In the second case, if $b = \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P} \rangle, \mathsf{m})$, it samples $\mathsf{ck} \leftarrow \mathsf{BGen}(\mathsf{par})$. In both cases, ck is returned as before. To see that $\mathbf{G}_3$ and $\mathbf{G}_4$ are indistinguishable, we first note that for the first case, the distribution of ck stays the same. This is because we can assume $(\mathsf{par}, x_{1,1}) \in \mathsf{Good}$ due to the previous change. The keys returned in the second case are indistinguishable by the multi-key indistinguishability of CMT. More precisely, we give a reduction $\mathcal{B}$ against the multi-key indistinguishability of CMT that interpolates between $\mathbf{G}_3$ and $\mathbf{G}_4$. The reduction gets as input $\mathsf{par}, x_{1,1}$ and Q_{H} commitment keys $\mathsf{ck}_1, \ldots, \mathsf{ck}_{Q_{\mathsf{H}}}$. It simulates $\mathbf{G}_3$ for $\mathcal{A}$ with par while embedding the commitment keys in random oracle responses for queries $\mathsf{H}(b, \langle \mathcal{P} \rangle, \mathsf{m})$ with $b = 1 - \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P} \rangle, \mathsf{m})$. In the end, it outputs whatever the game outputs[4] . We have

$$|\mathsf{Adv}_3 - \mathsf{Adv}_4| \leq \mathsf{Adv}^{Q_{\mathsf{H}}\text{-}\mathsf{keydist}}_{\mathcal{B},\mathsf{CMT}}(\lambda).$$

Game $\mathbf{G}_5$: In game $\mathbf{G}_5$, we change the signing oracles $\textsc{Sig}_0, \textsc{Sig}_1$. Our goal is to eliminate the use of the secret key component $x_{1,1}$. Recall that in previous games, oracle $\textsc{Sig}_0$ derived a bit $b_1 := \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P} \rangle, \mathsf{m})$ and sampled random $r_{1,0}, r_{1,1}$ and $\varphi_{1,0}, \varphi_{1,1}$. Then, these were used to compute commitments $\mathsf{com}_{1,0}, \mathsf{com}_{1,1}$, which where then output together with b_1. Then, in oracle $\textsc{Sig}_1$ the values $s_{1,0}, s_{1,1}$ were computed using the secret keys $x_{1,b_1}, x_{1,1-b_1}$, respectively.

In this game, we change how the commitment $\varphi_{1,1-b_1}$ and the value $s_{1,1-b_1}$ is computed to eliminate the dependency on $x_{1,1}$. Namely, in oracle $\textsc{Sig}_0$, we do not compute $r_{1,1-b_1}, \varphi_{1,1-b_1}$ and $R_{1,1-b_1}$ anymore. Instead, we compute the commitment $\mathsf{com}_{1,1-b_1}$ via

$$\mathsf{td} := tr[\langle \mathcal{P} \rangle, , \mathsf{m}], \quad (\mathsf{com}_{1,1-b_1}, St) \leftarrow \mathsf{TCom}(\mathsf{ck}_{1-b_1}, \mathsf{td}).$$

[4]Note that at this point, it was important that we introduced the oracle $\bar{\mathsf{H}}_b$. This is because otherwise, if we queried $\mathsf{H}_b(\mathsf{seed}_1, \cdot)$ in oracle H, game $\mathbf{G}_3$ would always output 0 and the games would not be indistinguishable.

Note that $\mathsf{ck}_{1-b_1} = \mathsf{H}(1-b_1, \langle\mathcal{P}\rangle, \mathsf{m})$, and therefore ck_{1-b_1} and td were generated using $\mathsf{TGen}(\mathsf{par}, X_{1,1})$ due to the change in $\mathbf{G}_4$. Later, in oracle $\textsc{Sig}_1$, we derive

$$(\varphi_{1,1-b_1}, R_{1-b_1}, s_{1,1-b_1}) \leftarrow \mathsf{TCol}(St, c_{1,1-b_1}).$$

Then, message $\mathsf{pm}_{2,1} := (s_{1,0}, s_{1,1}, \varphi_{1,0}, \varphi_{1,1})$ is output as before.

We can easily argue indistinguishability by using the special trapdoor property of CMT Q_{S_0} many times and get

$$|\mathsf{Adv}_4 - \mathsf{Adv}_5| \leq Q_S \varepsilon_{\mathtt{t}}.$$

Game $\mathbf{G}_6$: Here we do not abort if $(\mathsf{par}, x_{1,1}) \notin \mathsf{Good}$ anymore. That is, we revert the change introduced in $\mathbf{G}_3$. It is clear that

$$|\mathsf{Adv}_5 - \mathsf{Adv}_6| \leq \Pr\left[(\mathsf{par}, x_{1,1}) \notin \mathsf{Good}\right] \leq \varepsilon_{\mathtt{g}}.$$

Game $\mathbf{G}_7$: In game $\mathbf{G}_7$, we change how the public key component $X_{1,1}$ is computed. Recall that before, $X_{1,1}$ is computed as $X_{1,1} := \mathsf{F}(x_{1,1})$ for $x_{1,1} \xleftarrow{\$} \mathcal{D}$. Also, note that due to the previous changes, the value $x_{1,1}$ is not used anymore. In $\mathbf{G}_7$, we sample $X_{1,1} \xleftarrow{\$} \mathcal{R}$. A direct reduction $\mathcal{B}'$ against the key indistinguishability of the lossiness admitting linear function family LF shows indistinguishability of $\mathbf{G}_6$ and $\mathbf{G}_7$. Concretely, $\mathcal{B}'$ gets par and $X_{1,1}$ as input, and simulates $\mathbf{G}_6$ for $\mathcal{A}$. In the end, it outputs whatever the game outputs. We have

$$|\mathsf{Adv}_6 - \mathsf{Adv}_7| \leq \mathsf{Adv}^{\mathsf{keydist}}_{\mathcal{B}',\mathsf{LF}}(\lambda).$$

Game $\mathbf{G}_8$: In $\mathbf{G}_8$, we change how H_c is executed. Concretely, consider a query $\mathsf{H}_c(\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B, b)$ with $\mathsf{pk} = \mathsf{pk}^*$ and $b = \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle\mathcal{P}\rangle, \mathsf{m})$. For these queries, the game now runs $R \leftarrow \mathsf{Ext}(\mathsf{H}(b, \langle\mathcal{P}\rangle, \mathsf{m}), \mathsf{com})$ and stores $r[\mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B] := R$, where r is another map. Here, Ext is the (unbounded) extractor for the statistical binding property of CMT. The rest of the oracle does not change. Note that for b of this form, the value $\mathsf{ck} = \mathsf{H}(b, \langle\mathcal{P}\rangle, \mathsf{m})$ is sampled as $\mathsf{ck} \leftarrow \mathsf{BGen}(\mathsf{par})$ (cf. $\mathbf{G}_4$). We also slightly change the winning condition of the game. Namely, in $\mathbf{G}_8$, consider the forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$ with $\sigma^* = (\sigma_0^*, \sigma_1^*, B^*)$, $B^* = b_1^* \dots b_N^*$, and let $R_0^*, R_1^* \in \mathcal{R}$ be the values that are computed during the execution of $\mathsf{Ver}(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$. The game returns 0 if $R^*_{1-b_1^*} \neq r[\mathsf{com}^*_{1-b_1^*}, \mathsf{m}^*, \langle\mathcal{P}^*\rangle, B^*]$.

We claim that indistinguishability of $\mathbf{G}_7$ and $\mathbf{G}_8$ can be argued using the statistical binding property of CMT. To see this, assume that $\mathbf{G}_7$ outputs 1. Then, due to the change in $\mathbf{G}_2$, we know that $1 - b_1^* = \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle\mathcal{P}^*\rangle, \mathsf{m}^*)$. Therefore, in the corresponding query $\mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}^*_{1-b_1^*}, \mathsf{m}^*, \langle\mathcal{P}^*\rangle, B^*, 1-b_1^*)$ algorithm Ext was run and the value $r[\mathsf{com}^*_{1-b_1^*}, \mathsf{m}^*, \langle\mathcal{P}^*\rangle, B^*]$ is defined. Next, by definition of Ver, we have $\mathsf{Com}(\mathsf{ck}_{1-b_1^*}, R^*_{1-b_1^*}; \varphi^*_{1-b_1^*}) = \mathsf{com}^*_{1-b_1^*}$. Therefore, if $R^*_{1-b_1^*} \neq r[\mathsf{com}^*_{1-b_1^*}, \mathsf{m}^*, \langle\mathcal{P}^*\rangle, B^*]$, we have a contradiction to the statistical binding property of CMT. More precisely, we sketch an (unbounded) reduction from the statistical binding property. Namely, this reduction gets as input par and a commitment key ck^*. Then, the reduction guesses $i_{\mathsf{H}} \xleftarrow{\$} [Q_{\mathsf{H}}]$ and $i_{\mathsf{H}_c} \xleftarrow{\$} [Q_{\mathsf{H}_c}]$. It simulates game $\mathbf{G}_8$ honestly, except for query i_{H} to random oracle H and query i_{H_c} to random oracle H_c. If it had to sample a $\mathsf{ck} \leftarrow \mathsf{BGen}(\mathsf{par})$ in the former query, it instead responds with ck^*. Similarly, if it had to run Ext in the latter query, it outputs com to the binding experiment. If these queries are the queries of interest (i.e. query i_{H} was used to derive $\mathsf{ck}_{1-b_1^*}$ and query i_{H_c} was used to derive $c^*_{1,1-b_1^*}$) for the forgery, and $R^*_{1-b_1^*} \neq r[\mathsf{com}^*_{1-b_1^*}, \mathsf{m}^*, \langle\mathcal{P}^*\rangle, B^*]$, then the reduction outputs $R^*_{1-b_1^*}; \varphi^*_{1-b_1^*}$. Otherwise, it outputs $\perp$. It is easy to see that if the reduction guesses the correct queries and the bad event separating $\mathbf{G}_7$ and $\mathbf{G}_8$ occurs, then it breaks the statistical binding property. As the view of $\mathcal{A}$ is as in $\mathbf{G}_8$, and independent of $(i_{\mathsf{H}}, i_{\mathsf{H}_c})$, we obtain

$$|\mathsf{Adv}_7 - \mathsf{Adv}_8| \leq Q_{\mathsf{H}} Q_{\mathsf{H}_c} \varepsilon_{\mathtt{b}}.$$

Finally, we use lossy soundness of LF to bound the probability that $\mathbf{G}_8$ outputs 1. To do that, we give an unbounded reduction from the lossy soundness experiment, which is as follows.

- The reduction gets $\mathsf{par}, X_{1,1}$ as input. It samples $\hat{i} \xleftarrow{\$} [Q_{\mathsf{H}_c}]$. Then, it simulates $\mathbf{G}_8$ honestly until $\mathcal{A}$ outputs a forgery, except for query $\hat{i}$ to oracle H_c.
- Consider this query $\mathsf{H}_c(\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B, b)$. The reduction aborts its execution, if the hash value for this query is already defined, or if $\mathsf{pk} \neq \mathsf{pk}^* \vee b \neq \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle\mathcal{P}\rangle, \mathsf{m})$. Otherwise, it runs $\hat{R} \leftarrow \mathsf{Ext}(\mathsf{H}(b, \langle\mathcal{P}\rangle, \mathsf{m}), \mathsf{com})$ as in $\mathbf{G}_8$. Then, it parses $\mathcal{P} = \{\mathsf{pk}_1, \dots, \mathsf{pk}_N\}$ and $B = b_1 \dots b_N$. It

parses $\mathsf{pk}_i = (X_{i,0}, X_{i,1})$ for each $i \in [N]$, and it sets $c_{i,b} = \mathsf{H}_c(\mathsf{pk}_i, \mathsf{com}, \mathsf{m}, \langle \mathcal{P} \rangle, B, b)$ for each $i \in [N] \setminus \{1\}$. Next, it defines

$$R := \hat{R} + \sum_{i=2}^{N} c_{i,b} \cdot X_{i,\hat{b}_i},$$

where $\hat{b}_i := (b + b_i) \mod 2$. It outputs R to the lossy soundness game and obtains a value c in return. Then, it sets $h_c[\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle \mathcal{P} \rangle, B, b] := c$ and continues the simulation as in $\mathbf{G}_8$.

- When the reduction gets the forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$ from $\mathcal{A}$, it runs all the verification steps in $\mathbf{G}_8$. Additionally, it checks if the value $\mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}^*_{1-b_1^*}, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*, 1 - b_1^*)$ was defined during query $\hat{i}$ to H_c. If this is not the case, it aborts its execution. Otherwise, it returns $s := s^*_{1-b_1^*}$ to the lossy soundness game.

It is clear that the view of $\mathcal{A}$ is independent of the index $\hat{i}$ until a potential abort of the reduction. Also, if the reduction does not abort its execution, it perfectly simulates game $\mathbf{G}_8$ for $\mathcal{A}$. Thus, it remains to show that if $\mathbf{G}_8$ outputs 1, then the values output by the reduction satisfy $\mathsf{F}(s) - c \cdot X_{1,1} = R$. Once we have shown this, it follows that

$$\mathsf{Adv}_8 \leq Q_{\mathsf{H}_c} \varepsilon_{\mathsf{I}}.$$

To show the desired property, assume that the reduction does not abort and $\mathbf{G}_8$ outputs 1. Then, define $\hat{b}_i^* = (1 - b_1^* + b_i) \mod 2$ for all $i \in [N]$. Note that $\hat{b}_i^* = 1$. Due to the change in $\mathbf{G}_2$, we have

$$b = 1 - b_1^* = \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P}^* \rangle, \mathsf{m}^*).$$

As the reduction guessed the right query and does not abort, we have

$$c^*_{1,1-b_1^*} = \mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}^*_{1-b_1^*}, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*, 1 - b_1^*) = c.$$

Due to the change in $\mathbf{G}_8$, we have

$$\mathsf{F}(s^*_{1-b_1^*}) - \sum_{i=1}^{N} c^*_{i,1-b_1^*} \cdot X_{i,\hat{b}_i^*} = R^*_{1-b_1^*} = \hat{R}.$$

Therefore, we have

$$\begin{aligned}
\mathsf{F}(s) - c \cdot X_{1,1} &= \mathsf{F}(s^*_{1-b_1^*}) - c^*_{1,1-b_1^*} \cdot X_{1,1} \\
&= \mathsf{F}(s^*_{1-b_1^*}) - \sum_{i=1}^{N} c^*_{i,1-b_1^*} \cdot X_{i,\hat{b}_i^*} + \sum_{i=2}^{N} c^*_{i,1-b_1^*} \cdot X_{i,\hat{b}_i^*} \\
&= \hat{R} + \sum_{i=2}^{N} c^*_{i,1-b_1^*} \cdot X_{i,\hat{b}_i^*} = R.
\end{aligned}$$

Concluded. □

4 Instantiation

In this section, we show how to instantiate the building blocks that are needed for our constructions in the previous section. Concretely, we give a linear function family and a commitment scheme based on the DDH assumption. Then, we also discuss the efficiency of the resulting multi-signature schemes.

4.1 Linear Function Family

We make use of the well-known [KMP16] linear function family $\mathsf{LF}_{\mathsf{DDH}} = (\mathsf{Gen}, \mathsf{F})$ based on the DDH assumption. Precisely, let GGen be an algorithm that on input 1^λ outputs the description of a prime order group $\mathbb{G}$ of order p with generator g. Then, Gen runs GGen and outputs[5] $\mathsf{par} := (g, h) \in \mathbb{G}^2$ for $h \xleftarrow{\$} \mathbb{G}$. Then, the set of scalars, domain, range, and function $\mathsf{F}(\mathsf{par}, \cdot)$ are given as follows:

$$\mathcal{S} := \mathbb{Z}_p, \quad \mathcal{D} := \mathbb{Z}_p, \quad \mathcal{R} := \mathbb{G} \times \mathbb{G}, \quad \mathsf{F}(\mathsf{par}, x) := (g^x, h^x).$$

It is easily verified that this constitutes a linear function family.

[5] We omit the description of $\mathbb{G}$ from par to make the presentation concise.

Lemma 3. *Assuming that the* DDH *assumption holds relative to* GGen*, the linear function family* $\mathsf{LF}_{\mathsf{DDH}}$ *is* ε_{l}*-lossiness admitting, with* $\varepsilon_{\mathsf{l}} \leq 3/p$*. 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*

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

Proof. First, note that the definition of key indistinguishability matches exactly the DDH assumption relative to GGen. Next, we argue that lossy soundness holds. We have to bound the probability

$$\Pr\left[(g^s \cdot X_1^{-c}, h^s \cdot X_2^{-c}) = (R_1, R_2) \;\middle|\; \begin{array}{l}(g,h) \leftarrow \mathsf{Gen}(1^\lambda),\ (X_1, X_2) \xleftarrow{\$} \mathbb{G}^2,\\ (St, (R_1, R_2)) \leftarrow \mathcal{A}((g,h),(X_1,X_2)),\\ c \xleftarrow{\$} \mathbb{Z}_p,\ s \leftarrow \mathcal{A}(St, c)\end{array}\right].$$

The probability that $h = g^0$ is at most $1/p$. Thus, we assume that h is a generator of $\mathbb{G}$. Write $X_1 = g^{x_1}$ and $X_2 = h^{x_2}$. With probability at most $1/p$ we have $x_1 = x_2$. Assume that $x_1 \neq x_2$. We claim that with these assumptions, the probability that we have to bound is at most $1/p$. To see this, assume that there is some (R_1, R_2) such that there exist two different $c \neq c'$ in $\mathbb{Z}_p$, such that there exists a $s, s' \in \mathbb{Z}_p$ with

$$\left(g^s \cdot X_1^{-c}, h^s \cdot X_2^{-c}\right) = (R_1, R_2) \text{ and } \left(g^{s'} \cdot X_1^{-c'}, h^{s'} \cdot X_2^{-c'}\right) = (R_1, R_2).$$

Then, we can combine both equations and rearrange terms to get

$$\left(g^{(s-s')/(c-c')}, h^{(s-s')/(c-c')}\right) = (X_1, X_2),$$

contradicting our assumption that $x_1 \neq x_2$. The claim follows. □

Lemma 4. *Linear function family* $\mathsf{LF}_{\mathsf{DDH}}$ *satisfies* $\varepsilon_{\mathsf{al}}$*-aggregation lossy soundness with* $\varepsilon_{\mathsf{al}} \leq 4/p$*.*

Proof. Let $\mathcal{A}$ be any unbounded algorithm. We have to bound the probability that

$$\left(g^s \cdot \tilde{X}_1^{-c}, h^s \cdot \tilde{X}_2^{-c}\right) = (R_1, R_2),$$

where we consider the following experiment. First, $(g,h) \leftarrow \mathsf{Gen}(1^\lambda)$, $(X_1, X_2) \xleftarrow{\$} \mathbb{G}^2$ is sampled and g, h, X_1, X_2 are given to $\mathcal{A}$. Then, $\mathcal{A}$ outputs pairs of group elements and exponents $((X_{2,1}, X_{2,2}), a_2)$, $\ldots, ((X_{N,1}, X_{N,2}), a_N)$. Next, exponent $a_1 \xleftarrow{\$} \mathbb{Z}_p$ are sampled and $\tilde{X}_1, \tilde{X}_2$ are defined as

$$\left(\tilde{X}_1, \tilde{X}_2\right) := \left(\prod_{i=1}^{N} X_{i,1}^{a_i}, \prod_{i=1}^{N} X_{i,2}^{a_i}\right).$$

Then, $\mathcal{A}$ outputs (R_1, R_2) on input a_1. A challenge $c \xleftarrow{\$} \mathbb{Z}_p$ is sampled and $\mathcal{A}$ outputs s on input c.

The probability that $h = g^0$ is at most $1/p$. Thus, we assume that h is a generator of $\mathbb{G}$. Looking at the proof of Lemma 3, we see that it is sufficient to argue that with high probability, $\left(\tilde{X}_1, \tilde{X}_2\right)$ is not of the form $(g^{\tilde{x}}, h^{\tilde{x}})$ for any $\tilde{x} \in \mathbb{Z}_p$. In other words, we have to show that with high probability, the pair $\left(\tilde{X}_1, \tilde{X}_2\right)$ is not in the image of F. Conditioned on that, as in the proof of Lemma 3, the probability above can be bounded by $1/p$.

To show this, we fix the exponents $x_{i,j} \in \mathbb{Z}_p$ such that $X_{i,1} = g^{x_{i,1}}$ and $X_{i,2} = h^{x_{i,2}}$. The probability that $x_{1,1} = x_{1,2}$ is at most $1/p$. From now on, we condition on $x_{1,1} \neq x_{1,2}$. The pair $\left(\tilde{X}_1, \tilde{X}_2\right)$ is not in the image of F if and only if

$$\sum_{i=1}^{N} a_i x_{i,1} = \sum_{i=1}^{N} a_i x_{i,2}.$$

This is equivalent to

$$a_1 = \frac{\sum_{i=2}^{N} a_i x_{i,2} - \sum_{i=2}^{N} a_i x_{i,2}}{x_{1,1} - x_{1,2}}.$$

As a_1 is sampled uniformly over $\mathbb{Z}_p$ after $\mathcal{A}$ choses the $x_{i,j}$ and a_i, $i > 2$, the above holds with probability at most $1/p$, and the claim follows. □

4.2 Commitment Scheme

We give a special trapdoor commitment scheme $\mathsf{CMT}_{\mathsf{DDH}} = (\mathsf{BGen}, \mathsf{TGen}, \mathsf{Com}, \mathsf{TCom}, \mathsf{TCol})$ for the linear function family $\mathsf{LF}_{\mathsf{DDH}}$. For given parameters of $\mathsf{LF}_{\mathsf{DDH}}$, the commitment scheme has key space $\mathcal{K} := \mathbb{G}^{3\times 3}$ and message space $\mathcal{D} = \mathbb{G} \times \mathbb{G}$. It has randomness space $\mathcal{G} = \mathbb{Z}_p^3$ and commitment space $\mathcal{H} = \mathbb{G}^3$. Both are associated with the natural componentwise group operations. We describe the algorithms of the 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

$$\mathsf{ck} := \mathbf{A} := \begin{pmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} := \begin{pmatrix} g_1 & g_1^a & g_1^b \\ g_2 & g_2^a & g_2^b \\ g_3 & g_3^a & g_3^b \end{pmatrix} \in \mathbb{G}^{3\times 3}.$$

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

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

  Next, set

$$\mathsf{td} := (\mathbf{D}, X_1, X_2), \text{ for } \mathbf{D} := \begin{pmatrix} d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,1} & d_{2,2} & d_{2,3} \\ d_{3,1} & d_{3,2} & d_{3,3} \end{pmatrix} \in \mathbb{Z}_p^{3\times 3}.$$

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

$$\mathsf{com} := (C_0, C_1, C_2), \text{ for } \begin{pmatrix} C_0 \\ C_1 \\ C_2 \end{pmatrix} := \begin{pmatrix} & A_{1,1}^{\alpha} \cdot A_{1,2}^{\beta} \cdot A_{1,3}^{\gamma} \\ R_1 \cdot & A_{2,1}^{\alpha} \cdot A_{2,2}^{\beta} \cdot A_{2,3}^{\gamma} \\ R_2 \cdot & A_{3,1}^{\alpha} \cdot A_{3,2}^{\beta} \cdot A_{3,3}^{\gamma} \end{pmatrix}.$$

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

$$\mathsf{com} := (C_0, C_1, C_2), \text{ for } \begin{pmatrix} C_0 \\ C_1 \\ C_2 \end{pmatrix} := \begin{pmatrix} g^{\tau} \\ X_1^{\rho_1} \cdot g^s \\ X_2^{\rho_2} \cdot h^s \end{pmatrix}.$$

- $\mathsf{TCol}(St, c) \to (\varphi, R, s)$: Set $R := (R_1, R_2) := \left(g^s \cdot X_1^{-c}, h^s \cdot X_2^{-c}\right)$. Then, if $\mathbf{D}$ is not invertible, return $\perp$. Otherwise, compute

$$\varphi := (\alpha, \beta, \gamma), \text{ for } \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \mathbf{D}^{-1} \cdot \begin{pmatrix} \tau \\ \rho_1 + c \\ \rho_2 + c \end{pmatrix}.$$

Theorem 3. *If the* DDH *assumption holds relative to* GGen*, then* $\mathsf{CMT}_{\mathsf{DDH}}$ *is a* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-special commitment scheme for* $\mathsf{LF}_{\mathsf{DDH}}$*, with*

$$\varepsilon_{\mathsf{b}} \le 1/p, \quad \varepsilon_{\mathsf{g}} \le 2/p, \quad \varepsilon_{\mathsf{t}} \le 6/p.$$

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*

$$\mathsf{Adv}^{Q\text{-keydist}}_{\mathcal{A},\mathsf{CMT}_{\mathsf{DDH}}}(\lambda) \le \mathsf{Adv}^{\mathsf{uDDH3}}_{\mathcal{B},\mathsf{GGen}}(\lambda) + \frac{6}{p}.$$

The homomorphism property is trivial to check. Next, we define the set Good as in the definition of a special commitment scheme. Namely, we define

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

Clearly, for $(g,h) \leftarrow \mathsf{LF.Gen}(1^\lambda)$ and $x \xleftarrow{\$} \mathbb{Z}_p$, the probability that $((g,h),x) \notin \mathsf{Good}$ is at most $2/p$. Therefore, $\varepsilon_{\mathsf{g}} \le 2/p$. In the following we also need the following observation: If $((g,h),x) \in \mathsf{Good}$, then the elements g, h, g^x, h^x are all generators of $\mathbb{G}$. The rest of proof of the theorem is given in separate lemmas.

Lemma 5. $\mathsf{CMT}_{\mathsf{DDH}}$ *satisfies the uniform keys property of an* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-special commitment scheme for* $\mathsf{LF}_{\mathsf{DDH}}$.

Proof. Let $(\mathsf{par}, x) \in \mathsf{Good}$ for $\mathsf{par} = (g, h)$. Let $(X_1, X_2) = \mathsf{F}(x) = (g^x, h^x)$. Consider the distribution of ck for $(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, (X_1, X_2))$. Then ck has the form

$$\begin{pmatrix} g^{d_{1,1}} & g^{d_{1,2}} & g^{d_{1,3}} \\ X_1^{d_{2,1}} & X_1^{d_{2,2}} & X_1^{d_{2,3}} \\ X_2^{d_{3,1}} & X_2^{d_{3,2}} & X_2^{d_{3,3}} \end{pmatrix} \in \mathbb{G}^{3\times 3}$$

for uniformly random and independent exponents $d_{i,j} \in \mathbb{Z}_p$ $(i, j \in [3])$. As g, X_1, X_2 are generators, we see that ck is uniform over $\mathbb{G}^{3\times 3}$, proving the claim. □

Lemma 6. $\mathsf{CMT}_{\mathsf{DDH}}$ *satisfies the special trapdoor property of an* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-special commitment scheme for* $\mathsf{LF}_{\mathsf{DDH}}$*, where* $\varepsilon_{\mathsf{t}} \leq 6/p$.

Proof. Let $((g, h), x) \in \mathsf{Good}$ and $c \in \mathbb{Z}_p$. Set $(X_1, X_2) := (g^x, h^x)$. We have to show that the distributions $\mathcal{T}_0$ and $\mathcal{T}_1$ of tuples

$$((g, h), \mathbf{A}, \mathbf{D}, X_1, X_2, x, c, (C_0, C_1, C_2), \alpha, \beta, \gamma, R_1, R_2, s)$$

are identical. Here, we have $(\mathbf{A}, \mathbf{D}, X_1, X_2) \leftarrow \mathsf{TGen}(\mathsf{par}, (X_1, X_2))$. The remaining components in $\mathcal{T}_0$ are generated via

$$((C_0, C_1, C_2), St) \leftarrow \mathsf{TCom}(\mathsf{ck}, \mathsf{td}),\ ((\alpha, \beta, \gamma), (R_1, R_2), s) \leftarrow \mathsf{TCol}(St, c),$$

and in $\mathcal{T}_1$ via

$$\begin{aligned} r \xleftarrow{\$} \mathbb{Z}_p,\ R_1 := g^r,\ R_2 := h^r,\ s := c \cdot x + r \\ \alpha, \beta, \gamma \xleftarrow{\$} \mathbb{Z}_p,\ (C_0, C_1, C_2) := \mathsf{Com}(\mathbf{A}, (R_1, R_2); (\alpha, \beta, \gamma)). \end{aligned}$$

First, we make the assumption that in both distributions, the matrix $\mathbf{D}$ has full rank. The probability that this does not hold can easily be bounded by $3/p$.

We can equivalently[6] write $\mathcal{T}_1$ as

$$\begin{aligned} s \xleftarrow{\$} \mathbb{Z}_p,\ R_1 := g^s \cdot X_1^{-c},\ R_2 := h^s \cdot X_2^{-c}, \\ \alpha, \beta, \gamma \xleftarrow{\$} \mathbb{Z}_p,\ (C_0, C_1, C_2) := \mathsf{Com}(\mathbf{A}, (R_1, R_2); (\alpha, \beta, \gamma)). \end{aligned}$$

Using that $\mathbf{D}$ is full rank and g, X_1, X_2 are generators of $\mathbb{G}$, we see that in this distribution, (C_0, C_1, C_2) is uniform over $\mathbb{G}^3$. Therefore, this is identically distributed to the distribution that we get from

$$\begin{aligned} s \xleftarrow{\$} \mathbb{Z}_p,\ R_1 := g^s \cdot X_1^{-c},\ R_2 := h^s \cdot X_2^{-c}, \\ \tau, \rho_1, \rho_2 \xleftarrow{\$} \mathbb{Z}_p,\ (C_0, C_1, C_2) := (g^\tau, X_1^{\rho_1} g^s, X_2^{\rho_2} h^s), \end{aligned}$$

and then finding the unique values (α, β, γ) that satisfy $(C_0, C_1, C_2) = \mathsf{Com}(\mathbf{A}, (R_1, R_2); (\alpha, \beta, \gamma))$. We claim that this can be done using $(\alpha, \beta, \gamma)^t := \mathbf{D}^{-1}(\tau, \rho_1 + c, \rho_2 + c)^t$, which is equivalent to distribution $\mathcal{T}_0$.

To see this, note that $(C_0, C_1, C_2) = \mathsf{Com}(\mathbf{A}, (R_1, R_2); (\alpha, \beta, \gamma))$ is equivalent to

$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} & A_{1,1}^{\alpha} \cdot A_{1,2}^{\beta} \cdot A_{1,3}^{\gamma} \\ R_1 \cdot & A_{2,1}^{\alpha} \cdot A_{2,2}^{\beta} \cdot A_{2,3}^{\gamma} \\ R_1 \cdot & A_{3,1}^{\alpha} \cdot A_{3,2}^{\beta} \cdot A_{3,3}^{\gamma} \end{pmatrix} = \begin{pmatrix} & g^{d_{1,1}\alpha} \cdot g^{d_{1,2}\beta} \cdot g^{d_{1,3}\gamma} \\ g^s \cdot X_1^{-c} \cdot & X_1^{d_{2,1}\alpha} \cdot X_1^{d_{2,2}\beta} \cdot X_1^{d_{2,3}\gamma} \\ h^s \cdot X_2^{-c} \cdot & X_2^{d_{3,1}\alpha} \cdot X_2^{d_{3,2}\beta} \cdot X_2^{d_{3,3}\gamma} \end{pmatrix}.$$

Using the way we generate (C_0, C_1, C_2), we see that the g^s and h^s terms cancel out, and this is equivalent to

$$\begin{pmatrix} g^\tau \\ X_1^{\rho_1} \\ X_2^{\rho_2} \end{pmatrix} = \begin{pmatrix} g^{d_{1,1}\alpha} \cdot g^{d_{1,2}\beta} \cdot g^{d_{1,3}\gamma} \\ X_1^{d_{2,1}\alpha} \cdot X_1^{d_{2,2}\beta} \cdot X_1^{d_{2,3}\gamma} \\ X_2^{d_{3,1}\alpha} \cdot X_2^{d_{3,2}\beta} \cdot X_2^{d_{3,3}\gamma} \end{pmatrix} \iff \begin{pmatrix} \tau \\ \rho_1 + c \\ \rho_2 + c \end{pmatrix} = \mathbf{D} \cdot \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}.$$

This concludes the proof. □

[6] This corresponds to the HVZK property of linear identification protocols.

Lemma 7. $\mathsf{CMT}_{\mathsf{DDH}}$ *satisfies the statistically binding property of an* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-special commitment scheme for* $\mathsf{LF}_{\mathsf{DDH}}$*, with* $\varepsilon_{\mathsf{b}} \leq 1/p$.

Proof. We describe an unbounded algorithm Ext, that takes as input a commitment key $\mathsf{ck} = \mathbf{A} = (A_{i,j})_{i,j} \in \mathbb{G}^{3\times 3}$, and a commitment $\mathsf{com} = (C_0, C_1, C_2) \in \mathbb{G}^3$, and outputs a tuple $R = (R_1, R_2) \in \mathbb{G} \times \mathbb{G}$. It is given as follows:

1. Extract discrete logarithms $\mathbf{c} = (c_0, c_1, c_2)^t \in \mathbb{Z}_p^3$ and $\mathbf{a} = (a_0, a_1, a_2)^t \in \mathbb{Z}_p^3$ such that
$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} g^{c_0} \\ g^{c_1} \\ g^{c_2} \end{pmatrix} \text{ and } \begin{pmatrix} A_{1,1} \\ A_{2,1} \\ A_{3,1} \end{pmatrix} = \begin{pmatrix} g^{a_0} \\ g^{a_1} \\ g^{a_2} \end{pmatrix}.$$
2. If $a_0 = 0$, return $\perp$. Otherwise, let $\mathbf{e}_2 = (0,1,0)^t$ and $\mathbf{e}_3 = (0,0,1)^t$. Note that $\mathbf{a}, \mathbf{e}_2, \mathbf{e}_3$ form a basis of $\mathbb{Z}_p^3$.
3. Write $\mathbf{c}$ as $\mathbf{c} = t\mathbf{a} + r_1\mathbf{e}_2 + r_2\mathbf{e}_3$ for $t, r_1, r_2 \in \mathbb{Z}_p$, and return $(R_1, R_2) := (g^{r_1}, g^{r_2})$.

To finish the proof, let $\mathcal{A}$ be any algorithm. We have to bound the probability

$$\Pr\left[\begin{array}{l|l} \mathsf{Com}(\mathbf{A}, (R_1', R_2'); \varphi') = (C_0, C_1, C_2) & (g,h) \leftarrow \mathsf{LF.Gen}(1^\lambda), \\ \quad \wedge \ (R_1, R_2) \neq (R_1', R_2') & \mathbf{A} \leftarrow \mathsf{BGen}(\mathsf{par}), \\ & ((C_0, C_1, C_2), St) \leftarrow \mathcal{A}(\mathbf{A}), \\ & (R_1, R_2) \leftarrow \mathsf{Ext}(\mathbf{A}, (C_0, C_1, C_2)), \\ & (R_1, R_2', \varphi') \leftarrow \mathcal{A}(St) \end{array}\right].$$

Note that the probability that Ext outputs $\perp$ in this experiment is $1/p$, as $A_{1,1}$ is uniform in $\mathbb{G}$. We assume that Ext does not output $\perp$, and want to show that the above probability conditioned on this event is zero. First, it is easy to see that we have $\mathsf{Com}(\mathbf{A}, (R_1, R_2); (t,0,0)) = (C_0, C_1, C_2)$. Further, assume that $\mathcal{A}$ outputs $(R_1', R_2') = (g^{r_1'}, g^{r_2'})$ and $\varphi' = (\alpha, \beta, \gamma)$ such that

$$\mathsf{Com}(\mathbf{A}, (R_1', R_2'); \varphi') = (C_0, C_1, C_2) = \mathsf{Com}(\mathbf{A}, (R_1, R_2); (t,0,0)).$$

Using the definition of Com and BGen, we see that this implies the vector $(0, r_1 - r_1', r_2 - r_2')^t$ is in the span of $\mathbf{a}$. As $a_0 \neq 0$ this implies that it is the zero vector, showing that $R_1 = R_1'$ and $R_2 = R_2'$. □

Lemma 8. *For any PPT algorithm* $\mathcal{A}$*, there is a PPT algorithm* $\mathcal{B}$ *with* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$ *and*

$$\mathsf{Adv}^{Q\text{-}\mathsf{keydist}}_{\mathcal{A},\mathsf{CMT}_{\mathsf{DDH}}}(\lambda) \leq \mathsf{Adv}^{\mathsf{uDDH3}}_{\mathcal{B},\mathsf{GGen}}(\lambda) + \frac{6}{p}.$$

Proof. Let $\mathcal{A}$ be a PPT algorithm and $Q = \mathsf{poly}(\lambda)$. We have to bound

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

Note that in game $\mathbf{KEYDIST}_{1,\mathsf{CMT}_{\mathsf{DDH}}}$, all commitment keys $\mathbf{A}_i$ are sampled uniformly at random. Therefore, looking at one fixed commitment key $\mathbf{A}_i$, indistinguishability would directly follow from the $\mathsf{uDDH3}$ assumption. To give a tight reduction for any $Q = \mathsf{poly}(\lambda)$, we use the random self-reducibility of $\mathsf{uDDH3}$. Our reduction $\mathcal{B}$ is as follows:

1. $\mathcal{B}$ gets as input $\mathbb{G}, p, g$ and group elements $(h_{i,j})_{i,j\in[3]}$.
2. $\mathcal{B}$ samples $h \xleftarrow{\$} \mathbb{G}$, sets $\mathsf{par} := (g,h)$, and samples $x \xleftarrow{\$} \mathbb{Z}_p$.
3. If $(\mathsf{par}, x) \notin \mathsf{Good}$, $\mathcal{B}$ returns 0, as game $\mathbf{KEYDIST}_{b,\mathsf{CMT}_{\mathsf{DDH}}}$ does.
4. Otherwise, $\mathcal{B}$ prepares commitment keys $\mathsf{ck}_i := \mathbf{A}_i \in \mathbb{G}^{3\times 3}$ for all $i \in [Q]$ as follows:
   (a) For simplicity of notation, write $h_{i,j} := g^{\mathbf{H}_{i,j}}$, i.e. let $\mathbf{H} \in \mathbb{Z}_p^{3\times 3}$ denote the matrix of exponents of $h_{i,j}$. Partition $\mathbf{H}$ into $\mathbf{H} = [\mathbf{H}_0 \mid \mathbf{H}_1]$ for $\mathbf{H}_0 \in \mathbb{Z}_p^{3\times 1}$ and $\mathbf{H}_1 \in \mathbb{Z}_p^{3\times 2}$.
   (b) Reduction $\mathcal{B}$ samples $\mathbf{T}_i \xleftarrow{\$} \mathbb{Z}_p^{3\times 3}$, and $\mathbf{S}_i \xleftarrow{\$} \mathbb{Z}_p^{1\times 2}$. We define
$$\mathbf{D}_i := \mathbf{T}_i\mathbf{H} + [\mathbf{0} \mid \mathbf{T}_i\mathbf{H}_0\mathbf{S}_i].$$

(c) Reduction $\mathcal{B}$ computes $\mathbf{A}_i := g^{\mathbf{D}_i}$, which should be understood componentwise. It is easy to see 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 $\beta \leftarrow \mathcal{A}(\mathsf{par}, x, (\mathsf{ck}_i)_{i\in[Q]})$ and returns β.

Apart from the distribution of the commitment keys $\mathbf{A}_i$, it is clear that $\mathcal{B}$ simulates the games perfectly for $\mathcal{A}$. We claim that if the $h_{i,j}$ are uniform and independent, then $\mathcal{B}$ provides a simulation statistically close to $\mathbf{KEYDIST}_{1,\mathsf{CMT}_{\mathsf{DDH}}}$, i.e. all $\mathbf{A}_i$ are uniform and independent. If on the other hand, there are a, b such that $h_{i,2} := h_{i,1}^a, h_{i,3} := h_{i,1}^b$ for all $i \in [3]$, then $\mathcal{B}$ does the same for $\mathbf{KEYDIST}_{1,\mathsf{CMT}_{\mathsf{DDH}}}$.

For the first claim, assume that the $h_{i,j}$ are uniform and independent, i.e. $\mathbf{H}$ is uniform over $\mathbb{Z}_p^{3\times 3}$. Then with probability at least $1 - 3/p$ it has full rank, i.e. is invertible. Assuming that it has full rank, we see that even if we fix the matrix $\mathbf{H}$, the matrix $\mathbf{T}_i\mathbf{H}$ is uniform over $\mathbb{Z}_p^{3\times 3}$. This implies that $\mathbf{D}_i$ is uniform, showing the first claim.

For the second claim, assume that there are a, b such that $h_{i,2} := h_{i,1}^a, h_{i,3} := h_{i,1}^b$ for all $i \in [3]$. This is equivalent to writing $\mathbf{H} = [\mathbf{H}_0 \mid \mathbf{H}_0\mathbf{R}]$, where $\mathbf{R} = [a, b] \in \mathbb{Z}_p^{1\times 2}$. With probability at least $1 - 1/p^3$, the matrix $\mathbf{H}_0 \in \mathbb{Z}_p^{3\times 1}$ is full rank. We see that

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

If $\mathbf{H}_0$ has full rank, we see that (even for fixed $\mathbf{H}$ of this form) the key $\mathbf{A}_i$ is distributed exactly as a commitment key in $\mathbf{KEYDIST}_{0,\mathsf{CMT}_{\mathsf{DDH}}}$, which finishes the proof. □

4.3 Efficiency

We briefly discuss efficiency of our schemes $\mathsf{MS}_t[\mathsf{LF}, \mathsf{CMT}]$ and $\mathsf{MS}_a[\mathsf{LF}, \mathsf{CMT}]$ presented in Sections 3.2 and 3.3, when instantiated with the linear function family and commitment scheme presented here.

Asymptotics. Denote the size of an element in $\mathbb{G}$ by $\mathsf{size}(\mathbb{G})$, and the size of an element in $\mathbb{Z}_p$ by $\mathsf{size}(\mathbb{Z}_p)$. For our scheme with key aggregation $\mathsf{MS}_a[\mathsf{LF}, \mathsf{CMT}]$ we get the following signature size $|\sigma|$ and communication $|\mathsf{Comm}|$ per signer.

$$|\mathsf{Comm}| = 3\mathsf{size}(\mathbb{G}) + 4\mathsf{size}(\mathbb{Z}_p), \quad |\sigma| = 3\mathsf{size}(\mathbb{G}) + 4\mathsf{size}(\mathbb{Z}_p).$$

For our tight scheme $\mathsf{MS}_t[\mathsf{LF}, \mathsf{CMT}]$ we obtain the following sizes for N signers.

$$|\mathsf{Comm}| = 1 + 6\mathsf{size}(\mathbb{G}) + 8\mathsf{size}(\mathbb{Z}_p), \quad |\sigma| = 6\mathsf{size}(\mathbb{G}) + 8\mathsf{size}(\mathbb{Z}_p) + N.$$

For both schemes, we can further reduce the communication complexity. Namely, instead of sampling the commitment randomness $\varphi_i \in \mathbb{Z}_p^3$ directly, each signer i samples a short seed $\mathsf{seed}_i \xleftarrow{\$} \{0,1\}^\lambda$ and defines $\varphi_i = \mathsf{H}(\mathsf{seed}_i)$, where H is a random oracle. Later, seed_i is sent as an opening instead of φ_i. Our security proofs still go through, using the entropy of seed_1 and by programming $\mathsf{H}(\mathsf{seed}_1)$ after using the equivocation trapdoor. This reduces the per-signer communication complexity to

$$\begin{aligned} \mathsf{MS}_a[\mathsf{LF}, \mathsf{CMT}]: &\quad |\mathsf{Comm}| = 3\mathsf{size}(\mathbb{G}) + \mathsf{size}(\mathbb{Z}_p) + \lambda, \\ \mathsf{MS}_t[\mathsf{LF}, \mathsf{CMT}]: &\quad |\mathsf{Comm}| = 6\mathsf{size}(\mathbb{G}) + 2\mathsf{size}(\mathbb{Z}_p) + \lambda + 1. \end{aligned}$$

Concrete Parameters. We estimate concrete sizes for keys, communication, and signatures for existing two-round multi-signatures and our schemes. The results are computed using Python scripts (cf. Supplementary Material Section D) and are presented in Table 2. Concretely, we assume 2^{20} signing queries and 2^{30} hash queries. We compute (1) the security level that is provided for the schemes assuming that the underlying assumption is 128 bit hard (see Table 2, Column "Security"), and (2) the sizes of groups, keys, signatures, and per-signer communication if we want to achieve 128 bit security for the scheme, and instantiate the underlying group based on the security loss (see Table 2, other columns). For (1), the table shows that our schemes are the only ones providing meaningful security guarantees when instantiated with standardized groups. In addition, note that Musig2 comes at the cost of relying on a stronger one-more style assumption. For the setting in (2), our schemes have slightly worse concrete parameter sizes. However, we argue that in practice, the setting in (1) is much more important than (2), because schemes are mostly implemented using standardized groups. The approach in (2) should be

Scheme	Security	size($\mathbb{G}$)	$\lvert\mathsf{pk}\rvert$	$\lvert\mathsf{Comm}\rvert$	$\lvert\sigma\rvert$
Musig2 [NRS21]	9	1209	32.13	160.63	64.25
HBMS [BD21]	-11	1369	32.13	96.38	96.37
Ours (Section 3.2)	106	301	64.25	144.50	224.88
Ours (Section 3.3)	126	261	128.50	273.13	481.88

Table 2: Comparison of concrete parameters for existing two-round multi-signature schemes (top) in the random oracle model with our schemes (bottom) in terms. The column "Security" shows the security level that is provided for the schemes assuming that the underlying assumption is 128 bit hard. Other columns show sizes of keys, per-signer communication, and signatures in bytes assuming the schemes are instantiated (using non-standard groups) to have 128 bit security based on the security loss.

avoided, since it leads to the use of groups that are not optimized for computation, and not well-studied in terms of (concrete) security. In terms of computation, consider for example Musig2 compared to our scheme from Section 3.2. Musig2 uses one multi-exponentiation of size two for verification. In our scheme, signatures can be verified using one multi-exponentiation of size three, and two multi-exponentiations of size five. Taking into account that Musig2 would have to use a 1209 bit size group, and our scheme can use standardized groups for which multi-exponentiations are optimized, we expect that our scheme is computationally equally or more efficient.

Acknowledgments. We thank the anonymous reviewers from Eurocrypt 2023 for their useful feedback and suggestions. In particular, it was pointed out the similarity between the commitment scheme of Bagherzandi, Cheon, and Jarecki in [BCJ08] and ours.

References

[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, Heidelberg. (Cited on page 2.)

[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, Heidelberg, April 2012. (Cited on page 4, 9.)

[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 3, 23.)

[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, Heidelberg, December 2021. (Cited on page 2, 3, 23.)

[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, Heidelberg, December 2018. (Cited on page 1, 2, 3, 4.)

[BKP14] Olivier Blazy, Eike Kiltz, and Jiaxin Pan. (Hierarchical) identity-based encryption from affine message authentication. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 408–425. Springer, Heidelberg, August 2014. (Cited on page 2.)

[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 1, 2, 3, 4, 7.)

[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, Heidelberg, January 2003. (Cited on page 2.)

[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, Heidelberg, August 2022. (Cited on page 2.)

[CAHL+22] Rutchathon Chairattana-Apirom, Lucjan Hanzlik, Julian Loss, Anna Lysyanskaya, and Benedikt Wagner. PI-cut-choo and friends: Compact blind signatures via parallel instance cut-and-choose and more. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 3–31. Springer, Heidelberg, August 2022. (Cited on page 8.)

[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. https://eprint.iacr.org/2021/1375. (Cited on page 1, 2, 7.)

[Dam00] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 418–430. Springer, Heidelberg, May 2000. (Cited on page 4.)

[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*, pages 1084–1101. IEEE Computer Society Press, May 2019. (Cited on page 1, 2, 3, 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, Heidelberg, May 2021. (Cited on page 2, 5, 7.)

[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, Heidelberg, August 2013. (Cited on page 9.)

[FH21] Masayuki Fukumitsu and Shingo Hasegawa. A tightly secure ddh-based multisignature with public-key aggregation. *Int. J. Netw. Comput.*, 11(2):319–337, 2021. (Cited on page 2, 3, 4.)

[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 5.)

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 97–111. Springer, Heidelberg, August 2006. (Cited on page 4.)

[GQ88] Louis C. Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both trasmission and memory. In C. G. Günther, editor, *EUROCRYPT'88*, volume 330 of *LNCS*, pages 123–128. Springer, Heidelberg, May 1988. (Cited on page 4.)

[Gro09] Jens Groth. Homomorphic trapdoor commitments to group elements. Cryptology ePrint Archive, Report 2009/007, 2009. https://eprint.iacr.org/2009/007. (Cited on page 4, 6.)

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008. (Cited on page 4, 6.)

[HJK+21] Shuai Han, Tibor Jager, Eike Kiltz, Shengli Liu, Jiaxin Pan, Doreen Riepel, and Sven Schäge. Authenticated key exchange and signatures with tight security in the standard model. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 670–700, Virtual Event, August 2021. Springer, Heidelberg. (Cited on page 2.)

[HKL19] Eduard Hauck, Eike Kiltz, and Julian Loss. A modular treatment of blind signatures from identification schemes. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 345–375. Springer, Heidelberg, May 2019. (Cited on page 8.)

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

[KLR21] Jonathan Katz, Julian Loss, and Michael Rosenberg. Boosting the security of blind signature schemes. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 468–492. Springer, Heidelberg, December 2021. (Cited on page 8.)

[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, Heidelberg, August 2016. (Cited on page 4, 17.)

[KW03] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM CCS 2003*, pages 155–164. ACM Press, October 2003. (Cited on page 2, 4.)

[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, Heidelberg, May / June 2006. (Cited on page 2.)

[LP20] Roman Langrehr and Jiaxin Pan. Unbounded HIBE with tight security. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 129–159. Springer, Heidelberg, December 2020. (Cited on page 2, 9.)

[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 1.)

[MPSW19] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple schnorr multi-signatures with applications to bitcoin. *Des. Codes Cryptogr.*, 87(9):2139–2164, 2019. (Cited on page 2, 3, 4.)

[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, Heidelberg. (Cited on page 2, 3, 23.)

[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 2.)

[Ped92] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992. (Cited on page 4.)

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991. (Cited on page 4.)

[TZ23] Stefano Tessaro and Chenzhi Zhu. Threshold and multi-signature schemes from linear hash functions. In *Eurocrypt 2023 (to appear)*, LNCS. Springer, Heidelberg, 2023. (Cited on page 4.)

Supplementary Material

A Omitted Proof from Section 3.2

Proof of Theorem 1. The proof can be understood as a simplified version of the proof of Theorem 2. Set $\mathsf{MS} := \mathsf{MS}_a[\mathsf{LF}, \mathsf{CMT}]$, let $\mathcal{A}$ be a PPT algorithm. In the following, we present a sequence of games $\mathbf{G}_0$-$\mathbf{G}_8$ proving the statement. The games are presented in Figures 4 and 5. We fix the notation

$$\mathsf{Adv}_i := \Pr[\mathbf{G}_i \Rightarrow 1] \text{ for } i \in \{0\} \cup [8].$$

Game $\mathbf{G}_0$: Game $\mathbf{G}_0$ is defined as $\mathbf{G}_0 := \mathbf{MS\text{-}EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{MS}}$. To fix notation, we recall this game. First, the game samples $\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda)$ and a pair $(\mathsf{pk}, \mathsf{sk})$ with $\mathsf{sk} := x_1 \xleftarrow{\$} \mathcal{D}$ and $\mathsf{pk} := X_1 := \mathsf{F}(x)$. Then, $\mathcal{A}$ gets $\mathsf{par}, \mathsf{pk}$ as input, and access to oracles $\textsc{Sig}_0, \textsc{Sig}_1$. We omit signing oracle $\textsc{Sig}_2$. As in the proof of Theorem 2 this does not change the advantage of $\mathcal{A}$, as algorithm Sig_2 does not make any use of the secret key or a secret state and can be publicly run using the messages output in Sig_0 and Sig_1. Further, $\mathcal{A}$ gets access to random oracles $\mathsf{H}, \mathsf{H}_a, \mathsf{H}_c$, simulated by the game in a lazy manner, using maps h, h_a, h_c, respectively. Finally, $\mathcal{A}$ outputs a forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$. The game outputs 1 if and only if $\mathsf{pk}^* \in \mathcal{P}^*$,$(\mathcal{P}^*, \mathsf{m}^*) \notin \mathcal{L}$, and $\mathsf{Ver}(\mathcal{P}^*, \mathsf{m}^*, \sigma^*) = 1$. We assume that the public key pk^* is equal to pk_1 for $\mathcal{P}^* = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$. We write $\sigma^* = (\mathsf{com}^*, s^*, \varphi^*)$, and denote the aggregated key for $\mathcal{P}^*$ by $\tilde{\mathsf{pk}} := \tilde{X} := \mathsf{Agg}(\mathcal{P}^*)$. By definition, we have

$$\mathsf{Adv}_0 = \mathsf{Adv}^{\mathsf{MS\text{-}EUF\text{-}CMA}}_{\mathcal{A},\mathsf{MS}_t[\mathsf{LF},\mathsf{CMT}]}(\lambda).$$

Game $\mathbf{G}_1$: In this game $\mathbf{G}_1$, we add a bad event and let the game abort if it occurs. Concretely, consider par, x_1 sampled by the game as described above, and let Good be as in the definition of a special commitment scheme. The game aborts if $(\mathsf{par}, x_1) \notin \mathsf{Good}$. By definition of the special commitment scheme, we have

$$|\mathsf{Adv}_0 - \mathsf{Adv}_1| \leq \Pr[(\mathsf{par}, x_1) \notin \mathsf{Good}] \leq \varepsilon_{\mathsf{g}}.$$

Game $\mathbf{G}_2$: In this game $\mathbf{G}_2$, we introduce a map b, that maps inputs to random oracle H to bits. For each new input $(\tilde{\mathsf{pk}}, \mathsf{m})$ to H, the bit $b[\tilde{\mathsf{pk}}, \mathsf{m}]$ is sampled from a Bernoulli distribution with parameters $\gamma := 1/(Q_S + 1)$. Further, the game aborts if any of the follow occurs:

- For a signing query $\textsc{Sig}_0(\mathcal{P}, \mathsf{m})$ and $\tilde{\mathsf{pk}} := \mathsf{Agg}(\mathcal{P})$, it holds that $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 1$, or
- for the forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$ and $\tilde{\mathsf{pk}} := \mathsf{Agg}(\mathcal{P}^*)$, it holds that $b[\tilde{\mathsf{pk}}, \mathsf{m}^*] = 0$.

Note that the view of $\mathcal{A}$ is independent of the map b until an abort occurs. If the game does not abort, it is exactly like $\mathbf{G}_1$. Therefore, we can use the fact $(1 - 1/z)^z \geq 1/4$ for all $z \geq 2$ and get

$$\begin{aligned}\mathsf{Adv}_2 = \gamma(1-\gamma)^{Q_S} \cdot \mathsf{Adv}_1 &= \frac{1}{Q_S+1}\left(1 - \frac{1}{Q_S+1}\right)^{Q_S} \cdot \mathsf{Adv}_1 \\ &= \frac{1}{Q_S}\left(1 - \frac{1}{Q_S+1}\right)^{Q_S+1} \cdot \mathsf{Adv}_1 \geq \frac{1}{4Q_S} \cdot \mathsf{Adv}_1.\end{aligned}$$

Game $\mathbf{G}_3$: In game $\mathbf{G}_3$, we change how random oracle H is executed. Consider a query $\mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$ for which the hash value is not yet defined. Recall that in this case, a bit $b[\tilde{\mathsf{pk}}, \mathsf{m}]$ is sampled. Then, a commitment key ck has to be returned. In previous games, ck was sampled uniformly via $\mathsf{ck} \xleftarrow{\$} \mathcal{K}$. Now, depending on this bit, we change how ck is computed. Namely, if $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 0$, we sample $(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, X_1)$ and store the trapdoor td in another map $tr[\tilde{\mathsf{pk}}, \mathsf{m}] := \mathsf{td}$. On the other hand, if $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 1$, we sample $\mathsf{ck} \leftarrow \mathsf{BGen}(\mathsf{par})$.

We argue that games $\mathbf{G}_2$ and $\mathbf{G}_3$ are indistinguishable as follows. First, note that for case $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 0$, the distribution of ck stays the same, because we can assume $(\mathsf{par}, x_1) \in \mathsf{Good}$ due to previous changes.

For the case $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 1$, we use a reduction $\mathcal{B}$ against the multi-key indistinguishability of CMT interpolating between $\mathbf{G}_2$ and $\mathbf{G}_3$. Precisely, $\mathcal{B}$ gets as input par, x_1 and Q_{H} commitment keys $\mathsf{ck}_1, \ldots, \mathsf{ck}_{Q_{\mathsf{H}}}$. It simulates $\mathbf{G}_2$ for $\mathcal{A}$ with par while embedding the commitment keys in random oracle responses for queries $\mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$ with $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 1$. In the end, it outputs whatever the game outputs. Clearly, we have

$$|\mathsf{Adv}_2 - \mathsf{Adv}_3| \leq \mathsf{Adv}^{Q_{\mathsf{H}}\text{-}\mathsf{keydist}}_{\mathcal{B},\mathsf{CMT}}(\lambda).$$

Game $\mathbf{G}_4$: Game $\mathbf{G}_4$ is as $\mathbf{G}_3$, but we change the execution of oracles $\text{SIG}_0, \text{SIG}_1$. Concretely, after this change, the secret key x_1 is no longer needed. Consider a query $\text{SIG}_0(\mathcal{P}, \mathsf{m})$. Recall that for previous games, in such a query, a commitment key $\mathsf{ck} := \mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$ is computed. Then, values r_1, φ_1 are sampled, and $R_1 := \mathsf{F}(r_1)$ and a commitment $\mathsf{com}_1 := \mathsf{Com}(\mathsf{ck}, R_1; \varphi_1)$ is computed. Later, in SIG_1, s_1 is computed as $s_1 := c \cdot a_1 \cdot x_1 + r_1$, where c and a_1 are output by H_c and H_a as in the scheme. Assuming that the game does not abort in this query, we can assume that $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 0$, due to the change in $\mathbf{G}_2$. This means that the entry $\mathsf{td} := tr[\tilde{\mathsf{pk}}, \mathsf{m}]$ is defined, and was sampled together with ck using $\mathsf{TGen}(\mathsf{par}, X_1)$. We use this in game $\mathbf{G}_4$ as follows: The game no longer samples r_1 and φ_1. Instead, the commitment com_1 is computed via $(\mathsf{com}_1, St) \leftarrow \mathsf{TCom}(\mathsf{ck}, \mathsf{td})$. Later, in SIG_1, s_1 and φ_1 are computed using $(\varphi_1, R_1, s_1) \leftarrow \mathsf{TCol}(St, c \cdot a_1)$. Applying the special trapdoor property of CMT Q_S many times we obtain

$$|\mathsf{Adv}_4 - \mathsf{Adv}_5| \leq Q_S \varepsilon_{\mathsf{t}}.$$

Game $\mathbf{G}_5$: In game $\mathbf{G}_5$, we revert the change we introduced in $\mathbf{G}_1$. Concretely, the game no longer aborts if $(\mathsf{par}, x_1) \notin \mathsf{Good}$. As before, we get

$$|\mathsf{Adv}_4 - \mathsf{Adv}_5| \leq \Pr\left[(\mathsf{par}, x_1) \notin \mathsf{Good}\right] \leq \varepsilon_{\mathsf{g}}.$$

Game $\mathbf{G}_6$: In game $\mathbf{G}_6$, we change how the public key X_1 is generated. Recall that it was generated as $\mathsf{F}(x_1)$ before, where $x_1 \xleftarrow{\$} \mathcal{D}$. In this game, we sample $X_1 \xleftarrow{\$} \mathcal{R}$ instead. Note that due to the change in $\mathbf{G}_4$, we do not need x_1 anymore. We sketch a simple reduction $\mathcal{B}'$ against the key indistinguishability of the lossiness admitting linear function family LF to show indistinguishability of $\mathbf{G}_5$ and $\mathbf{G}_6$. Namely, $\mathcal{B}'$ gets par and X_1 as input, and simulates $\mathbf{G}_5$ for $\mathcal{A}$. In the end, it outputs whatever the game outputs. We have

$$|\mathsf{Adv}_5 - \mathsf{Adv}_6| \leq \mathsf{Adv}^{\mathsf{keydist}}_{\mathcal{B}', \mathsf{LF}}(\lambda).$$

Game $\mathbf{G}_7$: In game $\mathbf{G}_7$, we want to use the binding property of CMT. To do that, we introduce two changes. First, in oracle queries of the form $\mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m})$ we first set $\mathsf{ck} := \mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$. Then, if $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 0$, we simulate H_c as before. If $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 1$, we run the (unbounded) extraction algorithm Ext that exists according to the statistical binding property of CMT. Concretely, we run $R \leftarrow \mathsf{Ext}(\mathsf{ck}, \mathsf{com})$ and store $r[\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m}] := R$, where r is another map. Then, we continue the simulation of H_c as before. Second, we change the winning condition of the game. Concretely, after $\mathcal{A}$ outputs forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$, we parse $\sigma^* = (\mathsf{com}^*, s^*, \varphi^*)$ and compute the aggregated key $\tilde{\mathsf{pk}} := \tilde{X} := \mathsf{Agg}(\mathcal{P}^*)$ as before. In addition to the verification steps that we had before, we now also compute $c^* := \mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}^*, \mathsf{m}^*)$ and $R^* := \mathsf{F}(s^*) - c^* \cdot \tilde{X}$, and check if $R^* = r[\tilde{\mathsf{pk}}, \mathsf{com}^*, \mathsf{m}^*]$. If this does not hold, the game outputs 0.

Intuitively, these changes accomplish the following. The game extracts the values R from every commitment that is given by $\mathcal{A}$ via random oracle H_c, for which the commitment key ck was generated using algorithm BGen (cf. game $\mathbf{G}_3$). Then, we force the adversary into using the extracted value for its forgery.

Formally, we argue indistinguishability of $\mathbf{G}_6$ and $\mathbf{G}_7$ using an unbounded reduction to the statistical binding property of CMT. This reduction gets as input par and ck^*. It guesses $i_{\mathsf{H}} \xleftarrow{\$} [Q_{\mathsf{H}}]$ and $i_{\mathsf{H}_c} \xleftarrow{\$} [Q_{\mathsf{H}_c}]$. The reduction simulates game $\mathbf{G}_7$ for $\mathcal{A}$ honestly, except for query i_{H} to random oracle H and query i_{H_c} to random oracle H_c. If it had to sample a $\mathsf{ck} \leftarrow \mathsf{BGen}(\mathsf{par})$ in the former query, it instead responds with ck^*. If it had to run Ext in the latter query, it outputs com to the experiment. If query i_{H} was used to derive the commitment key used in the forgery and query i_{H_c} was used to derive the challenge c^* for the forgery, and $R^* \neq r[\tilde{\mathsf{pk}}, \mathsf{com}^*, \mathsf{m}^*]$, then the reduction outputs $R^*; \varphi^*$. Otherwise, it outputs $\perp$. Clearly, if the reduction guesses the correct queries and the bad event separating $\mathbf{G}_6$ and $\mathbf{G}_7$ occurs, then it breaks the statistical binding property. The view of $\mathcal{A}$ is as in $\mathbf{G}_7$, and independent of $(i_{\mathsf{H}}, i_{\mathsf{H}_c})$. Therefore, we obtain

$$|\mathsf{Adv}_6 - \mathsf{Adv}_7| \leq Q_{\mathsf{H}} Q_{\mathsf{H}_c} \varepsilon_{\mathsf{b}}.$$

Game $\mathbf{G}_8$: In game $\mathbf{G}_8$, we introduce another abort. Namely, the game aborts in a query $\mathsf{H}_a(\langle \mathcal{P} \rangle, \mathsf{pk})$, for which $\mathsf{pk} = \mathsf{pk}_1$ and the hash value is not yet defined, but for $\tilde{\mathsf{pk}} := \mathsf{Agg}(\mathcal{P})$, there is some com, m such that $\mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m})$ is already defined. The probability of this bad event is easily bounded. First, assume that $\mathsf{pk}_1 = X_1$ is not the zero vector in $\mathcal{R}$. The probability that X_1 is the zero vector is at most $1/|\mathcal{R}|$. Then, fix such a query $\mathsf{H}_a(\langle \mathcal{P} \rangle, \mathsf{pk} = X_1)$, and any previous query to oracle H_c. The bad event can only occur if the input of the latter query starts with $\tilde{X}$, where $a_1 X_1 = \sum_{i=2}^{N} a_i X_i - \tilde{X}$. As X_1 is not

the zero vector, the value $a_1 X_1$ is uniform over the span of X_1. Further the values on the right-hand side are fixed before a_1 is sampled, assuming that the bad event occurs. Thus, the probability of the bad event for this pair of queries is at most $1/|\mathcal{S}|$. We get

$$|\mathsf{Adv}_7 - \mathsf{Adv}_8| \leq \frac{1}{|\mathcal{R}|} + \frac{Q_{\mathsf{H}_a} Q_{\mathsf{H}_c}}{|\mathcal{S}|}.$$

Note that this change ensured that for the forgery output by $\mathcal{A}$, the query defining coefficient a_1 occurred before the query defining the challenge c^*.

To bound the probability that $\mathbf{G}_8$ outputs 1, we give an unbounded reduction from the aggregation lossy soundness of LF.

- The reduction gets as input parameters par and an element X_1. It samples $\hat{i}_{\mathsf{H}_c} \xleftarrow{\$} [Q_{\mathsf{H}_c}]$ and $\hat{i}_{\mathsf{H}_a} \xleftarrow{\$} [Q_{\mathsf{H}_a}]$. Then, it simulates $\mathbf{G}_8$ honestly until $\mathcal{A}$ outputs a forgery, except for queries $\hat{i}_{\mathsf{H}_c}$ to oracle H_c and $\hat{i}_{\mathsf{H}_a}$ to oracle H_a.
- If the query $\hat{i}_{\mathsf{H}_c}$ to oracle H_c occurs before the query $\hat{i}_{\mathsf{H}_a}$ to oracle H_a, the reduction aborts its execution.
- Consider the query $\hat{i}_{\mathsf{H}_a}$ to oracle H_a. If the hash value is already defined, the reduction aborts its execution. Else, let this query be $\mathsf{H}_a(\langle \mathcal{P} \rangle, \mathsf{pk})$. If $\mathsf{pk} \neq \mathsf{pk}_1$, the reduction aborts. Otherwise, it first parses $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ and queries $a_i := \mathsf{H}_a(\langle \mathcal{P} \rangle, \mathsf{pk}_i)$ for all $2 \leq i \leq N$. Then it outputs the pairs $(\mathsf{pk}_2, a_2), \ldots, (\mathsf{pk}_N, a_N)$ to the aggregation lossy soundness experiment. It gets as input a_1, sets $h_a[\langle \mathcal{P} \rangle, \mathsf{pk}] := a_1$, and continues the simulation as in $\mathbf{G}_8$.
- Consider the query $\hat{i}_{\mathsf{H}_c}$ to oracle H_c. Let this query be $\mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m})$. The reduction aborts its execution, if the hash value for this query is already defined. Else, it queries $\mathsf{ck} := \mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$. If $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 0$, it aborts its execution. Otherwise, it runs $R \leftarrow \mathsf{Ext}(\mathsf{ck}, \mathsf{com})$ as in $\mathbf{G}_8$. It outputs R to the aggregation lossy soundness experiment and obtains a value c in return. Then, it sets $h_c[\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m}] := c$ and continues the simulation as in $\mathbf{G}_8$.
- When $\mathcal{A}$ outputs the forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$, the reduction runs all the verification steps in $\mathbf{G}_8$. Additionally, it checks if the value $\mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}^*, \mathsf{m}^*)$ was defined during query $\hat{i}_{\mathsf{H}_c}$ to H_c, and the value $\mathsf{H}_a(\langle \mathcal{P}^* \rangle, \mathsf{pk}_1)$ was defined during $\hat{i}_{\mathsf{H}_a}$ to oracle H_a. If this is not the case, it aborts its execution. Otherwise, it returns $s := s^*$ to the aggregation lossy soundness experiment.

Clearly, unless the reduction aborts due to wrong guessing of the indices $\hat{i}_{\mathsf{H}_a}, \hat{i}_{\mathsf{H}_c}$, the view of $\mathcal{A}$ is exactly as in $\mathbf{G}_8$. Before any such abort, $\mathcal{A}$'s view is independent of the indices $\hat{i}_{\mathsf{H}_a}, \hat{i}_{\mathsf{H}_c}$. Also, it is clear that if the reduction does not abort, it outputs a valid solution to the aggregation lossy soundness experiment. Therefore, we get

$$\mathsf{Adv}_8 \leq Q_{\mathsf{H}_a} Q_{\mathsf{H}_c} \varepsilon_{\mathsf{al}},$$

and the statement is proven. □

Game $\mathbf{G}_0$-$\mathbf{G}_8$

$\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda)$
$\mathsf{sk} := x_1 \xleftarrow{\$} \mathcal{D},\ \mathsf{pk} := X_1 := \mathsf{F}(x)$ // $\mathbf{G}_0$-$\mathbf{G}_5$
$\mathsf{pk}_1 := X_1 \xleftarrow{\$} \mathcal{R}$ // $\mathbf{G}_6$-$\mathbf{G}_8$
if $(\mathsf{par}, x_1) \notin \mathsf{Good}$: **abort** // $\mathbf{G}_1$-$\mathbf{G}_4$
$(\mathcal{P}^*, \mathsf{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathrm{SIG}_0, \mathrm{SIG}_1, \mathsf{H}, \mathsf{H}_a, \mathsf{H}_c}(\mathsf{par}, \mathsf{pk}_1)$
if $\mathsf{pk} \notin \mathcal{P}^* \vee (\mathcal{P}^*, \mathsf{m}^*) \in \mathcal{L}$: **return** 0
$\tilde{\mathsf{pk}} := \mathsf{Agg}(\mathcal{P}^*)$ // $\mathbf{G}_2$-$\mathbf{G}_8$
if $b[\tilde{\mathsf{pk}}, \mathsf{m}^*] = 0$: **return** 0 // $\mathbf{G}_2$-$\mathbf{G}_8$
let $\tilde{\mathsf{pk}} = \tilde{X},\ \sigma^* = (\mathsf{com}^*, s^*, \varphi^*)$ // $\mathbf{G}_7$-$\mathbf{G}_8$
$c^* := \mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}^*, \mathsf{m}^*),\ R^* := \mathsf{F}(s^*) - c^* \cdot \tilde{X}$ // $\mathbf{G}_7$-$\mathbf{G}_8$
if $R^* \neq r[\tilde{\mathsf{pk}}, \mathsf{com}^*, \mathsf{m}^*]$: **return** 0 // $\mathbf{G}_7$-$\mathbf{G}_8$
return $\mathsf{Ver}(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$

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

let $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$
if $\mathsf{pk}_1 \neq \mathsf{pk}$: **return** $\perp$
$\mathcal{L} := \mathcal{L} \cup \{(\mathcal{P}, \mathsf{m})\},\ sid := sid + 1,\ ctr[sid] := 1$
$\tilde{\mathsf{pk}} := \mathsf{Agg}(\mathcal{P}),\ \mathsf{ck} := \mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$
if $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 1$: **abort** // $\mathbf{G}_2$-$\mathbf{G}_8$
$r_1 \xleftarrow{\$} \mathcal{D},\ R_1 := \mathsf{F}(r_1),\ \varphi_1 \xleftarrow{\$} \mathcal{G}$ // $\mathbf{G}_0$-$\mathbf{G}_3$
$\mathsf{com}_1 := \mathsf{Com}(\mathsf{ck}, R_1; \varphi_1)$ // $\mathbf{G}_0$-$\mathbf{G}_3$
$St_1 := (\tilde{\mathsf{pk}}, x_1, r_1, \varphi_1)$ // $\mathbf{G}_0$-$\mathbf{G}_3$
$(\mathsf{com}_1, St) \leftarrow \mathsf{TCom}(\mathsf{ck}, tr[\tilde{\mathsf{pk}}, \mathsf{m}])$ // $\mathbf{G}_4$-$\mathbf{G}_8$
$St_1 := St$ // $\mathbf{G}_4$-$\mathbf{G}_8$
$(\mathsf{pm}_1[sid], St_1[sid]) := (\mathsf{pm}_{1,1} := \mathsf{com}_1, St_1)$
return $(\mathsf{pm}_1[sid], sid)$

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

if $ctr[sid] \neq 1$: **return** $\perp$
let $\mathcal{M}_1 = (\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N})$
if $\mathsf{pm}_1[sid] \neq \mathsf{pm}_{1,1}$: **return** $\perp$
$ctr[sid] := ctr[sid] + 1$
let $St_1 = (x_1, r_1, \varphi_1)$ // $\mathbf{G}_0$-$\mathbf{G}_3$
let $St_1 = St$ // $\mathbf{G}_4$-$\mathbf{G}_8$
for $i \in [N]$: **let** $\mathsf{pm}_{1,i} = \mathsf{com}_i$
$\mathsf{com} := \bigotimes_{i \in [N]} \mathsf{com}_i,\ c := \mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m}),\ a_1 := \mathsf{H}_a(\langle \mathcal{P} \rangle, \mathsf{pk}_1)$
$s_1 := c \cdot a_1 \cdot x_1 + r_1$ // $\mathbf{G}_0$-$\mathbf{G}_3$
$(\varphi_1, R_1, s_1) \leftarrow \mathsf{TCol}(St, c \cdot a_1)$ // $\mathbf{G}_4$-$\mathbf{G}_8$
$(\mathsf{pm}_2[sid], St_2[sid]) := (\mathsf{pm}_{2,1} := (s_1, \varphi_1), St_2 := \mathsf{com})$
return $\mathsf{pm}_2[sid]$

Figure 4: The games $\mathbf{G}_0$-$\mathbf{G}_8$ used in the proof of Theorem 1. Lines with highlighted comments are only executed in the respective games. The random oracles are defined in Figure 5.

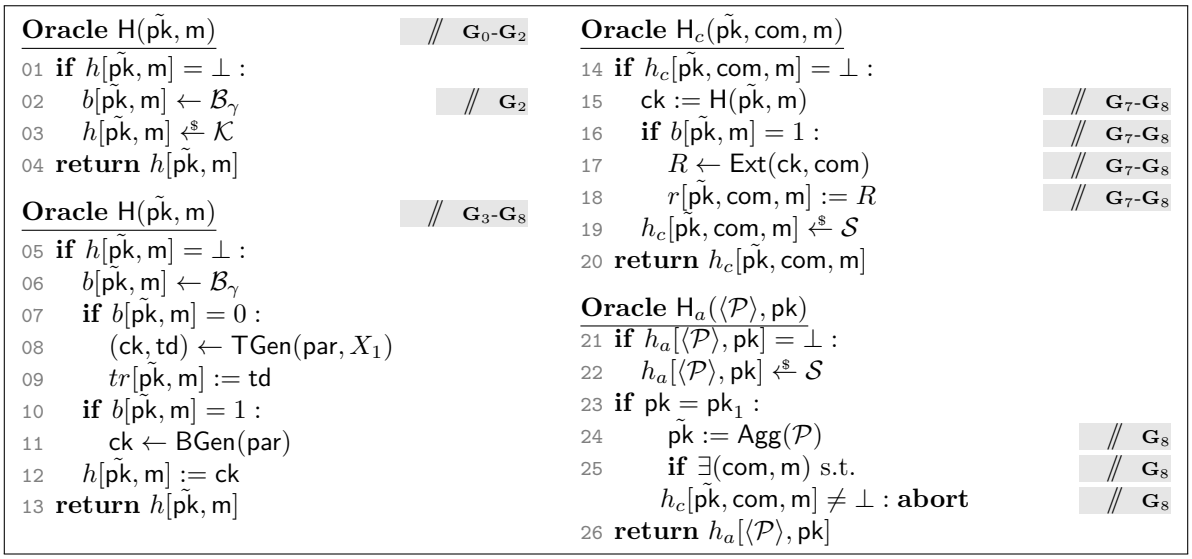

Figure 5: The random oracles that are used in the proof of Theorem 1. Lines with highlighted comments are only executed in the respective games. Algorithm Ext is the (unbounded) extractor for the statistical binding property of CMT.

B Omitted Proofs and Figures from Section 3.3

Proof of Lemma 2. Consider the variables given in verification and an honest execution of the protocol. Concretely, let $\mathcal{P} = \{\mathsf{pk}_1, \dots, \mathsf{pk}_N\}$ be a set of public keys, $\mathsf{m} \in \{0,1\}^*$ be a message, and $\sigma = (\sigma_0, \sigma_1, B)$ be a signature computed by an honest execution of the signing protocol specified by algorithms Sig_0, $\mathsf{Sig}_1, \mathsf{Sig}_2$. Write $B = b_1 \dots b_N, \sigma_0 = (\mathsf{com}_0, \varphi_0, s_0)$ and $\sigma_1 = (\mathsf{com}_1, \varphi_1, s_1)$. Write the public keys pk_i as $\mathsf{pk}_i = (X_{i,0}, X_{i,1})$. Then, we can use the homomorphic properties of F to obtain

$$\begin{aligned}\mathsf{F}(s_0) - \sum_{i=1}^{N} c_{i,0} \cdot X_{i,b_i} &= \mathsf{F}\left(\sum_{i=1}^{N} s_{i,0}\right) - \sum_{i=1}^{N} c_{i,0} \cdot X_{i,b_i} \\ &= \sum_{i=1}^{N} \mathsf{F}(s_{i,0}) - c_{i,0} \cdot \mathsf{F}(x_{i,b_i}) \\ &= \sum_{i=1}^{N} \mathsf{F}(s_{i,0} - c_{i,0} \cdot x_{i,b_i}) = \sum_{i=1}^{N} \mathsf{F}(r_{i,0}) = \sum_{i=1}^{N} R_{i,0}.\end{aligned}$$

Using this, the homomorphic properties of Com, and the definition of φ_0, it follows that

$$\begin{aligned}\mathsf{Com}\left(\mathsf{ck}_0, \mathsf{F}(s_0) - \sum_{i=1}^{N} c_{i,0} \cdot X_{i,b_i}; \varphi_0\right) &= \mathsf{Com}\left(\mathsf{ck}_0, \sum_{i=1}^{N} R_{i,0}; \bigoplus_{i=1}^{N} \varphi_{i,0}\right) \\ &= \bigotimes_{i=1}^{N} \mathsf{Com}\left(\mathsf{ck}_0, R_{i,0}; \varphi_{i,0}\right) \\ &= \bigotimes_{i=1}^{N} \mathsf{com}_{i,0} = \mathsf{com}_0.\end{aligned}$$

This shows that the first verification equation holds. The proof for the second equation is similar. □

Game $\mathbf{G}_0$-$\mathbf{G}_8$

$\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda),\ x_{1,0}, x_{1,1} \xleftarrow{\$} \mathcal{D},\ \mathsf{seed}_1 \xleftarrow{\$} \{0,1\}^\lambda,\ X_{1,0} := \mathsf{F}(x_{1,0})$
if $(\mathsf{par}, x_{1,1}) \notin \mathsf{Good}$: **abort** // $\mathbf{G}_3$-$\mathbf{G}_5$
$X_{1,1} := \mathsf{F}(x_{1,1})$ // $\mathbf{G}_0$-$\mathbf{G}_6$
$X_{1,1} \xleftarrow{\$} \mathcal{R}$ // $\mathbf{G}_7$-$\mathbf{G}_8$
$\mathsf{pk}^* := (X_{1,0}, X_{1,1})$
$(\mathcal{P}^*, \mathsf{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{H},\mathsf{H}_b,\mathsf{H}_c,\mathrm{SIG}_0,\mathrm{SIG}_1}(\mathsf{par}, \mathsf{pk}^*)$
if $\mathsf{pk}^* \notin \mathcal{P}^* \vee (\mathcal{P}^*, \mathsf{m}^*) \in \mathcal{L}$: **return** 0
let $\sigma^* = (\sigma_0^*, \sigma_1^*, B^*),\ B^* = b_1^* \dots b_N^* \in \{0,1\}^N$
let $\sigma_0^* = (\mathsf{com}_0^*, \varphi_0^*, s_0^*),\ \sigma_1^* = (\mathsf{com}_1^*, \varphi_1^*, s_1^*)$
if $\mathsf{bad} = 1$: **return** 0 // $\mathbf{G}_1$-$\mathbf{G}_8$
if $b_1^* \neq 1 - \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P}^* \rangle, \mathsf{m}^*)$: **return** 0 // $\mathbf{G}_2$-$\mathbf{G}_8$
let $\mathcal{P}^* = \{\mathsf{pk}_1 = \mathsf{pk}^*, \dots, \mathsf{pk}_N\}$ // $\mathbf{G}_8$
for $i \in [N]$: // $\mathbf{G}_8$
  let $\mathsf{pk}_i = (X_{i,0}, X_{i,1})$ // $\mathbf{G}_8$
  $c_{i,0}^* := \mathsf{H}_c(\mathsf{pk}_i, \mathsf{com}_0^*, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*, 0)$ // $\mathbf{G}_8$
  $c_{i,1}^* := \mathsf{H}_c(\mathsf{pk}_i, \mathsf{com}_1^*, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*, 1)$ // $\mathbf{G}_8$
$R_0^* := \mathsf{F}(s_0^*) - \sum_{i=1}^N c_{i,0}^* \cdot X_{i,b_i^*},\ R_1^* := \mathsf{F}(s_1^*) - \sum_{i=1}^N c_{i,1}^* \cdot X_{i,1-b_i^*}$ // $\mathbf{G}_8$
if $R_{1-b_1^*}^* \neq r[\mathsf{com}_{1-b_1^*}^*, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*]$: **return** 0 // $\mathbf{G}_8$
return $\mathsf{Ver}(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$

Figure 6: The games $\mathbf{G}_0$-$\mathbf{G}_8$ used in the proof of Theorem 2. Lines with highlighted comments are only executed in the respective games. The signing oracles are defined in Figure 7, and the random oracles are defined in Figure 8.

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

01 **let** $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$
02 **if** $\mathsf{pk}_1 \neq \mathsf{pk}^*$: **return** $\bot$
03 $\mathcal{L} := \mathcal{L} \cup \{(\mathcal{P}, \mathsf{m})\},\ sid := sid + 1,\ ctr[sid] := 1$
04 $\mathsf{ck}_0 := \mathsf{H}(0, \langle\mathcal{P}\rangle, \mathsf{m}),\ \mathsf{ck}_1 := \mathsf{H}(1, \langle\mathcal{P}\rangle, \mathsf{m})$
05 $b_1 := \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle\mathcal{P}\rangle, \mathsf{m})$
06 $r_{1,b_1} \xleftarrow{\$} \mathcal{D},\ \varphi_{1,b_1} \xleftarrow{\$} \mathcal{G},\ R_{1,b_1} := \mathsf{F}(r_{1,b_1})$
07 $\mathsf{com}_{1,b_1} := \mathsf{Com}(\mathsf{ck}_{b_1}, R_{1,b_1}; \varphi_{1,b_1})$
08 $r_{1,1-b_1} \xleftarrow{\$} \mathcal{D},\ \varphi_{1,1-b_1} \xleftarrow{\$} \mathcal{G},\ R_{1,1-b_1} := \mathsf{F}(r_{1,1-b_1})$ // $\mathbf{G}_0$-$\mathbf{G}_4$
09 $\mathsf{com}_{1,1-b_1} := \mathsf{Com}(\mathsf{ck}_{1-b_1}, R_{1,1-b_1}; \varphi_{1,1-b_1})$ // $\mathbf{G}_0$-$\mathbf{G}_4$
10 $St_1 := (r_{1,0}, r_{1,1}, \varphi_{1,0}, \varphi_{1,1})$ // $\mathbf{G}_0$-$\mathbf{G}_4$
11 $(\mathsf{com}_{1,1-b_1}, St) \leftarrow \mathsf{TCom}(\mathsf{ck}_{1-b_1}, tr[\langle\mathcal{P}\rangle, \mathsf{m}])$ // $\mathbf{G}_5$-$\mathbf{G}_8$
12 $St_1 := (r_{1,b_1}, \varphi_{1,b_1}, St)$ // $\mathbf{G}_5$-$\mathbf{G}_8$
13 $(\mathsf{pm}_1[sid], St_1[sid]) := (\mathsf{pm}_{1,1} := (b_1, \mathsf{com}_{1,0}, \mathsf{com}_{1,1}), St_1)$
14 **return** $(\mathsf{pm}_1[sid], sid)$

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

15 **if** $ctr[sid] \neq 1$: **return** $\bot$
16 **let** $\mathcal{M}_1 = (\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N})$
17 **if** $\mathsf{pm}_1[sid] \neq \mathsf{pm}_{1,1}$: **return** $\bot$
18 $ctr[sid] := ctr[sid] + 1,\ St_1 := St_1[sid]$
19 **let** $St_1 = (r_{1,0}, r_{1,1}, \varphi_{1,0}, \varphi_{1,1})$ // $\mathbf{G}_0$-$\mathbf{G}_4$
20 **let** $St_1 := (r_{1,b_1}, \varphi_{1,b_1}, St)$ // $\mathbf{G}_5$-$\mathbf{G}_8$
21 **for** $i \in [N]$: **let** $\mathsf{pm}_{1,i} = (b_i, \mathsf{com}_{i,0}, \mathsf{com}_{i,1})$
22 $B := b_1 \ldots b_N \in \{0,1\}^N$
23 $\mathsf{com}_0 := \bigotimes_{i\in[N]} \mathsf{com}_{i,0},\ \mathsf{com}_1 := \bigotimes_{i\in[N]} \mathsf{com}_{i,1}$
24 $c_{1,0} := \mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}_0, \mathsf{m}, \langle\mathcal{P}\rangle, B, 0),\ c_{1,1} := \mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}_1, \mathsf{m}, \langle\mathcal{P}\rangle, B, 1)$
25 $s_{1,b_1} := c_{1,b_1} \cdot x_{1,0} + r_{1,b_1}$
26 $s_{1,1-b_1} := c_{1,1-b_1} \cdot x_{1,1} + r_{1,1-b_1}$ // $\mathbf{G}_0$-$\mathbf{G}_4$
27 $(\varphi_{1,1-b_1}, R_{1-b_1}, s_{1,1-b_1}) \leftarrow \mathsf{TCol}(St, c_{1,1-b_1})$ // $\mathbf{G}_5$-$\mathbf{G}_8$
28 $St_2 := (\mathsf{com}_0, \mathsf{com}_1)$
29 $(\mathsf{pm}_2[sid], St_2[sid]) := (\mathsf{pm}_{2,1} := (s_{1,0}, s_{1,1}, \varphi_{1,0}, \varphi_{1,1}), St_2)$
30 **return** $\mathsf{pm}_2[sid]$

Figure 7: The signing oracles that are used in the proof of Theorem 2. Lines with highlighted comments are only executed in the respective games.

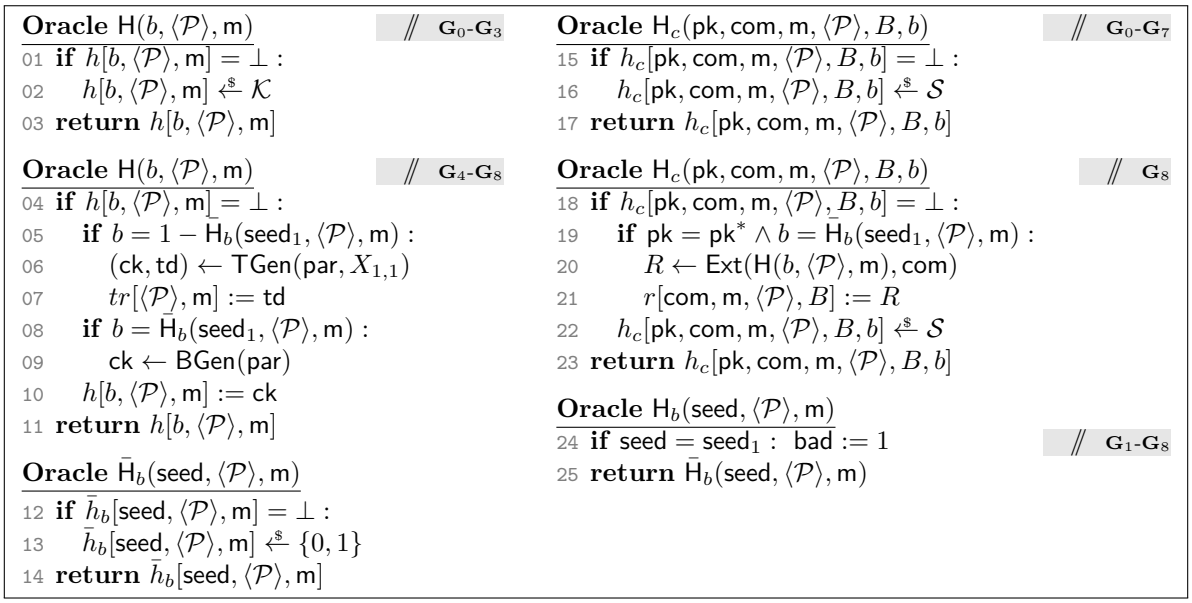

Figure 8: The random oracles that are used in the proof of Theorem 2. Lines with highlighted comments are only executed in the respective games. Algorithm Ext is the (unbounded) extractor for the statistical binding property of CMT.

C Pseudocode for Our Schemes

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

Alg $\mathsf{Gen}(\mathsf{par})$
02 $\mathsf{sk} := x \xleftarrow{\$} \mathcal{D},\ \mathsf{pk} := X := \mathsf{F}(x)$
03 **return** $(\mathsf{pk}, \mathsf{sk})$

Alg $\mathsf{Agg}(\mathcal{P})$
04 **let** $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$
05 **for** $i \in [N]$: **let** $\mathsf{pk}_i = X_i$
06 **for** $i \in [N] : a_i := \mathsf{H}_a(\langle \mathcal{P} \rangle, \mathsf{pk}_i)$
07 **return** $\tilde{\mathsf{pk}} := \tilde{X} := \sum_{i=1}^{N} a_i \cdot X_i$

Alg $\mathsf{VerAgg}(\tilde{\mathsf{pk}}, \mathsf{m}, \sigma)$
08 **let** $\tilde{\mathsf{pk}} = \tilde{X},\ \sigma = (\mathsf{com}, s, \varphi)$
09 $c := \mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m})$
10 $R := \mathsf{F}(s) - c \cdot \tilde{X}$
11 $\mathsf{ck} := \mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$
12 **if** $\mathsf{com} \neq \mathsf{Com}(\mathsf{ck}, R; \varphi)$: **return** 0
13 **return** 1

Alg $\mathsf{Ver}(\mathcal{P}, \mathsf{m}, \sigma)$
14 $\tilde{\mathsf{pk}} := \mathsf{Agg}(\mathcal{P})$
15 **return** $\mathsf{VerAgg}(\tilde{\mathsf{pk}}, \mathsf{m}, \sigma)$

Alg $\mathsf{Sig}_0(\mathcal{P}, \mathsf{sk}_1, \mathsf{m})$
16 **let** $\mathsf{sk}_1 = x_1$
17 $\tilde{\mathsf{pk}} := \mathsf{Agg}(\mathcal{P}),\ \mathsf{ck} := \mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$
18 $r_1 \xleftarrow{\$} \mathcal{D},\ R_1 := \mathsf{F}(r_1),\ \varphi_1 \xleftarrow{\$} \mathcal{G}$
19 $\mathsf{pm}_{1,1} := \mathsf{com}_1 := \mathsf{Com}(\mathsf{ck}, R_1; \varphi_1)$
20 $St_1 := (\tilde{\mathsf{pk}}, x_1, r_1, \varphi_1, \mathsf{m})$
21 **return** $(\mathsf{pm}_{1,1}, St_1)$

Alg $\mathsf{Sig}_1(St_1, \mathcal{M}_1)$
22 **let** $\mathcal{M}_1 = (\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N})$
23 **let** $St_1 = (\tilde{\mathsf{pk}}, x_1, r_1, \varphi_1, \mathsf{m})$
24 **for** $i \in [N]$: **let** $\mathsf{pm}_{1,i} = \mathsf{com}_i$
25 $\mathsf{com} := \bigotimes_{i \in [N]} \mathsf{com}_i$
26 $c := \mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m})$
27 $a_1 := \mathsf{H}_a(\langle \mathcal{P} \rangle, \mathsf{pk}_1)$
28 $s_1 := c \cdot a_1 \cdot x_1 + r_1$
29 $\mathsf{pm}_{2,1} := (s_1, \varphi_1)$
30 **return** $(\mathsf{pm}_{2,1}, St_2 := \mathsf{com})$

Alg $\mathsf{Sig}_2(St_2, \mathcal{M}_2)$
31 **let** $St_2 = \mathsf{com}$
32 **let** $\mathcal{M}_2 = (\mathsf{pm}_{2,1}, \ldots, \mathsf{pm}_{2,N})$
33 **for** $i \in [N]$: **let** $\mathsf{pm}_{2,i} = (s_i, \varphi_i)$
34 $s := \sum_{i=1}^{N} s_i,\ \varphi := \bigoplus_{i=1}^{N} \varphi_i$
35 **return** $\sigma := (\mathsf{com}, s, \varphi)$

Figure 9: The multi-signature scheme $\mathsf{MS}_a[\mathsf{LF}, \mathsf{CMT}] = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ with key aggregation for a linear function family $\mathsf{LF} = (\mathsf{LF.Gen}, \mathsf{F})$ and a special commitment scheme $\mathsf{CMT} = (\mathsf{BGen}, \mathsf{TGen}, \mathsf{Com}, \mathsf{TCom}, \mathsf{TCol})$.

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

Alg $\mathsf{Gen}(\mathsf{par})$
02 $x_0, x_1 \xleftarrow{\$} \mathcal{D}, \mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$
03 $X_0 := \mathsf{F}(x_0), X_1 := \mathsf{F}(x_1)$
04 $\mathsf{pk} := (X_0, X_1), \mathsf{sk} := (x_0, x_1, \mathsf{seed})$
05 **return** $(\mathsf{pk}, \mathsf{sk})$

Alg $\mathsf{Ver}(\mathcal{P}, \mathsf{m}, \sigma)$
06 **let** $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$
07 **let** $\sigma = (\sigma_0, \sigma_1, B)$
08 **let** $\sigma_0 = (\mathsf{com}_0, s_0, \varphi_0)$
09 **let** $\sigma_1 = (\mathsf{com}_1, s_1, \varphi_1)$
10 **let** $B = b_1 \ldots b_N \in \{0,1\}^N$
11 **for** $i \in [N]$:
12 **let** $\mathsf{pk}_i = (X_{i,0}, X_{i,1})$
13 $c_{i,0} := \mathsf{H}_c(\mathsf{pk}_i, \mathsf{com}_0, \mathsf{m}, \langle\mathcal{P}\rangle, B, 0)$
14 $c_{i,1} := \mathsf{H}_c(\mathsf{pk}_i, \mathsf{com}_1, \mathsf{m}, \langle\mathcal{P}\rangle, B, 1)$
15 $R_0 := \mathsf{F}(s_0) - \sum_{i=1}^N c_{i,0} \cdot X_{i,b_i}$
16 $R_1 := \mathsf{F}(s_1) - \sum_{i=1}^N c_{i,1} \cdot X_{i,1-b_i}$
17 $\mathsf{ck}_0 := \mathsf{H}(0, \langle\mathcal{P}\rangle, \mathsf{m})$
18 $\mathsf{ck}_1 := \mathsf{H}(1, \langle\mathcal{P}\rangle, \mathsf{m})$
19 **if** $\mathsf{com}_0 \neq \mathsf{Com}(\mathsf{ck}_0, R_0; \varphi_0)$:
20 **return** 0
21 **if** $\mathsf{com}_1 \neq \mathsf{Com}(\mathsf{ck}_1, R_1; \varphi_1)$:
22 **return** 0
23 **return** 1

Alg $\mathsf{Sig}_0(\mathcal{P}, \mathsf{sk}_1, \mathsf{m})$
24 **let** $\mathsf{sk}_1 = (x_{1,0}, x_{1,1}, \mathsf{seed}_1)$
25 $\mathsf{ck}_0 := \mathsf{H}(0, \langle\mathcal{P}\rangle, \mathsf{m})$
26 $\mathsf{ck}_1 := \mathsf{H}(1, \langle\mathcal{P}\rangle, \mathsf{m})$
27 $b_1 := \mathsf{H}_b(\mathsf{seed}_1, \langle\mathcal{P}\rangle, \mathsf{m})$
28 $r_{1,0}, r_{1,1} \xleftarrow{\$} \mathcal{D},\ \varphi_{1,0}, \varphi_{1,1} \xleftarrow{\$} \mathcal{G}$
29 $R_{1,0} := \mathsf{F}(r_{1,0}), R_{1,1} := \mathsf{F}(r_{1,1})$
30 $\mathsf{com}_{1,0} := \mathsf{Com}(\mathsf{ck}_0, R_{1,0}; \varphi_{1,0})$
31 $\mathsf{com}_{1,1} := \mathsf{Com}(\mathsf{ck}_1, R_{1,1}; \varphi_{1,1})$
32 $\mathsf{pm}_{1,1} := (b_1, \mathsf{com}_{1,0}, \mathsf{com}_{1,1})$
33 $St_1 := (\mathsf{sk}_1, r_{1,0}, r_{1,1}, \varphi_{1,0}, \varphi_{1,1})$
34 **return** $(\mathsf{pm}_{1,1}, St_1)$

Alg $\mathsf{Sig}_1(St_1, \mathcal{M}_1)$
35 **let** $\mathcal{M}_1 = (\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N})$
36 **let** $St_1 = (\mathsf{sk}_1, r_{1,0}, r_{1,1}, \varphi_{1,0}, \varphi_{1,1})$
37 **for** $i \in [N]$:
38 **let** $\mathsf{pm}_{1,i} = (b_i, \mathsf{com}_{i,0}, \mathsf{com}_{i,1})$
39 $B := b_1 \ldots b_N \in \{0,1\}^N$
40 $\mathsf{com}_0 := \bigotimes_{i\in[N]} \mathsf{com}_{i,0}$
41 $\mathsf{com}_1 := \bigotimes_{i\in[N]} \mathsf{com}_{i,1}$
42 $c_{1,0} := \mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}_0, \mathsf{m}, \langle\mathcal{P}\rangle, B, 0)$
43 $c_{1,1} := \mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}_1, \mathsf{m}, \langle\mathcal{P}\rangle, B, 1)$
44 $s_{1,0} := c_{1,0} \cdot x_{1,b_1} + r_{1,0}$
45 $s_{1,1} := c_{1,1} \cdot x_{1,1-b_1} + r_{1,1}$
46 $\mathsf{pm}_{2,1} := (s_{1,0}, s_{1,1}, \varphi_{1,0}, \varphi_{1,1})$
47 $St_2 := (\mathsf{com}_0, \mathsf{com}_1)$
48 **return** $(\mathsf{pm}_{2,1}, St_2)$

Alg $\mathsf{Sig}_2(St_2, \mathcal{M}_2)$
49 **let** $St_2 = (\mathsf{com}_0, \mathsf{com}_1)$
50 **let** $\mathcal{M}_2 = (\mathsf{pm}_{2,1}, \ldots, \mathsf{pm}_{2,N})$
51 **for** $i \in [N]$:
52 **let** $\mathsf{pm}_{2,i} = (s_{i,0}, s_{i,1}, \varphi_{i,0}, \varphi_{i,1})$
53 $s_0 := \sum_{i=1}^N s_{i,0},\ \varphi_0 := \bigoplus_{i=1}^N \varphi_{i,0}$
54 $s_1 := \sum_{i=1}^N s_{i,1},\ \varphi_1 := \bigoplus_{i=1}^N \varphi_{i,1}$
55 $\sigma_0 := (\mathsf{com}_0, \varphi_0, s_0)$
56 $\sigma_1 := (\mathsf{com}_1, \varphi_1, s_1)$
57 $\sigma := (\sigma_0, \sigma_1, B)$
58 **return** σ

Figure 10: The multi-signature scheme $\mathsf{MS}_t[\mathsf{LF}, \mathsf{CMT}] = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ for a linear function family $\mathsf{LF} = (\mathsf{LF.Gen}, \mathsf{F})$ and a special commitment scheme $\mathsf{CMT} = (\mathsf{TGen}, \mathsf{Com}, \mathsf{TCom}, \mathsf{TCol})$.

D Scripts for Parameter Computation

Listing 1: Python Script to compute security levels of two-round multi-signatures for a fixed group sizes. A discussion is given in Section 4.3.

```
#!/usr/bin/env python



################PURPOSE#OF#THIS#SCRIPT##################
# For each scheme, we estimate the security level      #
# that is guaranteed by the security proof, assuming   #
# a certain number of hash queries and signing queries #
# and given the hardness of the underlying problem     #
########################################################



import math
from tabulate import tabulate



#number of hash queries and signing queries
log_q_h = 30
log_q_s = 20

#assumed hardness of the underlying assumption
sec_level_assumption = 128


#################Define Schemes#######################################
# Note: we estimate an upper bound on epsilon, assuming unit time    #
# One can see that this favors schemes with rewinding due to the sqrt#
######################################################################


musigtwo = {
        "name": "Musig2",
        "loss": lambda kappa : 0.25 * (kappa-2-3*log_q_h)
}

hbms = {
        "name": "HBMS",
        "loss": lambda kappa : 0.25 * (kappa-2-3*log_q_h - 4*log_q_s)
}

oursone = {
        "name": "Ours 1",
        "loss": lambda kappa : kappa - 2 - log_q_s
}

ourstwo = {
        "name": "Ours 2",
        "loss": lambda kappa : kappa - 2
}

schemes = [musigtwo,hbms,oursone,ourstwo]



####################Main Part########################


print("Q_H = 2^" +str(log_q_h))
print("Q_S = 2^" +str(log_q_s))
print("Security Level of Assumption = " + str(sec_level_assumption))
print("")

#tabulate preparation
data = [["Scheme", "Security"]]

for s in schemes:
        name = s["name"]
        sec_level = int(s["loss"](sec_level_assumption))
        data.append([name,sec_level])


#print(tabulate(data,headers='firstrow',tablefmt='fancy_grid'))
print(tabulate(data,headers='firstrow'))
```

Listing 2: Python Script to compute sizes of groups, keys, signatures, and communcation of two-round multi-signatures, assuming groups to be instantiated such that the scheme has a fixed security leve. A discussion is given in Section 4.3.

```
#!/usr/bin/env python



################PURPOSE#OF#THIS#SCRIPT####################
# For each scheme, we estimate the required group size   #
# to guarantee 128 bit security for the scheme using the #
# bound of the reduction. Then, we compute the signature #
# and communication size                                 #
##########################################################



import math
from tabulate import tabulate
```

```
#number of hash queries and signing queries
log_q_h = 30
log_q_s = 20

#target hardness for the scheme
sec_level = 128



#################Define Schemes#########################################
# Note: we estimate an upper bound on epsilon, assuming unit time     #
# One can see that this favors schemes with rewinding due to the sqrt#
########################################################################


musigtwo = {
        "name": "Musig2",
        "inv_loss": lambda kappa : 4*kappa + 2+3*log_q_h,
        "pk_size": lambda kappa : 1*(2*kappa+1),
        "com_size": lambda kappa : 5*(2*kappa+1),
        "sig_size": lambda kappa : 2*(2*kappa+1)
}

hbms = {
        "name": "HBMS",
        "inv_loss": lambda kappa : 4*kappa+2+3*log_q_h+4*log_q_s,
        "pk_size": lambda kappa : 1*(2*kappa+1),
        "com_size": lambda kappa : 3*(2*kappa+1),
        "sig_size": lambda kappa : 3*(2*kappa+1)
}

oursone = {
        "name": "Ours 1",
        "inv_loss": lambda kappa : kappa + 2 + log_q_s,
        "pk_size":  lambda kappa : 2*(2*kappa+1),
        "com_size": lambda kappa : 4*(2*kappa+1) +128,
        "sig_size": lambda kappa : 7*(2*kappa+1)
}

ourstwo = {
        "name": "Ours 2",
        "inv_loss": lambda kappa : kappa + 2,
        "pk_size":  lambda kappa : 4*(2*kappa+1),
        "com_size": lambda kappa : 8*(2*kappa+1) +128+1,
        "sig_size": lambda kappa : 15*(2*kappa+1) #assuming number of signers <= 2*kappa+1
}

schemes = [musigtwo,hbms,oursone,ourstwo]


####################Main Part########################


print("Q_H = 2^" +str(log_q_h))
print("Q_S = 2^" +str(log_q_s))
print("Target Security Level For Scheme = " + str(sec_level))
print("")



data = [["Scheme", "Group [bits]", "Pk [Bytes]", "Comm. [Bytes]", "Sig [Bytes]"]]

for s in schemes:
        name = s["name"]
        assumption_sec_level = s["inv_loss"](sec_level)
        group_size = 2*assumption_sec_level+1
        size_pk = s["pk_size"](sec_level)
        size_com = s["com_size"](sec_level)
        size_sig = s["sig_size"](sec_level)
        data.append([name,group_size,size_pk/8,size_com/8,size_sig/8])


#print(tabulate(data,headers='firstrow',tablefmt='fancy_grid'))
print(tabulate(data,headers='firstrow'))
```