Two-Round Trip Schnorr Multi-Signatures via Delinearized Witnesses

Handan Kılınç Alper and Jeffrey Burdges

Web3 Foundation

Abstract. We construct a two-round Schnorr-based signature scheme (DWMS) by delinearizing two pre-commitments supplied by each signer. DWMS is a secure signature scheme in the algebraic group model (AGM) and the random oracle model (ROM) under the assumption of the hardness of the one-more discrete logarithm problem and the 2-entwined sum problem that we introduce in this paper. Our new *m-entwined sum* problem tweaks the k-sum problem in a scalar field using the associated group. We prove the hardness of our new problem in the AGM assuming the hardness of the discrete logarithm problem in the associated group. We believe that our new problem simplifies the security proofs of multi-signature schemes that use the delinearization of commitments.

1 Introduction

A multi-signature scheme is a signature scheme that allows multiple parties collaboratively to sign a message so that the final signature can be verified with the public keys of the signers by a user. The trivial solution for this is that each entity signs the data individually and provides it to the user. However, this is not a space and time-efficient solution since the user needs to verify each signature separately and keep a larger amount of signatures. Hence, we require more elegant multi-signature schemes which save time and space on the user and the signer side.

To this end, multi-signature schemes [3, 6, 9, 15, 17–19, 22] have been studied for a long time. Increased operational security demands have driven a growth spurt in multi-signer implementations in recent years especially for Schnorr-based multi-signature schemes [2, 9–11, 17, 24]. At their core, any multi-signature scheme should protect each honest signer who participates against forgeries by an adversary who controls all the other signers and interacts extensively with our one honest signer. Yet in [11], Drijvers, et al. broke all previously known Schnorr-based two-round multi-signature protocols [3, 18, 19, 26], using the traumatic ROS [8] or k-SUM attack [27] that break blind Schnorr signatures [23, 25]. In short, these attacks are executed by the adversary who engages in enough parallel signing sessions to reply with appropriate witnesses against honest witnesses in the first round of each session. Then, the linear combination of signatures in each session constitutes a forgery.

In this paper, we propose a simple and lightweight two-round Schnorr-based multi-signature protocol that we call delinearized witness multi-signatures

(DWMS) and prove its security. We call the linear combination of the witnesses with random oracle outputs delinearization. The reason of this naming is that the coefficients of the linear combinations are random and cannot be known by the adversary before the adversary selects its own witnesses. The security of DWMS is based on the hardness assumptions of entwined sum problem that we define and one more discrete logarithm (OMDL) problem [5, 7]. In more detail, our contributions in this paper are as follows:

- We define a new computationally hard problem that we called m-entwined sum problem. We show that this problem is hard as long as $m > 1$ and the discrete logarithm is hard in the algebraic group model (AGM) [12] and the random oracle model (ROM). Thanks to this problem, we prove the security of our signature scheme without any complicated linear algebra analysis. Thus, we avoid possible mistakes in the proof. We believe that our new problem improves and simplifies the security proof of multi-signature [20] and threshold signature schemes [16] based delinearization of witnesses as ours. Beyond the simplified proof, we wonder if it is possible to improve Clause Schnorr-based blind signature [13] invoking the m-entwined sum problem instead of the modified-ROS problem [13] which does not have any extensive cryptanalysis yet.
- We construct our new protocol DWMS which consists of two-rounds: first, all signers generate two witnesses and propose two pre-commitment of them in the prime-order group, and second, after obtaining all pre-commitments, all signers compute the Schnorr commitment by delinearizing these pre-commitment with a random oracle and produce their signature share using their portion of the combined witnesses. As we mentioned above, most of the two-round Schnorr-based multi-signature schemes can be broken by solving the k-sum problem with respect to the session witnesses. Similarly, our new protocol DWMS can be broken by solving more complex k-sum problem that we call the m-entwined sum problem. Since we show that the m-entwined sum problem is hard when $m > 1$, DWMS is not vulnerable to this attack as long as the number of witnesses is at least two. After making sure that the 2-entwined sum problem is hard, we prove the security of DWMS in the plain public-key model [3], in the AGM and the ROM under the assumption that the OMDL problem and the 2-entwined sum problem is hard.

We note that DWMS is implemented in the cryptographic library 'schnorrkel/sr25519' [1] and it is used by substrate based blockchains since January 2020 as an option for multi-signatures.

1.1 Related Work

Insecure Multi-Signatures: Drijvers et al. [11] invalidated the security of some Schnorr-based two-round multi-signature schemes [3, 18, 19, 26] by showing an attack based on the k-sum problem [27]. The key observation that Drijvers et

al. [11] made was that a multi-signature participant choosing her signature randomness (nonce) after other participants could launch a parallel attack by initiating multiple concurrent signature sessions (say k of them) and let the honest participants commit to their randomness across all those sessions before choosing its own randomness for these sessions. It could then choose her randomness so that the hash on the sum of session randomness across the k sessions equals the sum of hashes on each individual session randomness, thereby helping it to derive a multi-signature session on a message of its own choice. They also noted a subtle flaw in the security proofs of these schemes, where the reduction to DL (or ODML depending on the scheme) on rewinding the multi-signature forger was inadvertently giving two signatures of the honest signer on the same randomness and two different challenges to the forger, thereby letting the latter potentially derive the honest signer's secret key. Note, all the earlier multi-signature schemes used different techniques: BCJ1, BCJ2 [2, 3] used a multiplicatively homomorphic equivocal commitment scheme to prove security under the DL assumption, MWLD [18] used Okamoto signatures [22] and the double-hashing technique to prove security under the DL assumption and MuSig [19] used regular Schnorr signatures and proved security under the OMDL assumption, but all of them crucially relied on the rewinding of the multi-signature forger to derive its security. Thus, mBCJ [11] ruled out the possibility of the existence of secure two-round multi-signatures based on Schnorr signatures via an impossibility result that formalized the above inconsistency in the proof to construct a meta-reduction (which now simulated the multi-signature forger for the DL/OMDL reduction) to solve the OMDL problem itself.

A recent work of Bellare et al. [4] introduced a new problem called the Multi-Base Discrete Log (MBDL) problem that enables the security of several schemes including Schnorr signatures, Okamoto signatures, Bellare-Neven multi-signatures, and other Schnorr based ring and threshold signatures to be proved using a non-rewinding reduction.

Random inhomogeneities in a Overdetermined Solvable system of linear equations (ROS): Earlier blind signatures constructed from Schnorr signatures [25] were shown to be insecure via a k-SUM attack [27]. The security of these blind signatures had been proven under the ROS assumption which was shown to be false as Wagner's (sub-exponential) algorithm on the k-SUM problem was used to break the ROS problem. Fuchsbauer et. al [14] was able to overcome this attack and constructed a two-round blind signature scheme that is proved secure under a (new) modified-ROS assumption that does not seem to fall prey to a k-SUM attack. Recently, Benhamouda et al. [8] find an algorithm that solves the ROS problem in polynomial time for large enough dimensions.

Secure 2-round Schnorr-based Multi-Signatures: We compare the existing 2-round Schnorr-based Multi-Signatures in Table 1. We note that we do not give the key aggregation operations in Table 1 to obtain the aggregated public key PK because this is necessary step for all multi-signature protocols. Some of them [11] first verifies the proof of knowledge of each public key with

Table 1. Review of the efficiency and security of existing Schnorr-based multi-signature schemes with n-signers. Optim. is the optimized DWMS where signers are organised in a tree structure: i.e., each multi-signature round takes 2-sub-rounds: parent-children communication only [11]. $\mathbb{G}$ is the prime p order group that the schemes work. $k$exp shows k-exponentiation in $\mathbb{G}$. NIZK is non-interactive zero knowledge and NIZK proof and NIZK verify corresponds to the number of exp in order to execute the prove and verify algorithm. pk is the individual public key and PK is the aggregated public key.

Protocol	**Sign**		**Verify**	**Domain**			**Security**
				pk	**signature**	PK	
mBCJ [11]	5exp		6exp	$\mathbb{G} \times \mathbb{Z}_p^2$	$\mathbb{G}^2 \times \mathbb{Z}_p^3$	$\mathbb{G}$	DL, ROM
MuSig-DN [21]	NIZK proof + n NIZK verification		2exp	$\mathbb{G}$	$\mathbb{G} \times \mathbb{Z}_p$ or $\mathbb{Z}_p \times \mathbb{Z}_p$	$\mathbb{G}$	DL, DDH, ZK, PRF,ROM
Musig2 $v = 4$ **[20]**	7exp		2exp	$\mathbb{G}$	$\mathbb{G} \times \mathbb{Z}_p$ or $\mathbb{Z}_p \times \mathbb{Z}_p$	$\mathbb{G}$	$4q_s$-OMDL, ROM
Musig2 $v = 2$ **[20]**	3exp		2exp	$\mathbb{G}$	$\mathbb{G} \times \mathbb{Z}_p$ or $\mathbb{Z}_p \times \mathbb{Z}_p$	$\mathbb{G}$	$2q_s$-OMDL, AGM, ROM
Ours (DWMS $m = 2$**)**	(Worst Case)	$(2n+2)$exp	2exp	$\mathbb{G}$	$\mathbb{G} \times \mathbb{Z}_p$ or $\mathbb{Z}_p \times \mathbb{Z}_p$	$\mathbb{G}$	q_s-OMDL, 2-entwined sum, AGM, ROM
	(Optim.)	$(2n+2)$exp by the root, 2exp by others					

two-exponentiation and then obtains the aggregated public key and some of them [20, 21] including DWMS delinearize each public key by one exponentiation and then obtain the aggregated public key.

mBCJ [11] is one of the few existing two-round Schnorr-based signature schemes. mBCJ is more efficient in terms of signing operations than DWMS, but DWMS is more efficient in terms of verification operations and has also shorter signature size (See Table 1). mBCJ's first round messages can be spread by aggregating. Thus, the signers use the network bandwidth more efficiently. One advantage of mBCJ over DWMS is that it is secure in the random oracle model while DWMS is secure in AGM and the random oracle model.

Aside from mBCJ, MuSig-DN [21] is another Schnorr multi-signature protocol which provides deterministic witnesses, a nice property previously unavailable in a Schnorr multi-signature. It achieves determinism using several novel bulletproof optimizations which require a suitable group for an efficient instantiation. In MuSig-DN, the first round messages require only 1124 bytes per signer, but their participant-only benchmarks show 0.9 second proving times.

In DWMS, all signers incur a per signer cost of only 64 bytes and only two scalar multiplications in a prime order-elliptic curve group. Besides, DWMS asks no special features of the underlying group as MuSig-DN.

FROST [16] is a recent Schnorr-based threshold signature scheme that uses the delinearization of witnesses in the first round similar to DWMS. However,

the security proof of FROST applies an ad-hoc heuristic that resembles a Fiat-Shamir transform which adds an additional round the FROST. Therefore, the proven FROST and the original FROST protocol are different. We hope that the hardness of our new problem 'the m-entwined sum problem' or its extensions improve the security proof of FROST.

There exists also simultaneous and independent protocol MuSig2 [20] which applies a similar idea as our DWMS protocol. DWMS's pre-commitments that signers send in the first round are delinearized first and then aggregated while Musig2's pre-commitments can be aggregated before delinearizing the pre-commitments. Therefore, the first round messages in Musig2 can be spread by aggregating. Thus, signers use the network bandwidth more efficiently. Besides, the signers in DWMS exponentiate each pre-commitment by a random oracle output for the delinearization while signers in Musig2 exponentiate only the summation of pre-commitments by a random oracle output. As a result of this, Musig2 is more efficient in terms of signing operations than DWMS (See Table 1). MuSig2 is secure in AGM with two pre-commitments as DWMS. They also show that its security with four pre-commitments without AGM. They deal with the case, where we need the hardness of the 2-entwined sum problem in our security proof in AGM, inside their signature proof with complex linear algebra analysis. It makes the proof much longer and hard to follow and verify. This also shows the ease that the m-entwined sum problem provides.

2 Preliminaries

2.1 Notations

We denote by $\mathbb{G}$ a prime p-order group. For the sake of representation, we consider *additive operation* in $\mathbb{G}$ in this paper. The notations with capital letters represent the elements of $\mathbb{G}$ and with small letters represent the elements in the scalar field $\mathbb{Z}_p$. The addition and multiplication operation between elements of $\mathbb{Z}_p$ is always in $\mod p$ even though we do not specifically write it.

We use superscript (i) on any notation where $i \in \mathbb{N}$ to distinguish values generated in a session i e.g, $X^{(i)}$.

We use bold style for the vectors. If elements of the vector are from $\mathbb{G}$ then we represent the vector with the capital bold letter e.g. $\boldsymbol{V}$. If they are from $\mathbb{Z}_p$ then we represent the vector with the small bold letter e.g. $\boldsymbol{v}$.

2.2 Security Definitions

Multi-Signature Schemes: We describe the multi-signature scheme and the security model that we consider for DWMS.

Definition 1 (Multi-Signature Scheme). *Multi-signature scheme with the security parameter λ consists of the following algorithms.*

- $\mathsf{ParamGen}(\lambda) \to par$: *It generates the parameters of the signature scheme par with respect to the security parameter λ.*

- $\mathsf{KeyGen}(par) \rightarrow (\mathsf{sk}, \mathsf{pk})$*: It generates a secret/public key pair* $(\mathsf{sk}, \mathsf{pk})$ *with the input par.*
- $\mathsf{Sign}(par, \mathsf{sk}, msg) \rightarrow \sigma$*: It is an interactive algorithm which is run between other signers to sign a message* $msg \in \mathcal{M}$ *where* $\mathcal{M}$ *is the message space.*
- $\mathsf{KeyAg}(par, \{\mathsf{pk}_i\}_{i=1}^n) \rightarrow PK$*: It receives public key* pk_i *of each signer and generates the aggregated public key* PK*.*
- $\mathsf{Verify}(par, PK, \sigma, msg) \rightarrow 1/0$*: It verifies whether the signature* σ *is signed by the parties with public key* pk_i *for the message* msg*.*

We consider the plain public-key model [3] for the security of our multi-signature scheme as described below. In this model, $\mathsf{KeyVerify}(par, \mathsf{pk}_i)$ outputs always 1.

Definition 2 (Multi-Signature Security in the Plain Public-key Model [3]). *The challenger generates the parameters par with* $\mathsf{ParamGen}(\lambda)$ *and generates a secret/public key pair* $(\mathsf{sk}, \mathsf{pk})$ *and gives* par, pk *to the adversary* $\mathcal{A}$*.* $\mathcal{A}$ *has access to the signing oracle* Σ *which returns* $\mathsf{Sign}(par, \mathsf{sk}, msg)$ *given input* msg *by* $\mathcal{A}$*.*

In the end, $\mathcal{A}$ *outputs a signature* σ^**, message* $msg^* \in \mathcal{M}$ *and* $\mathcal{PK} = \{\mathsf{pk}_1, \mathsf{pk}_2, \ldots, \mathsf{pk}_n\}$*.* $\mathcal{A}$ *wins if*

- *the public key given by the challenger is in* $\mathcal{PK}$ *i.e.,* $\mathsf{pk} \in \mathcal{PK}$*,*
- $\mathcal{A}$ *never queries with the input* msg^* *to the signing oracle* Σ *before,*
- *the signature is valid i.e.,* $\mathsf{Verify}(par, PK, \sigma, msg) \rightarrow 1$ *where* $\mathsf{KeyAg}(par, \mathcal{PK}) \rightarrow PK$*.*

We say that a multi-signature scheme is secure in the plain public-key model if for all probabilistic polynomial time (PPT) adversary $\mathcal{A}$ *with* q_s*-signing oracle queries, the probability of winning the above game is* ϵ *which is negligible in terms of* λ*.*

There is also a weaker security definition called the knowledge of secret key (KOSK) model where the adversary outputs its secret keys in the end of the game.

Algebraic Group Model (AGM) The AGM is a model between the standard model and the generic group model, i.e., the security in the standard model implies security in the AGM and the security in the AGM implies the security in the GGM [12]. In short, the AGM is a computational model that considers only algebraic algorithms (corresponds to the adversaries in the security proofs) as described below. As in the standard model, it is possible to have security proofs with reductions [12].

Definition 3 (Algebraic Adversary). *Let* $\mathbb{G}$ *be a group with order* p *and* P *be the element of* $\mathbb{G}$*. Informally, we call an algorithm* $\mathcal{A}$ *is algebraic if it fulfills the following requirement: whenever* $\mathcal{A}$ *outputs* $Z \in \mathbb{G}$*, "representation"* $\boldsymbol{z} = (z_1, \ldots, z_t) \in \mathbb{Z}_p^t$ *such that* $Z = \boldsymbol{z} \cdot \boldsymbol{L} = \sum_{i=1}^t z_i L_i$ *where* $\boldsymbol{L} = (L_1, \ldots, L_t)$ *is the vector in* $\mathbb{G}^t$ *and each element of* $\boldsymbol{L}$ *is a group element that* $\mathcal{A}$ *has seen during its execution so far.*

Now, we give the OMDL problem [5, 7] since DWMS's security relies on the hardness of the OMDL problem.

Definition 4 (n-OMDL Problem [5, 7]). *Given a prime p-order group $\mathbb{G}$ generated by P, and random $Y_1, Y_2, \ldots, Y_{n+1} \in \mathbb{G}$ and a discrete logarithm oracle* $\mathsf{DLOracle}$ *which returns discrete logarithm of any given input in $\mathbb{G}$, if a PPT adversary $\mathcal{A}$ outputs the discrete logarithms of $Y_1, Y_2, \ldots, Y_{n+1}$ with the access of at most n-times to the* $\mathsf{DLOracle}$*, then $\mathcal{A}$ solves the n-OMDL problem. We say that n-OMDL problem is hard in $\mathbb{G}$, if for all PPT adversaries, the probability of solving the n-OMDL problem is ϵ_{omdl} which is negligible in terms of the security parameter.*

2.3 Multi-signature Schemes vs. the k-sum Problem

In this section, we show an example attack to one of the broken schemes in [11] to explain why a solution of the k-sum problem helps an adversary to break some multi-signature schemes. This section will make more clear the necessity of defining a new k-sum-like problem for our new scheme DWMS and proving its hardness.

Definition 5 (k-sum Problem). *Given k-lists $\mathsf{L}_1, \mathsf{L}_2, \ldots, \mathsf{L}_k$ of size s_L where each elements of L_i is from $\mathbb{Z}_p$, the problem requires to find $x_1, x_2, \ldots, x_k$ where $x_i \in \mathsf{L}_i$ such that $x_1 + x_2 + \ldots + x_k \equiv 0 \mod p$.*

Wagner [27] described an algorithm which solves the k-sum problem within $O(k2^{n/(1+\log n)})$ when $s_L = 2^{n/(1+\log n)}$ where n is the bit length of the elements in lists. Benhamouda et al. [8] recently showed that with the list of dimensions $s_L = k > \log p$, the ROS-problem can be solved in polynomial time. Recall that the ROS problem is a generalization of the k-sum problem. Solving the ROS problem is sufficient to break security of the schemes [3, 18, 19, 26].

Drijvers et al. [11] showed an attack on many two-round Schnorr-based multi-signatures [3, 18, 19, 26] based on a solution of the k-sum problem. We next show the attack on CoSi [26] described in [11] to illustrate the relation between the k-sum problem and multi-signature schemes. The other attacks on other schemes [3, 18, 19] can be found in [11].

CoSi Protocol [26]: CoSi works in a prime p-order group $\mathbb{G}$ with the hash function $H : \{0,1\}^* \to \mathbb{Z}_p$. Public parameters are the group structure $(\mathbb{G}, P, p)$ where P is a generator of $\mathbb{G}$. Normally, each public key of the signers are associated with the proof-of-knowledge of the secret key to prevent the rogue key attacks. We omit this part in the description of the protocol because the attack is still applicable even if the public keys are generated as described in the protocol.
Key Generation: Each signer generates a random private key x and computes the public key $X = xP$.
Signing: Each signer i does the following to sign a message msg with public keys $\mathcal{PK} = \{X_1, X_2, \ldots, X_n\}$:

- **[Round 1:]** Each signer i picks $r_i \in \mathbb{Z}_p$ and computes $T_i = r_i P$. Then, each publishes the witness T_i.
- **[Round 2:]** After receiving all witnesses, each signer i aggregates all witnesses and obtains $T = \sum_{j=1}^{n} T_j$ and computes the partial signature $s_i = r_i + cx_i$ where $c = H(T, msg)$. In the end, each publishes s_i.

The signature of the message msg is $\sigma = (c, s)$ where $s = \sum_{j=1}^{n} s_i$.
Key Aggregation: Given list of public keys $\mathcal{PK} = \{X_1, X_2, \ldots, X_n\}$, the aggregated public key is $X = \sum_{j=1}^{n} X_j$.
Verification: The verification algorithm outputs 1 if $c = H(sP - cX, msg)$. Otherwise, it outputs 0.

Now, we explain the k-sum attack on the CoSi protocol described in [11].

The k-sum Attack on CoSi [11]: For simplicity, we describe the attack with one honest party having the public key X_1 and one adversary having the public key X_2 and we let $\mathcal{PK} = \{X_1, X_2\}$. In a nutshell, the adversary aims in this attack to find an appropriate adversarial witnesses for each parallel session that is started with an honest party such that the linear combination of the honest partial signatures lets the adversary obtain a forgery. It consists of the following steps:

1. The adversary starts q_s concurrent session for an arbitrary message(s) $msg^{(i)}$ with an honest signer and obtains q_s-witnesses $T_1^{(1)}, T_1^{(2)}, \ldots, T_1^{(q_s)}$ of the honest signer for each session $i \in [1, q_s]$. Now, the adversary needs to select corresponding witnesses for each session. For this selection, it continues with the next step.
2. The adversary creates $q_s + 1$ lists of size s_L. It creates the first q_s list L_i as follows: picks a random element $r_2^{(i)} \in \mathbb{Z}_p$ and lets $T_2^{(i)} = r_2^{(i)} P$ as a possible adversarial witness against the honest witness $T_1^{(i)}$, computes $c^{(i)} = H(\sum T_1^{(i)} + T_2^{(i)}, msg^{(i)})$ and adds $c^{(i)}$ to the list L_i. The adversary repeats this process s_L times until the size of L_i is s_L. It creates the last list L_{q_s+1} differently. For each element of L_{q_s+1}, it picks a message msg^* and adds $-H(\sum_i^{q_s} T_1^{(i)}, msg^*)$ to L_{q_s+1}. The adversary repeats this process s_L times until the size of L_{q_s+1} is s_L.
3. Adversary finds elements $c^{(i)}$ in each list L_i such that $c^{(1)} + c^{(2)} + \ldots c^{(q_s)} + c^{(q_s+1)} = 0$ as in the k-sum problem (Definition 5) i.e.,

$$c^{(1)} + c^{(2)} + \ldots + c^{(q_s)} = H(\sum_{i=1}^{q_s} T_1^{(i)}, msg^*) \tag{1}$$

Then the adversary gets the corresponding adversarial witness $T_2^{(i)}$ that was selected in step 2 for each found element $c^{(i)}$ for $i \in [1, q_s]$. In the end, it responds with $T_2^{(1)}, T_2^{(2)}, \ldots, T_2^{(q_s)}$ respectively to each q_s session that was initiated in the beginning and receives partial signatures for all sessions

$s_1^{(1)}, \ldots, s_{q_s}^{(q_s)}$ where $s_1^{(i)} = r_1^{(i)} + x_1 c^{(i)}$ and $c^{(i)} = H(T_1^{(i)} + T_2^{(i)}, msg^{(i)})$. Thus, all sessions end.
This step of the attack is equivalent to solving the k-sum problem [27] with the lists $\mathsf{L}_1, \mathsf{L}_2, \ldots, \mathsf{L}_{q_s+1}$ where $k = q_s + 1$.

4. Now, the adversary outputs (c^*, s^*), as a forgery of the message msg^* signed by $\mathcal{PK}$ where

$$s^* = \sum_{i=1}^{q_s} s_1^{(i)} + x_2 c^*$$

and $c^* = H(\sum_{i=1}^{q_s} T_1^{(i)}, msg^*)$. This is a valid forgery because

$$\begin{aligned} s^*P - c^*X &= (\sum_{i=1}^{q_s} s_1^{(i)})P + c^* x_2 P - c^*(X_1 + X_2) \\ &= (\sum_{i=1}^{q_s} T_1^{(i)} + c^{(i)} X_1) - c^* X_1 \qquad (2) \\ &= \sum_{i=1}^{q_s} T_1^{(i)} \qquad (3) \end{aligned}$$

We obtain Equation (3) by using Equation (1) that the adversary obtained in the third step.

The main reason for the above attack comes from the fact that the adversary selects its witnesses after seeing the honest witnesses. Therefore, three-round Schnorr-based multi-signatures are not vulnerable to this type of attack because the adversary has to commit its witness in the first round before seeing the honest witness. Existing two-round multi-signature protocols overcome this issue in different ways. Witnesses in mBCJ [11] are generated with different group elements for each message. This makes the adversary's job hard to generate compatible list L_{q_s+1}. In Musig-DN [21], the witnesses are generated deterministically so that the adversary cannot choose an appropriate witness as in the above attack. We consider a different solution. We observe that the independence of the last list L_{q_s+1} from the selection of the adversarial witnesses is causing to apply this attack. Therefore, we decide to solve the issue by preventing this independence in our scheme. Accordingly, we solve this problem in our scheme by delinearizing each witness by a corresponding random value generated with the adversarial and honest witnesses. We will explain this in the next sections in more detail.

3 Delinearized Witness Multi-Signature (DWMS)

We give our two-round delinearized witness multi-signature (DWMS) protocol that replaces the witness sharing and combination steps in multi-signer Schnorr protocol with a delinearization phase inspired by the delinearization defense [6] against rogue key attacks. The DWMS protocol works in prime p-order group $\mathbb{G}$

with the functions $H : \{0,1\}^* \times \mathbb{G} \times \mathbb{G} \to \mathbb{Z}_p$, $H_1 : \{0,1\}^* \times \mathbb{G}^{mn} \times \mathbb{N} \times \mathbb{N} \to \mathbb{Z}_p$ and $H_2 : \mathbb{G}^n \times \mathbb{G} \to \mathbb{Z}_p$ where $n \geq 1$ is the number of signers and m is a parameter.

Public Parameter Generation ($\mathsf{ParamGen}(\lambda)$) Given λ, $\mathsf{ParamGen}$ generates a prime p order group $\mathbb{G}$ and a generator $P \in \mathbb{G}$. In the end, it outputs $par = (\mathbb{G}, P, p)$.

Key generation ($\mathsf{KeyGen}(par) \to (\mathsf{sk}, \mathsf{pk})$) Each signer generates a random private key $x \in \mathbb{Z}_q$ and computes the public key $X = xP$.

Signing ($\mathsf{Sign}(par, x_i, msg) \to \sigma$)**.** It consists of two rounds where in the first round signers exchange their pre-commitments and in the second round signers generate their signature by delinearizing their pre-commitments.

- **[Round 1]:** Each signer i with the secret key $x_i \in \{x_1, x_2, \ldots, x_n\}$ generates random witnesses $r_{i1}, r_{i2}, \ldots, r_{im} \in \mathbb{Z}_p$ and computes the pre-commitments $T_{i1} = r_{i1}P, T_{i2} = r_{i2}P, \ldots, T_{im} = r_{im}P$ and broadcasts $(T_{i1}, T_{i2}, \ldots, T_{im})$ together with their public key X_i[1]. This round ends when each signer i receives all pre-commitments. Let $\mathcal{PK} = \{X_1, \ldots, X_n\}$ be the multi-set of all public keys involved in the session.
- **[Round 2]:** On receiving $(T_{j1}, T_{j2}, \ldots, T_{jm})$ from the co-signers, each signer i sets the list of public keys $\mathcal{PK} = \{X_1, X_2, \ldots, X_n\}$, it computes the delinearization parameter of each key $X_j \in \mathcal{PK}$: $a_j = H_2(\mathcal{PK}, X_j)$ and finds the aggregated public key $X = \sum_{j=1}^{n} a_j X_j$. This step is necessary to prevent rogue-key attacks. It lets the session identifier be

$$\mathsf{SID} = (\mathcal{PK}, msg, \{T_{1j}\}_{j=1}^{m}, \ldots, \{T_{nj}\}_{j=1}^{m}).$$

  Then, it computes delinearization scalars for each pre-commitments T_{uj} where $u \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots, m\}$ which are

$$\alpha_{ij} = H_1(\mathsf{SID}, i, j)$$

  and computes the individual delinearized commitment of each signer u:

$$T_u = \sum_{j=1}^{m} \alpha_{uj} T_{uj}, \text{for } u \in [1, n]$$

  Signer i further computes $T = \sum_{u=1}^{n} T_u$ and $c = H(msg, X, T)$ and the signature

$$s_i = (\sum_{j=1}^{m} \alpha_{ij} r_{ij}) + c a_i x_i$$

  and broadcasts s_i as its signature.
  On receiving other signatures s_j's from the co-signers, signer can compute $s = \sum_{j=1}^{n} s_i$. The multi-signature on message msg under the aggregated public key X is $\sigma = (c, s)$. We remark that the signature is the same as the Schnorr signature.

[1] There can be a specified structure (e.g., tree structure [11]) between parties for more efficient communication.

Key Aggregation ($\mathsf{KeyAg}(par, \mathcal{PK})$)**.** It outputs the aggregated public key $X = \sum_{i=1}^{n} a_i X_i$ where $a_i = H_2(\mathcal{PK}, X_i)$
Verification ($\mathsf{Verify}(par, X, \sigma, msg)$)**.** It accepts the signature if c equals to $H(msg, X, sP - cX)$ as in the standard Schnorr signature scheme.

Network and Computation Optimization: In the worst case, each signer sends the first round messages (m-exponentiations) to all other signers, each signer computes T (mn-exponentiation) and sends the partial signature to all other signers. So, the communication complexity is $O(n^2)$ and computation complexity is $O(n)$ i.e., the number of exponentiations made by each signer is $mn + m$. We can optimize this with the tree based network topology suggested in [11]. The leaf nodes start the protocol by sending their pre-commitments to their parents. When a party P_i receives pre-commitments from its children, it relays them and its own pre-commitments $T_{i1}, \ldots, T_{im}$ to its parent. In the end, the root node receives all of them. Then, the root node computes $T = \sum_{u=1}^{n} T_u$ and sends T to its children so that the leaf nodes receives them. Finally, each leaf node gives its partial signature to its parent to be added parent's partial signature and the root node obtains the multi-signature. The communication complexity is $O(n)$ and the computation complexity is $O(n)$ ($mn + m$-exponentiations) for the root node and $O(1)$ (m-exponentiations) for the other parties. The computation can be distributed among other nodes where each node computes m-exponentiation by additional 2-sub-rounds. In this case, the root node should send the pre-commitments to the children so that the computation of T can be done from leaves to root by aggregation.

We prove in Section 5 the security of DWMS when $m = 2$. However, we show below that it is not secure when $m = 1$ by solving a k-sum problem. Then, we discuss whether such an attack is possible when $m > 1$ to illustrate a relation between our new problem 'the m-entwined sum problem' and these attacks in DWMS.

Forgery Attack When $m = 1$: For simplicity, we describe the attack with one honest party with the public key X_1 and one adversary with the public key X_2 and we let $\mathcal{PK} = \{X_1, X_2\}$, $a_1 = H_2(\mathcal{PK}, X_1)$ and $a_2 = H_2(\mathcal{PK}, X_2)$, the aggregated key $X = a_1 X_1 + a_2 X_2$. As in the CoSi attack in Section 2.3, the adversary wants to find adversarial witnesses for each parallel session to obtain a forgery with the linear combinations of honest partial signatures. It consists of the following steps:

1. The adversary starts q_s-concurrent session for an arbitrary message(s) $msg^{(i)}$ with an honest signer and obtain q_s-pre-commitments $T_{11}^{(1)}, T_{11}^{(2)}, \ldots, T_{11}^{(q_s)}$ of the honest signer for each session $i \in [1, q_s]$. Now, the adversary needs to select the corresponding witnesses for each session. For this selection, it continues with the next step.
2. The adversary creates $q_s + 1$ lists of size s_L. It creates the first q_s list L_i as follows: picks a random element $r_{21}^{(i)} \in \mathbb{Z}_p$ and lets $T_{21}^{(i)} = r_{21}^{(i)} P$ as a possible adversarial pre-commitments against the honest pre-commitment

$T_{11}^{(i)}$, computes the corresponding $\alpha_{21}^{(i)}$ and $c^{(i)}$ as in the second round of DWMS and adds $\frac{c^{(i)}}{\alpha_{11}^{(i)}}$ to the list L_i. The adversary repeats this process s_L times until the size of L_i is s_L. It creates the last list L_{q_s+1} differently. For each element of L_{q_s+1}, it picks randomly an element $\beta' \in \mathbb{Z}_p$ and a forgery message msg^* and adds $-\frac{H(msg^*, X, \beta' \sum_{i=1}^{q_s} T_{11}^{(i)})}{\beta'}$ to L_{q_s+1}. We note that selecting β' is necessary to populate L_{q_s+1} with random elements if the forgery msg^* is fixed. The adversary repeats this process s_L times until the size of L_{q_s+1} is s_L.

3. The adversary finds elements $v^{(i)}$ in each list L_i such that $v^{(1)} + v^{(2)} + \ldots v^{(q_s)} + v^{(q_s+1)} = 0$ i.e.,

$$\frac{c^{(1)}}{\alpha_{11}^{(1)}} + \frac{c^{(2)}}{\alpha_{11}^{(2)}} + \ldots + \frac{c^{(q_s)}}{\alpha_{11}^{(q_s)}} = \frac{H(msg^*, X, \beta' \sum_{i=1}^{q_s} T_{11}^{(i)})}{\beta'} \tag{4}$$

Then the adversary gets the corresponding adversarial pre-commitment $T_{21}^{(i)}$ that was selected in step 2 for each found element $v^{(i)}$ for $i \in [1, q_s]$. In the end, it responds with $T_{21}^{(1)}, T_{21}^{(2)}, \ldots, T_{21}^{(q_s)}$ respectively to each q_s session that is initiated in the beginning and receives partial signatures $s_1^{(1)}, \ldots, s_1^{(q_s)}$ where $s_1^{(i)} = \alpha_{11}^{(i)} r_{11}^{(i)} + x_1 a_1 c^{(i)}$ and $c^{(i)} = H(msg^{(i)}, X, \alpha_{11}^{(i)} T_{11}^{(i)} + \alpha_{21}^{(i)} T_{21}^{(i)})$. Thus, all sessions end.
This step of the attack is equivalent to solving the k-sum problem [27] with the lists $\mathsf{L}_1, \mathsf{L}_2, \ldots, \mathsf{L}_{q_s+1}$ where $k = q_s + 1$.

4. Now, the adversary outputs (c^*, s^*), as a forgery of the message msg^* signed by $\mathcal{PK}$ where

$$s^* = \beta'(\frac{s_1^{(1)}}{\alpha_{11}^{(1)}} + \frac{s_1^{(2)}}{\alpha_{11}^{(2)}} + \ldots + \frac{s_1^{(q_s)}}{\alpha_{11}^{(q_s)}}) + c^* a_2 x_2$$

and $c^* = H(msg^*, X, T^*)$ where $T^* = \beta' \sum_{i=1}^{q_s} T_{11}^{(i)}$.. This is a valid forgery because $c^* = H(msg^*, X, T^*)$ and

$$\begin{aligned} s^* P - c^* X &= (\beta'(\frac{s_1^{(1)}}{\alpha_{11}^{(1)}} + \frac{s_1^{(2)}}{\alpha_{11}^{(2)}} + \ldots + \frac{s_1^{(q_s)}}{\alpha_{11}^{(q_s)}}) + c^* a_2 x_2) P - (c^* a_1 X_1 - c^* a_2 X_2) \\ &= \beta'(T_{11}^{(1)} + T_{11}^{(2)} + \ldots + T_{11}^{(q_s)}) + (\beta' \sum_{i=1}^{q_s} \frac{x_1 a_1 c^{(i)}}{\alpha_{11}^{(i)}}) P - c^* a_1 X_1 \qquad (5) \\ &= T^* \qquad (6) \end{aligned}$$

We obtain Equation (6) from Equation (5) by using Equation (4) that the adversary obtained in the third step.

The adversary's attack described above is equivalent to finding a message msg^*, a vector $\boldsymbol{\beta} = (\beta^{(1)}, \ldots, \beta^{(q_s)})$ and $T_{21}^{(1)}, T_{21}^{(2)}, \ldots, T_{21}^{(q_s)}$ such that $c^* = \sum \beta^{(i)} c^{(i)} = H(msg^*, X, \sum_{i=1}^{q_s} \beta^{(i)} \alpha_{11}^{(i)} T_{11}^{(i)})$. In this case, the forgery $\sigma^* = (c^*, s^*)$ of msg^* is a valid signature signed by X_1 and X_2 where $s^* = \sum_{i=1}^{q_s} \beta^{(i)} s_1^{(i)} + c^* a_2 x_2$. So, in the attack that we describe, $\boldsymbol{\beta} = (\frac{\beta'}{\alpha_{11}^{(1)}}, \frac{\beta'}{\alpha_{11}^{(2)}}, \ldots, \frac{\beta'}{\alpha_{11}^{(q_s)}})$.

Is a similar attack possible when $m > 1$?: Now, we informally discuss whether an adversary can do the similar attack when $m > 1$. We believe that these discussions give a motivation for our new problem 'the m-entwined sum problem' which we introduce in the next section.

The question we should examine is after initiating q_s parallel sessions with an honest signer and receiving honest pre-commitments $\boldsymbol{T_h^{(i)}} = (T_{11}^{(i)}, T_{12}^{(i)}, \ldots, T_{1m}^{(i)})$ for each session $i \in [1, q_s]$ whether the adversary can find a message msg^*, a vector $\boldsymbol{\beta} = (\beta^{(1)}, \ldots, \beta^{(q_s)})$ and $(T_{21}^{(1)}, \ldots, T_{2m}^{(1)}), \ldots, (T_{21}^{(q_s)}, \ldots, T_{2m}^{(q_s)})$ such that

$$c^* = \sum \beta^{(i)} c^{(i)} = H(msg^*, X, \sum_{i=1}^{q_s} \beta^{(i)} (\sum_{j=1}^{m} \alpha_{1j}^{(i)} T_{1j}^{(i)})). \tag{7}$$

In this case, the forgery $\sigma^* = (c^*, s^*)$ of msg^* is a valid signature signed by X_1 and X_2 where $s^* = \sum_{i=1}^{q_s} \beta^{(i)} s_1^{(i)} + c^* a_2 x_2$ and $s_1^{(i)} = (\sum_{i=1}^{m} \alpha_{1j}^{(i)} r_{1j}^{(i)}) + c^{(i)} a_1 x_1$.

In Definition 8, we formalize this attack as a new problem and prove its hardness when $m > 1$ in the AGM. We show that an algebraic adversary (Definition 3) cannot find $\boldsymbol{\beta} \in \mathbb{Z}_p^{q_s+1}$ and $\boldsymbol{T_{\mathcal{A}}^{(i)}}$ for $i \in [1, q_s]$ satisfying Equation (7) except with the negligible probability as long as the discrete logarithm problem is hard and $m > 1$.

In the next section, we have a break on DWMS and introduce the simple m-entwined sum and the m-entwined sum problem and show their hardness. We later prove the security of DWMS when $m > 1$ under the assumption that the q_s-OMDL and the m-entwined sum problems are hard.

4 Entwined Sum Problem

In this section, we introduce our new k-sum [27] like problem 'the entwined sum problem' and show its hardness in AGM. Before giving the problem, we introduce a version of the discrete logarithm problem (the DLR problem) that we use in the hardness proof of our entwined sum problem. DLR is equivalent to the discrete logarithm problem. We employ DLR because it is a more flexible variant of the discrete logarithm problem in which an adversary finding any new relationship among group elements interests us.

Definition 6 (Discrete Logarithm Relation (DLR) problem). *Given prime p-order group $\mathbb{G}$ generated by P and random $X_i \in \mathbb{G}$ for $i \in \{1, 2, \ldots, k\}$ with $k \geq 1$ then find $y_0, y_1, \ldots, y_k \in \mathbb{Z}_p$ such that $y_0 P + \sum_{i=1}^{k} y_i X_i = 0$ where not all y_i's are 0.*

Proposition 1. *The discrete logarithm (DLOG) problem in the group structure* $(\mathbb{G}, P, p)$ *is equivalent to the DLR problem in the same group structure.*

Proof. It is clear that if DLOG problem is easy then DLR is easy. Now, we show that if DLR is easy then DLOG is easy. Let X denote our DLOG challenge over prime p order group $\mathbb{G}$ generated by P. We compute the Pedersen commitment $X_i = a_i X + b_i P$ for random $a_i, b_i \in \mathbb{Z}_p$ for $i \in \{1, 2, \ldots, k\}$ and give $X_1, X_2, \ldots, X_k, (\mathbb{G}, P, p)$ to the DLR solver. In the end, the DLR solver outputs a DLR solution $y_0, y_1, y_2, \ldots, y_k \in \mathbb{Z}_p$ such that $y_0 P + \sum_{i=1}^{k} y_i X_i = y_0 P + \sum_{i=1}^{k} y_i (a_i X + b_i P) = 0$. It follows that $X = \frac{-(y_0 + \sum_i y_i b_i)}{\sum_i y_i a_i} P$, as desired. We remark that $\sum_i y_i a_i \neq 0$ except with probability $\frac{1}{p}$ because these Pedersen commitments X_i's are perfectly hiding.

We first introduce a simple version of our problem that we call 'the simple m-entwined sum problem'. Then, we extend the simple version and give the m-entwined sum problem. The simple m-entwined sum problem has fewer variables. When we introduce the m-entwined sum problem, we show its hardness based on the results in the proof of the simple m-entwined sum problem so that we avoid dealing with more variables that may complicate the proof.

The classical k-sum problem [27] works in the field $\mathbb{Z}_p$ (See Definition 5). Differently, an attacker in our problem works not only in the field $\mathbb{Z}_p$ but also in its associated group $\mathbb{G}$. This difference makes our problem hard as long as the DLR problem is hard in the AGM.

Definition 7 (Simple m-entwined sum Problem). *The challenger generates a prime p order group $\mathbb{G}$ with the security parameter λ and selects a generator $P \in \mathbb{G}$. As an input, the challenger supplies the group description $(p, \mathbb{G}, P)$ and vectors $\boldsymbol{T_h^{(1)}}, \boldsymbol{T_h^{(2)}}, \ldots, \boldsymbol{T_h^{(q_s)}} \in \mathbb{G}^m$ to the the adversary $\mathcal{A}$ where $\boldsymbol{T_h^{(i)}} = (T_{11}^{(i)}, T_{12}^{(i)}, \ldots, T_{1m}^{(i)})$. $\mathcal{A}$ has access to the random oracles $H, H' : \Omega \times \mathbb{G} \to \mathbb{Z}_p$ and $H_1 : \Omega \times \mathbb{G}^m \times \mathbb{N} \to \mathbb{Z}_p$ where Ω is an arbitrary set.*

In the end, the adversary outputs the following vectors $\boldsymbol{\beta} = (\beta^{(0)}, \beta^{(1)}, \ldots, \beta^{(q_s)}) \in \mathbb{Z}_p^{q_s+1}$, $\mathcal{T}_{\mathsf{out}} = (\mu^{(1)}, \mu^{(2)}, \ldots, \mu^{(q_s)}) \in \Omega^{q_s}$ and $\omega^* \in \Omega$. If the following holds, then the adversary wins the simple entwined sum game:

$$\sum_{u=1}^{q_s} \beta^{(u)} H'(\mu^{(u)}, T_h^{(u)}) = H(\omega^*, \beta^{(0)} P + \sum_{u=1}^{q_s} \beta^{(u)} T_h^{(u)}) \tag{8}$$

where

$$T_h^{(u)} = \sum_{j=1}^{m} \alpha_{1j}^{(u)} T_{1j}^{(u)} \text{ and } \alpha_{1j}^{(u)} = H_1(\mu^{(u)}, \boldsymbol{T_h^{(u)}}, j) \tag{9}$$

We call that the simple m-entwined sum problem is hard in $\mathbb{G}$, if for all PPT adversaries who access the random oracles H, H' and H_1 at most q_h,$q_{h'}$,q_{h_1}-times respectively, the probability of solving the above problem is ϵ_{eSum} which is negligible in terms of λ.

We named our problem *entwined* because the adversary should satisfy the same linear relationship in $\mathbb{Z}_p$ (see the left hand side of Equation 8) and in $\mathbb{G}$ (see the right hand side of Equation 8). Both linear relationships in $\mathbb{Z}_p$ and in $\mathbb{G}$ are constructed with respect to the challenges and adversarial outputs. We note that if the simple m-entwined sum was not a hard problem then an adversary could obtain adversarial witnesses that satisfy the Equation 7 and construct a forgery in DWMS.

We prove that the simple m-entwined sum problem when $m > 1$ is hard under the assumption that DLR problem is hard. Our hardness proof is based on the lemma (Lemma 1) that for any choice of $W \in \mathbb{G}$ as an input to the right hand side random oracle input of Equation 8, the adversary can obtain only one possible solution $(\boldsymbol{\beta}, \mathcal{T}_{\mathsf{out}})$ and vice versa. This fact makes the adversary's job hard to solve the problem.

Theorem 1. *Assume that there exits a PPT,* ***algebraic*** *adversary $\mathcal{A}$ that solves the simple m-entwined sum problem for $m > 1$ with probability ϵ_{eSum} in the group structure $(\mathbb{G}, P, p)$ with $q_h \geq q_{h'}$ and q_{h_1} random oracle queries. Then, there exists a PPT adversary $\mathcal{B}$ that solves the DLR problem in the group structure $(\mathbb{G}, P, p)$ with probability $\epsilon_{dlr} \geq \epsilon_{eSum} - \frac{q_h}{p} - \frac{q_{h_1}}{\sqrt{p}p^{m-2}}$.*

Proof. We construct an adversary $\mathcal{B}$ which simulates the simple m-entwined sum problem against an *algebraic* adversary $\mathcal{A}$. We denote by $\Pr[eSum \rightarrow 1] = \epsilon_{eSum}$ the probability that $\mathcal{A}$ solves the simple m-entwined sum problem. The DLR challenger gives the group $(\mathbb{G}, P, p)$, the challenges $T_{11}^{(1)}, \ldots, T_{1m}^{(1)}, T_{11}^{(2)}, \ldots, T_{1m}^{(2)}, \ldots, T_{11}^{(q_s)}, \ldots, T_{1m}^{(q_s)} \in \mathbb{G}$ to $\mathcal{B}$. $\mathcal{B}$ forwards them to $\mathcal{A}$ as m-entwined sum challenges. $\mathcal{B}$ simulates the random oracles H, H' and H_1 against $\mathcal{A}$ as a usual random oracle. We let

$$\boldsymbol{T}_{\mathsf{inp}} = (P, T_{11}^{(1)}, \ldots, T_{1m}^{(1)}, \ldots, T_{11}^{(q_s)}, \ldots, T_{1m}^{(q_s)})$$

which is the vector including the group elements given to $\mathcal{A}$ during the simulation and we also let $\boldsymbol{T}_{\boldsymbol{h}}^{(\boldsymbol{i})} = (T_{11}^{(i)}, T_{12}^{(i)}, \ldots, T_{1m}^{(i)})$.

Whenever $\mathcal{A}$ queries the oracle H, H' or H_1 with an input including a group element $Y \in \mathbb{G}$, it gives the representation of it $\boldsymbol{z} = (z_0, z_{11}^{(1)}, z_{12}^{(1)}, \ldots, z_{1m}^{(1)}, \ldots, z_{11}^{(q_s)}, z_{12}^{(q_s)}, \ldots, z_{1m}^{(q_s)}) \in \mathbb{Z}_p^{mq_s+1}$ such that $\boldsymbol{z} \cdot \boldsymbol{T}_{\mathsf{inp}} = Y$ because $\mathcal{A}$ is algebraic (Definition 3) and $\boldsymbol{T}_{\mathsf{inp}}$ are the group elements that $\mathcal{A}$ has seen. If $\mathcal{A}$ gives two different representation of an element $Y \in \mathbb{G}$ which are $\boldsymbol{z} \in \mathbb{Z}_p^{mq_s+1}$ and $\hat{\boldsymbol{z}} \in \mathbb{Z}_p^{mq_s+1}$ such that $\boldsymbol{z} \neq \hat{\boldsymbol{z}}$, $\mathcal{B}$ gives the vector $\boldsymbol{v} = \boldsymbol{z} - \hat{\boldsymbol{z}}$ to the DLR challenger as a solution and ends the simulation against $\mathcal{A}$. We remark that $\boldsymbol{v} \cdot \boldsymbol{T}_{\mathsf{inp}} = 0$ so it is a valid solution to the DLR problem. Otherwise, in the end, $\mathcal{A}$ outputs $\boldsymbol{\beta} = (\beta^{(0)}, \beta^{(1)}, \beta^{(2)}, \ldots, \beta^{(q_s)})$, ω^* and $\mathcal{T}_{\mathsf{out}} = (\mu^{(1)}, \mu^{(2)}, \ldots, \mu^{(q_s)})$ as a solution of the simple entwined sum problem which satisfies the below equation and the simulation against $\mathcal{A}$ ends.

$$\sum_{u=1}^{q_s} \beta^{(u)} H'(\mu^{(u)}, T_h^{(u)}) = H(\omega^*, \beta^{(0)} P + \sum_{u=1}^{(q_s)} \beta^{(u)} T_h^{(u)}) \tag{10}$$

Then, $\mathcal{B}$ computes $W = \beta^{(0)}P + \sum_{u=1}^{(q_s)} \beta^{(u)} T_h^{(u)}$. By using the fact that $T_h^{(u)} = \sum_{j=1}^{m} \alpha_{1j}^{(u)} T_{1j}^{(u)}$ (See Equation 9), $\mathcal{B}$ reorganizes W and obtains $W = \beta^{(0)}P + \sum_{u=1}^{q_s}\sum_{j=1}^{m} \beta^{(u)}\alpha_{1j}^{(u)} T_{1j}^{(u)}$. Thus, $\mathcal{B}$ obtains a representation $\boldsymbol{b} = (\beta^{(0)}, b_{11}^{(1)}, \ldots, b_{1m}^{(1)}, \ldots, b_{11}^{(q_s)}, \ldots, b_{1m}^{(q_s)})$ of W in terms of $\boldsymbol{T_{\mathsf{inp}}}$ where $b_{1j}^{(u)} = \beta^{(u)}\alpha_{1j}^{(u)}$ i.e. $W = \boldsymbol{b} \cdot \boldsymbol{T_{\mathsf{inp}}}$. Then, $\mathcal{B}$ checks the oracle queries of H with $(., W)$. We remark that there should be such query, otherwise $\mathcal{A}$ cannot check the correctness of its solution. When $\mathcal{A}$ queries $(., W)$ to H, it gives a representation $\boldsymbol{a} = (a, a_{11}^{(1)}, \ldots, a_{1m}^{(1)}, \ldots, a_{11}^{(q_s)}, \ldots, a_{1m}^{(q_s)})$ of W such that $W = \boldsymbol{a} \cdot \boldsymbol{T_{\mathsf{inp}}}$. If $\boldsymbol{b} \neq \boldsymbol{a}$, then $\mathcal{B}$ finds a solution of the DLR problem which is $\boldsymbol{b} - \boldsymbol{a}$ and wins. Otherwise, $\mathcal{B}$ aborts.

So, the success probability of $\mathcal{B}$ is

$$\epsilon_{dlr} = \Pr[\mathcal{A} \text{ wins}] - \Pr[\mathcal{A} \text{ wins} | \boldsymbol{a} = \boldsymbol{b}]$$

Next, we find $\Pr[\mathcal{A} \text{ wins} | \boldsymbol{a} = \boldsymbol{b}]$. We distinguish the session indexes i where $\beta^{(i)} \neq 0$. For this, we define an index set $\mathcal{I} = \{i : \beta^{(i)} \neq 0, i \in [1, q_s]\}$ which includes the session indexes of non-zero elements of the vector $\boldsymbol{\beta}$. We note that $\mu^{(j)} \in \mathcal{T}_{\mathsf{out}}$ where $j \notin \mathcal{I}$ does not play any role in the correctness of the m-entwined sum solution because $\beta^{(j)} = 0$. Therefore, we consider $\mathcal{T}_{\mathsf{out}} = \{\mu^{(j)}\}_{i \in \mathcal{I}}$ in the rest of the proof.

We call that $\hat{\mu} \in \Omega$ is an i^{th} potential session where $i \in \mathcal{I}$, for all $1 \leq j \leq m$ if $(\hat{\mu}, \boldsymbol{T_h^{(i)}}, j)$ is queried to the random oracle H_1. In other words, $\hat{\mu} \in \Omega$ is the i^{th} potential session, if $\mathcal{A}$ obtained all $\hat{\alpha}_{1j}^{(i)} = H_1(\hat{\mu}, \boldsymbol{T_h^{(i)}}, j)$ values to be able to compute $\hat{T}_h^{(i)} = \sum_{j=1}^{m} \hat{\alpha}_{1j}^{(i)} T_{1j}^{(i)}$. We denote by $\mathbb{O}_{H_1}^{(i)}$ the set of i^{th} potential sessions.

We define a function $\phi : \mathbb{Z}_p^{q_s+1} \times (\times_{i \in \mathcal{I}} \mathbb{O}_{H_1}^{(i)}) \rightarrow \mathbb{Z}_p^{q_s+1}$. Given input $\hat{\boldsymbol{\beta}} = (\hat{\beta}^{(0)}, \hat{\beta}^{(1)}, \ldots, \hat{\beta}^{(q_s)}) \in \mathbb{Z}_p^{q_s+1}$ and $\hat{\mathcal{T}_{\mathsf{out}}} = \{\hat{\mu}^{(i)}\}_{i \in \mathcal{I}} \in (\times_{i \in \mathcal{I}} \mathbb{O}_{H_1}^{(i)})$, we define the function ϕ as $\phi(\hat{\boldsymbol{\beta}}, \hat{\mathcal{T}_{\mathsf{out}}}) = \boldsymbol{y}$ where $\boldsymbol{y} = (y^{(0)}, y_{11}^{(1)}, \ldots, y_{1m}^{(1)}, \ldots, y_{11}^{(q_s)}, \ldots, y_{1m}^{(q_s)})$

$$\phi(\hat{\boldsymbol{\beta}}, \hat{\mathcal{T}_{\mathsf{out}}}) = \boldsymbol{y} = \begin{cases} y^{(0)} = \beta^{(0)} & \\ y_{1j}^{(i)} = \hat{\beta}^{(i)}\hat{\alpha}_{1j}^{(i)} & \text{if } i \in \mathcal{I} \\ y_{1j}^{(i)} = 0 & \text{otherwise} \end{cases} \tag{11}$$

We remark that $(\boldsymbol{\beta}, \mathcal{T}_{\mathsf{out}})$ provided by $\mathcal{A}$ as a solution to the simple m-entwined sum problem satisfies $\phi(\boldsymbol{\beta}, \mathcal{T}_{\mathsf{out}}) = \boldsymbol{b}$.

Lemma 1. *ϕ is an injective function for $m > 1$ except with the probability less than $\frac{q_{h_1}}{\sqrt{p} p^{m-2}}$.*

Proof. Let's assume that ϕ is not injective. Then, there exists $(\tilde{\boldsymbol{\beta}}, \tilde{\mathcal{T}_{\mathsf{out}}}) \neq (\hat{\boldsymbol{\beta}}, \hat{\mathcal{T}_{\mathsf{out}}})$ such that $\phi(\tilde{\boldsymbol{\beta}}, \tilde{\mathcal{T}_{\mathsf{out}}}) = \phi(\hat{\boldsymbol{\beta}}, \hat{\mathcal{T}_{\mathsf{out}}}) = \boldsymbol{y} \in \mathbb{Z}_p^{q_s+1}$. In this case, $\tilde{\beta}^{(i)}\tilde{\alpha}_{1j}^{(i)} = y_{1j}^{(i)}$ and

$\hat{\beta}^{(i)}\hat{\alpha}_{1j}^{(i)} = y_{1j}^{(i)}$ for all $i \in \mathcal{I}$ and $1 \leq j \leq m$ which implies that

$$\frac{\tilde{\alpha}_{11}^{(i)}}{\hat{\alpha}_{11}^{(i)}} = \ldots = \frac{\tilde{\alpha}_{1m}^{(i)}}{\hat{\alpha}_{1m}^{(i)}} \tag{12}$$

So, if ϕ is not injective, there should exist two different potential sessions $\hat{\mu}^{(i)}$ and $\tilde{\mu}^{(i)}$ in a potential session set $\mathbb{O}_{H_1}^{(i)}$ satisfies Equation (12). The probability that it happens is for $m > 1$, given that $|\mathbb{O}_{H_1}^{(i)}| \leq q_{h_1}$

$$\Pr\left[\frac{\alpha_{11}^{(i)}}{\hat{\alpha}_{11}^{(1)}} = \ldots = \frac{\alpha_{1m}^{(i)}}{\hat{\alpha}_{1m}^{(1)}}\right] \leq \frac{q_{h_1}}{\sqrt{p}p^{m-2}}$$

□

As a result of the lemma, for each representation $\boldsymbol{c} \in \mathbb{Z}_p^{mq_s+1}$ of W such that $W = \boldsymbol{c} \cdot \boldsymbol{T}_{\mathsf{inp}}$, $\mathcal{A}$ can have only one possible $(\tilde{\boldsymbol{\beta}}, \tilde{\mathcal{T}_{\mathsf{out}}})$ except with the probability $\frac{q_{h_1}}{\sqrt{p}p^{m-2}}$ that satisfies $W = \tilde{\beta}^{(0)}P + \sum_{u=1}^{(q_s)} \tilde{\beta}^{(u)} \sum_{j=1}^{m} \tilde{\alpha}_{1j}^{(u)} T_{1j}^{(u)}$ because $\phi(\tilde{\boldsymbol{\beta}}, \tilde{\mathcal{T}_{\mathsf{out}}}) = \boldsymbol{c}$ is injective. So, when $\mathcal{A}$ queries $(., W)$ to the oracle H and gives its representation $\boldsymbol{a}$, it can have only one appropriate $(\boldsymbol{\beta}, \mathcal{T}_{\mathsf{out}})$. Therefore, if $\mathcal{A}$ finds a solution with respect to the representation $\boldsymbol{a}$, in other words, if $\boldsymbol{a} = \boldsymbol{b}$, the probability that $\mathcal{A}$ satisfies Equation (10) with only one possible $\boldsymbol{\beta}, \mathcal{T}_{\mathsf{out}}$ for $W \in \mathbb{G}$ is $\frac{q_h}{p}$. The reason of this is that the left hand side of the Equation 8 is fixed when $\mathcal{A}$ gives the representation $\boldsymbol{a}$ because ϕ is injective. Since the probability that the fixed left hand side equals to $H(\bar{\omega}, W)$ for $\bar{\omega} \in \Omega$ is $\frac{1}{p}$, the adversary's success probability to win when $\boldsymbol{a} = \boldsymbol{b}$ is $\frac{q_h}{p}$.

As a result of this, $\epsilon_{dlr} \geq \epsilon_{eSum} - \frac{q_h}{p} - \frac{q_{h_1}}{\sqrt{p}p^{m-2}}$ which implies that ϵ_{eSum} is negligible in terms of λ.

□

We next give the m-entwined sum problem which is a version of the simple m-entwined sum problem with one more variable.

Definition 8 (m-entwined sum problem). *It is the same as the simple m-entwined sum problem in Definition 7 except that the challenger gives additionally $Y \in \mathbb{G}^n$ to the adversary and the adversary additionally outputs $v \in \mathbb{Z}_p$. Adversary wins the m-entwined sum game if*

$$\sum_{u=1}^{q_s} \beta^{(u)} H'(\mu^{(u)}, T^{(u)}) = H(\omega^*, \underline{vY} + \beta^{(0)}P + \sum_{u=1}^{(q_s)} \beta^{(u)} T_h^{(i)}) \tag{13}$$

We call that the m-entwined sum problem is hard in $\mathbb{G}$, if for all PPT adversaries who access the random oracles H, H' and H_1 at most $q_h, q_{h1'} q_{h_1}$-times respectively, the probability of solving above problem is ϵ_{eSum} which is negligible in terms of λ.

Theorem 2. *If the simple m-entwined sum problem is hard then the m-entwined sum problem is hard.*

Proof. We construct an adversary $\mathcal{B}$ that breaks the simple m-entwined sum problem given that there is another adversary $\mathcal{A}$ that breaks the m-entwined sum problem. The simulation against $\mathcal{A}$ is trivial. $\mathcal{B}$ receives the simple m-entwined sum challenges $(p, \mathbb{G}, P)$ and vectors $\boldsymbol{T_h^{(1)}}, \boldsymbol{T_h^{(2)}}, \ldots, \boldsymbol{T_h^{(q_s)}} \in \mathbb{G}^m$. Then, $\mathcal{B}$ picks randomly $y \in \mathbb{Z}_p$ and sends the simple m-entwined sum challenges and *additionally* $Y = yP$ to $\mathcal{A}$. Whenever $\mathcal{A}$ queries any random oracle with an input, $\mathcal{B}$ queries the same to the corresponding simple m-entwined sum oracle and forwards the answer to $\mathcal{A}$. In the end, $\mathcal{A}$ outputs a solution $\boldsymbol{\beta}, \mathcal{T}_{\mathsf{out}}$ and ω^*, v. $\mathcal{B}$ lets $\beta = \beta^{(0)} + vy$ and sends $\bar{\boldsymbol{\beta}} = (\beta, \beta^{(1)}, \ldots, \beta^{(q_s)})$ and the same $\mathcal{T}_{\mathsf{out}}, \omega^*$ that $\mathcal{A}$ outputted as a solution of the m-entwined sum problem. The simulation against $\mathcal{A}$ is perfect. Therefore, the probability that $\mathcal{B}$ wins the simple m-entwined sum problem is the same as the probability that $\mathcal{A}$ wins the m-entwined sum problem. Since we know that $\mathcal{B}$'s success probability is negligible, the success probability of $\mathcal{A}$ is negligible in the m-entwined sum problem. □

5 Security Proof of DWMS

In the next theorem, we prove that DWMS is a secure multi-signature scheme in the ROM and the AGM.

Theorem 3. *Suppose there exists a PPT* ***algebraic*** *adversary $\mathcal{A}$ in the AGM against DWMS with parameters $(\mathbb{G}, P, p)$ and $m > 1$ who accesses random oracles H, H_1, H_2 at most q_h, q_{h_1}, q_{h_2} times respectively and breaks the security of DWMS in the plain public-key model (Definition 2) with probability ϵ. Then, under the 2-entwined sum assumption, there exists a PPT reduction $\mathcal{R}$ that solves the q_s-OMDL problem with probability $\epsilon_{omdl} \geq \epsilon - \frac{2q_s - 2q_{h_1} - q_{h_2} + q_s q_{h_1}}{p} - \frac{q_{h_2}}{\sqrt{p}} - \frac{q_s}{p^2} - \epsilon_{eSum}$ where $\epsilon_{eSum} \leq \epsilon_{dlr} + \frac{q_h}{p} + \frac{q_{h_1}}{\sqrt{p}}$.*

Before giving the proof, we give the main ideas about how to construct a reduction that solves the OMDL problem given that an adversary outputs a forgery in DWMS. The q_s-OMDL challenger gives $q_s + 1$ challenges. The reduction selects the last challenge as its public key. In each signing query, the reduction sends a random linear combination of the first q_s OMDL challenges as pre-commitments. Since the reduction does not know its secret key and the discrete logarithm of pre-commitments, it obtains the partial signature while simulating round 2 of DWMS from the $\mathsf{DLOracle}$. Thus, the reduction obtains a linear equation with $q_s + 1$ unknowns (discrete logarithm of the OMDL challenges) in each signing session simulation. After at most q_s signing oracle calls, $\mathcal{A}$ outputs a forgery. Since we are in the AGM, $\mathcal{A}$ gives the representation of each group element during the simulation. From these representations, the reduction obtains $q_s + 1^{th}$ linear equation with the $q_s + 1$ unknowns which are the discrete logarithm of the OMDL challenges. If the last linear equation obtained from the

forgery is not linearly dependent on other equations obtained from DLOracle, then the reduction solves the linear system of equations and obtains the discrete logarithm of the OMDL challenges. If it is linearly dependent, it means that the adversary obtained the forgery with a linear combination of the partial signatures generated in each signing session so the forgery does not give any new information and OMDL challenges cannot be found. During the proof, we see that if it is the case, the adversary actually solves the 2-entwined sum problem which can happen with negligible probability. Therefore, the reduction obtains the OMDL solution by solving $q_s + 1$ linear equations as long as the 2-entwined sum problem is hard.

Proof. We will show that given a forger $\mathcal{A}$ on the multi-signature scheme DWMS, there exists a reduction $\mathcal{R}$ that can solve the q_s-OMDL problem under the entwined sum assumption. We will use the following three notations for the adversary in the proof: $\mathcal{A}$ is an abbreviated notation that we will use often in the text description. A more formal and expanded notation is $\mathcal{A}^{H,H_1,H_2,\Sigma_1,\Sigma_2}(X_1, par; \rho)$, which accesses various oracles (all explained later) H, H_1, H_2, Σ_1 and Σ_2 that are simulated by the reduction $\mathcal{R}$ and receives the two inputs, namely X_1 - the honest signer's public key, *par*- the parameters of the multi-signature scheme and ρ - any random coins, provided by the reduction.

The formal security definition for multi-signatures is defined in Definition 2 and we denote it as **Game 0**. The challenger (simulated by the reduction $\mathcal{R}$) publishes the public parameters using the ParamGen algorithm and shares the honest signer's public key generated using the key generation algorithm KeyGen with the adversary. The reduction also provides any random coins ρ used by the adversary. Thereafter, the reduction simulates the random oracles H, H_1, H_2 for the adversary. The reduction also simulates the honest signer for the adversary using two oracles Σ_1 and Σ_2 that execute the two rounds of the Sign algorithm. The adversary can make up to q_s queries to Oracles Σ_1 and Σ_2. The adversary's challenge is to output a tuple $(\mathcal{PK}^*, msg^*, \sigma^*)$, such that the honest signer's public key X_1 is part of the multiset $\mathcal{PK}^*$, and the message msg^* was never queried to Oracle Σ_1 and the signature σ^* verifies for $(\mathcal{PK}^*, msg^*)$ while making fewer than q_h, q_{h_1}, q_{h_2} queries to the random oracles H, H_1, H_2, respectively and q_s queries to oracles Σ_1 and Σ_2.

The oracles that $\mathcal{A}$ can access are in more detail as follows in **Game 0**:

Oracle H, H_1, H_2**:** The reduction simulates a perfect random oracle by responding with a random group element from $\mathbb{Z}_p$ for each previously unseen query.

Oracle Σ_1: The reduction maintains a counter ℓ to track each query made by the adversary. We use the notation superscript $^{(\ell)}$ to distinguish the values specific to ℓ^{th} signature query. The reduction simulates a perfect response to a signature initiation query msg from the adversary by generating randomly chosen witnesses $r_{11}^{(\ell)}, r_{12}^{(\ell)}$. It stores all responses in a list L_{Σ_1}. In more detail, Σ_1 in **Game 0** is as follows:

Oracle Σ_2: The reduction simulates a perfect response to the ℓ^{th} signature query from the adversary by retrieving witnesses $r_{11}^{(\ell)}$ and $r_{12}^{(\ell)}$ from List L_{Σ_1} and using the pre-commitments of other session participants to derive a signature

$\Sigma_1(\mathbf{msg}^{(\ell)})$ in Game 0

1: $\ell := \ell + 1$
2: $r_{11}^{(\ell)}, r_{12}^{(\ell)} \leftarrow_R \mathbb{Z}_p, T_{11}^{(\ell)} = r_{11}^{(\ell)} P, T_{12}^{(\ell)} = r_{12}^{(\ell)} P$
3: $\mathsf{L}_{\Sigma_1} := \mathsf{L}_{\Sigma_1} \cup \{msg, T_{11}^{(\ell)}, T_{12}^{(\ell)}, r_{11}^{(\ell)}, r_{12}^{(\ell)}\}$
4: **return** $(msg^{(\ell)}, T_{11}^{(\ell)}, T_{12}^{(\ell)})$

contribution. It stores all necessary elements related to the session in List L_{Σ_2}. In more detail, Σ_2 in **Game 0** is as described below:

$\Sigma_2(\mathcal{PK}^{(\ell)}, \mathbf{msg}^{(\ell)}, (\mathbf{T_{i1}^{(\ell)}}, \mathbf{T_{i2}^{(\ell)}})_{\mathbf{i}\in[\mathbf{1},\mathbf{n}]})$ in Game 0

if $(msg, T_{11}^{(\ell)}, T_{12}^{(\ell)}, ., .) \notin \mathsf{L}_{\Sigma_1}$ **or** $|\mathcal{PK}^{(\ell)}| \neq n-1$ **then return** 0
retrieve $(r_{11}^{(\ell)}, r_{12}^{(\ell)})$ **from** L_{Σ_1}
$\mathsf{SID} = (\mathcal{PK}^{(\ell)}, msg^{(\ell)}, \{T_{11}^{(\ell)}, T_{12}^{(\ell)}\}, \ldots, \{T_{n1}^{(\ell)}, T_{n2}^{(\ell)}\})$
$\alpha_{i1}^{(\ell)} \leftarrow H_1(\mathsf{SID}, i, 1), \alpha_{i2}^{(\ell)} \leftarrow H_1(\mathsf{SID}, i, 2), \forall i \in [1, n]$
$T_h^{(\ell)} = \alpha_{11}^{(\ell)} T_{11}^{(\ell)} + \alpha_{12}^{(\ell)} T_{12}^{(\ell)}$
$T_i^{(\ell)} = (\alpha_{i1}^{(\ell)} T_{i1}^{(\ell)} + \alpha_{i2}^{(\ell)} T_{i2}^{(\ell)}), \forall i \in [2, n]$
$T^{(\ell)} = T_h^{(\ell)} + \sum_{i=2}^{n} T_i^{(\ell)}$
$\mathcal{PK}^{(\ell)} := \mathcal{PK}^{(\ell)} \cup \{X_1\}$
$X^{(\ell)} \leftarrow \mathsf{KeyAgg}(par, \mathcal{PK})$
$a_1^{(\ell)} = H_2(\mathcal{PK}^{(\ell)}, X_1)$
$c^{(\ell)} \leftarrow H(msg^{(\ell)}, X^{(\ell)}, T^{(\ell)})$
$s_1^{(\ell)} = \alpha_{11}^{(\ell)} r_{11}^{(\ell)} + \alpha_{12}^{(\ell)} r_{12}^{(\ell)} + c^{(\ell)} a_1^{(\ell)} x_1$
$\mathsf{L}_{\Sigma_2} := \mathsf{L}_{\Sigma_2} \cup \{\ell, \mathsf{SID}, (T^{(\ell)}, s_1^{(\ell)}, c^{(\ell)}), r_{11}^{(\ell)}, r_{12}^{(\ell)}, \alpha_{11}^{(\ell)}, \alpha_{12}^{(\ell)}\}$
return $s_1^{(\ell)}$

Remember that $\mathcal{A}$ is an algebraic adversary (See Definition 3). Therefore, whenever it queries to oracles with a group element $Z \in \mathbb{G}$, it also gives the representation of them $\boldsymbol{z} \in \mathbb{Z}_p^{|\boldsymbol{V}|}$ such that $Z = \boldsymbol{z}.\boldsymbol{V}$ in terms of the group elements that it has seen so far which we denote in vector $\boldsymbol{V}$. For the sake of the presentation, we do not specify the representation vector of each group element in the oracle descriptions.

In the end, we define the **Game 0** which is equivalent to the game in Definition 2 as follows:

Game 0
$par = (\mathbb{G}, P, p) \leftarrow \mathsf{ParamGen}(\lambda)$
$(x_1, X_1) \leftarrow \mathsf{KeyGen}(par)$
$\ell := 0$
$(\mathcal{PK}^*, msg^*, \sigma^* = (s^*, c^*)) \leftarrow \mathcal{A}^{H, H_1, H_2, \Sigma_1, \Sigma_2}(X_1, par; \rho)$
return $(X_1 \in \mathcal{PK}^* \wedge msg^* \notin \mathsf{L}_{\Sigma_2} \wedge \mathsf{Verify}(par, \mathcal{PK}^*, msg^*, \sigma^*))$

The adversary is said to succeed in **Game 0** if **Game 0** returns 1 and thus the success probability of the adversary, $\epsilon = \Pr[\textbf{Game 0} \rightarrow 1]$.

In the following, we will show how the reduction $\mathcal{R}$ solves the q_s-OMDL problem given a multi-signature adversary $\mathcal{A}$ against the DWMS scheme. To simplify the presentation of the proof, we define a sequence of security games: **Game 1**, ..., **Game 9**, each game differing *slightly* from the previous one. We relate the probability of the output being 1 in any two consecutive games, and finally show that probability of solving the q_s-OMDL problem is close to the probability of the output being 1 in **Game 9**.

*In **Game 1**,* the reduction $\mathcal{R}$ uses its OMDL challenge $\{Y_1, \ldots, Y_{q_s+1}\}$ to derive the honest signer's public key as well as the pre-commitments in the simulation of Oracle Σ_1. $\mathcal{R}$ sets its public key X_1 as Y_{q_s+1}. The details of **Game 1** is given below. The gray color lines are the same as **Game 0**.

Game 1
$(Y_1, Y_2, \ldots, Y_{q_s}, Y_{q_s+1}) \leftarrow$ OMDL-Game
$X_1 := Y_{q_s+1}$
$\ell := 0$
$(\mathcal{PK}^*, msg^*, \sigma^* = (s^*, c^*)) \leftarrow \mathcal{A}^{H, H_1, H_2, \Sigma_1, \Sigma_2}(X_1, par; \rho)$
return$(\mathsf{pk} \in \mathcal{PK}^* \wedge msg^* \notin \mathsf{L}_{\Sigma_2} \wedge \mathsf{Verify}(\mathcal{PK}^*, msg^*, \sigma^*))$

We also change the way of generating pre-commitments in Σ_1 **Game 1**. Instead of generating fresh witnesses r_{11}, r_{12} to generate the honest signer's pre-commitments in Oracle Σ_1, $\mathcal{R}$ uses the first q_s OMDL challenges as follows: $T_{11} = \sum_{j=1}^{q_s} \eta_{1j} Y_j, T_{12} = \sum_{j=1}^{q_s} \eta_{2j} Y_j$ for randomly chosen $\{\eta_{1j}, \eta_{2j}\}_{j \in [1,q_s]}$ from $\mathbb{Z}_p$. In more detail, the oracle Σ_1 in **Game 1** is modified as below:

$\Sigma_1(\mathbf{msg}^{(\ell)})$ in Game 1

1: $\ell := \ell + 1$
2: $\eta_{ij}^{(\ell)} \leftarrow_R \mathbb{Z}_p, \forall i \in [1,2], \forall j \in [1, q_s]$
3: $T_{11}^{(\ell)} = \sum_{j=1}^{q_s} \eta_{1j}^{(\ell)} Y_j, T_{12}^{(\ell)} = \sum_{j=1}^{q_s} \eta_{2j}^{(\ell)} Y_j$
4: $\mathsf{L}_{\Sigma_1} := \mathsf{L}_{\Sigma_1} \cup \{msg^{(\ell)}, T_{11}^{(\ell)}, T_{12}^{(\ell)}, \{\eta_{1j}^{(\ell)}\}_{j \in [1,q_s]}, \{\eta_{2j}^{(\ell)}\}_{j \in [1,q_s]}\}$
5: **return** $(msg^{(\ell)}, T_{11}^{(\ell)}, T_{12}^{(\ell)})$

$\mathcal{R}$ also modifies the oracle Σ_2 in **Game 1** because the discrete logarithms of the OMDL challenges are not known to the reduction $\mathcal{R}$. So, it uses the available DL oracle from the OMDL problem, to which it can make up to q_s queries, to simulate Oracle Σ_2. The reduction queries to the DL oracle on $T_h^{(\ell)} + c^{(\ell)} a_1^{(\ell)} Y_{q_s+1} = \alpha_{11}^{(\ell)} (\sum_{j=1}^{q_s} \eta_{1j}^{(\ell)} Y_j) + \alpha_{12}^{(\ell)} (\sum_{j=1}^{q_s} \eta_{2j}^{(\ell)} Y_j) + c^{(\ell)} a_1^{(\ell)} Y_{q_s+1}$ to generate the honest signer's signature towards the multi-signature. The details of Σ_2 is below. The different lines from the Σ_2 of the previous game are in color black.

Since all η_{ij}'s are uniformly distributed in $\mathbb{Z}_p$ and Y_i's are randomly chosen elements in $\mathbb{G}$ (received as part of the OMDL challenge), both $T_{11} = \sum_{j=1}^{q_s} \beta_{1j} Y_j$ and $T_{12} = \sum_{j=1}^{q_s} \beta_{2j} Y_j\}$ are uniformly distributed over $\mathbb{G}$ as in Game 1. So, **Game 0** and **Game 1** are identical. Therefore Pr[**Game 0**] = Pr[**Game 1**].

$\Sigma_2(\mathcal{PK}, \mathbf{msg}, (\mathbf{T}^{(\ell)}_{\mathbf{i1}}, \mathbf{T}^{(\ell)}_{\mathbf{i2}})_{\mathbf{i}\in[\mathbf{1},\mathbf{n}]})$ in Game 1

if $(msg, T^{(\ell)}_{11}, T^{(\ell)}_{12}) \notin \mathsf{L}_{\Sigma_1}$ **or** $|\mathcal{PK}^{(\ell)}| \neq n-1$ **then return** 0
retrieve $(\{\eta^{(\ell)}_{1j}\}_{j\in[1,q_s]}, \{\eta^{(\ell)}_{2j}\}_{j\in[1,q_s]})$ **from** L_{Σ_1}
$\mathsf{SID}^{(\ell)} = (\mathcal{PK}^{(\ell)}, msg^{(\ell)}, \{T^{(\ell)}_{11}, T^{(\ell)}_{12}\}, \ldots, \{T^{(\ell)}_{n1}, T^{(\ell)}_{n2}\})$
$\alpha^{(\ell)}_{i1} \leftarrow H_1(\mathsf{SID}, i, 1), \alpha^{(\ell)}_{i2} \leftarrow H_1(\mathsf{SID}, i, 2), \forall i \in [1, n]$
$T^{(\ell)}_h = \alpha^{(\ell)}_{11} T^{(\ell)}_{11} + \alpha^{(\ell)}_{12} T^{(\ell)}_{12}$
$T^{(\ell)}_i = (\alpha^{(\ell)}_{i1} T^{(\ell)}_{i1} + \alpha^{(\ell)}_{i2} T^{(\ell)}_{i2}), \forall i \in [2, n]$
$T^{(\ell)} = T^{(\ell)}_h + \sum_{i=2}^{n} T^{(\ell)}_i$
$\mathcal{PK}^{(\ell)} := \mathcal{PK}^{(\ell)} \cup \{X_1\}$
$X^{(\ell)} \leftarrow \mathsf{KeyGen}(par, \mathcal{PK}^{(\ell)})$
$a^{(\ell)}_1 = H_2(\mathcal{PK}^{(\ell)}, X_1)$
$c^{(\ell)} \leftarrow H(msg^{(\ell)}, X^{(\ell)}, T^{(\ell)})$
$s^{(\ell)}_1 \leftarrow \mathsf{DLOracle}(T^{(\ell)}_h + c^{(\ell)} a^{(\ell)}_1 Y_{q_s+1})$
$\mathsf{L}_{\Sigma_2} := \mathsf{L}_{\Sigma_2} \cup \{\ell, \mathsf{SID}^{(\ell)}, (T^{(\ell)}, s^{(\ell)}_1, c^{(\ell)}), \{\eta^{(\ell)}_{1j}\}_{j\in[1,q_s]}, \{\eta^{(\ell)}_{2j}\}_{j\in[1,q_s]}, \alpha^{(\ell)}_{11}, \alpha^{(\ell)}_{12}\}$
return $s^{(\ell)}_1$

Remember that, given the signing counter equals to ℓ, when $\mathcal{A}$ outputs a group element $B \in \mathbb{G}$ with its representation vector $\boldsymbol{b} = (b_0, b^{(1)}_{11}, b^{(1)}_{12}, \ldots, b^{(\ell)}_{12}, b^{(\ell)}_{12}, b_{q_s+1}) \in \mathbb{Z}^{2\ell+2}_p$ in terms of the group elements $\boldsymbol{V} = (P, T^{(1)}_{11}, T^{(1)}_{12}, \ldots, T^{(\ell)}_{11}, T^{(\ell)}_{12}, Y_{q_s+1})$ that it has seen so far. In the rest of the proof, we consider another representation of $Z \in \mathbb{G}$ in terms of vector $\boldsymbol{Y} = (P, Y_1, Y_2, \ldots, Y_{q_s+1})$ whenever $\mathcal{A}$ gives a representation of Z. Since all group elements that $\mathcal{A}$ sees are the the linear combinations of $Y_1, Y_2, \ldots, Y_{q_s}$, the another representation of Z in terms $\boldsymbol{Y}$ is as follows:

$$
\begin{aligned}
B &= b_0 P + b^{(1)}_{11} T^{(1)}_{11} + b^{(1)}_{12} T^{(1)}_{12} + \ldots + b^{(\ell)}_{11} T^{(\ell)}_{11} + b^{(\ell)}_{12} T^{(\ell)}_{12} + b_{q_s+1} Y_{q_s+1} \\
&= b_0 P + b^{(1)}_{11} \sum_{j=1}^{q_s} \eta^{(1)}_{1j} Y_j + b^{(1)}_{12} \sum_{j=1}^{q_s} \eta^{(1)}_{2j} Y_j + \ldots + b^{(\ell)}_{11} \sum_{j=1}^{q_s} \eta^{(\ell)}_{1j} Y_j + b^{(\ell)}_{12} \sum_{j=1}^{q_s} \eta^{(\ell)}_{2j} Y_j + b_{q_s+1} Y_{q_s+1} \\
&= b_0 P + \sum_{j=1}^{q_s} \underbrace{(\sum_{i=1}^{\ell} b^{(i)}_{11} \eta^{(i)}_{1j} + b^{(i)}_{12} \eta^{(i)}_{2j})}_{b_j} Y_j + b_{q_s+1} Y_{q_s+1} = b_0 P + \sum_{i=1}^{q_s+1} b_j Y_j \qquad (14)
\end{aligned}
$$

In ***Game 2,*** $\mathcal{R}$ works as in the previous game except that it outputs abort if H_1 ever outputs 0. Since H_1 is a random oracle, the probability of aborting in **Game 2** is $\frac{q_{h_1}}{p}$. From the difference lemma, $\Pr[\text{Output}(\textbf{Game 2}) = 1] \geq \Pr[\text{Output}(\textbf{Game 1}) = 1] - \frac{q_{h_1}}{p}$.

In ***Game 3,*** $\mathcal{R}$ works as in the previous game except that it outputs abort if $\alpha^{(i)}_{11} T^{(i)}_{11} + \alpha^{(i)}_{11} T^{(i)}_{11} = 0$ for a session i. Remark that $\alpha^{(i)}_{11} T^{(i)}_{11} + \alpha^{(i)}_{12} T^{(i)}_{12} = (\alpha^{(i)}_{11} (\sum_{j=1}^{q_s} \eta^{(i)}_{1j} y_j) + \alpha^{(i)}_{12} (\sum_{j=1}^{q_s} \eta^{(i)}_{2j} y_j)) P = \tau^{(i)} P$. Therefore, if

$\alpha_{11}^{(i)}T_{11}^{(i)} + \alpha_{12}^{(i)}T_{12}^{(i)} = 0$ for a session i, it means that either $T_{11}^{(i)} = T_{12}^{(i)} = 0$ or $\alpha_{11}^{(i)}(\sum_{j=1}^{q_s}\eta_{1j}^{(i)}y_j) = -\alpha_{12}^{(i)}(\sum_{j=1}^{q_s}\eta_{2j}^{(i)}y_j)$ given that $T_{11}^{(i)} \neq 0, T_{12}^{(i)} \neq 0$. Therefore, the probability of this event for a session i is $(1-(1-\frac{1}{p^2})^{q_s}) + (1-(1-\frac{q_{h_1}}{p})^{q_s}) \leq (1-(1-\frac{q_s}{p^2})) + (1-(1-\frac{q_{h_1}q_s}{p})) = \frac{q_s}{p^2} + \frac{q_{h_1}q_s}{p}$[2]. From the difference lemma, $\Pr[\text{Output}(\textbf{Game 3}) = 1] \geq \Pr[\text{Output}(\textbf{Game 2}) = 1] - \frac{q_s q_{h_1}}{p} - \frac{q_s}{p^2}$.

*In **Game 4**,* $\mathcal{R}$ works as in the previous game except that it also generates a matrix $\mathtt{M}$ of size $\ell \leq q_s$ after receiving the forgery and it aborts the game if the rank of $\mathtt{M}$ is not ℓ. In more details, $\mathcal{R}$ obtains $\{\eta_{1j}^{(i)}\}_{j\in[1,q_s]}, \{\eta_{2j}^{(i)}\}_{j\in[1,q_s]}, \alpha_{11}^{(i)}, \alpha_{12}^{(i)}, a_1^{(i)}, c^{(i)}$ from $\mathsf{L}_{\Sigma_2}[i]$ for all $i \in [1,\ell]$. Then, $\mathcal{R}$ constructs a matrix $\mathtt{M}$ of size $(\ell \times q_s + 1)$ as below:

$$\mathtt{M} = \begin{bmatrix} \alpha_{11}^{(1)}\eta_{11}^{(1)} + \alpha_{12}^{(1)}\eta_{21}^{(1)} & \dots & \alpha_{11}^{(1)}\eta_{1q_s}^{(1)} + \alpha_{12}^{(1)}\eta_{2q_s}^{(1)} & a_1^{(1)}c^{(1)} \\ \alpha_{11}^{(2)}\eta_{11}^{(2)} + \alpha_{12}^{(2)}\eta_{21}^{(2)} & \dots & \alpha_{11}^{(2)}\eta_{1q_s}^{(2)} + \alpha_{12}^{(2)}\eta_{2q_s}^{(2)} & a_1^{(2)}c^{(2)} \\ \vdots & \dots & \vdots & \vdots \\ \alpha_{11}^{(\ell)}\eta_{11}^{(\ell)} + \alpha_{12}^{(\ell)}\eta_{21}^{(\ell)} & \dots & \alpha_{11}^{(q_s)}\eta_{1q_s}^{(\ell)} + \alpha_{12}^{(\ell)}\eta_{2q_s}^{(\ell)} & a_1^{(\ell)}c^{(\ell)} \end{bmatrix}$$

After the construction of $\mathtt{M}$, if the rank of $\mathtt{M}$ is not ℓ, $\mathcal{R}$ aborts. Clearly, the only difference of **Game 4** from **Game 3** is aborting when the rank of $\mathtt{M}$ is not ℓ. Therefore, we analyse the probability of abort in **Game 4**.

We assume next that the q_s-OMDL challenger never selects $y_i = 0$. If there was $y_i = 0$, $\mathcal{R}$ wins the q_s-OMDL game without the forgery by $\mathcal{A}$ i.e., $\mathcal{R}$ knows the discrete logarithm of $Y_i = 0$ and receives discrete logarithm of rest of the challenges from the DL- oracle.

We first consider the case where $\ell < q_s$. We remark that given that $y_1 \neq 0$, $T_{11}^{(i)}$ and $T_{12}^{(i)}$ are uniformly random even if we fix $\eta_{1j}^{(i)}, \eta_{2j}^{(i)}$ for $j > 1$ (i.e., fix all $\eta_{1j}^{(i)}, \eta_{2j}^{(i)}$'s except $\eta_{11}^{(i)}, \eta_{21}^{(i)}$) and all $y_1, y_2, \dots, y_\ell$. values. Therefore, conditioned on $T_{11}^{(i)}, T_{12}^{(i)}$ and $y_1 \neq 0$, all $\eta_{1j}^{(i)}, \eta_{2j}^{(i)}$-values for $\mathbf{j > 1}$ are independent and uniformly distributed i.e.,

$$\Pr\left[\eta_{1j}^{(i)}, \eta_{2j}^{(i)}, \forall j > 1 | T_{11}^{(i)}, T_{12}^{(i)}\right] = \frac{\Pr\left[T_{11}^{(i)}, T_{12}^{(i)} | \eta_{1j}^{(i)}, \eta_{2j}^{(i)}, \forall j > 1\right] \Pr\left[\eta_{1j}^{(i)}, \eta_{2j}^{(i)}, \forall j > 1\right]}{\Pr\left[T_{11}^{(i)}, T_{12}^{(i)}\right]}$$
$$= \frac{\frac{1}{p^2}\frac{1}{p^{2q_s-2}}}{\frac{1}{p^2}} = \frac{1}{p^{2q_s-2}}$$

We remark that we can say only $\eta_{1j}^{(i)}, \eta_{2j}^{(i)}$ values for $\mathbf{j > 1}$ are independent and uniformly distributed because $\Pr\left[\eta_{1j}^{(i)}, \eta_{2j}^{(i)}, \forall j \geq 1 | T_{11}^{(i)}, T_{12}^{(i)}\right] = \frac{\frac{1}{p^{q_s}}}{\frac{1}{p}} = \frac{1}{p^{2q_s-2}} \neq \frac{1}{p^{2q_s}}$. Therefore, all column vectors from 2^{nd} column vector to the q_s^{th} column

[2] Remark that $\alpha_{11}^{(i)} = H_1(\mathsf{SID}^{(i)}, 1, 1), \alpha_{12}^{(i)} = H_1(\mathsf{SID}^{(i)}, 1, 2)$. So, the probability of having $\alpha_{11}^{(i)}(\sum_{j=1}^{q_s}\eta_{1j}^{(i)}y_j) = -\alpha_{12}^{(i)}(\sum_{j=1}^{q_s}\eta_{2j}^{(i)}y_j)$ given that $T_{11}^{(i)} \neq 0, T_{11}^{(i)} \neq 0$ is not a collision probability.

vector are uniformly random and independent random variables given that α-values are not 0 (See **Game 2**). In other words, we have a random submatrix $\bar{\mathtt{M}}$ of size $(\ell \times q_s - 1)$ which is $\mathtt{M}$ without the first and the last column. It is a known fact that random $\bar{\mathtt{M}}$'s rank is ℓ except with the probability $\frac{\ell}{p}$ where $\ell \leq q_s - 1$ (See Appendix A). Therefore, the rank of $\mathtt{M}$ is $\ell \leq q_s - 1$ except with the probability $\frac{\ell}{p}$ because the rank of a matrix is defined as the maximum number of linearly independent columns (or rows) and the columns of $\bar{\mathtt{M}}$ are also $\mathtt{M}$'s columns. Remember that the rank of $\mathtt{M}$ is at most ℓ. In other words, $\Pr[\textsc{Rank}(\mathtt{M}) < \ell | \ell < q_s] \leq \frac{\ell}{p} \leq \frac{q_s - 1}{p}$.

Now, we assume that $\ell = q_s$. In this case, $\bar{\mathtt{M}}$ without the first and last column of $\mathtt{M}$ consists of independent and random column vectors because of the same reasoning as above. Let's define a vector $\boldsymbol{\tau} = (\tau^{(1)}, \tau^{(2)}, \ldots, \tau^{(q_s)}) \in \mathbb{Z}_p^{q_s}$ where $\tau^{(i)}P = \alpha_{11}^{(i)} T_{11}^{(i)} + \alpha_{11}^{(i)} T_{11}^{(i)} = (\alpha_{11}^{(i)} (\sum_{j=1}^{q_s} \eta_{1j}^{(i)} y_j) + \alpha_{12}^{(i)} (\sum_{j=1}^{q_s} \eta_{2j}^{(i)} y_j))P$. Remember that $\boldsymbol{\tau}$ is a non-zero vector i.e., $\boldsymbol{\tau} \neq \mathbf{0}$ because of **Game 3**.

We can write the first column of $\mathtt{M}$ as a linear combination of $\boldsymbol{\tau}$ and the column vectors of $\bar{\mathtt{M}}$ i.e.,

$$\alpha_{11}^{(i)} \eta_{11}^{(i)} + \alpha_{12}^{(i)} \eta_{21}^{(i)} = \frac{-1}{y_1} \tau^{(i)} - \sum_{j=2}^{q_s} \frac{y_j}{y_1} (\alpha_{11}^{(i)} \eta_{1j}^{(i)} + \alpha_{12}^{(i)} \eta_{2j}^{(i)}) \tag{15}$$

since $\frac{1}{y_1} \neq 0$ and $y_1 \neq 0$. Next, we prove a lemma that implies that if $\boldsymbol{\tau}$ and the column vectors of $\bar{\mathtt{M}}$ are linearly independent, then the first column vector and the column vectors of $\bar{\mathtt{M}}$ (i.e, the first q_s columns of $\mathtt{M}$) are linearly independent.

Lemma 2. *Assume that there exist vectors $\boldsymbol{v_1}, \boldsymbol{v_2}, \boldsymbol{v_3}, \ldots, \boldsymbol{v_{q_s}}$ in $\mathbb{Z}_p^{q_s}$ and another vector $\boldsymbol{v_1'} \in \mathbb{Z}_p^{q_s}$ such that $\boldsymbol{v_1'} = \sum_{j=1}^{q_s} \lambda_j \boldsymbol{v_j}$ where $\lambda_j \in \mathbb{Z}_p \setminus \{0\}$. If $\boldsymbol{v_1}, \boldsymbol{v_2}, \boldsymbol{v_3}, \ldots, \boldsymbol{v_{q_s}}$ are linearly independent, then $\boldsymbol{v_1'}, \boldsymbol{v_2}, \boldsymbol{v_3}, \ldots, \boldsymbol{v_{q_s}}$ are linearly independent,*

Proof. Assume that it is not the case to prove by contradiction i.e., $\boldsymbol{v_1'}, \boldsymbol{v_2}, \boldsymbol{v_3}, \ldots, \boldsymbol{v_{q_s}}$ are not linearly independent while $\boldsymbol{v_1}, \boldsymbol{v_2}, \boldsymbol{v_3}, \ldots, \boldsymbol{v_{q_s}}$ are linearly independent. In this case, there exists a non-zero vector $\boldsymbol{\theta} = (\theta_2, \theta_3, \ldots, \theta_{q_s}) \in \mathbb{Z}_p^{q_s - 1}$ such that $\boldsymbol{v_1'} = \sum_{i=2}^{q_s} \theta_i \boldsymbol{v_j}$. We also know that $\boldsymbol{v_1'} = \sum_{j=1}^{q_s} \lambda_j \boldsymbol{v_j}$. These two imply that $\boldsymbol{v_1} = \sum_{j=2}^{q_s} \frac{\theta_j - \lambda_j}{\lambda_1} \boldsymbol{v_j}$ which is a contradiction with the linear independence of $\boldsymbol{v_1}, \boldsymbol{v_2}, \boldsymbol{v_3}, \ldots, \boldsymbol{v_{q_s}}$. Remark that not all $\frac{\theta_j - \lambda_j}{\lambda_1} = 0$ because $\boldsymbol{v_1} \neq \mathbf{0}$ due to the fact that $\boldsymbol{v_1}, \boldsymbol{v_2}, \boldsymbol{v_3}, \ldots, \boldsymbol{v_{q_s}}$ are linearly independent. □

Lemma 2 shows that if we show $\boldsymbol{\tau}$ and the column vectors of $\bar{\mathtt{M}}$ are linearly independent, then the first column vector and the column vectors of $\bar{\mathtt{M}}$ are linearly independent because of Equation (15). Next, we show another lemma to relate the linear independence of $\boldsymbol{\tau}$ and the column vectors of $\bar{\mathtt{M}}$.

Lemma 3. *Given a fixed vector $\boldsymbol{\tau} \in \mathbb{Z}_p^{q_s}$ where $\boldsymbol{\tau} \neq \mathbf{0}$ and uniformly and independently chosen vectors $\boldsymbol{v}_2, \boldsymbol{v}_3, \ldots, \boldsymbol{v}_{q_s}$ in $\mathbb{Z}_p^{q_s}$, $\boldsymbol{\tau}$ and $\boldsymbol{v}_2, \boldsymbol{v}_3, \ldots, \boldsymbol{v}_{q_s}$ are linearly independent except with the probability $\frac{q_s}{p}$.*

Proof. $\boldsymbol{\tau}$ and $\boldsymbol{v}_2, \boldsymbol{v}_3, \ldots, \boldsymbol{v}_{q_s}$ are linearly dependent if either

1. $\boldsymbol{v}_2, \boldsymbol{v}_3, \ldots, \boldsymbol{v}_{q_s}$ are linearly dependent or
2. $\boldsymbol{\tau}$ and $\boldsymbol{v}_2, \boldsymbol{v}_3, \ldots, \boldsymbol{v}_{q_s}$ are linearly dependent given that $\boldsymbol{v}_2, \boldsymbol{v}_3, \ldots, \boldsymbol{v}_{q_s}$ are linearly independent.

The probability of condition 1 is $1 - \frac{q_s - 1}{p}$ since $\boldsymbol{v}_2, \boldsymbol{v}_3, \ldots, \boldsymbol{v}_{q_s}$ are randomly and independently selected (See Appendix A). Therefore, they span a random vector space $\mathcal{V}$ with the size $p^{q_s - 1}$ except with the probability $\frac{q_s - 1}{p}$.

If condition 2 does not hold, it means that $\boldsymbol{\tau}$ is not in the vector space $\mathcal{V}$. Since $\mathcal{V}$ is a random vector space of size $p^{q_s - 1}$, the probability that a fixed $\boldsymbol{\tau}$ is not in it is $\frac{p^{q_s} - p^{q_s - 1}}{p^{q_s}} = 1 - \frac{1}{p}$. So, condition 2 holds with the probability $\frac{1}{p}$.

Hence, the probability that they are linearly dependent is $\frac{q_s - 1}{p} + \frac{1}{p} = \frac{q_s}{p}$. □

When $\mathcal{A}$ gives a forgery, we set the first q_s columns of $\mathtt{M}$ and we fix $\boldsymbol{\tau}$. Thanks to Lemma 3, fixed $\boldsymbol{\tau}$ and independent and random column vectors of $\bar{\mathtt{M}}$ are linearly independent except with the probability $\frac{q_s}{p}$. By applying Lemma 2 thanks to the Equation (15), we conclude that the first q_s columns of $\mathtt{M}$ (equivalently the first column of $\mathtt{M}$ and the column vectors of $\bar{\mathtt{M}}$) are linearly independent. It means that the rank of $\mathtt{M}$ is q_s except with the probability $\frac{q_s}{p}$. In other words, $\Pr[\textsc{Rank}(M) < \ell | \ell = q_s] \leq \frac{q_s}{p}$.

As a result, the rank of $\mathtt{M}$ is ℓ except with the probability

$$\begin{aligned}\Pr[\textsc{Rank}(M) < \ell] &= \Pr[\textsc{Rank}(M) < \ell | \ell < q_s] + \Pr[\textsc{Rank}(M) < \ell | \ell = q_s] \\ &\leq \frac{q_s - 1}{p} + \frac{q_s}{p}\end{aligned}$$

By the difference lemma, $\Pr[\text{Output}(\textbf{Game 4}) = 1] \geq \Pr[\text{Output}(\textbf{Game 3}) = 1] - \frac{2q_s - 1}{p}$.

In ***Game 5****,* the reduction simulates all the oracles as in the previous game except H_2. $\mathcal{R}$ defines a map map_{key} from $\mathbb{Z}_p \to \mathbb{Z}_p^*$. When $(\mathcal{PK}, X_i) \in \mathbb{Z}_p^{n'} \times \mathbb{Z}_p$ is queried to H_2 and $X_i \in \mathcal{PK}$, $\mathcal{R}$ finds the aggregated public key which is $X \leftarrow \mathsf{KeyAgg}(par, \mathcal{PK})$ i.e, $X = \sum_{X_i \in \mathcal{PK}} H_2(\mathcal{PK}, X_i) X_i$. Then, it checks whether X is mapped to a set in map_{key}. If it is not mapped, it lets $\mathsf{map}_{key}(X) = \mathcal{PK}$. Otherwise, it checks whether $\mathsf{map}_{key}(X) = \mathcal{PK}$. If $\mathsf{map}_{key}(X) \neq \mathcal{PK}$, $\mathcal{R}$ aborts. When $\mathcal{A}$ first time queries with a list of $\mathcal{PK}$ to H_2, it actually commits $\mathcal{PK}$ to X without knowing it. Therefore, the probability that an aggregated public key maps to two list of public key sets $\mathcal{PK}, \mathcal{PK}'$ is $\frac{b}{p} \leq \frac{q_{h_2}}{p}$ where b is the number of elements that is mapped. By the difference lemma, $\Pr[\text{Output}(\textbf{Game 5}) = 1] \geq \Pr[\text{Output}(\textbf{Game 4}) = 1] - \frac{q_{h_2}}{p}$.

In ***Game 6****,* the reduction simulates all the oracles as in the previous game except H_1. $\mathcal{R}$ defines a map map_{commit} from $\mathbb{Z}_p \to \{0,1\}^*$. In this game, we call (SID, i, j) is a legit input for H_1 if $\mathsf{SID} = (\mathcal{PK}, msg, \{T_{11}, T_{12}\}, \ldots, \{T_{n1}, T_{n2}\}) \in \mathbb{G}^n \times \{0,1\}^* \times \mathbb{G}^{2n}$, $(msg, T_{11}, T_{12}) \in \mathsf{list}_{\Sigma_1}$, $1 \leq i \leq n$ and $1 \leq j \leq 2$. In

short, (SID, i, j) is legit if SID is a valid session id for the second round of the signing the message msg by Σ_2. When legit (SID, i, j) is queried to H_1, $\mathcal{R}$ finds the commitment $T = \sum_{i=1}^{n} \alpha_{i1} T_{i1} + \alpha_{i2} T_{i2}$ where $\alpha_{ij} = H_1(\mathsf{SID}, i, j)$. Then, it checks whether T is mapped to a value in map_{commit}. If it is not mapped, it lets $\mathsf{map}_{commit}(T) = \mathsf{SID}$. Otherwise, it checks whether $\mathsf{map}_{commit}(T) = \mathsf{SID}$. If $\mathsf{map}_{commit}(T) \neq \mathsf{SID}$, $\mathcal{R}$ aborts. Similarly to **Game 5**, when $\mathcal{A}$ first time queries with legit SID to H_1, it actually commits SID to the value T without knowing it. Therefore, the probability that T maps to two session ids $\mathsf{SID}, \mathsf{SID}'$ is $\frac{b}{p} \leq \frac{q_{h_1}}{p}$ where b is the number of elements that is mapped. By the difference lemma, $\Pr[\text{Output}(\textbf{Game 6}) = 1] \geq \Pr[\text{Output}(\textbf{Game 5}) = 1] - \frac{q_{h_1}}{p}$

In ***Game 7****,* the reduction simulates all the oracles as in the previous game except H_2. When the adversary queries with $\mathcal{PK} = \{X_1, X_2, \ldots, X_n\}$ and $X_j \in \mathbb{G}$ with the representation vectors of each group element in $\mathcal{PK}$, $\mathcal{R}$ obtains the representation $\boldsymbol{x_i} \in \mathbb{Z}_p^{|\boldsymbol{Y}|}$ of the each group element $X_i \in \mathcal{PK}$ in terms of $\boldsymbol{Y}$. If $Y_{q_s+1}, X_j \in \mathcal{PK}$ (the valid key aggregation input), the reduction simulates H_2 as follows: We assume that $X_1 = Y_{q_s+1}$ without loss of generality and there exists a key database DB_{key} which stores the aggregated key and its representation. $\mathcal{R}$ computes $X \leftarrow \mathsf{KeyAgg}(par, \mathcal{PK})$ and if $\mathsf{DB}_{key}[X]$ is empty, it does the following: $\mathcal{R}$ first obtains the representation of adversarial aggregated key $\tilde{X} = \sum_{i=2}^{n} a_i X_i$ which is $\tilde{z} = \sum_{i=2}^{n} a_i \boldsymbol{x_i}$ in terms of $\boldsymbol{Y}$. Then, it lets $\boldsymbol{z} = \tilde{\boldsymbol{z}} + (0, 0, \ldots, 0, a_1)$ be the representation of the aggregated public key of $\mathcal{PK}$ (i.e., $X = \mathsf{KeyAgg}(par, \mathcal{PK}) = \tilde{X} + a_1 Y_{q_s+1}$) in terms of $\boldsymbol{Y}$ where $(0, 0, \ldots, 0, a_1)$ is a representation $a_1 Y_{q_s+1}$. If $\tilde{z}_{q_s+1} = -a_1$, $\mathcal{R}$ aborts and the simulation ends. If it is not the case, $\mathcal{R}$ stores $\boldsymbol{z}$ to the valid key database $\mathsf{DB}_{key}[X] = \boldsymbol{z}$. We remark that $\tilde{z}_{q_s+1} = \sum_{i=2}^{n} a_i \boldsymbol{x_i}[q_s + 1]$ is random because $\mathcal{R}$ does this check when $\mathcal{A}$ queries first time with the valid key aggregation input $\mathcal{PK}, X_j$ and its representations to H_2. In other words, $\mathcal{R}$ does this check when $\mathcal{A}$ does not know $a_1, a_2, \ldots, a_n$. The later queries of the same input with different representation of keys are not considered in this check. Since a_1 is the random oracle output and $\tilde{z}_{q_s+1}$ is random, the probability that $a_1 = -\tilde{z}_{q_s+1}$ less than or equal to $\frac{1}{\sqrt{p}}$. By the difference lemma, $\Pr[\text{Output}(\textbf{Game 7}) = 1] \geq \Pr[\text{Output}(\textbf{Game 6}) = 1] - \frac{q_{h_2}}{\sqrt{p}}$.

In ***Game 8****,* the reduction simulates all the oracles as in the previous game. Differently, after receiving the forgery, it constructs a new matrix $\mathtt{M}'$ by adding a new row $\boldsymbol{v} = (t_1 + c^* z_1, t_2 + c^* z_2, \ldots, t_{q_s+1} + c^* z_{q_s+1})$ to $\mathtt{M}$ and it aborts the game if the rank of the new matrix $\mathtt{M}'$ is less than $\ell + 1$. Here, $\boldsymbol{t} = (t_0, t_1, \ldots, t_{q_s+1})$ is the representation of $T^* = s^* P - c^* X^*$ where $X^* = \mathsf{KeyAgg}(par, \mathcal{PK}^*)$ is the aggregated public key of the forgery. The vector $\boldsymbol{z} = (z_0, z_1, z_2, \ldots, z_{q_s+1})$ is $\mathsf{DB}_{key}[X^*]$ which is a representation of X^* as defined in **Game 7**.

$\mathcal{R}$ can obtain the representation $\boldsymbol{t}$ by checking the H-oracle queries with the input $(msg^*, X^*, s^* P - c^* P)$ where $\boldsymbol{t}$ must be given. We remark that $\mathcal{A}$ has to query with the input $(msg^*, X^*, s^* P - c^* P)$ to output c^* as a part of the forgery. $\mathsf{DB}_{key}[X^*]$ cannot be null because $\mathcal{A}$ needs the aggregated public key for the forgery. We remark that $s^* P - c^* X^* = \boldsymbol{t} \cdot \boldsymbol{Y}$ and $c^* X^* = c^* (\boldsymbol{z} \cdot \boldsymbol{Y})$

$$\mathtt{M}' = \begin{bmatrix} \alpha_{11}^{(1)}\eta_{11}^{(1)} + \alpha_{12}^{(1)}\eta_{21}^{(1)} & \dots & \alpha_{11}^{(1)}\eta_{1q_s}^{(1)} + \alpha_{12}^{(1)}\eta_{2q_s}^{(1)} & a_1^{(1)}c^{(1)} \\ \vdots & \dots & \vdots & \vdots \\ \alpha_{11}^{(\ell)}\eta_{11}^{(\ell)} + \alpha_{12}^{(\ell)}\eta_{21}^{(\ell)} & \dots & \alpha_{11}^{(\ell)}\eta_{1q_s}^{(\ell)} + \alpha_{12}^{(\ell)}\eta_{2q_s}^{(\ell)} & a_1^{(\ell)}c^{(\ell)} \\ t_1 + c^* z_1 & \dots & t_{q_s} + c^* z_{q_s} & t_{q_s+1} + c^* z_{q_s+1} \end{bmatrix}$$

Let's assume that the rank of $\mathtt{M}'$ is less than $\ell + 1$ to analyse the probability that it happens. We show next that if it happens, $\mathcal{A}$ solves the 2-entwined sum problem (Definition 8) with the challenges $(p, \mathbb{G}, P)$, $\boldsymbol{T_h^{(i)}} = (T_{11}^{(i)}, T_{12}^{(i)})$ for $i \in [1, \ell]$ and $Y = Y_{q_s+1}$, the random oracles $\bar{H} : (\{0,1\}^* \times \mathbb{G}) \times \mathbb{G} \to \mathbb{Z}_p$, $\bar{H}' : (\mathbb{G}^n \times \{0,1\}^* \times \mathbb{G}^{2n}) \times \mathbb{G} \to \mathbb{Z}_p$ and $\bar{H}_1 : (\mathbb{G}^n \times \{0,1\}^* \times \mathbb{G}^{2n}) \times \mathbb{G}^2 \times \mathbb{N} \to \mathbb{Z}_p$ for $n \geq 1$. The random oracles $\bar{H}, \bar{H}', \bar{H}_1$ are defined as follows where each stores their responses in the database $\mathsf{DB}_{\bar{H}}, \mathsf{DB}_{\bar{H}'}$ and $\mathsf{DB}_{\bar{H}_1}$, respectively:

$\underline{\bar{H}(\omega, T)}$:
input: $\omega = (msg, X) \in \{0,1\}^* \times \mathbb{G}$ and $\boldsymbol{t} = (t_0, t_1, t_2, \dots, t_{q_s+1})$ which is the representation of T
if $(\omega, W) \notin \mathsf{DB}_{\bar{H}}$:
  if $\mathsf{DB}_{key}[X] \neq \mathsf{null}$:
    $\boldsymbol{z} \leftarrow \mathsf{DB}_{key}[X]$
    $\rho_1 \leftarrow z_{q_s+1}$
    $\rho_2 \leftarrow t_{q_s+1}$
    $\mathsf{DB}_{\bar{H}}[(\omega, T)] \leftarrow \rho_1 H(\omega, T) + \rho_2$
  else:
    $\mathsf{DB}_{\bar{H}}[(\omega, T)] \leftarrow_{\$} \mathbb{Z}_p$
return $\mathsf{DB}_{\bar{H}}[(\omega, T)]$

$\bar{H}$ is a random oracle because H is a random oracle and ρ_1 is not 0 in this game (See **Game 7**).

$\underline{\bar{H}'(\mu, T_h)}$:
input: $\mu = (\mathcal{PK}, msg, \{T_{11}, T_{12}\}, \dots, \{T_{n1}, T_{n2}\}) \in \mathbb{G}^n \times \{0,1\}^* \times \mathbb{G}^{2n}$
if $(\mu, T_h) \notin \mathsf{DB}_{\bar{H}'}$:
  $\alpha_{11} \leftarrow H_1(\mu, 1, 1), \alpha_{12} \leftarrow H_1(\mu, 1, 2)$
  if $T_h = \sum_{j=1}^{2} \alpha_{1j} T_{1j}$:
    $\alpha_{i1}^{(\ell)} \leftarrow H_1(\mu, i, 1), \alpha_{i2}^{(\ell)} \leftarrow H_1(\mu, i, 2), \forall i \in [2, n]$
    $T = T_h + \sum_{i=2}^{n} (\alpha_{i1}^{(\ell)} T_{i1} + \alpha_{i2}^{(\ell)} T_{i2})$
    $X \leftarrow \mathsf{KeyAgg}(par, \mathcal{PK})$
    $a_1 \leftarrow H_2(\mathcal{PK}, X_1)$
    $c = H(msg, X, T)$
    $\mathsf{DB}_{\bar{H}'}[(\mu, T_h)] \leftarrow a_1 c$
  else: $\mathsf{DB}_{\bar{H}'}[(\mu, T_h)] \leftarrow_{\$} \mathbb{Z}_p$
return $\mathsf{DB}_{\bar{H}'}[(\mu, T_h)]$

$\bar{H}'$ is a random oracle because H_1, H_2 and H are random oracles and also there exists no $\mathcal{PK} \neq \mathcal{PK}'$ that aggregates to same X (See **Game 5**) and there exists no $\mu \neq \mu'$ that maps to same T (See **Game 6**).

$\underline{\bar{H}_1(\mu, T, j)}$:
input: $\mu = (\mathcal{PK}, msg, \{T_{11}, T_{12}\}, \ldots, \{T_{n1}, T_{n2}\}) \in \mathbb{G}^n \times \{0,1\}^* \times \mathbb{G}^{2n}$
if $(\mu, T, j) \notin \mathsf{DB}_{\bar{H}_1}$
  if $T = \{T_{11}, T_{12}\}$
    $\mathsf{DB}_{\bar{H}_1}[(\mu, T, j)] \leftarrow H_1(\mu, 1, j)$
  else: $\mathsf{DB}_{\bar{H}_1}[(\mu, T, j)] \leftarrow \mathbb{Z}_p$
else: $\mathsf{DB}_{\bar{H}_1}[(\mu, T, j)] \leftarrow \mathbb{Z}_p$
return $\mathsf{DB}_{\bar{H}_1}[(\mu, T, j)]$

$\bar{H}_1$ is a random oracle because H_1 is a random oracle.

Now, we show why $\mathsf{Rank}(\mathtt{M}') < \ell$ implies that $\mathcal{A}$ finds a 2-entwined sum solution: We remark that the rank of $\mathtt{M}'$ must be ℓ if it is less than $\ell+1$ thanks to **Game 4**. Therefore, if the last row of $\mathtt{M}'$ is linearly dependent, then there exists a **unique** vector $\boldsymbol{\beta} = (\beta^{(1)}, \beta^{(2)}, \ldots, \beta^{(\ell)})$ such that $t_j + c^* z_j = \sum_{i=1}^{\ell} \beta^{(i)} (\alpha_{11}^{(i)} \eta_{1j}^{(i)} + \alpha_{12}^{(i)} \eta_{2j}^{(i)})$ for $j \in [1, q_s]$ and $t_{q_s+1} + c^* z_{q_s+1} = \sum_{i=1}^{\ell} \beta^{(i)} a_1^{(i)} c^{(i)}$. We remark that in this case $s^* = t^* + c^* (\sum_{i=2}^{n} a_i^* x_i + a_1^* y_{q_s+1}) = t_0 + (\sum_{i=1}^{q_s+1} t_i y_i) + c^* z_0 + c^* (\sum_{i=1}^{q_s+1} z_i y_i) = t_0 + c^* z_0 + \sum_{i=1}^{q_s} \beta^{(i)} s^{(i)}$ as in the m-entwined sum attack that we show in Section 3. Accordingly, if we reorganize $T^* = s^* P - c^* X$, we obtain the following:

$$
\begin{aligned}
T^* &= (t_0 + \sum_{j=1}^{q_s+1} y_j t_j) P \\
&= t_0 P + (\sum_{j=1}^{q_s} y_j (\sum_{i=1}^{\ell} \beta^{(i)} (\alpha_{11}^{(i)} \eta_{1j}^{(i)} + \alpha_{12}^{(i)} \eta_{2j}) - c^* z_j)) P + t_{q_s+1} Y_{q_s+1} \\
&= t_0 P + \sum_{i=1}^{\ell} \beta^{(i)} (\alpha_{11}^{(i)} T_{11}^{(i)} + \alpha_{12}^{(i)} T_{12}^{(i)}) - \sum_{j=1}^{q_s} c^* z_j Y_j + t_{q_s+1} Y_{q_s+1} \qquad (16) \\
&= (t_0 - c^* (\sum_{i=2}^{n} a_i x_i - z_0)) P + \sum_{i=1}^{\ell} \beta^{(i)} T_h^{(i)} + (t_{q_s+1} - c^* (a_1^* - z_{q_s+1})) Y_{q_s+1} \qquad (17) \\
&= \beta^{(0)} P + \sum_{i=1}^{\ell} \beta^{(i)} T_h^{(i)} + v Y_{q_s+1} \qquad (18)
\end{aligned}
$$

We obtain from Equation (16) to Equation (17) by using the fact that $\sum_{j=1}^{q_s} c^* z_j Y_j = c^* (X^* - z_{q_s+1} Y_{q_s+1} - z_0 P) = c^* ((\sum_{i=2}^{n} a_i^* x_i) P + a_1^* Y_{q_s+1} -$

$z_{q_s+1}Y_{q_s+1} - z_0 P$). Since the forgery is a valid signature, $c^* = H(msg^*, X^*, T^*)$. c^* satisfies the following because $\text{RANK}(\mathtt{M}') = \ell$ that

$$z_{q_s+1}c^* + t_{q_s+1} = \sum_{i=1}^{\ell} \beta^{(i)} a_1^{(i)} c^{(i)}.$$

These imply that

$$\bar{H}(msg^*, X^*, T^*) = \sum_{i=1}^{\ell} \beta^{(i)} \bar{H}'(SID^{(i)}, T_h^{(i)})$$

Therefore, if the rank of $\mathtt{M}'$ is ℓ, it means that $\mathcal{A}$ generates the forgery by solving 2-entwined sum problem with the solution $\boldsymbol{\beta} = (\beta^{(0)}, \beta^{(1)}, \ldots, \beta^{(\ell)})$, $\mathcal{T}_{\mathsf{out}} = (SID^{(1)}, SID^{(2)}, \ldots, SID^{(\ell)})$, $\omega = (msg^*, X^*)$ and v. We remark that the adversary knows the solution $\boldsymbol{\beta}$ because $\beta^{(0)} = t_0 - c^*(\sum_{i=2}^{n} a_i x_i - z_0)$ and for $i \in [1, \ell]$, $\beta^{(i)} = \frac{t_{11}^{(i)} + z_{11}^{(i)}}{\alpha_{11}^{(i)}} = \frac{t_{12}^{(i)} + z_{12}^{(i)}}{\alpha_{12}^{(i)}}$ (See Equation (14)), $v = t_{q_s+1} - c^*(a_1^* - z_{q_s+1})$ which are generated by the parameters selected by the adversary.

Since the probability of having 2-entwined sum problem solution is ϵ_{eSum}, the probability that the rank of $\mathtt{M}'$ is ℓ is ϵ_{eSum}. Therefore, $\Pr[\textbf{Game 8} = 1] \geq \Pr[\textbf{Game 7} = 1] - \epsilon_{eSum}$.

*In **Game 9**,* $\mathcal{R}$ obtains the OMDL solution by solving a linear system of equations. If $\ell < q_s$, $\mathcal{R}$ can make $q_s - \ell$ more DL-query. So, it queries $Y_{\ell+1}, Y_{\ell+2}, \ldots, Y_{q_s}$ to the DL-oracle and obtain $y_{\ell+1}, y_{\ell+2}, \ldots, y_{q_s}$. Now, it needs to learn the DL of $Y_1, Y_2, \ldots, Y_\ell, Y_{q_s+1}$.

Given that $s^* = t^* + c^*(a_1 y_{q_s+1} + \sum_{i=2}^{n} a_i x_i) = t_0 + \sum_{j=1}^{q_s+1} t_j y_j + c^*(z_0 + \sum_{j=1}^{q_s+1} z_j y_j) = t_0 + c^* z_0 + \sum_{j=1}^{q_s+1}(t_j + c^* z_j) y_j$, $\mathcal{R}$ obtains a linear equation with $\ell + 1$ unknowns $y_1, y_2, \ldots, y_\ell, y_{q_s+1}$ i.e, $\bar{s} = s^* - t_0 - c^* z_0 - \sum_{j=\ell+1, \text{if } \ell<q_s}^{q_s}(t_j + z_j) y_j = \sum_{j=1}^{\ell}(t_j + c^* z_j) y_j + (t_{q_s+1} + c^* z_{q_s+1}) y_{q_s+1}$ Similarly, given that for $u \in [1, \ell]$, $s_1^{(u)} = \sum_{i=1}^{q_s} y_i(\alpha_{11}^{(u)} \eta_{1i}^{(u)} + \alpha_{12}^{(u)} \eta_{2i}^{(u)}) + c^{(u)} a_1^{(u)} y_{q_s+1}$ $\mathcal{R}$ also obtains ℓ-more linear equations such that $\bar{s}_1^{(u)} = s_1^{(u)} - \sum_{j=\ell+1, \text{if } \ell<q_s}^{q_s}(\alpha_{11}^{(u)} \eta_{1j}^{(u)} + \alpha_{12}^{(u)} \eta_{2j}^{(u)}) y_j = \sum_{j=1}^{\ell}(\alpha_{11}^{(u)} \eta_{1j}^{(u)} + \alpha_{12}^{(u)} \eta_{2j}^{(u)}) y_j + c^{(u)} a_1^{(u)} y_{q_s+1}$.

In the end, $\mathcal{R}$ obtains a unique solution $y_1, y_2, \ldots, y_\ell, y_{q_s+1}$ by solving the following linear equation system $\mathtt{M}'\boldsymbol{y} = \bar{\boldsymbol{s}}$ where $\boldsymbol{y} = (y_1, y_2, \ldots, y_\ell, y_{q_s+1})$ and $\bar{\boldsymbol{s}} = (\bar{s}_1^{(1)}, \bar{s}_1^{(2)}, \ldots, \bar{s}_1^{(\ell)}, \bar{s})$

We remark that matrix of the linear system of equations is $\mathtt{M}'$ without columns $\ell + 1, \ell + 2, \ldots, q_s$. Therefore, its rank is $\ell + 1$ which is the reason of the unique solution. Hence, $\Pr[\textbf{Game 9} = 1] = \Pr[\textbf{Game 8} = 1] = \epsilon_{omdl}$.

□

6 Conclusion

In this paper, we introduce our new Schnorr-based two-round multi-signature scheme DWMS. Our protocol is one of the few provably secure protocols among

the existing secure Schnorr-based two-round multi-signature schemes [11,20,21]. Drawing upon the lessons learned from the k-sum attack [11], we proved the security of our scheme with special care. We introduced the m-entwined sum problem that simplifies the security proof of DWMS. We showed that the m-entwined sum problem is hard in the AGM as long as the DLOG problem is hard. We believe that the m-entwined sum problem shows a way to improve and simplify the security proofs which require excluding a specific relationship between the group and the field in the ROM. As future work, it would be interesting to show the hardness of the m-entwined sum problem in the standard model.

Acknowledgement

We thank Raghav Bhaskar and Alistair Stewart for their extensive advise and extremely insightful conversations throughout the effort. We warmly thank Michele Orrù for his helpful conversations, especially around understanding the algebraic group models.

References

1. Schnorrkel library. `https://github.com/w3f/schnorrkel/commit/fa6c35f832`, January 2020.
2. Ali Bagherzandi, Jung-Hee Cheon, and Stanislaw Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 449–458, 2008.
3. Ali Bagherzandi and Stanisław Jarecki. Multisignatures using proofs of secret key possession, as secure as the Diffie-Hellman problem. In *International Conference on Security and Cryptography for Networks*, pages 218–235. Springer, 2008.
4. Mihir Bellare and Wei Dai. The multi-base discrete logarithm problem: Concrete security improvements for Schnorr identification, signatures and multi-signatures. *IACR Cryptol. ePrint Arch.*, 2020:416, 2020.
5. Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3), 2003.
6. Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 390–399, 2006.
7. Mihir Bellare and Adriana Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In *Annual International Cryptology Conference*, pages 162–177. Springer, 2002.
8. Fabrice Benhamouda, Tancrède Lepoint, Michele Orrù, and Mariana Raykova. On the (in)security of ROS. Cryptology ePrint Archive, Report 2020/945, 2020. `https://eprint.iacr.org/2020/945`.
9. Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In *International Workshop on Public Key Cryptography*, pages 31–46. Springer, 2003.

10. Dan Boneh, Manu Drijvers, and Gregory Neven. *Compact multi-signatures for smaller blockchains*, pages 435–464. 2018.
11. Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs. On the security of two-round multi-signatures. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1084–1101. IEEE, 2019.
12. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In *Advances in Cryptology – CRYPTO 2018*, volume 10992 of *Lecture Notes in Computer Science*, pages 33–62. Springer, 2018.
13. Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind Schnorr signatures in the algebraic group model. Cryptology ePrint Archive, Report 2019/877, 2019. `https://eprint.iacr.org/2019/877`.
14. Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind Schnorr signatures and signed elgamal encryption in the algebraic group model. In *International Conference on the Theory and Applications of Cryptographic Techniques*, volume 12105 of *Lecture Notes in Computer Science*. Springer, 2020.
15. Kazuharu Itakura and Katsuhiro Nakamura. A public-key cryptosystem suitable for digital multisignatures. *NEC Research & Development*, (71):1–8, 1983.
16. Chelsea Komlo and Ian Goldberg. FROST: Flexible round-optimized Schnorr threshold signatures. 2020.
17. Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 465–485. Springer, 2006.
18. Changshe Ma, Jian Weng, Yingjiu Li, and Robert Deng. Efficient discrete logarithm based multi-signature scheme in the plain public key model. *Designs, Codes and Cryptography*, 54(2):121–133, 2010.
19. Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple Schnorr multi-signatures with applications to bitcoin. *Designs, Codes and Cryptography*, 87(9):2139–2164, 2019.
20. Jonas Nick, Tim Ruffing, and Yannick Seurin. Musig2: Simple two-round Schnorr multi-signatures. Cryptology ePrint Archive, Report 2020/1261, 2020. `https://eprint.iacr.org/2020/1261`.
21. Jonas Nick, Tim Ruffing, Yannick Seurin, and Pieter Wuille. MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. Cryptology ePrint Archive, Report 2020/1057, 2020. `https://eprint.iacr.org/2020/1057`.
22. Kazuo Ohta and Tatsuaki Okamoto. A digital multisignature scheme based on the Fiat-Shamir scheme. In *International Conference on the Theory and Application of Cryptology*, pages 139–148. Springer, 1991.
23. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind linsignatures. In *In Journal of Cryptology 13*, pages 361–396, 2000.
24. Thomas Ristenpart and Scott Yilek. The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 228–245. Springer, 2007.
25. Claus Peter Schnorr. Security of blind discrete log signatures against interactive attacks. In *ICICS 2001, LNCS 2229*, pages 1–12. Springer-Verlag, 2001.
26. Ewa Syta, Iulia Tamas, Dylan Visher, David Isaac Wolinsky, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ismail Khoffi, and Bryan Ford. Keeping authorities "honest or bust" with decentralized witness cosigning. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 526–545. IEEE, 2016.

27. David A. Wagner. A generalized birthday problem. In *CRYPTO*, 2002.

A Rank of a Random Matrix

Assume that we have a random matrix $\mathtt{M}$ of size $(\ell \times \ell')$. Given that $\ell \leq \ell'$, the rank of $\mathtt{M}$ can be at most ℓ.

Let's define another event E_i where the first i row vectors of $\mathtt{M}$ are linearly independent. In this case, $\Pr[E_1] = 1 - \frac{1}{p^\ell}$ which is the probability that a random vector equals to $\mathbf{0}$ (vector consisting of 0). In this case,

$$\begin{aligned}
\Pr[E_\ell] &= \Pr[E_\ell | E_{\ell-1}] \Pr[E_{\ell-1}] + \underbrace{\Pr[E_\ell | \neg E_{\ell-1}] \Pr[\neg E_{\ell-1}]}_{0} \\
&= \Pr[E_1] \prod_{k=2}^{\ell} \underbrace{\Pr[E_k | E_{k-1}]}_{(1 - \frac{p^{k-1}}{p^\ell})} \\
&= \prod_{k=1}^{\ell} (1 - \frac{p^{k-1}}{p^\ell}) \leq \prod_{k=1}^{\ell} (1 - \frac{p^{k-1}}{p^\ell}) \leq (1 - \frac{1}{p})^\ell \leq 1 - \frac{\ell}{p}
\end{aligned}$$

So, the probability of $\mathtt{M}$'s rank is less than ℓ is at most $\frac{\ell}{p}$.