Key-and-Signature Compact Multi-Signatures: A Compiler with Realizations

Shaoquann Jiang, Dima Alhadidi and Hamid Fazli Khojir

Abstract—Multi-signature is a protocol where a set of signatures jointly sign a message so that the final signature is significantly shorter than concatenating individual signatures together. Recently, it finds applications in blockchain, where several users want to jointly authorize a payment through a multi-signature. However, in this setting, there is no centralized authority and it could suffer from a rogue key attack where the attacker can generate his own keys arbitrarily. Further, to minimize the storage on blockchain, it is desired that the aggregated public-key and the aggregated signature are both as short as possible. In this paper, we find a compiler that converts a kind of identification (ID) scheme (which we call a linear ID) to a multi-signature so that both the aggregated public-key and the aggregated signature have a size independent of the number of signers. Our compiler is provably secure. The advantage of our results is that we reduce a multi-party problem to a weakly secure two-party problem. We realize our compiler with two ID schemes. The first is Schnorr ID. The second is a new lattice-based ID scheme, which via our compiler gives the first regular lattice-based multi-signature scheme with key-and-signature compact without a restart during signing process.

Index Terms—Blockchain, Multi-Signature, Identification, Lattice, Random Oracle

1 Introduction

A multi-signature scheme allows a group of signers to jointly generate a signature while no subset of them can represent all the members to generate it. It was first introduced by Itakura and Nakamura [26]. A trivial method is to ask each signer to generate a signature on the message and concatenate their signatures together. However, this is not efficient: (1) the signature size is linear in the number of signers n; (2) we need to provide n signer public-keys to verifier; (3) the verification needs to verify n signatures; (4) all the n public-keys need to be provided to the verifier; (5) the communication and storage complexity for the signature are both linear in n. With applications in blockchain, these problems are crucial as the signature will be transmitted, verified and stored in the blockchain network. Hence, it is desired to find multi-signature that has a signature with these efficiency measure independent of n.

Early multi-signarture schemes [25], [30], [44] assumed the signer keys are chosen honestly. In Bitcoin [41], every user can choose his own public-key. However, this might raise a very serious issue. For example, if a user wants to generate a multi-signature with users of 3 pubic-keys $g^{x_1}, g^{x_2}, g^{x_3}$, he could choose s randomly and compute his public-key as $pk = g^s(g^{x_1+x_2+x_3})^{-1}$. If the aggregated public-key (which is the only public-key provided to the verifier) is the multiplication of the four public-keys, then attacker knows its secret and hence can forge a multi-signature. This is called *a rogue key attack.* How to construct a key-and-signature compact multi-signature scheme secure against a rogue key attack is an important question.

• *Authors are all with School of Computer Science, University of Windsor, ON, Canada, N9B 3P4.*
E-mail: {jiangshq,Dima.Alhadidi,fazlikh}@uwindsor.ca

Manuscript received Jan 1, 2023; revised Jan 1, 2023.

1.1 Related Works

A multi-signature scheme [26] is a special case of aggregate signature [12] where each signer of the latter can sign a possibly different message. In this work, we only discuss a multi-signature scheme with a motivation of blockchain application where the public-key is arbitrary and the target is to minimize the signature and the aggregated public-key size. Micali et al. [39] requires an interactive key generation among signers and hence is not suitable. Boldyreva [11] and Lu et al. [32] require signers to add *proof of possession* (PoP) to their public-keys, which is typically a signature of the user's public-key. The main disadvantage of this assumption is the increase of the public-key size. In the signing process, it also requires a signer to verify the PoP of all the other signers. In addition, this assumption is not compatible with an ordinary signature where PoP is not required.

Bellare and Neven [8] converted the Schnorr signature [48] into a multi-signature by linearly adding the signature together. Their protocol is of 3-round but without the aggregated key aggregation. Bagherzandi et al. [3], Ma et al. [36], Syta et al. [51] and Maxwell et al. [38] attempted to construct a 2-round multi-signature scheme which essentially tries to remove the preliminary committing message which is a hash of the first message in an ID scheme (see [8] for example). However, Drijvers et al. [17] pointed out that all these schemes have proof flaws. They then proved that a slightly modified scheme of Bagherzandi et al. [3] is secure under the PoP assumption. Other 2-round proposals that support the key-and-signature aggregation are due to Alper and Burdges [2] and Nick et al. [42], [43], where Nick et al. [43] employed a generic NIZK proof while the other two proposals [2], [42] are efficient in aggregated key and signature and verification cost (similar to the original Schnorr signature). Boneh et al. [13] proved the security of a modified version of Maxwell et al. [38] via an added

	Key Compact	Round Comp	♯ Restart	Assump/ Remark
[21]	No	3	exp	R-LWE
[22]	No	3	exp	non-standard
[14]	**Yes**	2	exp	R-MLWE& R-MSIS
[16]	No	2	exp	R-MLWE& R-MSIS
[20]	No	1	**0**	R-SIS limited-sign
ours	**Yes**	3	**0**	R-SIS & R-LWE

Fig. 1. Performance of Lattice-based Secure Multi-Signature Schemes: compact means the size independent of ♯ signers; all schemes have compact signatures; schemes requiring a honest key generations are not included; ♯ restart is ♯ of repeated runs of signing algorithm (in case it aborts); exp means exponential in either ♯ signers or the security parameter; limited-sign restricts each user to have a bounded number of signings.

preliminary committing message and hence it is a 3-round scheme. Bellare and Dai [4] proposed a 2-round multi-signature scheme with a tight reduction without the key aggregation.

The above constructions are all based on variants of the discrete logarithm assumption. It is important to find out quantum-resistant schemes while this is not easy. For instance, lattice-based scheme [28] is insecure [31]. Also, the proof for a ring-SIS based scheme [27] is invalid. They reduced to find a short W for ring-SIS problem $AW = 0$ with public parameter A. However, their obtained W is trivially zero which does not contradict the ring-SIS assumption. Some schemes [14], [21], [22], [37] need an exponential number of restarts of the signing algorithm, due to a noticeable probability of abort event. Some schemes [19], [37] are provably secure only when all the user keys are generated honestly which is not suitable for blockchain. Damgård et al. [16] and Fleischhacker et al. [20] do not support key aggregations while the latter can only allow a signer to sign a predefined number of signatures. Thus, currently no multi-signature scheme can support a key-and-signature aggregation without a restart and allow an unlimited number of signing. Our work is to study this question in details.

1.2 Contribution

In this paper, we consider the key-and-signature compact multi-signature. That is, both key and signature support aggregation and have a size independent of the number of signers. Toward this, we formulate the *linear* identification scheme (ID) and propose a compiler that transforms a linear ID to a key-and-signature compact multi-signature scheme, where the signature size and the aggregated public-key are independent of the number of signers. The advantage of our compiler is that we reduce the multi-party signature problem to a two-party identification problem and hence it is much easier to deal with and also the security proof for latter should be simpler. We formulate the linearity of ID via the $\mathcal{R}$-module from algebra. Our compiler is provably secure. We realize our compiler with two ID schemes. The first is Schnorr ID scheme. The second one is a new ID scheme over ring that is secure under ring-LWE and ring-SIS assumptions. Our ID scheme via the compiler gives the first key-and-signature compact multi-signature without a restart during the signing process (see Fig. 1 for a comparison with other schemes), where a signer can do an unlimited number of signing (unlike [20], which can only do a predetermined number of signings). The security of ID schemes is formulated in terms of unforgeability against an aggregated key of multi-users with at least one of them honest. Our ID schemes are proven secure through a new forking lemma (called nested forking lemma). Our forking algorithm has a nested rewinding and is more effective than the previous algorithms which fork at two or more spots sequentially.

2 Preliminaries

Notations. We will use the following notations.

- $x \leftarrow S$ samples x uniformly random from a set S.
- For a randomized algorithm A, $u = A(x; r)$ denotes the output of A with input x and randomness r, while $u \leftarrow A(x)$ denotes the random output (with unspecified randomness).
- We use $P_R(r)$ to denote the probability $\Pr(R = r)$; for Boolean variable G, $\Pr(G)$ means $\Pr(G = 1)$.
- PPT stands for probabilistic polynomial time.
- Min-entropy $H_\infty(X) = -\log(\max_x \log P_X(x))$.
- $A|B$ stands for A concatenating with B.
- $\mathsf{negl}(\lambda)$ is *negligible*: $\lim_{\lambda\to\infty} poly(\lambda)\mathsf{negl}(\lambda) = 0$ for any polynomial $poly(\lambda)$.
- $[\nu]$ denotes set $\{1, \cdots, \nu\}$.

2.1 Ring and Module

In this section, we review a math concept: module (for details, see [29]). We start with the concept of ring. A **ring** A is a set, associated with multiplication and addition operators, respectively written as a product and a sum, satisfying the following conditions:

- **R-1.** A is a commutative group under addition operator + with identity element **0**.
- **R-2.** A is associative under multiplication operator: for $a, b, c \in A$, (ab)c=a(bc). Also, it has a unit element **1**: **1**a=a.
- **R-3.** It satisfies the distributive law: for $a, b, c \in A$, $a(b + c) = ab + ac$ and $(b + c)a = ba + ca$.

In this paper, we only consider a *commutative ring*: if $a, b \in A$, then $ab = ba$. That is, when we say ring, it always means a commutative ring. Note that a non-zero element in a ring does not necessarily have a (multiplicative) inverse, where b is an inverse of a if $ab = \mathbf{1}$. For instance, in $\mathbb{Z}_{10}$, 3 is an inverse of 7 while 5 does not have an inverse. If A is a commutative ring with $\mathbf{0} \neq \mathbf{1}$ and every non-zero element in A has an inverse, then A is a **field**.

Now we introduce the concept *module*.

Definition 1. Let R be a ring. An Abelian group M (with group operator $\boxplus$) is a R**-module**, if (1) it has defined a multiplication operator $\bullet$ between R and M: for any $r \in R, m \in M$, $r \bullet m \in M$; (2) the following conditions are satisfied: for any $r, s \in R$ and $x, y \in M$,

1. $r \bullet (x \boxplus y) = (r \bullet x) \boxplus (r \bullet y)$;
2. $(r + s) \bullet x = (r \bullet x) \boxplus (s \bullet x)$
3. $(rs) \bullet x = r \bullet (s \bullet x)$
4. $1_R \bullet x = x$, where 1_R is the multiplicative identity of R.

We remark that the group operator $\boxplus$ for M is not necessarily the regular number addition (e.g., it can be the integer multiplication).

In the following, we give some R-module examples.

Example 1. Let q be a prime and M is a group of order q with generator g (i.e., $M = \langle g \rangle$). Examples of M are a subgroup of $\mathbb{Z}_p^*$ or an elliptic curve group. $x, y \in M$, xy denotes its group operation. Then, M is a $\mathbb{Z}_q$-module with $\bullet$ defined as $r \bullet m \stackrel{def}{=} m^r$, for $r \in \mathbb{Z}_q$ and $m \in M$. It is well-defined: since $m^q = 1$, any representative r in $\mathbb{Z}_q$ such as $r, r + q$ gives the same result $r \bullet m$. For $r, s \in \mathbb{Z}_q$ and $x, y \in M$, we check the module conditions: (1) $s \bullet (xy) = (xy)^s = x^s y^s = (s \bullet x)(s \bullet y)$; (2) $(r + s) \bullet m = m^{r+s} = m^r m^s = (r \bullet m)(s \bullet m)$; (3) $(rs) \bullet x = x^{rs} = (x^s)^r = r \bullet (s \bullet x)$; (4) $\mathbf{1} \bullet x = x^1 = x$.

Example 2. For any integer $n > 0$, $M = \mathbb{Z}_n$ (as an additive group) is a $\mathbb{Z}_n$-module, where $\bullet$ is simply the modular multiplication. The verification of module properties is s-traightforward.

Example 3. Let n be a positive integer. Then, the polynomial ring $M = \mathbb{Z}_n[x]$ (as an additive group) is a $\mathbb{Z}_n$-module with $\bullet$ being the modular n multiplication: for $s \in \mathbb{Z}_n, m = \sum_{i=0}^t u_i x^i$, $s \bullet m = \sum_{i=0}^t u_i s x^i$, where $u_i s$ is the multiplication over $\mathbb{Z}_n$. All the other verifications of the properties are straightforward.

3 Nested Forking Lemma

The original forking lemma was formulated by Pointcheval and Stern [46] to analyze Schnorr signature [48]. It basically shows that if the attacker can forge a Schnorr signature in the random oracle model [7] with a non-negligible probability, then it can generate two forgeries when reminding to the place where the random oracle value was revised. Bellare and Neven [8] generalized the forking lemma to a general algorithm A, without resorting to a signature scheme. This was further generalized by Bagherzandi et al. [3] so that A is rewound to many places. However, the algorithm needs $O(n^2 q/\epsilon)$ rewindings, where q is the number of random values in one run of A (which is the number of random oracle queries in typical cryptographic applications) and ϵ is the successful probability of A while n is the number of rewinding spots. However, this is not efficient and can even be essentially exponential for a non-negligible ϵ. The main issue comes from the fact the rewinding for each spot is repeated independently until a new success is achieved. But it does not relate different rewindings. In this section, we give a new forking lemma for two rewinding spots (say at index i, j with $i < j$) while it can be generalized to n rewinding spots. The new feature here is that the rewinding is *nested*. To see this, suppose that the first run of A uses the list of random values: $h_1, \cdots, h_{i-1}, h_i, \cdots, h_{j-1}, h_j, \cdots, h_q$ and the rewinding spots are chosen at index i and j. Then, we execute A for another 3 runs with rewindings that respectively the following lists of random values:

$$h_1, \cdots, h_{i-1}, h_i, \cdots, h_{j-1}, h'_j, \cdots, h'_q; \tag{1}$$
$$h_1, \cdots, h_{i-1}, \bar{h}_i, \cdots, \bar{h}_{j-1}, \bar{h}_j, \cdots, \bar{h}_q; \tag{2}$$
$$h_1, \cdots, h_{i-1}, \bar{h}_i, \cdots, \bar{h}_{j-1}, \underline{h}_j, \cdots, \underline{h}'_q. \tag{3}$$

That is, execution (1) rewinds the initial execution to index j; execution (2) rewinds the initial execution to index i while execution (3) rewinds the (rewound) execution (2) to index j. With these related executions, we are able to claim the outputs are all successful with probability at least $O(\epsilon^4)$, which is still non-negligible. The advantage of this nested forking is that it can be *directly* used to extract a secret hidden in recursive random oracle evaluations. Our algorithm will use the following notations.

$h[\![1, \cdots, q]\!] \stackrel{def}{=} h_1, \cdots, h_q$ (a sequence of elements);
$h[\![1, \cdots, i, \widehat{\cdots, q}]\!] \stackrel{def}{=} h_1, \cdots, h_{i-1}, \hat{h}_i, \cdots, \hat{h}_q$;
$h[\![1, \cdots, i, \widehat{\cdots, j}, \overline{j+1, \cdots, q}]\!]$
$= h_1 \cdots, h_{i-1}, \hat{h}_i, \cdots, \hat{h}_j, \bar{h}_{j+1}, \underline{\cdots, \bar{h}_q}$.

Other variants such as $h[\![1, \cdots, \overline{i, \cdots, j}, \underline{j+1, \cdots, q}]\!]$) can be defined similarly. Our forking algorithm is in Fig. 2.

Algorithm $F_{\mathsf{A}}(x)$

---

pick coin ρ for A at random
$h_1, \cdots, h_q \leftarrow H$
$(I_0, J_0, \sigma_0) \leftarrow \mathsf{A}(x,\ h[\![1, \cdots, q]\!];\ \rho)$
If $I_0 = 0$ or $J_0 = 0$ or $I_0 \geq J_0$, return Fail
$\hat{h}_{J_0}, \cdots, \hat{h}_q \leftarrow H$
$(I_1, J_1, \sigma_1) \leftarrow \mathsf{A}(x,\ h[\![1, \cdots, J_0, \widehat{\cdots, q}]\!];\ \rho)$
If $I_1 = 0$ or $J_1 = 0$, return Fail
$\bar{h}_{I_0}, \cdots, \bar{h}_q \leftarrow H$
$(I_2, J_2, \sigma_2) \leftarrow \mathsf{A}(x,\ h[\![1, \cdots, \overline{I_0, \cdots, q}]\!];\ \rho)$
If $I_2 = 0$ or $J_2 = 0$, return Fail
$\underline{h}_{J_0}, \cdots, \underline{h}_q \leftarrow H$
$(I_3, J_3, \sigma_3) \leftarrow \mathsf{A}(x, h[\![1, \cdots, \overline{I_0, \cdots, J_0 - 1}, \underline{J_0, \cdots, q}]\!]; \rho)$
If $I_3 = 0$ or $J_3 = 0$, return Fail
Let $\mathsf{Flag}_1 = (I_0 = I_1 = I_2 = I_3) \wedge (J_0 = J_1 = J_2 = J_3)$
Let $\mathsf{Flag}_2 = (h_{I_0} \neq \bar{h}_{I_0}) \wedge (h_{J_0} \neq \hat{h}_{J_0}) \wedge (\bar{h}_{J_0} \neq \underline{h}_{J_0})$
If $\mathsf{Flag}_1 \wedge \mathsf{Flag}_2$, return $(I_0, J_0, \{\sigma_i\}_{i=0}^3)$
else return Fail.

Fig. 2. Forking Algorithm F_{A}

Before introducing our lemma, we give two facts.

Fact 1. *For any random variable I, R and any function $F()$ on I, R, we have*

$$\Pr(I = i \wedge F(I, R) = f) = \Pr(I = i \wedge F(i, R) = f).$$

Proof. For any function G and any random variable W, $\Pr(G(W) = g) = \sum_{w: G(w) = g} P_W(w)$. Applying this to $W = (I, R)$ and $G = (I, F)$, a simple calculation gives the result as $(I, F) = (i, f)$ is $I = i \wedge F = f$. □

Fact 2. *Let B', B, R be independent random variables with B', B identically distributed. Let G be a fixed boolean function. Then,*

$$\Pr(G(R, B) \wedge G(R, B')) = \sum_r P_R(r) \cdot \Pr^2(G(r, B)).$$

Proof. Notice $\Pr(X = x) = \sum_r \Pr(R = r, X = x)$ for variable R, X. Together with Fact 1, we have

$$\begin{aligned}
&\Pr(G(R,B) \wedge G(R,B')) \\
=&\sum_r \Pr(R=r, \{G(R,B) \wedge G(R,B')\} = 1) \\
=&\sum_r \Pr(R=r, \{G(r,B) \wedge G(r,B')\} = 1) \\
=&\sum_r P_R(r) \cdot \Pr(G(r,B)) \cdot \Pr(G(r,B')) \\
=&\sum_r P_R(r) \cdot \Pr^2(G(r,B)),
\end{aligned}$$

where the third equality uses the independence of R, B, B' and the last equality uses the fact that B' and B are identically distributed. □

Now we are ready to present our forking lemma.

Lemma 1. Let $q \geq 2$ be a fixed integer and H be a set of size $N \geq 2$. Let A be a randomized algorithm that on input $x, h_1, \cdots, h_q$ returns a triple, the first two elements of which are integers from $\{0, 1, \cdots, q\}$ and the last element of which is a side output. Let IG be a randomized algorithm (called input generator). The accepting probability of A, denoted by acc, is defined as the probability that $I, J \geq 1$ in the experiment

$$x \leftarrow \mathsf{IG};\ h_1, \cdots, h_q \leftarrow H;$$
$$(I, J, \sigma) \leftarrow \mathsf{A}(x, h[\![1, \cdots, q]\!]).$$

The forking algorithm F_{A} associated with A is a randomized algorithm that takes x as input and proceeds as in Fig. 2. Let $frk = \Pr[F_{\mathsf{A}}(x) \neq \mathsf{Fail} : x \leftarrow \mathsf{IG}]$. Then,

$$frk \geq \frac{8 \cdot acc^4}{q^3(q-1)^3} - \frac{3}{N}. \tag{4}$$

Proof. With respect to Flag_1, we define Flag_1^* as event

$$(I_0 = \cdots = I_3 \geq 1) \wedge (J_0 = \cdots = J_3 \geq 1) \wedge (J_0 > I_0).$$

Then, it is easy to check that $F_A(x) \neq \mathsf{Fail}$ is equivalent to $\mathsf{Flag}_1^* \wedge \mathsf{Flag}_2 = 1$. Since $h_{I_0} = \bar{h}_{I_0}$ (resp. $h_{J_0} = \hat{h}_{J_0}$, or, $\bar{h}_{J_0} = \underline{h}_{J_0}$) in $\neg\mathsf{Flag}_2$ holds with probability $1/N$. It follows that

$$\begin{aligned}
frk =& \Pr(\mathsf{Flag}_1^* \wedge \mathsf{Flag}_2 = 1) \\
\geq& \Pr(\mathsf{Flag}_1^* = 1) - 3/N.
\end{aligned} \tag{5}$$

Notice that

$$\begin{aligned}
&\Pr(\mathsf{Flag}_1^* = 1) \\
=&\sum_{i=1}^{q} \sum_{j=i+1}^{q} \Pr(\wedge_{b=0}^{3}\{I_b = i\ \wedge\ J_b = j\}).
\end{aligned} \tag{6}$$

Let A_1 (resp. A_2, A_{12}) be three variants of algorithm A with the only difference in the output which is the first element (resp. the second element, the first two elements) of A's output. For instance,

$$J_1 = \mathsf{A}_2(x, h[\![1, \cdots, J_0 - 1, \widehat{J_0, \cdots, q}]\!]; \rho), \tag{7}$$
$$I_2 = \mathsf{A}_1(x, h[\![1, \cdots, I_0 - 1, \overline{I_0, \cdots, q}]\!]; \rho). \tag{8}$$

Assigning $I_0 = i$ and $J_0 = j$, we denote

$$J_1' = \mathsf{A}_2(x, h[\![1, \cdots, j-1, \widehat{j, \cdots, q}]\!]; \rho), \tag{9}$$
$$I_2' = \mathsf{A}_1(x, h[\![1, \cdots, i-1, \overline{i, \cdots, q}]\!]; \rho). \tag{10}$$

We can similarly define I_1', J_2', I_3', J_3'. So I_b, J_b for $b \geq 1$ are functions (of A's inputs and randomness) and when assigning $I_0 = i$ and $J_0 = j$, they become I_b', J_b'. Hence, we can apply fact 1 to evaluate Eq. (6). Then, assigning $I_0 = i$ and $J_0 = j$, applying Fact 1 to realize $I_0 = i$ and $J_0 = j$ in A_1 (for I_b) and A_2 (for J_b), I_b and J_b respectively become I_b' and J_b'. Hence, we have

$$\Pr(\mathsf{Flag}_1^* = 1) \tag{11}$$
$$= \sum_{i=1}^{q} \sum_{j=i+1}^{q} \Pr(\wedge_{b=0}^{3}\{I_b' = i\ \wedge\ J_b' = j\}), \tag{12}$$

where I_0, J_0 is rewritten as I_0', J_0' for brevity (so the term $\{I_0 = i\ \wedge\ J_0 = j\}$ becomes $\{I_0' = i\ \wedge\ J_0' = j\}$). Notice $\wedge_{b=0}^{1}(I_b' = i \wedge J_b' = j)$ is a random variable, with randomness $R = (x, \rho, h_1, \cdots, h_{i-1})$ and $B = (h_i, \cdots, h_q, \hat{h}_j, \cdots, \hat{h}_q)$. So we can define

$$\wedge_{b=0}^{1}(I_b' = i \wedge J_b' = j) = G(R, B) \tag{13}$$

for some boolean function G.

Besides, by verifying the definition of I_b', J_b', we can see that

$$\wedge_{b=2}^{3}(I_b' = i \wedge J_b' = j) = G(R, B') \tag{14}$$

with $B' = (\bar{h}_i, \cdots, \bar{h}_q, \underline{h}_j, \cdots, \underline{h}_q)$.

Hence, applying Fact 2 to Eq. (12), we have

$$\begin{aligned}
&\Pr(\mathsf{Flag}_1^* = 1) \\
=& \sum_{\substack{r \\ 1 \leq i < j \leq q}} P_R(r) \Pr^2(\wedge_{b=0}^{1}(I_{br}' = i\ \wedge\ J_{br}' = j)) \\
=& \sum_{\substack{r \\ 1 \leq i < j \leq q}} P_R(r) \Pr^2(\wedge_{b=0}^{1}(I_{br}', J_{br}') = (i, j)).
\end{aligned} \tag{15}$$

where I_{br}' (resp. J_{br}') is I_b' (resp. J_b') with $R = r$.

Notice that $(I_{0r}', J_{0r}') = (i, j)$ is a boolean random variable (i.e., the result is true only if the equality holds), determined by $h_i, \cdots, h_q$. We can define

$$G'(S, C) \stackrel{def}{=} \{(I_{0r}', J_{0r}') = (i, j)\} \tag{16}$$

for some function G', where $S = h_i, \cdots, h_{j-1}$ and $C = h_j, \cdots, h_q$.

Checking the definition of (I_{1r}', J_{1r}'), we can see

$$\{(I_{1r}', J_{1r}') = (i, j)\} = G'(S, C') \tag{17}$$

with $C' = \hat{h}_j, \cdots, \hat{h}_q$.

Thus, Eq. (15) is

$$\begin{aligned}
&\Pr(\mathsf{Flag}_1^* = 1) \\
=& \sum_r P_R(r) \Pr^2\Big(G'(S, C) \wedge G'(S, C')\Big).
\end{aligned} \tag{18}$$

Hence, we can apply Fact 2 to Eq. (18) and obtain

$$\begin{aligned}
&\Pr(\mathsf{Flag}_1^*=1)\\
=&\sum_{1\le i<j\le q}\sum_r P_R(r)[\sum_s P_S(s)\Pr^2((I'_{0rs},J'_{0rs})=(i,j))]^2\\
\ge&\sum_{1\le i<j\le q}[\sum_{r,s} P_R(r)P_S(s)\Pr^2((I'_{0rs},J'_{0rs})=(i,j))]^2\\
\ge&\sum_{1\le i<j\le q}[\sum_{r,s} P_R(r)P_S(s)\Pr((I'_{0rs},J'_{0rs})=(i,j))]^4\\
=&\sum_{1\le i<j\le q}[\Pr((I'_0,J'_0)=(i,j))]^4,\\
\ge&\left[\sum_{1\le i<j\le q}\Pr((I'_0,J'_0)=(i,j))\right]^4/(q^3(q-1)^3/2^3)
\end{aligned}$$

where (I'_{0rs},J'_{0rs}) is (I'_{0r},J'_{0r}) with $S=s$, the first two inequalities follow from Cauchy-Schwarz inequality[1] (the first one is over distribution $P_R(\cdot)$ and the second one is over distribution $P_R(\cdot)P_S(\cdot)$); the last inequality is to apply Cauchy-Schwarz inequality $\sum_{i=1}^n x_i^2 \ge (\sum_i x_i)^2/n$ twice by noticing that $y_i^4=(y_i^2)^2$ so that the first time we use $x_i=y_i^2$ for Cauchy-Schwarz inequality. Finally, notice that $I'_0=I_0$ and $J'_0=J_0$ by definition. Also, $\sum_{1\le i<j\le q}\Pr((I_0,J_0)=(i,j))$ is exactly acc by definition. It follows that $\Pr(\mathsf{Flag}_1^*=1)\ge \frac{acc^4}{q^3(q-1)^3/2^3}$. From Eq. (5), we have $frk \ge \frac{8\cdot acc^4}{q^3(q-1)^3}-3/N$. □

4 Model of Multi-Signature

In this section, we introduce the model of multi-signature. It consists of the multi-signature definition and the security formalization.

4.1 Syntax

Mult-signature is a signature with a group of signers, where each of them has a public-key and a private key. They jointly generate a signature. The interaction between them proceeds in rounds. Signers are pair-wise connected but the channel is not secure. The signing protocol is to generate a signature so that the successful verification would indicate that all signers have agreed to sign the message. The target is to generate a compact signature that is shorter than concatenating all signers' individual signatures together.

Definition 2. A **multi-signature** is a tuple of algorithms (**Setup, KeyGen, Sign, Verify**), described as follows.

Setup. Given security parameter λ, it generates a system parameter param that serves as part of the input for **KeyGen, Sign, Verify** (but for brevity, we omit it).

KeyGen. It takes param as input and outputs for a user a private key sk and a public-key pk.

Sign. Assume n users with public-keys $(pk_1,\cdots,pk_n)$ want to jointly sign a message M. Then, each user i takes its private key sk_i as input and interacts with other signers. Finally, each of them outputs a signature σ (note: this is for simplicity only; in literature, usually a designated leader outputs σ). Besides, there is a function F that aggregates $(pk_1,\cdots,pk_n)$ into a compact public-key $\overline{pk}=F(pk_1,\cdots,pk_n)$.

Verify. Upon (σ,M) with the aggregated public-key $\overline{pk}=F(pk_1,\cdots,pk_n)$, verifier takes σ, M and $\overline{pk}$ as input, outputs 1 (for accept) or 0 (for reject).

Remark. The verify algorithm *only* uses the aggregated key $\overline{pk}$ to verify the signature. This is important for blockchain, where the recipient only uses $\overline{pk}$ as the public-key. Also, the redeem signature only uses the multi-signature σ. It is desired that both $\overline{pk}$ and σ are independent of n while no attacker can forge a valid signature w.r.t. this short $\overline{pk}$. Even though, our definition generally does not make any restriction on $\overline{pk}$ and it especially can be $(pk_1,\cdots,pk_n)$.

4.2 Security Model

In this section, we introduce the security model [13] of a multi-signature. It formulates the existential unforgeability. Essentially, it says that no attacker can forge a valid signature on a new message as long as the signing group contains an honest member. Toward this, the attacker can access to a signing oracle and create fake public-keys at will. The security is defined through a game between a challenger CHAL and an attacker $\mathcal{A}$.

Initially, CHAL runs $\mathbf{Setup}(1^\lambda)$ to generate system parameter param and executes $\mathbf{KeyGen}$ to generate a public-key pk^* and a private key sk^*. It then provides $pk^*|\mathsf{param}$ to $\mathcal{A}$ who interacts with CHAL through signing oracle below.

Sign $\mathcal{O}_s(PK,M)$. Here PK is a set of *distinct* public-keys with $pk^*\in PK$. Upon this query, CHAL represents pk^* and $\mathcal{A}$ represents $PK-\{pk^*\}$ to run the signing protocol on message M. Finally, $\mathcal{O}_s$ outputs the multi-signature σ (if it succeeds) or $\perp$ (if it fails).

Forgery. Finally, $\mathcal{A}$ outputs a signature σ^* for a message M^*, w.r.t. a set of *distinct* public-keys $(pk_1^*,\cdots,pk_N^*)$ s.t. $pk^*=pk_i^*$ for some i. $\mathcal{A}$ succeeds if two conditions are met: (a) $\mathsf{Verify}(\overline{pk^*},\sigma^*,M^*)=1$ (where $\overline{pk^*}=F(pk_1^*,\cdots,pk_N^*)$); (b) no query $((pk_1^*,\cdots,pk_N^*),M^*)$ was issued to $\mathcal{O}_s$. Denote a success forgery event by **succ**.

Now we can define the security of a multi-signature.

Definition 3. A multi-signature scheme $(\mathbf{Setup},\mathbf{KeyGen},\mathbf{Sign},\mathbf{Verify})$ is existentially unforgeable against chosen message attack (or EU-CMA for short), if satisfies the correctness and existential unforgeability below.

- **Correctness.** For $(sk_1,pk_1),\cdots,(sk_n,pk_n)$ generated by KeyGen, the signature generated by signing algorithm on a message M will pass the verification, except for a negligible probability.
- **Existential Unforgeability.** For any PPT adversary $\mathcal{A}$, $\Pr(\mathbf{succ}(\mathcal{A}))$ is negligible.

The multi-signature scheme is said t-**EU-CMA**, if it is **EU-CMA** w.r.t. adversary $\mathcal{A}$ who always restricts the number of signers in each signing query and the final forgery to be at most t.

1. $\sum_i p_i x_i^2 \ge (\sum_i p_i x_i)^2$, if $p_i\ge 0$ and $\sum_i p_i = 1$

5 Model of Canonical Linear Identification

In this section, we introduce a variant model of canonical identification (ID) scheme and extend it with linearity. We label the ID scheme with a parameter τ. This is needed in order to include our lattice-based ID scheme as a realization for our multi-signature compiler.

Definition 4. A canonical identification scheme with parameter $\tau \in \mathbb{N}$ is a tuple of algorithms $\mathcal{ID} = (\mathbf{Setup}, \mathbf{KeyGen}, P, V_\tau, \Theta)$, where **Setup** takes security parameter λ as input and generates a system parameter param; **KeyGen** is a key generation algorithm that takes param as input and outputs a public key pk and a private key sk; P is an algorithm, executed by *prover*; V_τ is a verification algorithm parameterized by τ, executed by *Verifier*; Θ is a set. $\mathcal{ID}$ scheme is a three-round protocol depicted in Fig. 3, where Prover first generates a committing message CMT with $H_\infty(\mathrm{CMT}) = \omega(\log \lambda)$, and then Verifier replies with a challenge $\mathrm{CH} \leftarrow \Theta$ and finally Prover finishes with a response Rsp which will be either rejected or accepted by V_τ.

Denote the domain of sk, pk, CMT, Rsp respectively by $\mathcal{SK}, \mathcal{PK}, \mathcal{CMT}, \mathcal{RSP}$. In the following, we define linearity and simutability for an ID scheme. Simulatbility appeared before (e.g., [1]) while the linearity is new.

Linearity. A canonical ID scheme $\mathcal{ID} = (\mathbf{Setup}, \mathbf{KeyGen}, P, V_\tau, \Theta)$ is **linear** if it satisfies the following conditions.

i. $\mathcal{SK}, \mathcal{PK}, \mathcal{CMT}, \mathcal{RSP}$ are $\mathcal{R}$-modules for some ring $\mathcal{R}$ with $\Theta \subseteq \mathcal{R}$ (as a set);
ii. For any $\lambda_1, \cdots, \lambda_t \in \Theta$ and public/private pairs (sk_i, pk_i) $(i = 1, \cdots, t)$, we have that $\overline{sk} = \sum_{i=1}^t \lambda_i \bullet sk_i$ is a private key of $\overline{pk} = \sum_{i=1}^t \lambda_i \bullet pk_i$.
Note: Operator $\bullet$ between $\mathcal{R}$ and $\mathcal{SK}$ (resp. $\mathcal{PK}, \mathcal{CMT}, \mathcal{RSP}$) might be different (as long as it is clear from the context), even though we use the same symbol $\bullet$.
iii. Let $\lambda_i \leftarrow \Theta$ and $(pk_i, sk_i) \leftarrow \mathbf{KeyGen}(1^\kappa)$, for $i = 1, \cdots, t$. If $\mathrm{CMT}_i|\mathrm{CH}|\mathrm{Rsp}_i$ is a *faithfully* generated transcript of the ID scheme w.r.t. pk_i, then
$$V_\tau(\overline{pk}, \overline{\mathrm{CMT}}|\mathrm{CH}|\overline{\mathrm{Rsp}}) = 1, \tag{19}$$
where $\overline{pk} = \sum_{i=1}^t \lambda_i \bullet pk_i, \overline{\mathrm{CMT}} = \sum_{i=1}^t \lambda_i \bullet \mathrm{CMT}_i$ and $\overline{\mathrm{Rsp}} = \sum_{i=1}^t \lambda_i \bullet \mathrm{Rsp}_i$.

Simulability. $\mathcal{ID}$ is simulatable if there exists a PPT algorithm **SIM** s.t. for $(sk, pk) \leftarrow \mathbf{KeyGen}(1^\lambda)$, $\mathrm{CH} \leftarrow \Theta$ and $(\mathrm{CMT}, \mathrm{Rsp}) \leftarrow \mathbf{SIM}(\mathrm{CH}, pk, \mathsf{param})$, it holds that CMT|CH|Rsp is indistinguishable from a real transcript, even if the distinguisher is given $pk|$param and has access to oracle $\mathcal{O}_{id}(sk, pk)$, where $\mathcal{O}_{id}(sk, pk)$ is as follows: $(st, \mathrm{CMT}) \leftarrow P(\mathsf{param})$; $\mathrm{CH} \leftarrow \Theta$; $\mathrm{Rsp} \leftarrow P(st|sk|pk, \mathrm{CH})$; output CMT|CH|Rsp.

Now we define the security for an ID scheme. Essentially, it is desired that an attacker is unable to impersonate a prover w.r.t. an aggregated public-key, where at least one of the participating public-keys is not generated by attacker. Later we will use this definition to convert an ID scheme into a secure multi-signature. In our definition, the prover does not access to additional information. He is not given extra capability, either. Thus, our definition is rather weak.

Definition 5. A canonical identification scheme $\mathcal{ID} = (\mathbf{Setup}, \mathbf{KeyGen}, P, V_\tau, \Theta)$ with linearity and $\tau \in \mathbb{N}$ is **secure** if it satisfies correctness and security below.

Correctness. When no attack presents, Prover will convince Verifier, except for a negligible probability.

Security. For any PPT adversary $\mathcal{A}$, $\Pr(\mathrm{EXP}_{\mathcal{ID},\mathcal{A}} = 1)$ is negligible, where $\mathrm{Exp}_{\mathcal{ID},\mathcal{A}}$ is defined as follows, where $pk_i \in \mathcal{PK}$ for $i \in [t]$ and $\overline{pk} = \sum_{i=1}^t \lambda_i \bullet pk_i$.

Experiment $\mathrm{Exp}_{\mathcal{ID},\mathcal{A}}(\lambda)$
param$\leftarrow \mathbf{Setup}(1^\lambda)$;
$(pk_1, sk_1) \leftarrow \mathbf{KeyGen}(\mathsf{param})$;
$(st_0, pk_2, \cdots, pk_t) \leftarrow \mathcal{A}(\mathsf{param}, pk_1)$
$\lambda_1, \cdots, \lambda_t \leftarrow \Theta$
$st_1|\mathrm{CMT} \leftarrow \mathcal{A}(st_0, \lambda_1, \cdots, \lambda_t)$;
$\mathrm{CH} \leftarrow \Theta$; $\mathrm{Rsp} \leftarrow \mathcal{A}(st_1, \mathrm{CH})$;
$b \leftarrow V_t(\overline{pk}, \mathrm{CMT}|\mathrm{CH}|\mathrm{Rsp})$;
output b.

$\mathcal{ID}$ is said t^***-secure** if the security holds for any $t \le t^*$.

6 From canonical Linear ID Scheme to Key-and-Signature Compact Multi-signature

In this section, we show how to convert a linear ID scheme into a multi-sinagure so that the aggregated public-key and signature are both compact. The idea is to linearly add the member signatures (resp. public-keys) together with weights while the weight depends on all public-keys and is different for each user.

6.1 Construction

Let
$$\mathcal{ID} = (\mathbf{Setup}_{id}, \mathbf{KeyGen}_{id}, P, V_\tau, \Theta)$$
be a canonical linear ID with parameter $\tau \in \mathbb{N}$. H_0, H_1 are two random oracles from $\{0,1\}^*$ to Θ with $\Theta \subseteq \mathcal{R}$, where $\mathcal{R}$ is the ring defined for the linearity property of $\mathcal{ID}$. Our multi-signature scheme $\Pi = (\mathbf{Setup}, \mathbf{KeyGen}, \mathbf{Sign}, \mathbf{Verify})$ is as follows.

Setup. Sample and output $\mathsf{param} \leftarrow \mathbf{Setup}_{id}(1^\lambda)$. *Note:* param should be part of the input to the algorithms below. But for brevity, we omit it in the future.

KeyGen. Sample $(pk, sk) \leftarrow \mathbf{KeyGen}_{id}(\mathsf{param})$; output a public-key pk and private key sk.

Sign. Suppose that users with public-keys $pk_i, i = 1, \cdots, t$ want to jointly sign a message M. Let $\lambda_i = H_0(pk_i, PK)$ and $\overline{pk} = \sum_{i=1}^t \lambda_i \bullet pk_i$, where $PK = \{pk_1, \cdots, pk_t\}$. They run the following procedure.

- *R-1.* User i takes $(st_i, \mathrm{CMT}_i) \leftarrow P(\mathsf{param})$ and sends $r_i := H_0(\mathrm{CMT}_i|pk_i)$ to other users.
- *R-2.* Upon r_j for all j (we do not restrict $j \neq i$ for simplicity), user i verifies if $r_j = H_0(\mathrm{CMT}_j|pk_j)$. If no, it aborts; otherwise, it sends CMT_i to other users.
- *R-3.* Upon $\mathrm{CMT}_j, j = 1, \cdots, t$, user i computes $\overline{\mathrm{CMT}} = \sum_{j=1}^t \lambda_j \bullet \mathrm{CMT}_j$. It computes

$$
\begin{array}{lcl}
\boxed{\textbf{Prover}}(sk, pk|\tau) & & \boxed{\textbf{Verifier}}(pk|\tau) \\
(st, \text{CMT}) \leftarrow P(\mathsf{param}) & \xrightarrow{\text{CMT}} & \text{CH} \leftarrow \Theta \\
 & \xleftarrow{\text{CH}} & \\
\text{Rsp} \leftarrow P(st|sk|pk, \text{CH}) & \xrightarrow{\text{Rsp}} & V_\tau(pk, \text{CMT}|\text{CH}|\text{Rsp}) \stackrel{?}{=} 1
\end{array}
$$

Fig. 3. Canonical Identification Protocol

$\text{CH} = H_1(\overline{pk}|\overline{\text{CMT}}|M)$. Finally, it computes $\text{Rsp}_i = P(st_i|sk_i|pk_i, \text{CH})$ and sends it to other signers.

- *Output.* Upon $\text{Rsp}_j, j = 1, \cdots, t$, user i computes $\overline{\text{Rsp}} = \sum_{j=1}^{t} \lambda_j \bullet \text{Rsp}_j$, and outputs the aggregated public-key $\overline{pk}|t$ and multi-signature $\overline{\text{CMT}}|\overline{\text{Rsp}}$.

Verify. Upon signature $(\overline{\text{CMT}}, \overline{\text{Rsp}})$ on message M with the aggregated public key $\overline{pk}|t$, it outputs $V_t(\overline{pk}, \overline{\text{CMT}}|\text{CH}|\overline{\text{RSP}})$, where $\text{CH} = H_1(\overline{pk}|\overline{\text{CMT}}|M)$.

Remark. (1) Since $\overline{pk}|t$ is the aggregated public-key, we assume that it will be correctly computed and available to verifier, which is true for the Bitcoin application.
(2) The most damaging attack to a multi-signature is the rogue key attack, where an attacker chooses his public-key after seeing other signers' public-keys. By doing this, the attacker could manage to reach an aggregated key for which he knows the private key. In our construction, attacker can not achieve this. Indeed, notice that $\overline{pk} = H_0(pk_n, PK) \bullet pk_n + \sum_{i=1}^{n-1} H_0(pk_i, PK) \bullet pk_i$, where $PK = \{pk_1, \cdots, pk_n\}$. The hash-value weights can be computed only after PK has been determined. Also, if pk_n is the honest user's key, then it is quite random. So, $H_0(pk_n, PK)$ (hence $H_0(pk_n, PK) \bullet pk_n$ and also $\overline{pk}$) will be random, given other variables in $\overline{pk}$. So it is unlikely that attacker can predetermined $\overline{pk}$ and so the rogue key attack can not succeed.

6.2 Security Theorem

In this section, we prove the security of our scheme. The idea is as follows. We notice that the multi-signature is $(\overline{\text{CMT}}, \overline{\text{RSP}})$ that satisfies $V_t(\overline{pk}, \overline{\text{CMT}}|\text{CH}|\overline{\text{RSP}}) = 1$, where $\text{CH} = H_0(\overline{pk}|\overline{\text{CMT}}|M)$. Assume $PK = \{pk_1, \cdots, pk_t\}$, where pk_1 is an honest user's key and other keys are created by attacker. We want to reduce the multi-signature security to the security of ID scheme. In this case, $\overline{pk}$ will be the aggregated key with weights $\lambda_i = H_0(pk_i, PK)$. If an attacker can forge a multi-signature with respect to $\overline{pk}$, we want to convert it into an impersonate attack to the ID scheme w.r.t. $\overline{pk}$. There are two difficulties for this task. First, we need to simulate the signing oracle without sk_1, where we have to compute the response Rsp for user of pk_1 without sk_1. Our idea is to use the simulability of the ID scheme to help: take a random CH and simulate an ID transcript $\text{CMT}'|\text{CH}|\text{Rsp}'$. Then, we send $\text{CMT}_1 = \text{CMT}'$ as the committing message. The simulation will be well done if we can manage to define CH as $H_1(\overline{pk}|\overline{\text{CMT}}|M)$. This will be fine if $\overline{pk}|\overline{\text{CMT}}|M$ was never queried to H_1 oracle. Fortunately, this is true with high probability: due to the initial registration message at round *R-1*, attacker can not know CMT_1 before registering CMT_j using r_j (hence CMT_j is known to us through oracle H_0). Hence, $\overline{\text{CMT}}$ will have a min-entropy of $H_\infty(\text{CMT}_1)$, which is super-logarithmic and hence can not be guessed. That is, $\overline{pk}|\overline{\text{CMT}}|M$ was unlikely to be queried to H_1 before. Hence, the signing oracle will be simulated without difficulty. The second difficulty is how to convert the forgery into an impersonating attack. In the ID attack, CH is provided by challenger while in the forgery, CH is the hash value from H_1. The problem is the attacker could make a query $\overline{pk}|\overline{\text{CMT}}|M$ to H_1 oracle (we maintain) while we do not know whether this query is toward his final forgery output or not and so we do not know which CMT should be sent to our challenger and consequently we do not know which of such queries should be answered with our challenger's CH. Fortunately, this is not a big issue as we can guess which query will be used for the forgery. There are a polynomial number of such queries. Our random guess only degrades the success probability by a polynomial fraction. This completes our idea. Now we give full details below.

Theorem 1. Assume that $h \leftarrow \Theta$ is invertible in $\mathcal{R}$ with probability $1 - \mathsf{negl}(\lambda)$. Let $\mathcal{ID} = (\textbf{Setup}_{id}, \textbf{KeyGen}_{id}, P, V_\tau, \Theta)$ be a secure identification scheme with linearity and simulability. Then, our multi-signature scheme is **EU-CMA** secure.

Proof. We show that if the multi-signature is broken by $\mathcal{D}$ with non-negligible probability ϵ, then we can construct an attacker $\mathcal{B}$ to break $\mathcal{ID}$ scheme with a non-negligible probability ϵ'. Given the challenge public-key pk_1^*, $\mathcal{B}$ needs to come up with some other public-keys $pk_2^*, \cdots, pk_\nu^*$ for some ν of his choice and receives a list of random numbers $\lambda_i^* \leftarrow \Theta$ for $i = 1, \cdots, \nu$. Then, he needs to play as a prover in the $\mathcal{ID}$ protocol for public-key $\overline{pk^*} = \sum_{i=1}^{\nu} \lambda_i^* \bullet pk_i^*$ to convince the verifier (his challenger). Toward this, $\mathcal{B}$ will simulate an environment for $\mathcal{D}$ and use the responses from $\mathcal{D}$ to help complete his attack activity. The details follow.

Upon receiving the challenge public-key pk_1^* and system parameter param, $\mathcal{B}$ samples $\ell_{H_0}^* \leftarrow \{1, \cdots, q_{H_0}^*\}$, where $q_{H_0}^*$ is the upper bound on the number of *new* queries (i.e., not queried before) of form (pk, PK) to random oracle H_0 s.t. $pk, pk_1^* \in PK$ (call it a ***Type-I irregular query***). In addition, a *new* query of format $\text{CMT}|\overline{pk^*}|*$ to oracle H_1 *after* the $\ell_{H_0}^*$th Type-I irregular query will be called a ***Type-II irregular query***, where $\text{CMT} \in \mathcal{CMT}$, $\overline{pk^*} = \sum_{i=1}^{\nu} H_0(pk_i^*, PK^*) \bullet pk_i^*$ and $PK^* = \{pk_1^*, \cdots, pk_\nu^*\}$ is the public-key set for the $\ell_{H_0}^*$th Type-I irregular query. Let q_{ch}^* be the upper bound on the number of the Type-II irregular queries. It then samples $\ell_{ch}^* \leftarrow \{1, \cdots, q_{ch}^*\}$. $\mathcal{B}$ invokes $\mathcal{D}$ with pk_1^* and param and answers his random oracle queries and signing queries as follows.

Random Oracle $H(\cdot)$. For simplicity, we maintain one random oracle H with $H_0(x) = H(0, x)$ and $H_1(x) = H(1, x)$. The query x to H_b is automatically interpreted as query $b|x$ to H. With this in mind, it maintains a hash list L_H (initially empty), consisting of records of form (u, y), where $y = H(u)$. Upon a query $b|x$, it first checks if there was a record $(b|x, y)$ in L_H for some y. If yes, it returns y; otherwise, there are three cases (all irregular queries will be in these cases as they are unrecorded by definition).

- x is not a (Type-I or Type-II) irregular query to H_b. In this case, it takes $y \leftarrow \Theta$ and adds $(b|x, y)$ into L_H.
- x is a Type-I irregular query to H_b (thus $b = 0$). In this setting, there are two cases.
  - *x is not the $\ell^*_{H_0}$th irregular query.* In this case, for each $pk' \in PK$, it takes $h \leftarrow \Theta$ and adds $(0|(pk', PK), h)$ into L_H. Note for convenience, we treat each new record in L_H as created due to a hash query (from either simulator $\mathcal{B}$ or $\mathcal{D}$). For the technical reason, for given PK with $pk^*_1 \in PK$, we treat $(0|(pk^*_1, PK), *)$ as the last record created in L_H among all records of $(0|(pk', PK), *)$ with $pk' \in PK$. Our treatment is well-defined and perfectly consistent with random oracle, as by our convention, all records of (pk', PK) with $pk', pk^*_1 \in PK$ will be recorded in L_H simultaneously whenever it receives a Type-I irregular query (which is $0|x$ in our case).
  - *x is the $\ell^*_{H_0}$th irregular query.* In this case, let $0|x = 0|(pk, PK^*)$ with $PK^* = \{pk^*_1, \cdots, pk^*_\nu\}$ for some $\nu \geq 2$. $\mathcal{B}$ sends $\{pk^*_2, \cdots, pk^*_\nu\}$ to his challenger and receives $\lambda^*_1, \cdots, \lambda^*_\nu$ (each of which is uniformly random over Θ). Then, $\mathcal{B}$ inserts $(0|(pk^*_i, PK^*), \lambda^*_i)$ into L_H for $i = 1, \cdots, \nu$. This treatment is perfectly consistent with random oracles: a Type-I irregular query by definition is an *unrecorded* query (i.e., not queried before) and $0|(pk', PK^*)$ for each $pk' \in PK^*$ will be recorded in L_H within one hash query (thus none of them was queried before).
- x is a Type-II irregular query to H_b (thus $b = 1$). In this setting, there are two cases.
  - *x is not the ℓ^*_{ch}th Type-II irregular query.* In this case, it takes $y \leftarrow \Theta$ and adds $(1|x, y)$ into L_H.
  - *x is the ℓ^*_{ch}th Type-II irregular query.* In this case, it parses $x = \mathrm{CMT}^*|\overline{pk^*}|M^*$ with $\mathrm{CMT}^* \in \mathcal{CMT}$. Then, it sends CMT^* to its challenger and receive CH^*. Then, it adds $(1|x, \mathrm{CH}^*)$ to L_H.

After our treatment above, x now has been recorded in L_H. Then, the oracle returns y for $(b|x, y) \in L_H$.

Sign $\mathcal{O}_s$ $(pk_1, \cdots, pk_n, M)$. By our security model, it is assumed that $pk^*_1 = pk_t$ for some t. Then, $\mathcal{B}$ plays the role of user pk_t while $\mathcal{D}$ plays users of pk_j for $j \neq t$ in the signing algorithm. The action of $\mathcal{B}$ is as follows.

- *R-1.* $\mathcal{B}$ generates $r_t \leftarrow \Theta$ and sends to other signers (played by $\mathcal{D}$).
- *R-2.* Upon $\{r_j\}_{j \neq t}$ from $\mathcal{D}$, $\mathcal{B}$ first issues hash queries (pk_i, PK) for each $pk_i \in PK$ to compute $\lambda_i = H_0(pk_i, PK)$, where $PK = \{pk_1, \cdots, pk_n\}$. Then, it computes $\overline{pk}$, takes $h \leftarrow \Theta$ and runs $\mathbf{SIM}(h, pk^*, \mathsf{param})$ to simulate an ID transcript $(\mathrm{CMT}', h, \mathrm{Rsp}')$. Then, he defines $\mathrm{CMT}_t = \mathrm{CMT}'$. He also adds $(0|\mathrm{CMT}_t|pk_t, r_t)$ into L_H (in case $(0|\mathrm{CMT}_t|pk_t, *)$ not in L_H) and otherwise aborts with $\perp$ (denoted by event Bad_0). Next, for each $j \neq t$, it searches a record $(0|\mathrm{CMT}_j|pk_j, r_j)$ in L_H for some CMT_j which results in two cases.
  (i) If $(0|\mathrm{CMT}_j|pk_j, r_j)$ for all $j \neq t$ are found in L_H, it computes $\overline{\mathrm{CMT}} = \sum_{i=1}^n \lambda_i \bullet \mathrm{CMT}_i$ and checks whether $(1|\overline{pk}|\overline{\mathrm{CMT}}|M, y) \in L_H$ for some y. If this y does not exist, it records $(1|\overline{pk}|\overline{\mathrm{CMT}}|M, h)$ into L_H and defines $\mathrm{CH} = h$ and sends CMT_t to $\mathcal{D}$; otherwise (denote this event by Bad_1), $\mathcal{B}$ aborts with $\perp$.
  (ii) If $(0|\mathrm{CMT}_{j^*}|pk_{j^*}, r_{j^*})$ does not exist in L_H for some j^*, it sends CMT_t to $\mathcal{D}$ (normally). However, we remark that CMT_{j^*} later in Step *R-3* (from j^*) satisfies $H_0(\mathrm{CMT}_{j^*}|pk_{j^*}) = r_{j^*}$ (which will be checked there) only negligibly (so this case will not raise a simulation difficulty), as the hash value is even undefined yet and hence equals r_j with probability $1/|\Theta|$ only, which we ignore it now.
- *R-3.* Upon $\{\mathrm{CMT}_j\}_{j \neq t}$, $\mathcal{B}$ checks if $H_0(\mathrm{CMT}_j|pk_j) = r_j$ for each j. If it does not hold for some j, $\mathcal{O}_s$ outputs $\perp$ (normally); otherwise, it sends $\mathrm{Rsp}_t := \mathrm{Rsp}'$ to $\mathcal{D}$. We clarify two events: (1) some CMT_j found in *R-2*(i) is different from that received in the current step. In this case, the check in the current step is consistent with a negligible probability only as H for two different inputs are independent. (2) *R-2*(ii) occurs to some j^* (so CMT_{j^*} is not found there) while CMT_{j^*} received in the current step is consistent with r_j. As seen above, this holds with probability $1/|\Theta|$ only. Ignoring these events, CH and $\{\mathrm{CMT}_j\}_j$ are determined in *R-2*(i) and $\{\mathrm{CMT}_j\}_j$ are consistent with those received in the current step.
- *Output.* Upon Rsp_j for $j \neq t$, it computes $\overline{\mathrm{RSP}} = \sum_{j=1}^n \lambda_j \bullet \mathrm{Rsp}_j$. The final signature is $(\overline{\mathrm{CMT}}, \overline{\mathrm{RSP}})$ with the aggregated key $\overline{pk}|t$.

Finally, $\mathcal{D}$ outputs a forgery (α, β) for message M' and public keys PK'. If $\alpha|PK'|M' \neq \mathrm{CMT}^*|PK^*|M^*$ or $\alpha|\beta$ is invalid (when verified using $V(\cdot)$), $\mathcal{B}$ exits with $\perp$; otherwise, he verifies (α, β). If invalid, he outputs $\perp$; otherwise, he defines $\mathrm{Rsp}^* = \beta$ and sends it back to his challenger. This completes the description of $\mathcal{B}$.

We now analyze the success probability of $\mathcal{B}$. First, the view of $\mathcal{D}$ is identical to the real game, except for the following events.

a. In step *R-2* of $\mathcal{O}_s$, $(\mathrm{CMT}', h, \mathrm{Rsp}')$ is simulated by **SIM** (instead of being computed using sk^*_1). However, by hybrid reduction to simulability of $\mathcal{ID}$, the view of $\mathcal{D}$ is statistical close from his view when this transcript is generated using sk^*_1 (with the same h).
b. In step *R-2* of oracle $\mathcal{O}_s$, when $(0|\mathrm{CMT}_t|pk_t, y) \in L_H$, Bad_0 occurs for some y (hence the view of $\mathcal{D}$ is inconsistent if $y \neq r_t$). However, since CMT' (i.e., CMT_t) is just simulated in this oracle query and $H_\infty(\mathrm{CMT}') = \omega(\log \lambda)$, CMT' is independent of

current records in L_H. Hence, Bad_0 occurs with probability at most $Q/2^{H_\infty(\mathrm{CMT}')}$ (negligible), where Q is the number of records in L_H. We ignore this negligible probability from now on.

c. In step *R-2* (i), if $(1|\overline{pk}|\overline{\mathrm{CMT}}|M,y) \in L_H$ for some y, then event Bad_1 occurs. In this case, A can not define $\mathrm{CH}=h$ and the simulation can not continue. However, since $\overline{\mathrm{CMT}} = \lambda_t \bullet \mathrm{CMT}' + \sum_{j\neq t}\lambda_j \bullet \mathrm{CMT}_j$ and CMT' is simulated in the current oracle and hence independent of the rest variables in this equation. Hence, as long as λ_t is invertible (which is violated only negligibly), $\overline{\mathrm{CMT}}$ has a min-entropy at least $H_\infty(\mathrm{CMT}) = \omega(\log\lambda)$. Thus, similar to Bad_0 event, Bad_1 occurs negligibly only.

d. Finally, when $\mathcal{D}$ outputs (α,β) for message M' and public-key set PK', it has $\alpha|PK'|M' \neq \mathrm{CMT}^*|PK^*|M^*$. Since (α,β) has been verified, a Type-I irregular query (pk, PK') and a Type-II irregular query $\alpha|\overline{pk'}|M$ must have been issued (the first query (pk, PK') for some $pk \in PK'$ is the Type-I irregular query while the first query of $\alpha|\overline{pk'}|M$ is the Type-II query; the existence of such queries are guaranteed as the verification of (α,β) by $\mathcal{B}$ will certainly issue these queries). Since ℓ_H^* and ℓ_{ch}^* are chosen uniformly random, they happened to the foregoing two queries with probability $\frac{1}{q_H^* q_{ch}^*} \geq \frac{1}{q_0 q_1}$, where q_0 (resp. q_1) is the upper bound on $\sharp$ queries to H_0 (resp. H_1).

From the analysis of (a)(b)(c), their occurrence changes the adversary view negligibly. Ignoring this, from item d, when ℓ_H^* and ℓ_{ch}^* is chosen correctly, the view of $\mathcal{D}$ is indistinguishable from its view in the real game. On the other hand, it is easy to verify that conditional on this correct choice, a valid forgery indicates a successful attack by $\mathcal{B}$. Hence, $\mathcal{B}$ can break the ID security with probability at least ϵ/q_0q_1, non-negligible. This contradicts the security of our ID scheme. □

If the adversary always restricts the number of signers in the signing query and the forgery to be at most T, then Theorem immediately implies the following corollary.

Corollary 1. Let $T \geq 2$. Assume that $h \leftarrow \Theta$ is invertible in $\mathcal{R}$ with probability $1-\mathsf{negl}(\lambda)$. Let $\mathcal{ID} = (\mathbf{Setup}_{id}, \mathbf{KeyGen}_{id}, P, V_\tau, \Theta)$ be a T-secure identification scheme with linearity and simulability. Then, our multi-signature scheme is T-**EU-CMA** secure.

7 REALIZATIONS

In this section, we will realize our compiler with ID schemes: Schnorr ID scheme and a lattice-based ID scheme. The first scheme is similar to Boneh et al. [13]. But we keep it as it is very simple and efficient and can demonstrate the usage of our compiler. The second one is new and breaks a barrier that the previous schemes can not overcome.

7.1 Realization I: Schnorr Identification

In this section, we apply our compiler to the well-known Schnorr ID scheme [48]. Toward this, we only need to show that it is linear with simulability and security. For clarity, we first review this scheme.

Let q be a large prime. Consider a prime group of order q with a random generator g (e.g., the group on elliptic curve secp256k1 of $y^2 = x^3+7$ for Bitcoin). The Schnorr identification is depicted in Fig. 4. This scheme can be regarded as a realization of the parameterized ID scheme with parameter τ never used. In the following, we show that Schnorr ID scheme satisfies the three properties.

Linearity. Notice that $\mathcal{SK} = \mathcal{RSP} = \mathcal{R} = \Theta = \mathbb{Z}_q$, $\mathcal{CMT} = \mathcal{PK} = \langle g\rangle$. We now verify the linearity property.

i. As seen in Section 2.1, $\mathbb{Z}_q$ and $\langle g\rangle$ are both $\mathbb{Z}_q$-modules, where the multiplication $\bullet$ between $\mathcal{R} = \mathbb{Z}_q$ and $\mathbb{Z}_q$ is the multiplication of $\mathbb{Z}_q$, while $\bullet$ between $\mathcal{R}=\mathbb{Z}_q$ and $\langle g\rangle$ is exponentiation: $s\bullet m = m^s$. Hence, $\mathcal{SK}, \mathcal{PK}, \mathcal{CMT}, \mathcal{RSP}$ are $\mathcal{R}$-modules.

ii. Let $pk_i = g^{s_i}$ with $sk_i = s_i, i=1,\cdots,n$. Let $\lambda_1,\cdots,\lambda_n \in \mathcal{R}$. Then, $\overline{sk} = \sum_{i=1}^n \lambda_i \bullet sk_i = \sum_{i=1}^n \lambda_i s_i$, where the addition is the group operation for $\mathcal{SK}$ (i.e., addition in $\mathbb{Z}_q$). Note the group operation for $\mathcal{PK}$ is the multiplication in $\langle g\rangle$. Hence, $\overline{pk} = \prod_{i=1}^n \lambda_i \bullet pk_i = \prod_{i=1}^n pk_i^{\lambda_i} = g^{\sum_{i=1}^n \lambda_i s_i}$. Thus, $\overline{sk} \in \mathcal{SK}$ is the private key of $\overline{pk} \in \mathcal{PK}$.

iii. Let $X_i|c|z_i$ be a transcript of $\mathcal{ID}$ w.r.t., $pk_i = g^{s_i}$ and $sk_i = s_i, i=1,\cdots,n$. For $\lambda_i \in \mathcal{R}$, $\overline{X} = \prod_{i=1}^n \lambda_i \bullet X_i = \prod_{i=1}^n X_i^{\lambda_i}$ and $\overline{z} = \sum_{i=1}^n \lambda_i \bullet z_i = \sum_{i=1}^n \lambda_i z_i$. If $g^{z_i} = pk_i^c X_i$, then $\prod_{i=1}^n g^{\lambda_i z_i} = \prod_{i=1}^n (pk_i^c X_i)^{\lambda_i}$. Hence, $g^{\overline{z}} = \overline{pk}^c\,\overline{X}$, desired!

Simulability. Let $pk = g^s$ be the public-key and $sk = s$ be the private key. For $c \leftarrow \mathbb{Z}_q$, we define **SIM** by taking $z \leftarrow \mathbb{Z}_q$ and $X = g^z pk^{-c}$. The simulated ID transcript is $X|c|z$. Obviously, this transcript is valid (i.e., it passes the verification). Now we show that for any (even unbounded) distinguisher $\mathcal{D}$ that has oracle access to $\mathcal{O}_{id}$ can not distinguish the output of **SIM** from the real ID transcript. Notice for both simulated and real transcripts $X|c|z$, it satisfies $g^z = pk^c X$. Hence, $X = g^x$ for some $x \in \mathbb{Z}_q$ and $z = cs + x$. In the real transcript, $x \leftarrow \mathbb{Z}_q$ while the simulated transcript $z \leftarrow \mathbb{Z}_q$. Hence, given c, (x,z) (hence X,z) in both transcripts has the same distribution. Since c is uniformly random in $\mathbb{Z}_q$ in the simulation, the simulated and real transcripts have the same distribution (independent of adversary view before the challenge which includes the responses from $\mathcal{O}_{id}$). Thus, the adversary view, given oracle access to $\mathcal{O}_{id}$, in both cases has the same distribution. The simulability follows.

Security. We now prove the security of Schnorr ID scheme under Definition 5.

Lemma 2. Under discrete logarithm assumption, Schnorr ID scheme is secure w.r.t. Definition 5.

Proof. If there exists an adversary $\mathcal{D}$ that breaks the Schnorr ID scheme with non-negligible probability ϵ, then we construct an adversary $\mathcal{A}$ that breaks discrete logarithm in $\langle g\rangle$ with a non-negligible probability ϵ'. The idea is to make use of $\mathcal{D}$ to construct an algorithm A for the nested forking lemma and then use the output of the forking algorithm to derive the discrete logarithm for the challenge. Upon

$\boxed{\textbf{Prover}}(s, A = g^s)$		$\boxed{\textbf{Verifier}}(A = g^s)$
$x \leftarrow \mathbb{Z}_q, X = g^x$	$\xrightarrow{\quad X \quad}$	$c \leftarrow \mathbb{Z}_q$
	$\xleftarrow{\quad c \quad}$	
$z = sc + x \bmod q$	$\xrightarrow{\quad z \quad}$	$g^z \stackrel{?}{=} A^c X$

Fig. 4. Schnorr Identification Scheme

a challenge $A_1 = g^x$ and parameters q, g, $\mathcal{A}$ constructs $\mathsf{A}((A_1, g, q), \lambda_1, c; \rho)$ as follows (so $h_1|h_2 = \lambda_1|c$ with $q = 2$ in the forking algorithm), where $\overline{A} = \sum_{i=1}^t A_i^{\lambda_i}$.

Algorithm $\mathsf{A}((A_1, g, q), \lambda_1, c; \rho)$
 Parse ρ as two parts: $\rho = \rho_0|\rho_1$
 $(st_0, A_2, \cdots, A_t) \leftarrow \mathcal{D}(q, g, A_1; \rho_0)$
 $\lambda_2, \cdots, \lambda_t \leftarrow \mathbb{Z}_q$ using randomness ρ_1
 $st_1|X \leftarrow \mathcal{D}(st_0, \lambda_1, \cdots, \lambda_t)$;
 $z \leftarrow \mathcal{D}(st_1, c)$;
 If $g^z = \overline{A}^c \cdot X$, **then** $b = 1$;
 else $b = 0$;
 output $(b, 2b, \{A_i|\lambda_i\}_1^t|X|z|c|g|q)$.

From the description of A and the forking algorithm F_A (for the forking lemma), the rewinding in the forking algorithm F_A only changes λ_1 and/or c as well as those affected by (λ_1, c). In terms of forking lemma terminology, we have $(h_1, h_2) = (\lambda_1, c)$ and $I_0 = 1, J_0 = 2$ (for a successful execution; otherwise, A will abort when $I_0 \leq J_0$). Let us now analyze algorithm forking algorithm F_{A}. When four executions are executed successfully (i.e., $b = 1$ for all cases), then the output for each execution will be described as follows. Let $A_i = g^{a_i}$ for $i = 1, \cdots, t$.

- *Execution 0.* It outputs $(1, 2, \{A_i|\lambda_i\}_1^t|X|z|c|g|q)$. As the verification passes,

$$z = (\sum_{i=1}^t \lambda_i a_i)c + x, \tag{20}$$

 where $X = g^x$.
- *Execution 1.* Compared with execution 0, the input only changes c to $\hat{c}$. From the code of A, the output is $(1, 2, \{A_i|\lambda_i\}_1^t|X|\hat{z}|\hat{c}|g|q)$. As the verification passes,

$$\hat{z} = (\sum_{i=1}^t \lambda_i a_i)\hat{c} + x. \tag{21}$$

- *Execution 2.* Compared with execution 0, the input changes λ_1 to $\bar{\lambda}_1$ and c to $\bar{c}$. From the code of A, the output is $(1, 2, \{A_i|\lambda_i\}_2^t|A_1|\bar{\lambda}_1|X'|\bar{z}|\bar{c}|g|q)$. As the verification passes,

$$\bar{z} = (\bar{\lambda}_1 a_1 + \sum_{i=2}^t \lambda_i a_i)\bar{c} + x', \tag{22}$$

 where $X' = g^{x'}$.
- *Execution 3.* Compared with execution 0, the input changes λ_1 to $\bar{\lambda}_1$ and c to $\underline{c}$. From the code of A, the output is $(1, 2, \{A_i|\lambda_i\}_2^t|A_1|\bar{\lambda}_1|X'|\underline{z}|\underline{c}|g|q)$. As the verification passes,

$$\underline{z} = (\bar{\lambda}_1 a_1 + \sum_{i=2}^t \lambda_i a_i)\underline{c} + x'. \tag{23}$$

From Eqs. (23)(22), $\mathcal{A}$ can derive $\bar{\lambda}_1 a_1 + \sum_{i=2}^t \lambda_i a_i$, as long as $c \neq c'$ in $\mathbb{Z}_q$. Similarly, from Eqs. (21)(20), $\mathcal{A}$ can derive $\lambda_1 a_1 + \sum_{i=2}^t \lambda_i a_i$, as long as $\bar{c} \neq \underline{c}$. This can further give a_1, as long as $\lambda_1 \neq \bar{\lambda}_1$ in $\mathbb{Z}_q$. Finally, if the forking algorithm does not fail, then the four executions succeeds **and** $(c \neq c') \wedge (\bar{c} \neq \underline{c}) \wedge (\lambda_1 \neq \bar{\lambda}_1)$ =True. By forking lemma, it does not fail with probability at least $\epsilon^4/(1 \cdot 1) - 3/|\Theta| = \epsilon^4 - 3/q$. Hence, $\mathcal{A}$ can obtain a_1 with probability at least $\epsilon^4 - 3/q$, non-negligible. This contradicts to the discrete logarithm assumption. □

Key-and-Signature Compact Multi-Signature from Schnorr ID Scheme. Since Schnorr ID scheme satisfies the linearity, simulability and special soundness, the multi-signature from this scheme using our compiler is obtained. For clarity, we give the complete signature in the following. Let $pk_i = g^{s_i}$ be the public-key with private key $sk_i = s_i$ for $i = 1, \cdots, n$. When users $PK = \{pk_1, \cdots, pk_n\}$ want to jointly sign a message M, they act as follows.

- **R-1.** User i generates $X_i = g^{x_i}$ for $x_i \leftarrow \mathbb{Z}_q$ and sends $H_0(X_i|pk_i)$ to other users.
- **R-2.** Upon $\{r_j\}_{j=1}^n$, user i sends X_i to other users.
- **R-3.** Upon $\{X_j\}_{j=1}^n$, user i checks $r_j \stackrel{?}{=} H_0(X_j|pk_j)$ for all j. If not, he rejects; otherwise, he computes

$$\overline{pk} = \prod_{i=1}^n pk_i^{H_0(pk_i, PK)} \tag{24}$$

$$\overline{X} = \prod_{i=1}^n X_i^{H_0(pk_i, PK)}. \tag{25}$$

 Then, he computes

$$c = H_1(\overline{pk}|\overline{X}|M), \ z_i = s_i c + x_i \tag{26}$$

 and sends z_i to leader.
- *Output.* Receiving all z_j's, user i computes

$$\overline{z} = \sum_{j=1}^n H_0(pk_j, PK) z_j.$$

 Finally, it outputs $(\overline{X}, \overline{z})$ as the multi-signature of M with the aggregated public-key $\overline{pk}$ (note: the compiler protocol includes n in the aggregated key; we omit it here as it is not used in the verification).
- *Verification.* To verify signature $(\overline{X}, \overline{z})$ for M with the aggregated public-key $\overline{pk}$, it computes $c = H_1(\overline{pk}|\overline{X}|M)$. It accepts only if $g^{\overline{z}} = \overline{pk}^c \cdot \overline{X}$.

We denote this signature scheme by $\mathsf{Schnorr\text{-}MultiSig}$. Notice that $c \leftarrow \mathbb{Z}_q$ is invertible in $\mathcal{R}$ with probability $1 - 1/q$. As it satisfies linearity, simulability and security, by Theorem 1, we have the following.

Corollary 2. If Discrete logarithm assumption in $\langle g\rangle$ holds, then Schnorr-MultiSig is EU-CMA.

Remark. Boneh et al. [13] proposed a method that transforms Schnorr ID to a key-and-signature compact multi-signature. Their protocol is an improvement of Maxwell et al. [38] to overcome a simulation flaw. Their protocol is also 3-round but computationally more efficient in the signing process than ours. However, our sizes of aggregated (public-key, signature) as well as the verification cost are all the same as theirs (also identical to the original Schnorr signature case). Aggregated public-key and signature have impacts on the storage at a large number of blockchain nodes and the verification cost has the impact on the power consumption on these nodes. The signing cost is relatively not so important as it only has impact on the involved signers. Boneh et al. [13] uses $\lambda_i s_i$ as a secret for public-key $pk_i^{\lambda_i}$ to generate a member signature $X_i|c|z_i$ and the final multi-signature $\widetilde{X}=\prod_i X_i$ and $\tilde{z}=\sum_i z_i$. Their main saving (over us) is to avoid n exponentiations in computing our $\widetilde{X}$. One might be motivated to modify our general compiler so that it uses $\lambda_i\bullet pk_i$ (whose private key is $\lambda_i\bullet sk_i$) to generate a member signature $\mathrm{CMT}_i|\mathrm{Rsp}_i$ so that the final multi-signature is $\widetilde{\mathrm{CMT}}|\widetilde{\mathrm{RSP}}$ with $\widetilde{\mathrm{CMT}}=\sum_i \mathrm{CMT}_i$ and $\widetilde{\mathrm{RSP}}=\sum_i \mathrm{Rsp}_i$. However, this looking secure scheme has a simulation issue in general when we prove Theorem 1: it is required that $\{\mathbf{SIM}(\mathrm{CH},\lambda\bullet pk)\}_\lambda$ is indistinguishable from the list of real transcripts for a fixed but random pk while it is not clear how this can be proven *generally*.

7.2 Realization II: a new lattice-based ID scheme

In this section, we propose a new ID scheme from lattice and then apply our compiler to obtain a lattice-based multi-signature scheme. This is the first lattice-based multi-signature that has both a compact public-key and a compact signature without a restart during the signing process.

Notations. The following notations are specific for this section (see Section 2 for more).

- As a convention for lattice over ring, this section uses security parameter n (a power of 2), instead of λ;
- q is a prime with $q\equiv 3 \bmod 8$;
- $R=\mathbb{Z}[x]/(x^n+1)$; $R_q=\mathbb{Z}_q[x]/(x^n+1)$; R_q^* is the set of invertible elements in R_q;
- for a vector $\mathbf{w}$, we implicitly assume it is a column vector and the ith component is w_i or $\mathbf{w}[i]$;
- for a matrix or vector X, X^T is its transpose;
- $\mathbf{1}$ denotes the all-1 vector $(1,\cdots,1)^T$ of dimension that is clear from the context;
- for $u=\sum_{i=0}^{n-1}u_ix^i\in R$, $||u||_\infty=\max_i|u_i|$;
- $\alpha\in\mathbb{Z}_q$ always uses the default representative with $-(q-1)/2\le\alpha\le(q-1)/2$ and similarly, for $u\in R_q$, each coefficient of u by default belongs to this range;
- $e=2.71828\cdots$ is the Euler's number;
- $\mathcal{C}=\{c\in R\mid ||c||_\infty\le\log n,\deg(c)<n/2\}$
- $\mathcal{Y}=\{y\in R\mid ||y||_\infty\le n^{1.5}\sigma\log^3 n\}$
- $\mathcal{Z}=\{z\in R\mid ||z||_\infty\le(n-1)n^{1/2}\sigma\log^3 n\}$.

7.2.1 Ring-LWE and Ring-SIS

In this section, we introduce the ring-LWE amd ring-SIS assumptions (see [33], [35], [45] for details). For $\sigma>0$, distribution $D_{\mathbb{Z}^n,\sigma}$ assigns the probability proportional to $e^{-\pi||\mathbf{y}||^2/\sigma^2}$ for any $\mathbf{y}\in\mathbb{Z}^n$ and 0 for other cases. As in [1], $y\leftarrow D_{R,\sigma}$ samples $y=\sum_{i=0}^{n-1}y_ix^i$ from R with $y_i\leftarrow D_{\mathbb{Z},\sigma}$.

The Ring Learning With Error (Ring-LWE$_{q,\sigma,2n}$) problem over R with standard deviation σ is defined as follows. Initially, it takes $s\leftarrow D_{R,\sigma}$ as secret. It then takes $a\leftarrow R_q, e\leftarrow D_{R,\sigma}$ and outputs $(a,as+e)$. The problem is to distinguish $(a,as+e)$ from a tuple (a,b) for $a,b\leftarrow R_q$. The Ring-LWE$_{q,\sigma,2n}$ assumption is to say that no PPT algorithm can solve Ring-LWE$_{q,\sigma,2n}$ problem with a non-negligible advantage. According to [18], [34], ring-LWE assumption with $\sigma=\tilde{\Omega}(n^{3/4})$ is provably hard and so it is safe to assume $\sigma=\Omega(n)$.

The Small Integer Solution problem with parameters q,m,β over ring R (ring-SIS$_{q,m,\beta}$) is as follows: given m uniformly random elements $a_1,\cdots,a_m$ over R_q, find $(t_1,\cdots,t_m)$ so that $||t_i||_\infty\le\beta$ and $a_1t_1+\cdots+a_mt_m=0$ (**note**: here we use $||\cdot||_\infty$ norm while the literature regularly uses square-root norm $||\cdot||$. However, the gap is only a factor n on β and does not affect the validity of the assumption according to the current research status for ring-SIS). We consider the case $m=3$. As we use $q=3 \bmod 8$, by [9, Theorem 1], $x^n+1=\Phi_1(x)\Phi_2(x)$ for irreducible polynomials $\Phi_1(x),\Phi_2(x)$ of degree $n/2$. So by Chinese remainder theorem, a_i is invertible, except for probability $2q^{-n/2}$. Hence, ring-SIS is equivalent to the case of invertible a_2 which is further equivalent to problem $a_1t_1+t_2+a_3t_3=0$, as we can multiply it by a_2^{-1}. By [15], [33], the best *quantum* polynomial algorithm for ring-SIS problem with q,m can only solve $\beta=2^{\tilde{O}(\sqrt{n})}$ case. Thus, it is safe to assume Ring-SIS$_{q,m,\beta}$ for any polynomial β or even $\beta=2^{\sqrt[4]{n}}$.

7.2.2 Construction

We now describe our new ID scheme from ring R. Initially, take $s_1,s_2\leftarrow D_{R,\sigma}, a\leftarrow R_q^*$ and compute $u=as_1+s_2$. The system parameter is a; the public key is u and the private key is (s_1,s_2). Our ID scheme is as follows; also see Fig. 5.

1. Prover generates $\mathbf{y}_1,\mathbf{y}_2\leftarrow\mathcal{Y}^\mu$ and computes $\mathbf{v}=a\mathbf{y}_1+\mathbf{y}_2$ and sends $\mathbf{v}$ to Verifier, where $\mu\ge\log^2 n$.
2. Receiver samples $c\leftarrow\mathcal{C}$ and sends it to Prover.
3. Upon c, Prover does the following:
   a. Compute $\mathbf{z}_1=s_1c\cdot\mathbf{1}+\mathbf{y}_1,\ \mathbf{z}_2=s_2c\cdot\mathbf{1}+\mathbf{y}_2$;
   b. Let $A=\{j\mid z_{1j},z_{2j}\in\mathcal{Z}\}$. If $A=\emptyset$, abort; otherwise, take $j^*\leftarrow A$ and compute
   $$z_1=z_{1j^*}+\sum_{j\ne j^*}y_{1j},\ z_2=z_{2j^*}+\sum_{j\ne j^*}y_{2j}.$$
4. Upon z_1,z_2, Verifier checks
   $$\sum_{i=1}^{\mu}v_i\stackrel{?}{=}az_1+z_2-uc,\ ||z_b||_\infty\stackrel{?}{\le}\eta,b=1,2,$$
   where $\eta_t=5\sigma n^2\sqrt{t\mu}\log^6 n$ and t is a positive integer (see the remark below). If all are valid, it accepts; otherwise, it rejects.

Remark. We give two clarifications.
(1) The correctness does not need η_t to vary with t as it is defined so. Actually, $\eta_1=3\sigma n^{1.5}\sqrt{\mu}\log^4 n$ suffices for all t. However, we need the dependency on t for the linearity and

later for the multi-signature. Especially, for linearity with t transcripts, η_t is needed to depend on t. Further, for the multi-signature scenario, t stands for the number of signers. (2) It should be pointed out that the choice of j^* (if it exists) does not affect z_1, z_2 at all as $z_b = s_b c + \sum_{j=1}^{t} y_{bj}$ for $b = 1, 2$. In addition, the probability that j^* does not exist is exponentially small in n and so defining j^* is unnecessary. However, we keep it for ease of analysis later.

Correctness. We now prove the correctness with η_t replaced by a smaller value $\eta_1 = 3\sigma n^{1.5}\sqrt{\mu}\log^4 n$. When all signers are honest, the protocol is easily seen to be correct if we can show $A = \emptyset$ or $||z_b||_\infty > \eta_1$ has a negligible probability. The former is shown in Lemma 5 below. For the latter, notice that $z_1 = s_1 c + y_{11} + \cdots + y_{1\mu}$. If we use $\underline{w} \in R$ to denote the coefficient vector of the polynomial w. Then,

$$\underline{y_{11} + \cdots + y_{1\mu}} = \underline{y_{11}} + \cdots + \underline{y_{1\mu}}. \tag{27}$$

Notice each component of $\underline{y_{1j}}$ is uniformly random in $\{-\sigma n^{1.5}\log^3 n, \cdots, \sigma n^{1.5}\log^3 n\}$. By Hoeffding inequality on each of the vector component in Eq. (27), $||\sum_i \underline{y_{1i}}||_\infty > 2\sigma n^{1.5}\sqrt{\mu}\log^4 n$ only has a probability at most $2ne^{-\log^2 n}$. By Lemma 3 below, $||sc||_\infty > \sigma n^{1/2}\log^3 n$ with probability at most $e^{-\Omega(\log^2 n)}$. Hence, correctness holds for bound η_1, except for probability at most $e^{-\Omega(\log^2 n)}$ (**note**: for brevity, this quantity should be understood as there exists constant C so that the exception probability is at most $e^{-C\log^2 n}$; we will later keep this convention without a mention).

7.2.3 Analysis

In this section, we analyze our ID scheme. We start with some preparations. The following lemma is adapted from [1, Lemma 4], where our restriction that the element c of $\mathcal{C}$ has a degree at most $n/2$, does not affect the proof.

Lemma 3. [1] If $s \leftarrow D_{R,\sigma}$ and $c \leftarrow \mathcal{C}$, then

$$\Pr(||sc||_\infty \le \sigma n^{1/2}\log^3 n) \ge 1 - e^{-\Omega(\log^2 n)},$$

where $e = 2.71828\cdots$ is the Euler's number

The lemma below was in the proof of [1, Lemma 3].

Lemma 4. [1] Fix $\gamma \in R$ with $||\gamma||_\infty \le \sigma n^{1/2}\log^3 n$. Then, for $y \leftarrow \mathcal{Y}$, we have

$$\Pr(\gamma + y \in \mathcal{Z}) \ge \frac{1}{e} - \frac{1}{en}$$
$$\Pr(\gamma + y = g \mid \gamma + y \in \mathcal{Z}) = \frac{1}{|\mathcal{Z}|}, \forall g \in \mathcal{Z}.$$

Lemma 5. Let A be the index set in our ID scheme. Then, $\Pr(A = \emptyset) < e^{-\Omega(\log^2 n)}$ for $\mu \ge \Omega(\log^2 n)$.

Proof. Notice $z_{bj} = sc + y_{bj}$ for $b = 1, 2$. By Lemma 3, $||sc||_\infty \le \sigma n^{1/2}\log^3 n$ with probability $1 - e^{-\Omega(\log^2 n)}$. Fixing sc (that satisfies this condition), z_{bj} for $b = 1, 2, j = 1, \cdots, \mu$ are independent and thus by Lemma 4, $A = \emptyset$ with probability at most $(1 - \frac{1}{e^2}(1 - \frac{1}{n})^2)^\mu < (1 - \frac{1}{4e^2})^\mu$, exponentially small. Together with the probability for $||sc||_\infty \le \sigma n^{1/2}\log^3 n$, we conclude the lemma. □

Lemma 6. If $u \leftarrow \mathcal{C}$, then u is invertible in R_q with probability $1 - (1 + 2\log n)^{-n/2}$.

Proof. Recall that $q \equiv 3 \bmod 8$ in this section. By Blake et al. [9, Theorem 1], $x^n + 1 = \Phi_1(x)\Phi_2(x) \bmod q$, where $\Phi_1(x), \Phi_2(x)$ have degree $n/2$ and are irreducible over $\mathbb{Z}_q$. By Chinese remainder theorem, u is invertible in R_q if and only if it is non-zero mod $\Phi_b(x)$ for both $b = 1, 2$. Since u has a degree at most $n/2 - 1$, u remains unchanged after mod $\Phi_b(x)$. Hence, it is invertible in R_q if and only if u is non-zero. This has a probability $1 - (1 + 2\log n)^{-n/2}$. □

Simulability. We now show the simulability of our ID scheme. Given the public-key u and $c \leftarrow \mathcal{C}$, we define the simulator **SIM** as follows.

- Sample $j^* \leftarrow [\mu]$ and $z_{1j^*}, z_{2j^*} \leftarrow \mathcal{Z}$; compute $v_{j^*} = az_{1j^*} + z_{2j^*} - uc$;
- For $j \in [\mu] - \{j^*\}$, sample $y_{1j}, y_{2j} \leftarrow \mathcal{Y}$ and compute $v_j = ay_{1j} + y_{2j}$.
- Compute $z_b = z_{bj^*} + \sum_{j \ne j^*} y_{bj}, b = 1, 2$.
- Output $\mathbf{v} = (v_1, \cdots, v_\mu)^T$ and z_1, z_2.

This simulation is valid by the following lemma.

Lemma 7. The output of **SIM** is statistically close to the real transcript, even if the distinguisher has oracle access to $\mathcal{O}((s_1, s_2), u)$, where $(s_1, s_2) \leftarrow D^2_{R,\sigma}$ is the private key and $u = as_1 + s_2$ is the public-key.

Proof. First, we can assume $A \ne \emptyset$ for the real transcript as by Lemma 5 this is violated negligibly only. Then, by symmetry, j^* for the real transcript is uniformly random over $\{1, \cdots, \mu\}$. By the definition of j^*, we know that z_{1j^*}, z_{2j^*} both belong to $\mathcal{Z}$. In this case, by Lemma 4, $sc + y_{1j^*}, sc + y_{2j^*}$ for the real transcript with given sc satisfying $||sc||_\infty < \sigma n^{1/2}\log^3 n$, are independent and uniformly random over $\mathcal{Z}$. By lemma 3, we conclude that z_{1j^*} and z_{2j^*} are statistically close to uniform over $\mathcal{Z}$ if they belong to $\mathcal{Z}$. On the other hand, when z_{1j^*} and z_{2j^*} are given, v_{j^*} is fixed as $v_{j^*} = az_{1j^*} + z_{2j^*} - uc$. Thus, our simulation of $z_{1j^*}, z_{2j^*}, v_{j^*}$ is statistically close to that in the real transcript. On the other hand, our simulation of y_{1j}, y_{2j}, v_j for $j \ne j^*$ is exactly according to the real distribution. Thus, our simulation is statistically close to the real transcript. This closeness holds (even given adversary view, which includes the responses from $\mathcal{O}_{id}$). Hence, the simulability follows. □

Security. Now we prove the security of our ID scheme, where the attacker needs to generate z_1, z_2 (given challenge c) to pass the verification w.r.t. an aggregated public-key $\overline{u}$. We show that this is unlikely by the ring-SIS assumption.

Lemma 8. Under ring-LWE$_{q,\sigma,2n}$ and ring-SIS$_{3,q,\beta_{t^*}}$ assumptions, our scheme is t^*-secure (with respect to Definition 5), where $\beta_{t^*} = 16\eta_{t^*}\sqrt{n}\log^2 n$ and $\sigma = \Omega(n)$.

Proof. If there exists an adversary $\mathcal{D}$ that breaks our ring-based ID scheme with non-negligible probability ϵ, then we construct an adversary $\mathcal{A}$ that breaks ring-SIS assumption with a non-negligible probability ϵ'. The idea is to make use of $\mathcal{D}$ to construct an algorithm A for the nested forking lemma and then uses the output of the forking algorithm to obtain a solution for ring-SIS problem. Upon a challenge u_1 and a (both uniformly over R_q), $\mathcal{A}$ needs to find short

$$
\begin{array}{lcl}
\boxed{\textbf{Prover}}((s_1,s_2),u|t) & & \boxed{\textbf{Verifier}}(u|t) \\
\mathbf{y}_1,\mathbf{y}_2 \leftarrow \mathcal{Y}^{\mu}, \mathbf{v} = a\mathbf{y}_1 + \mathbf{y}_2 & \xrightarrow{\quad \mathbf{v} \quad} & \\
\underline{b=1,2}: \ \ \mathbf{z}_b = s_b c \cdot \mathbf{1} + \mathbf{y}_b & & \\
A = \{j \mid z_{1j}, z_{2j} \in \mathcal{Z}\},\ j^* \leftarrow A & \xleftarrow{\quad c \quad} & c \leftarrow \mathcal{C} \\
\underline{b=1,2}: \ z_b = z_{bj^*} + \sum_{j\neq j^*} y_{bj} & & \\
 & \xrightarrow{\quad z_1, z_2 \quad} & \underline{b=1,2}: \quad ||z_b||_\infty < \eta_t? \\
 & & \sum_{j=1}^{\mu} v_j \stackrel{?}{=} az_1 + z_2 - uc
\end{array}
$$

Fig. 5. Our Lattice-based ID Scheme (**Note:** Membership checks $c \in \mathcal{C}$ at Prover is important but omitted in the figure; $\mathbf{1}$ is the vector of all 1 of length μ.)

$\alpha_1, \alpha_2, \alpha_3 \in R$ so that $a\alpha_1 + \alpha_2 + u_1\alpha_3 = 0$. Toward this, $\mathcal{A}$ constructs an algorithm $\mathsf{A}((u_1, a), \lambda_1, c; \rho)$ as follows (so $q = 2$ in the forking algorithm), where $\lambda_i, c \leftarrow \mathcal{C}$ and $\overline{u} = \sum_{i=1}^{t} \lambda_i \cdot u_i$ with $u_i \in R_q$ (in the description of A).

Algorithm $\mathsf{A}((u_1, a), \lambda_1, c; \rho)$
  Parse ρ as two parts: $\rho = \rho_0|\rho_1$
  $(st_0, u_2, \cdots, u_t) \leftarrow \mathcal{D}(u_1, a; \rho_0)$
  $\lambda_2, \cdots, \lambda_t \leftarrow \mathcal{C}$ using randomness ρ_1
  $st_1|\mathbf{v} \leftarrow \mathcal{D}(st_0, \lambda_1, \cdots, \lambda_t)$;
  $(z_1, z_2) \leftarrow \mathcal{D}(st_1, c)$;
  If $||z_b||_\infty < \eta_t$ **and** $\sum_{j=1}^{\mu} v_j = az_1 + z_2 - \overline{u}c$, **then**
    $b = 1$;
  else $b = 0$;
  Output $(b, 2b, \{u_i|\lambda_i\}_1^t|\mathbf{v}|z_1|z_2|c|a)$.

From the description of A and the forking algorithm F_A (for the forking lemma), the rewinding in F_A only updates λ_1 and/or c as well as variables affected by (λ_1, c). In terms of forking lemma terminology, we have $(h_1, h_2) = (\lambda_1, c)$ and $I_0 = 1, J_0 = 2$ (for a successful execution; otherwise, A will abort when $I_0 \leq J_0$). Let us now analyze algorithm forking algorithm F_{A}. When four executions are executed successfully (i.e., $b = 1$ for all cases), then the output for each execution will be described as follows.

- *Execution 0.* It outputs $(1, 2, \{u_i|\lambda_i\}_1^t|\mathbf{v}|z_1|z_2|c|a)$. Since it succeeds, $||z_b||_\infty \leq \eta_t$ $(b = 1, 2)$ and
$$\sum_{i=1}^{\mu} v_i = az_1 + z_2 - \overline{u}c. \tag{28}$$
- *Execution 1.* Compared with execution 0, the input only changes c to $\hat{c}$. From the code of A, the output is $(1, 2, \{u_i|\lambda_i\}_1^t|\mathbf{v}|\hat{z}_1|\hat{z}_2|\hat{c}|a)$. Since it succeeds, $||\hat{z}_b||_\infty \leq \eta_t$ $(b = 1, 2)$ and
$$\sum_{i=1}^{\mu} v_i = a\hat{z}_1 + \hat{z}_2 - \overline{u}\hat{c}. \tag{29}$$
- *Execution 2.* Compared with execution 0, the input changes λ_1 to $\bar{\lambda}_1$ and changes c to $\bar{c}$. From the code of A, the output is $(1, 2, \{u_i|\lambda_i\}_2^t|u_1|\bar{\lambda}_1|\mathbf{v}'|\bar{z}_1|\bar{z}_2|\bar{c}|a)$. Since it succeeds, $||\bar{z}_b||_\infty \leq \eta_t$ $(b = 1, 2)$ and
$$\sum_{i=1}^{\mu} v_i' = a\bar{z}_1 + \bar{z}_2 - \overline{u'}\bar{c}, \tag{30}$$
where $\overline{u'} = \bar{\lambda}_1 u_1 + \sum_{i=2}^{t} \lambda_i u_i$.
- *Execution 3.* Compared with execution 0, the input changes λ_1 to $\bar{\lambda}_1$ and changes c to $\underline{c}$. From the code of A, the output is $(1, 2, \{u_i|\lambda_i\}_2^t|u_1|\bar{\lambda}_1|\mathbf{v}'|\underline{z}_1|\underline{z}_2|\underline{c}|a)$. Since it succeeds, $||\underline{z}_b||_\infty \leq \eta_t$ $(b = 1, 2)$ and
$$\sum_{i=1}^{\mu} v_i' = a\underline{z}_1 + \underline{z}_2 - \overline{u'}\underline{c}. \tag{31}$$

From Eqs. (31)(30), $\mathcal{A}$ can derive
$$a(\underline{z}_1 - \bar{z}_1) + (\underline{z}_2 - \bar{z}_2) - \overline{u'}(\underline{c} - \overline{c}) = 0. \tag{32}$$
From Eqs. (29)(28),
$$a(\hat{z}_1 - z_1) + (\hat{z}_2 - z_2) - \overline{u}(\hat{c} - c) = 0. \tag{33}$$
Notice that Eq. (32)$\times(\hat{c} - c)$-Eq. (33)$\times(\underline{c} - \overline{c})$ gives
$$a\alpha_1 + \alpha_2 - u_1\alpha_3 = 0, \tag{34}$$
where
$$\alpha_1 = (\underline{z}_1 - \bar{z}_1)(\hat{c} - c) - (\hat{z}_1 - z_1)(\underline{c} - \overline{c}) \tag{35}$$
$$\alpha_2 = (\underline{z}_2 - \bar{z}_2)(\hat{c} - c) - (\hat{z}_2 - z_2)(\underline{c} - \overline{c}) \tag{36}$$
$$\alpha_3 = (\lambda_1 - \bar{\lambda}_1)(\hat{c} - c)(\underline{c} - \overline{c}). \tag{37}$$

Hence, $(\alpha_1, \alpha_2, -\alpha_3)$ forms a solution to ring-SIS problem with parameter $(a, 1, u)$. It suffices to verify that each α_i is short and also at least one of them is non-zero. For the second condition, it suffices to make sure that the probability for $\alpha_3 = 0$ is small. Notice that by Chinese remainder theorem, $\alpha_3 = 0$ implies $\lambda_1 = \bar{\lambda}_1 \bmod \Phi_1(x)$ or $\underline{c} = \overline{c} \bmod \Phi_1(x)$ or $\hat{c} = c \bmod \Phi_1(x)$. Similarly, this must also hold for modular $\Phi_2(x)$ but it suffices to consider $\Phi_1(x)$ only. Since $\lambda_1, \bar{\lambda}_1, \underline{c}, \overline{c}, \hat{c}$ is uniformly random over $\mathcal{C}$, each of the equality holds with probability $(1 + 2\log n)^{-n/2}$ only and hence $\Pr(\alpha_3 = 0) \leq 3(1 + 2\log n)^{-n/2}$, negligible! For the first condition, notice that $||\hat{c} - c||_\infty \leq 2\log n$ and $||\underline{z}_1 - \overline{z}_1||_\infty \leq 2\eta_t$. Further, the constant term of $(\hat{c} - c)(\underline{z}_1 - \overline{z}_1)$ is
$$(\hat{c} - c)[0] \cdot (\underline{z}_1 - \overline{z}_1)[0] - \sum_{k=1}^{\frac{n}{2}-1} (\hat{c} - c)[k] \cdot (\underline{z}_1 - \overline{z}_1)[n - k]$$
which, by Heoffding inequality, has an absolute value at most $\sqrt{n/2}\log n \cdot 8\eta_t \log n \leq 8\eta_t\sqrt{n}\log^2 n$, with probability at least $1 - e^{-\Omega(\log^2 n)}$. The constant term of $(\hat{z}_1 - z_1)(\underline{c} - \overline{c})$ is similar. Hence, $|\alpha_1[0]| \leq 16\eta_t\sqrt{n}\log^2 n$, with probability at least $1 - e^{-\Omega(\log^2 n)}$. The general case of $\alpha_1[i]$ is similar. Hence, $||\alpha_1||_\infty \leq 16\eta_t\sqrt{n}\log^2 n$ with probability

$1 - e^{-\Omega(\log^2 n)}$. Similarly, $||\alpha_2||_\infty$ has the same property. We can use the above proof technique to show that $||(\hat{c} - c)(\underline{c} - \overline{c})||_\infty \leq 8\log n \cdot \sqrt{n}\log^2 n$ with probability $1 - e^{-\Omega(\log^2 n)}$. Since $\lambda_1, \overline{\lambda}_1$ is uniformly random over $\mathcal{C}$, using the same technique, we have $||\alpha_3||_\infty \leq \sqrt{n}\log n \cdot 32\sqrt{n}\log^4 n = 32n\log^5 n$, with probability $1 - e^{-\Omega(\log^2 n)}$. Thus, we find a ring-SIS solution $(\alpha_1, \alpha_2, -\alpha_3)$ of length at most $16\eta_t\sqrt{n}\log^2 n$. Assume that the probability that $\mathcal{D}$ succeeds in one execution is $\hat{\epsilon}$. Then, by forking lemma, it succeeds in four executions with probability $\hat{\epsilon}^4 - 3(1 + 2\log n)^{-n/2}$. This implies that $\mathcal{A}$ breaks the ring-SIS assumption with probability at least $\hat{\epsilon}^4 - 3(1+2\log n)^{-n/2} - e^{-\Omega(\log^2 n)}$.

Finally, notice that the input u_1 is uniformly random over R_q while in our ID scheme $u_1 = as_1 + s_2$ for $s_1, s_2 \leftarrow D_{R,\sigma}$. However, under ring-LWE assumption, it is immediate that $\hat{\epsilon} \geq \epsilon - \mathsf{negl}(n)$. Hence, $\mathcal{A}$ can succeed with probability at least $\epsilon^4 - \mathsf{negl}(n)$, this contradicts the assumption of ring-SIS. □

Linearity. Let $\mathcal{SK} = \mathcal{RSP} = (R_q, R_q), \mathcal{CMT} = R_q^\mu, \mathcal{PK} = R_q, \mathcal{R} = R_q$. We now verifies the linearity.

i. Obviously, $\mathcal{SK}$ is a $\mathcal{R}$-module under the operation $\bullet$: for $(s_1, s_2) \in \mathcal{SK}$ and $c \in \mathcal{R}$, $c \bullet (s_1, s_2) = (cs_1, cs_2)$, where cs_1 and cs_2 are multiplications in R_q. Other cases are similar.

ii. If $(s_{1i}, s_{2i}) \in \mathcal{SK}$ and $\lambda_i \in \mathcal{C}$ for $i = 1, \cdots, t$, then $\sum_{i=1}^t (\lambda_i s_{1i}, \lambda_i s_{2i}) = (\sum_{i=1}^t \lambda_i s_{1i}, \sum_{i=1}^t \lambda_i s_{2i})$ is obviously the private key of $\sum_{i=1}^t \lambda_i \cdot (as_{1i} + s_{2i}) = a(\sum_{i=1}^t \lambda_i s_{1i}) + (\sum_{i=1}^t \lambda_i s_{2i})$. However, we emphasize that this key is not necessarily short. But for randomly generated (pk_i, sk_i, λ_i)'s, Lemma 9 implicitly implies that the aggregated private key has length at most $2\sqrt{nt}\sigma\log^3 n$ (except for probability $e^{-\Omega(\log^2 n)}$); see $\max_v |S_v|$ with $|S_v|$ given in the proof of Lemma 9).

iii. If $\{(\mathbf{v}_i, c, z_{1i}, z_{2i})\}_{i=1}^t$ are *honestly* generated accepting transcripts w.r.t the honestly public/private key pairs $\{(u_i, (s_{1i}, s_{2i}))\}_i$, then

$$\sum_{j=1}^{\mu} v_{ij} = az_{1i} + z_{2i} - u_i c. \tag{38}$$

Together with Lemma 9 below, for $h_1, \cdots, h_t \leftarrow \mathcal{C}$, $(\sum_{i=1}^t h_i \mathbf{v}_i, c, \sum_{i=1}^t h_i z_{1i}, \sum_{i=1}^t h_i z_{2i})$ satisfies (except for probability $e^{-\Omega(\log^2 n)}$)

$$\begin{aligned} ||\sum_{i=1}^t h_i z_{1i}||_\infty \leq& \eta_t, \quad ||\sum_{i=1}^t h_i z_{1i}||_\infty \leq \eta_t, \\ \sum_{j=1}^{\mu}(\sum_{i=1}^t h_i v_{ij}) =& a(\sum_{i=1}^t h_i z_{1i}) + (\sum_{i=1}^t h_i z_{2i}) - (\sum_{i=1}^t h_i u_i)c, \end{aligned}$$

where $\eta_t = 5\sigma n^2\sqrt{t\mu}\log^6 n$. The linearity follows.

Lemma 9. Fix integer $t \geq 2$ and $\sigma \geq \omega(\log n)$. Assume $s_i \leftarrow D_{R,\sigma}, h_i \leftarrow \mathcal{C}, y_{ij} \leftarrow \mathcal{Y}$ for $i \in [t], j \in [\mu], c \leftarrow \mathcal{C}$. Let

$$Z = \sum_{i=1}^t h_i(s_i c + \sum_{j=1}^{\mu} y_{ij}). \tag{39}$$

Then, $||Z||_\infty \leq \eta_t$ with probability $1 - e^{-\Omega(\log^2 n)}$.

Proof. Notice

$$Z[0] = \sum_{v=0}^{n-1} S_v \cdot c[v] - \sum_{i=1}^t \sum_{k=0}^{n-1} h_i[n-k] \cdot Y_{ik},$$

where $Y_{ik} = \sum_{j=1}^{\mu} y_{ij}[k]$, $h_i[n] \overset{def}{=} -h_i[0]$ and

$$S_v = \sum_{i=1}^t \sum_{k=0}^{n-1} h_i[n-k]s_i[k-v].$$

By [24, Lemma 4.2], $||s_i||_\infty \leq \sigma\log n$, except for probability $e^{-\Omega(\log^2 n)}$. When this is satisfied, terms $h_i[n-k]s_i[k-v]$ in S_v are independent random variables in the range $[-\sigma\log^2 n, \sigma\log^2 n]$. By Heoffding inequality, $|S_v| \leq 2\sqrt{nt}\sigma\log^3 n$, except for probability $e^{-\Omega(\log^2 n)}$. Since $y_{ij}[k]$ is uniformly random over $[-\sigma n^{1.5}\log^3 n, \sigma n^{1.5}\log^3 n]$, by Heoffding inequality, $|Y_{ik}| \leq 2\sigma\sqrt{\mu}n^{1.5}\log^4 n$, except for a probability $e^{-\log^2 n}$. Assuming these inequalities for S_v and Y_{ik}, we know that from Heoffding inequality again,

$$\begin{aligned} &|\sum_{v=1}^{n-1} S_v \cdot c[v]| \leq 4\sigma n\sqrt{t}\log^5 n \\ &|\sum_{i,k} h_i[n-k] \cdot Y_{ik}| \leq \sqrt{nt}\log n \cdot 4\sigma\sqrt{\mu}n^{1.5}\log^5 n, \end{aligned}$$

except for probability $e^{-\Omega(\log^2 n)}$. Hence, we conclude that $|Z[0]| \leq 5\sigma n^2\sqrt{t\mu}\log^6 n$, except for $e^{-\Omega(\log^2 n)}$. We can similarly bound $Z[i]$ for $i \geq 1$ and so $||Z||_\infty \leq 5\sigma n^2\sqrt{t\mu}\log^6 n$, except for probability $e^{-\Omega(\log^2 n)}$. □

7.2.4 Key-and-Signature Compact Multi-signature Scheme from our ID scheme.

With the simulability, linearity and security for our ID, we can use our compiler to convert it into a secure multi-signature. We now describe this scheme as follows.

Let (s_{1i}, s_{2i}) be the private key of public-key $u_i = as_{1i} + s_{2i}$ for $i = 1, \cdots, t$. If the users of $u_1, \cdots, u_t$ want to jointly sign M, they compute the aggregated public-key $\overline{u}|t$ and execute the protocol as follows, where $H_0, H_1 : \{0,1\}^* \to \mathcal{C}$ and we define $\overline{w} = \sum_{i=1}^t H_0(u_i, U)w_i$ for any list of variables $w_1, \cdots, w_t$ in the description and $U = (u_1, \cdots, u_t)$ (e.g., $\overline{\mathbf{v}} = \sum_{i=1}^t H_0(u_i, U) \cdot \mathbf{v}_i$).

- **R-1**. User generates $\mathbf{y}_{1i}, \mathbf{y}_{2i} \leftarrow \mathcal{Y}^\mu$, computes $\mathbf{v}_i = a\mathbf{y}_{1i} + \mathbf{y}_{2i}$ and sends $H_0(\mathbf{v}_i|u_i)$ to other users.
- **R-2**. Upon receiving all $r_j, j = 1, \cdots, t$, user i sends $\mathbf{v}_i$ to other users.
- **R-3**. Upon all $\mathbf{v}_j$, user i verifies its consistency with r_j. If verification fails, it rejects; otherwise, it computes $\overline{\mathbf{v}}$ and $c = H_1(\overline{u}|\overline{\mathbf{v}}|M)$ as well as the response (z_{1i}, z_{2i}) for challenge c in the ID scheme with committing message $\mathbf{v}_i$.
- *output*. After receiving (z_{1j}, z_{2j}) for $j \in [t]$, user i computes multi-signature $(\overline{z}_1, \overline{z}_2, \overline{\mathbf{v}})$. The aggregated public-key is $\overline{u}|t$.
- *Verify*. Upon $(\overline{z}_1, \overline{z}_2, \overline{\mathbf{v}})$, it verifies the following with $\overline{u}|t$ and accepts only if it is valid:

$$||\overline{z}_1||_\infty \leq \eta_t, \quad ||\overline{z}_2||_\infty \leq \eta_t, \tag{40}$$

$$\sum_{j=1}^{\mu} \overline{v}_j = a\overline{z}_1 + \overline{z}_2 - \overline{u}c, \tag{41}$$

where $\eta_t = 5\sigma n^2\sqrt{t\mu}\log^6 n$. Denote this multi-signature scheme by RLWE-MultiSig. From our compiler and the properties of our ID scheme, we obtain the following.

Corollary 3. Let $\eta_{t^*} = 5\sigma n^2\sqrt{t^*\mu}\log^6 n$, $\sigma = \Omega(n)$ and $\beta_{t^*} = 16\eta_{t^*}\sqrt{n}\log^2 n$. Under Ring-LWE$_{q,\sigma,2n}$ and Ring-SIS$_{3,q,\beta_{t^*}}$ assumptions, RLWE-MultiSig is t^*-EU-CMA secure. Especially, if the assumptions hold for $t^* = 2^{\sqrt[4]{n}}$, then RLWE-MultiSig is EU-CMA secure.

Remark. As the best algorithm [15], [33] can only solve ring-SIS$_{q,3,\beta}$ with $\beta = 2^{\tilde{O}(\sqrt{n})}$, it is safe to assume ring-SIS$_{q,3,\beta}$ with any polynomial β. If the assumption is sound for $\beta = 2^{\sqrt[4]{n}}$, our multi-signature scheme is EU-CMA secure, as t^* can only be polynomial for a PPT adversary.

8 CONCLUSION

In this paper, we proposed a compiler that converts a type of identification scheme to a key-and-signature compact multi-signature. This special type of ID owns a linear property. The aggregated public-key and multi-signature are of size both independent of the number of signers. We formulated this compiler through linear ID via the language of $\mathcal{R}$-module and proved the security through a new forking lemma called nested forking lemma. Under our compiler, the compact multi-signature problem has been reduced from a multi-party problem to a two-party problem. We realized our compiler with Schnorr ID scheme and a new lattice-based scheme. Our lattice multi-signature is the first of its kind that is key-and-signature compact without a restart in the signing process.

REFERENCES

[1] Michel Abdalla, Pierre-Alain Fouque, Vadim Lyubashevsky, Mehdi Tibouchi: Tightly-Secure Signatures from Lossy Identification Schemes. EUROCRYPT 2012: 572-590

[2] H. K. Alper and J. Burdges. Two-round trip schnorr multi-signatures via delinearized witnesses. In T. Malkin and C. Peikert, editors, CRYPTO 2021, Part I, volume 12825 of LNCS, pages 157-188, Virtual Event, Aug. 2021. Springer, Heidelberg.

[3] Ali Bagherzandi, Jung Hee Cheon and Stanislaw Jarecki, Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. *CCS 2008*, pp. 449-458, 2008.

[4] Mihir Bellare, Wei Dai, Chain Reductions for Multi-signatures and the HBMS Scheme. *ASIACRYPT* 2021, Part IV: 650-678

[5] Mihir Bellare, Adriana Palacio, GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks. CRYPTO 2002: 162-177.

[6] M. Bellare and G. Neven, Identity-Based Multi-signatures from RSA, *CT-RSA 2007*, M. Abe (Ed.), LNCS 4377, pp. 145162, 2007.

[7] Mihir Bellare, Phillip Rogaway: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. *CCS 1993*: 62-73, 1993.

[8] Mihir Bellare, Gregory Neven: Multi-signatures in the plain public-Key model and a general forking lemma. CCS 2006: 390-399

[9] Ian F. Blake, Shuhong Gao and Ronald C. Mullin, Explicit Factorization of $x^{2^k}+1$ over F_p with Prime $p \equiv 3$ mod 4. *Appl. Algebra Eng. Commun. Comput.* 4: 89-94 (1993)

[10] Florian Böhl, Dennis Hofheinz, Tibor Jager, Jessica Koch, Christoph Striecks: Confined Guessing: New Signatures From Standard Assumptions. J. Cryptol. 28(1): 176-208 (2015)

[11] Alexandra Boldyreva, Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. Public Key Cryptography 2003: 31-46.

[12] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In E. Biham, editor, EUROCRYPT 2003, volume 2656 of LNCS, pages 416432. Springer-Verlag, 2003.

[13] Dan Boneh, Manu Drijvers, Gregory Neven: Compact Multi-signatures for Smaller Blockchains. ASIACRYPT (2) 2018: 435-464

[14] Cecilia Boschini, Akira Takahashi, and Mehdi Tibouchi. Musig-L: Lattice-based multi-signature with single-round online phase, *CRYPTO'22*.

[15] Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. Short stickelberger class relations and application to ideal-svp. *Eurocrypt 2017*.

[16] Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n-out-of-n and multisignatures and trapdoor commitment from lattices. *PKC 2021*, LNCS 12710, pages 99-130, 2021.

[17] Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, Igors Stepanovs, On the Security of Two-Round Multi-Signatures. *IEEE Symposium on Security and Privacy 2019*, pp. 1084-1101, IEEE, 2019.

[18] Léo Ducas and Alain Durmus. Ring-lwe in polynomial rings. In *PKC 2012*, LNCS 7293, pages 34–51. Springer, 2012.

[19] Rachid El Bansarkhani and Jan Sturm. An efficient lattice-based multisignature scheme with applications to bitcoins, *CANS'16*, pages 140-155.

[20] Nils Fleischhacker, Mark Simkin, Zhenfei Zhang: Squirrel: Efficient Synchronized Multi-Signatures from Lattices. *CCS 2022*, papges 1109-1123, 2022.

[21] Masayuki Fukumitsu and Shingo Hasegawa. A tightly-secure lattice-based multisignature. *The 6th Asia Public-Key Cryptography Workshop 2019*, page 3-11, 2019.

[22] Masayuki Fukumitsu and Shingo Hasegawa. A lattice-based provably secure multisignature scheme in quantum random oracle model, *ProvSec 2020*.

[23] C. Gentry and Z. Ramzan. Identity-based aggregate signatures. In M. Yung, editor, PKC 2006, volume 3958 of LNCS, pages 257-273. Springer-Verlag, 2006.

[24] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. *STOC'08*, pp. 197-206, 2008.

[25] L. Harn. Group-oriented (t, n) threshold digital signature scheme and digital multisignature. *IEE Proceedings-Computers and Digital Techniques*, 141(5):307-313, 1994.

[26] K. Itakura and K. akamura, A public-key cryptosystem suitable for digital multisignatures. *NEC Research & Development*, 71:1-8, 1983.

[27] Meenakshi Kansal, Amit Kumar Singh, Ratna Dutta, Efficient Multi-Signature Scheme Using Lattice. *Comput. J.* 65(9): 2421-2429 (2022)

[28] Meenakshi Kansal and Ratna Dutta, Round Optimal Secure Multisignature Schemes from Lattice with Public Key Aggregation and Signature Compression. *AFRICACRYPT 2020*, pages 281-300, 2020.

[29] Serge Lang, Algebra, GTM 211, Springer-Verlag, 2002.

[30] C. M. Li, T. Hwang, and N.-Y. Lee. Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders. In A. D. Santis, editor, EUROCRYPT'94, volume 950 of LNCS, pages 194-204. Springer, Heidelberg, May 1995

[31] Zi-Yuan Liu, Yi-Fan Tseng, and Raylin Tso. Cryptanalysis of a round optimal lattice-based multisignature scheme. Cryptology ePrint Archive, Report 2020/1172, 2020.

[32] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, Brent Waters: Sequential Aggregate Signatures and Multisignatures Without Random Oracles. EUROCRYPT 2006: 465-485

[33] Vadim Lyubashevsky and Daniele Micciancio, Generalized Compact Knapsacks Are Collision Resistant. *ICALP 2006*, part 2, pages 144-155, 2006.

[34] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43:1–43:35, 2013.

[35] V. Lyubashevsky, C. Peikert, and O. Regev. A toolkit for ring-LWE cryptography. *EUROCRYPT'13*, pages 35-54, 2013.

[36] Changshe Ma, Jian Weng, Yingjiu Li, Robert H. Deng: Efficient discrete logarithm based multi-signature scheme in the plain public key model. Des. Codes Cryptogr. 54(2): 121-133 (2010)

[37] Changshe Ma, Mei Jiang, Practical Lattice-Based Multisignature Schemes for Blockchains. *IEEE Access* 7: 179765-179778 (2019)

[38] Maxwell, G., Poelstra, A., Seurin, Y., Wuille, P.: Simple schnorr multi-signatures with applications to bitcoin. Cryptology ePrint Archive, Report 2018/068 (2018), https://eprint.iacr.org/2018/068/20180118:124757

[39] Silvio Micali, Kazuo Ohta, Leonid Reyzin: Accountable-subgroup multisignatures: extended abstract. CCS 2001: 245-254.

[40] D. Micciancio and O. Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1): 267-302, 2007.

[41] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, 2008. Available at http://bitcoin.org/bitcoin.pdf

[42] J. Nick, T. Ruffing, and Y. Seurin. MuSig2: Simple two-round Schnorr multi-signatures. *CRYPTO 2021*, Part I, LNCS 12825, pp. 189-221, Springer, 2021.

[43] J. Nick, T. Ruffing, Y. Seurin, and P. Wuille. MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. In J. Ligatti, X. Ou, J. Katz, and G. Vigna, editors, ACM CCS 2020, pages 1717-1731. ACM Press, Nov. 2020.

[44] K. Ohta and T. Okamoto. A digital multisignature scheme based on the Fiat-Shamir scheme. In H. Imai, R. L. Rivest, and T. Matsumoto, editors, ASIACRYPT91, volume 739 of LNCS, pages 139-148. Springer, Heidelberg, Nov. 1993.

[45] Chris Peikert, Alon Rosen, Efficient Collision-Resistant Hashing from Worst-Case Assumptions on Cyclic Lattices. *TCC 2006*, pages 145-166, 2006.

[46] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. Journal of Cryptology, 13(3):361-396, 2000.

[47] Ronald L. Rivest, Adi Shamir, Leonard M. Adleman: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM* 21(2): 120-126, 1978.

[48] C. P. Schnorr, Efficient Signature Generation by Smart Cards, *Journal of Cryptology*, vol 4, no. 3, pp. 161-174, 1991.

[49] Damien Stehlé and Ron Steinfeld, Making NTRU as secure as worst-case problems over ideal lattices, *EUROCRYPT 2011*, K. G. Paterson (ed.), LNCS 6632, pp. 27-47, 2011.

[50] Damien Stehlé and Ron Steinfeld, Making NTRUEncrypt and NTRUSign as secure as standard worst-case problems over ideal lattices, *Cryptology ePrint Archive*, Report 2013/004, 2013, http://eprint.iacr.org/. Full version of [49].

[51] 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. *IEEE Symposium on Security and Privacy 2016*, pp. 526-545. IEEE Computer Society Press, May 2016.