2021 Ninth International Symposium on Computing and Networking Workshops (CANDARW) | 978-1-6654-2835-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/CANDARW53999.2021.00068

Single-round Lattice-based Multisignatures

Kittiphon Phalakarn
Dept. of Electrical and Computer Eng.
University of Waterloo
kphalakarn@uwaterloo.ca

Vorapong Suppakitpaisarn
Dept. of Computer Science
The University of Tokyo
vorapong@is.s.u-tokyo.ac.jp

M. Anwar Hasan
Dept. of Electrical and Computer Eng.
University of Waterloo
ahasan@uwaterloo.ca

***Abstract*—This work presents a provably-secure lattice-based multisignature scheme which requires only a single round of communication, whereas the existing works need two or three rounds. The reduction in the number of rounds for the proposed scheme is achieved by utilizing lattice trapdoors. In order to generate multisignatures securely, our scheme however requires an honest centralized server that maintains the trapdoor of a shared matrix used in the scheme.**

***Index Terms*—multisignatures, trapdoors, lattice-based cryptography, post-quantum cryptography**

I. Introduction

Multisignatures [1] are proofs that a group of N users have agreed on and signed the same message. We can have each user sign the message independently, but that results in N signatures and we have to perform the verification N times. Using multisignature schemes, we require only one signature for the whole group and one execution of the verification algorithm. Multisignatures are very useful in blockchains and cryptocurrencies. For example, by having one signature for the whole group in a transaction, the amount of data written on the blockchain and sent in the network is reduced.

Currently, there exist a variety of multisignature schemes based on the hardness of integer factorization (i.e., RSA) [2] and discrete logarithm [3]. As the cryptographic schemes based on any of these hard problems can be broken by large quantum computers, many researchers are turning to post-quantum cryptography. Lattice-based cryptosystem [4] is considered quantum-safe and has received much interest in the research community due to its moderate-sized cipherthext/signatures and simple computations. Nevertheless, lattice-based multisignatures are still not practical for real-world use and have rooms for improvements.

The first lattice-based multisignature scheme was proposed by Bansarkhani and Sturm [5]. It is a three-round protocol that requires signers to broadcast their information. The protocol was improved by subsequent works [6], [7] and the recent scheme presented by Damgård et al. [8] has reduced the number of rounds to two. All of the aforementioned schemes are derived from the lattice-based signatures by Lyubashevsky et al. [9]–[11] based on the Fiat-Shamir transform.

Here, we point out that broadcasting limits the scalability of the schemes. It is specially expensive in blockchain applications with up to 100 signers [12]. To the best of our knowledge, no secure single-round lattice-based multisignature scheme without broadcasting has been proposed. There was an attempt to do so [13], but it was later proved to be insecure [14].

A. Contributions

In this work, we propose a *single-round non-broadcasting* lattice-based multisignature scheme based on the Fiat-Shamir transform. In order to restrict the number of rounds to one, we utilize lattice trapdoors by Micciancio and Peikert [15]. For easier understanding, we start by presenting a signature scheme and then extend it to a multisignature one. Security proofs based on the general forking lemma are included for both of our schemes.

Our proposed multisignature scheme has two advantages: (i) no broadcasting is required, and (ii) if any signer has to restart the protocol, other signers do not have to restart their parts, unlike the existing schemes where all signers are affected. However, an honest centralized server maintaining the trapdoor of a shared matrix is required in order to work securely.

II. Preliminaries

A. Signature Scheme

A signature scheme Sig consists of three algorithms:

- $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Sig.KeyGen}(1^\lambda)$: The key generation algorithm produces a key pair from a security parameter λ.
- $\sigma \xleftarrow{\$} \mathsf{Sig.Sign}(\mathsf{sk}, \mu)$: The signing algorithm uses a secret key sk to create a signature σ for a message μ.
- $b \leftarrow \mathsf{Sig.Verify}(\mathsf{pk}, \mu, \sigma)$: The verification algorithm outputs $b = 1$ if a signature σ is valid for a public key pk and a message μ. Otherwise, it outputs $b = 0$.

A signature scheme is *correct* if all signatures created by the signing algorithm pass the verification, i.e., for all $\lambda \in \mathbb{N}$ and $\mu \in \{0,1\}^*$, it holds, over all possible $(\mathsf{pk}, \mathsf{sk})$ generated from $\mathsf{Sig.KeyGen}(1^\lambda)$, that $\Pr[\mathsf{Sig.Verify}(\mathsf{pk}, \mu, \mathsf{Sig.Sign}(\mathsf{sk}, \mu)) = 1] = 1$.

A signature scheme is *existentially unforgeable under one-per-message chosen-message attacks* [16] if for any probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$, the advantage $\mathsf{Adv}^{\text{euf-cma}_1}_{\mathsf{Sig},\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\text{euf-cma}_1}_{\mathsf{Sig},\mathcal{A}}(\lambda)]$ is $\mathrm{negl}(\lambda)$.

B. Multisignature Scheme

In this setting, there are N signers agreeing to sign a message μ. A multisignature scheme MS consists of five algorithms:

- $(\mathsf{pk}_i, \mathsf{sk}_i) \xleftarrow{\$} \mathsf{MS.KeyGen}(1^\lambda)$: The key generation algorithm produces a key pair for each signer.
- $\sigma_i \xleftarrow{\$} \mathsf{MS.Sign}(\mathsf{sk}_i, \mathsf{PK}, \mu)$: The signing algorithm creates a signature σ_i of the i-th signer where $\mathsf{PK} = \{\mathsf{pk}_j\}_{j=1}^N$.

978-1-6654-2835-4/21/$31.00 ©2021 IEEE
DOI 10.1109/CANDARW53999.2021.00068

Experiment $\mathsf{Exp}^{\text{euf-cma}_1}_{\mathsf{Sig},\mathcal{A}}(\lambda)$

$\mathcal{M} \leftarrow \{\}$
$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Sig.KeyGen}(1^\lambda)$
$(\mu^*, \sigma^*) \xleftarrow{\$} \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(1^\lambda, \mathsf{pk})$
if $\mathsf{Sig.Verify}(\mathsf{pk}, \mu^*, \sigma^*) = 1$ **and** $\mu^* \notin \mathcal{M}$ **then**
  return 1
else return 0

$\mathsf{Sign}(\mathsf{sk}, \mu)$

if $\mu \in \mathcal{M}$ **then return** $\perp$
$\mathcal{M} \leftarrow \mathcal{M} \cup \{\mu\}$
return $\mathsf{Sig.Sign}(\mathsf{sk}, \mu)$

Experiment $\mathsf{Exp}^{\text{euf-cma}_1}_{\mathsf{MS},\mathcal{A}}(\lambda)$

$\mathcal{M} \leftarrow \{\}$
for $i = 1$ **to** N **do** $(\mathsf{pk}_i, \mathsf{sk}_i) \xleftarrow{\$} \mathsf{MS.KeyGen}(1^\lambda)$
$(\mu^*, \Sigma^*) \xleftarrow{\$} \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}_1,\mathsf{PK},\cdot)}(1^\lambda, \mathsf{PK}, \{\mathsf{sk}_j\}_{j=2}^N)$
if $\mathsf{MS.MVerify}(\mathsf{PK}, \mu^*, \Sigma^*) = 1$ **and** $\mu^* \notin \mathcal{M}$ **then**
  return 1
else return 0

$\mathsf{Sign}(\mathsf{sk}_1, \mathsf{PK}, \mu)$

if $\mu \in \mathcal{M}$ **then return** $\perp$
$\mathcal{M} \leftarrow \mathcal{M} \cup \{\mu\}$
return $\mathsf{MS.Sign}(\mathsf{sk}_1, \mathsf{PK}, \mu)$

- $b_i \leftarrow \mathsf{MS.Verify}(\mathsf{pk}_i, \mathsf{PK}, \mu, \sigma_i)$: The verification algorithm outputs $b_i = 1$ if σ_i is valid for pk_i and μ.
- $\Sigma \xleftarrow{\$} \mathsf{MS.Agg}(\mathsf{PK}, \mu, \{\sigma_j\}_{j=1}^N)$: The aggregation algorithm creates a multisignature Σ from $\{\sigma_j\}_{j=1}^N$.
- $b \leftarrow \mathsf{MS.MVerify}(\mathsf{PK}, \mu, \Sigma)$: The multisignature verification algorithm outputs $b = 1$ if Σ is valid for PK and μ.

A multisignature scheme is *correct* if all multisignatures created by the algorithms pass the verification, i.e., for all $\lambda, N \in \mathbb{N}$ and $\mu \in \{0,1\}^*$, it holds, over all possible $(\mathsf{pk}_i, \mathsf{sk}_i)$ generated from $\mathsf{MS.KeyGen}(1^\lambda)$, that $\Pr[\mathsf{MS.MVerify}(\mathsf{PK}, \mu, \Sigma) = 1] = 1$. where Σ is generated by $\mathsf{MS.Agg}(\mathsf{PK}, \mu, \{\mathsf{MS.Sign}(\mathsf{sk}_j, \mathsf{PK}, \mu)\}_{j=1}^N)$.

A multisignature scheme is *existentially unforgeable under one-per-message chosen-message attacks* if for any PPT algorithm $\mathcal{A}$, the advantage $\mathsf{Adv}^{\text{euf-cma}_1}_{\mathsf{MS},\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\text{euf-cma}_1}_{\mathsf{MS},\mathcal{A}}(\lambda)]$ is $\text{negl}(\lambda)$. Generally speaking, the multisignature scheme is secure if the adversary cannot forge a valid multisignature when there is at least one honest signer.

C. Lattices

A lattice is a set of multi-dimensional points with a periodic structure defined as follows.

Definition 1. *The* lattice *generated from ℓ linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_\ell \in \mathbb{R}^\ell$ is defined as $\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_\ell) = \left\{\sum_{i=1}^\ell x_i \mathbf{b}_i : x_i \in \mathbb{Z}, 1 \le i \le \ell\right\}$, or equivalently $\mathcal{L}(\mathbf{B}) = \left\{\mathbf{Bx} : \mathbf{x} \in \mathbb{Z}^\ell\right\}$ where $\mathbf{B} = [\mathbf{b}_1 | \cdots | \mathbf{b}_\ell]$.*

One particular type of lattices that is of interest is *q-ary lattices* (i.e., lattices $\mathcal{L}$ where $q\mathbb{Z}^\ell \subseteq \mathcal{L} \subseteq \mathbb{Z}^\ell$) for a prime number q. It is not hard to see that for any $\mathbf{v} \in \mathbb{Z}^\ell$, $\mathbf{v} \in \mathcal{L}$ if and only if $\mathbf{v} \bmod q \in \mathcal{L}$. This property is desirable as it is sufficient to work with q-ary lattices under mod-q arithmetic. In the remainder of this work, all operations involving matrices and vectors are assumed to be under mod-q arithmetic. To construct a q-ary lattice, one can follow Definition 2.

Definition 2. *Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ for some prime number q and some $n, m \in \mathbb{Z}$ with $m \ge n$, we define an* m-dimensional q-ary lattice $\Lambda^\perp(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{Ax} = \mathbf{0}\}$.

D. The Short Integer Solution (SIS) Problem

For a vector $\mathbf{v} = [v_1 \cdots v_m]^\top \in \mathbb{Z}^m$, the size ($\ell_2$-norm) of $\mathbf{v}$ is denoted by $\|\mathbf{v}\| = \sqrt{v_1^2 + \ldots + v_m^2}$. The SIS problem can be defined as follows.

Definition 3 ($\text{SIS}_{q,n,m,\beta}$ problem [10])**.** *Given a random matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, find a vector $\mathbf{v} \in \mathbb{Z}^m \setminus \{\mathbf{0}\}$ such that $\mathbf{Av} = \mathbf{0}$ and $\|\mathbf{v}\| \le \beta$.*

From the analysis in [4], [10], [17], the SIS problem is expected not to be solved in polynomial time when $\beta < \min\left(q, 2^{2\sqrt{n \log q \log \delta}}\right)$ where δ is a constant due to the state-of-the-art lattice reduction algorithms. It is expected that δ is not less than 1.01.

E. Lattice Trapdoors

First, we define the following notations

- $\Lambda^\perp_{\mathbf{u}}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{Ax} = \mathbf{u}\}$.
- D_s denotes the discrete Gaussian distribution with a standard deviation s.
- $D_{\Lambda^\perp_{\mathbf{u}}(\mathbf{A}),s}$ denotes D_s restricted to have support $\Lambda^\perp_{\mathbf{u}}(\mathbf{A})$.

In [15], Micciancio and Peikert proposed a construction of trapdoors for lattices and an algorithm for Gaussian preimage sampling for the function $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{Ax}$ as follows.

1) $(\mathbf{A}, \mathbf{R}) \xleftarrow{\$} \mathsf{GenTrap}(1^\lambda)$: The trapdoor generation algorithm outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ with its trapdoor $\mathbf{R}$ for a security parameter λ.
2) $\mathbf{x} \xleftarrow{\$} \mathsf{SampleD}(\mathbf{R}, \mathbf{A}, \mathbf{u}, s)$: The Gaussian preimage sampling algorithm outputs a vector drawn from a distribution close to $D_{\Lambda^\perp_{\mathbf{u}}(\mathbf{A}),s}$.

The properties of the above algorithms are stated next.

Theorem 4 ([15])**.** *For suitable values of parameters,*

1) *The distribution of $\mathbf{A}$ from $\mathsf{GenTrap}(1^\lambda)$ is $\text{negl}(n)$-far from uniform.*
2) *For any $\mathbf{u} \in \mathbb{Z}_q^n$, the algorithm $\mathsf{SampleD}(\mathbf{R}, \mathbf{A}, \mathbf{u}, s)$ samples from a distribution within $\text{negl}(n)$ statistical distance of $D_{\Lambda^\perp_{\mathbf{u}}(\mathbf{A}),s}$.*

Experiment $\mathsf{Exp}^{\text{break}}_{\mathcal{A},\mathcal{P},\mathcal{H},t}$

$p \xleftarrow{\$} \mathcal{P}$
$\alpha \xleftarrow{\$} \{0,1\}^c$
$h_1, \ldots, h_t \xleftarrow{\$} \mathcal{H}$
$(s,k) \xleftarrow{\$} \mathcal{A}(p, h_1, \ldots, h_t, \alpha)$
return k

$\mathcal{F}(p)$

$\alpha \xleftarrow{\$} \{0,1\}^c$
$h_1, \ldots, h_t \xleftarrow{\$} \mathcal{H}$
$(s,k) \xleftarrow{\$} \mathcal{A}(p, h_1, \ldots, h_t, \alpha)$
$h'_k, \ldots, h'_t \xleftarrow{\$} \mathcal{H}$
$(s', k') \xleftarrow{\$} \mathcal{A}(p, h_1, \ldots, h_{k-1}, h'_k, \ldots, h'_t, \alpha)$
if $k = k'$ **and** $k \in [t]$ **and** $h_k \neq h_{k'}$ **then**
  return $(1, s, s')$
else return $(0, \perp, \perp)$

Experiment $\mathsf{Exp}^{\text{fork}}_{\mathcal{F},\mathcal{A},\mathcal{P},\mathcal{H},t}$

$p \xleftarrow{\$} \mathcal{P}$
$(b, s, s') \xleftarrow{\$} \mathcal{F}(p)$
return b

Fig. 1. General forking lemma.

$\mathsf{Sig.KeyGen}(1^\lambda)$

$(\mathbf{A}, \mathbf{R}) \xleftarrow{\$} \mathsf{GenTrap}(1^\lambda)$
$\mathbf{S} \xleftarrow{\$} \{-d, \ldots, d\}^{m \times k}$
$\mathbf{T} \leftarrow \mathbf{AS}$
$\mathsf{pk} \leftarrow (\mathbf{A}, \mathbf{T})$
$\mathsf{sk} \leftarrow (\mathbf{R}, \mathbf{S})$
return $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{Sig.Sign}(\mathsf{sk}, \mu)$

$\mathbf{r} \leftarrow \mathsf{RO}_1(\mathsf{pk}, \mu)$
$\mathbf{c} \leftarrow \mathsf{RO}_2(\mathbf{r}, \mu)$
$\mathbf{u} \leftarrow \mathbf{Tc} + \mathbf{r}$
$\mathbf{z} \xleftarrow{\$} \mathsf{SampleD}(\mathbf{R}, \mathbf{A}, \mathbf{u}, s)$
$\sigma \leftarrow \mathbf{z}$
return σ

$\mathsf{Sig.Verify}(\mathsf{pk}, \mu, \sigma)$

$\mathbf{r} \leftarrow \mathsf{RO}_1(\mathsf{pk}, \mu)$
$\mathbf{c} \leftarrow \mathsf{RO}_2(\mathbf{r}, \mu)$
$\mathbf{u} \leftarrow \mathbf{Tc} + \mathbf{r}$
if $\mathbf{Az} = \mathbf{u}$ **and** $\|\mathbf{z}\| \leq 2s\sqrt{m}$ **then**
  return 1
else return 0

Fig. 2. The proposed signature scheme.

F. General Forking Lemma

The security proofs of signature schemes from the Fiat-Shamir transform are usually based on the general forking lemma. We refer to this lemma as appeared in [18]. Firstly, let $\mathcal{A}$ be an algorithm which receives p sampled from a distribution $\mathcal{P}$, t random elements $h_1, \ldots, h_t$ sampled uniformly from $\mathcal{H}$, and a random coin sampled from $\{0,1\}^c$. The task of $\mathcal{A}$ is to output (s,k) where s is some solution and k is the index of h_k used in the solution, or $k = 0$ in the case that $\mathcal{A}$ cannot construct s from any of $h_1, \ldots, h_t$. Based on the definition of $\mathcal{A}$, we define the experiment $\mathsf{Exp}^{\text{break}}_{\mathcal{A},\mathcal{P},\mathcal{H},t}$ as in Fig. 1. We define $\mathsf{break} = \Pr[\mathsf{Exp}^{\text{break}}_{\mathcal{A},\mathcal{P},\mathcal{H},t} \in [t]]$ to be the probability that $\mathcal{A}$ outputs (s,k) and $k \in [t]$.

Next, we consider the forking algorithm $\mathcal{F}$ and the experiment $\mathsf{Exp}^{\text{fork}}_{\mathcal{F},\mathcal{A},\mathcal{P},\mathcal{H},t}$ in Fig. 1. We note that two executions of $\mathcal{A}$ receive the same value of h_1 up to h_{k-1}. Let $\mathsf{fork} = \Pr[\mathsf{Exp}^{\text{fork}}_{\mathcal{F},\mathcal{A},\mathcal{P},\mathcal{H},t} = 1]$. The general forking lemma can be stated as follows.

Lemma 5 (General forking lemma [18])**.** *Let $\mathcal{P}$ be an arbitrary distribution and let $\mathcal{H}$ be the uniform distribution on some finite set H. Let $\mathcal{A}$ be an algorithm (consuming coins from the set $\{0,1\}^c$). Then,* $\mathsf{fork} \geq \frac{1}{t} \cdot \mathsf{break}^2 - \frac{1}{|H|}$.

III. Proposed Signature Scheme

A. Description

The proposed signature scheme (Fig. 2) modifies the work of [10] by utilizing lattice trapdoors of Micciancio and Peikert. The signing and verification algorithms invoke two random oracles $\mathsf{RO}_1 : \{0,1\}^* \to \mathbb{Z}_q^n$ and $\mathsf{RO}_2 : \{0,1\}^* \to \{\mathbf{w} : \mathbf{w} \in \{-1,0,1\}^k \wedge \|\mathbf{w}\|_1 \leq \kappa\}$.

The intuition of the proposed scheme is as follows: instead of constructing $\mathbf{r} \leftarrow \mathbf{Ay}$ and $\mathbf{z} \leftarrow \mathbf{Sc} + \mathbf{y}$ from $\mathbf{y}$ which give $\mathbf{Az} = \mathbf{Tc} + \mathbf{r}$ as done in [10], we invoke the random oracle for $\mathbf{r}$ and then find $\mathbf{z}$ such that $\mathbf{Az} = \mathbf{Tc} + \mathbf{r}$ without knowing $\mathbf{y}$. We can achieve this since we have the trapdoor $\mathbf{R}$ of $\mathbf{A}$. Two advantages of the proposed scheme are (i) there is no need to restart the protocol because the distribution of our $\mathbf{z}$ is close to $D_{\Lambda^\perp_{\mathbf{u}}(\mathbf{A}),s}$, and (ii) we do not need to include $\mathbf{c}$ in the signature as it can be computed using pk and μ.

B. Correctness Proof

For any signature created by the signing algorithm, we have $\mathbf{Az} = \mathbf{u}$ as $\mathbf{z}$ is an output of $\mathsf{SampleD}(\mathbf{R}, \mathbf{A}, \mathbf{u}, s)$. And by the following lemma from [10], the probability that $\|\mathbf{z}\| > 2s\sqrt{m}$ is negligible. Therefore, the probability that the valid signature is accepted is close to 1.

Lemma 6 ([10])**.** $\Pr[\|\mathbf{z}\| > 2s\sqrt{m} \mid \mathbf{z} \xleftarrow{\$} D_s] < 2^{-m}$.

C. Security Proof

We first define the advantage of a PPT algorithm $\mathcal{D}$ in solving $\mathrm{SIS}_{q,n,m,\beta}$ as

$$\mathsf{Adv}^{\mathrm{SIS}_{q,n,m,\beta}}_{\mathcal{D}} = \Pr[\mathbf{v} \neq \mathbf{0} \wedge \mathbf{Av} = \mathbf{0} \wedge \|\mathbf{v}\| \leq \beta \mid \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}; \mathbf{v} \xleftarrow{\$} \mathcal{D}(\mathbf{A})].$$

$\mathcal{B}^{\mathsf{RO}_1,\mathsf{RO}_2}(1^\lambda, \mathsf{pk})$

1 : $T_1 \leftarrow [\,]$
2 : $T_2 \leftarrow [\,]$
3 : $(\mu^*, \sigma^*) \xleftarrow{\$} \mathcal{C}^{\mathsf{ORO}_1,\mathsf{ORO}_2,\mathsf{OSign}}(1^\lambda, \mathsf{pk})$
4 : **return** (μ^*, σ^*)

$\mathsf{ORO}_1(x)$

1 : parse x as (pk, μ)
2 : **if** $T_1[(\mathsf{pk}, \mu)] = \perp$ **then**
3 : $\quad \mathbf{c} \xleftarrow{\$} H;\ \mathbf{z} \xleftarrow{\$} D_s$
4 : $\quad T_1[(\mathsf{pk}, \mu)] \leftarrow (\mathbf{c}, \mathbf{z})$
5 : $(\mathbf{c}, \mathbf{z}) \leftarrow T_1[(\mathsf{pk}, \mu)]$
6 : $\mathbf{r} \leftarrow \mathbf{Az} - \mathbf{Tc}$
7 : **if** $T_2[(\mathbf{r}, \mu)] \notin \{\perp, \mathbf{c}\}$ **then abort**
8 : $T_2[(\mathbf{r}, \mu)] \leftarrow \mathbf{c}$
9 : **return** $\mathbf{r}$

$\mathsf{ORO}_2(x)$

1 : **if** $T_2[x] = \perp$ **then**
2 : $\quad T_2[x] \leftarrow \mathsf{RO}_2(x)$
3 : **return** $T_2[x]$

$\mathsf{OSign}(\mu)$

1 : $\mathbf{r} \leftarrow \mathsf{ORO}_1(\mathsf{pk}, \mu)$
2 : $(\mathbf{c}, \mathbf{z}) \leftarrow T_1[(\mathsf{pk}, \mu)]$
3 : **return** $\mathbf{z}$

Fig. 3. The algorithms simulating signature queries.

The proof consists of two lemmas, following the method used in [18]. The first lemma shows the advantage of a key-only attacker $\mathcal{B}$ in terms of that of $\mathcal{D}$, and the second lemma shows that a key-only attacker $\mathcal{B}$ can be reduced to a one-per-message chosen-message attacker $\mathcal{C}$. The advantage of a key-only attacker is defined similar to that of a chosen-message attacker except that the key-only attacker cannot make a signing query:

$$\mathsf{Adv}^{\text{euf-ko}}_{\mathsf{Sig},\mathcal{B}}(\lambda) = \Pr[\mathsf{Sig.Verify}(\mathsf{pk}, \mu^*, \sigma^*) = 1 \mid (\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Sig.KeyGen}(1^\lambda); (\mu^*, \sigma^*) \xleftarrow{\$} \mathcal{B}(1^\lambda, \mathsf{pk})].$$

In all subsequent proofs, we denote the range of RO_2 by $H = \{\mathbf{w} : \mathbf{w} \in \{-1, 0, 1\}^k \wedge \|\mathbf{w}\|_1 \le \kappa\}$.

Lemma 7. *For any PPT adversary $\mathcal{B}$ against the proposed signature scheme, making at most t_2 random oracle queries to RO_2, there is a PPT algorithm $\mathcal{D}$ solving $\mathrm{SIS}_{q,n,m,\beta}$ for $\beta = (4s + 2d\kappa)\sqrt{m}$ such that*

$$\mathsf{Adv}^{\mathrm{SIS}_{q,n,m,\beta}}_{\mathcal{D}} \ge \frac{1}{2}\left(\frac{1}{t_2 + 1} \cdot \left(\mathsf{Adv}^{\text{euf-ko}}_{\mathsf{Sig},\mathcal{B}}(\lambda)\right)^2 - \frac{1}{|H|}\right).$$

Proof. Let $\mathbf{A}$ be a given instance of $\mathrm{SIS}_{q,n,m,\beta}$. We choose $\mathbf{S} \xleftarrow{\$} \{-d, \ldots, d\}^{m \times k}$, compute $\mathbf{T} \leftarrow \mathbf{AS}$, and send $\mathsf{pk} = (\mathbf{A}, \mathbf{T})$ to $\mathcal{B}$. Since the distribution of $\mathbf{A}$ from $\mathsf{GenTrap}(1^\lambda)$ is statistically close to uniform, $\mathcal{B}$ cannot distinguish between two cases and see pk as a valid public key.

Next, we use the result of the general forking lemma where $\mathcal{P}$ is the set of all public keys. For an output (μ^*, σ^*) of $\mathcal{B}$, we can assume that $\mathsf{RO}_2(\mathsf{RO}_1(\mathsf{pk}, \mu^*), \mu^*)$ is queried during its execution, otherwise we can make this extra query, resulting in at most $t_2 + 1$ random oracle queries to RO_2. We then apply Lemma 5 with $\mathsf{break} = \mathsf{Adv}^{\text{euf-ko}}_{\mathsf{Sig},\mathcal{B}}(\lambda)$ and $t = t_2 + 1$.

Suppose the outputs of two executions of $\mathcal{B}$ in the forking algorithm $\mathcal{F}$ are $(\mu^*_1, \sigma^*_1 = \mathbf{z})$ and $(\mu^*_2, \sigma^*_2 = \mathbf{z}')$. By the general forking lemma, two executions are the same up to, but not include, $\mathbf{c} = \mathsf{RO}_2(\mathsf{RO}_1(\mathsf{pk}, \mu^*_1), \mu^*_1)$ and $\mathbf{c}' = \mathsf{RO}_2(\mathsf{RO}_1(\mathsf{pk}, \mu^*_2), \mu^*_2)$ with $\mathbf{c} \ne \mathbf{c}'$. This implies that $\mathsf{RO}_1(\mathsf{pk}, \mu^*_1) = \mathsf{RO}_1(\mathsf{pk}, \mu^*_2) = \mathbf{r}$. Therefore, $\mathbf{Az} - \mathbf{Tc} = \mathbf{Az}' - \mathbf{Tc}'$. From $\mathbf{T} = \mathbf{AS}$, we have $\mathbf{A}(\mathbf{z} - \mathbf{z}' - \mathbf{S}(\mathbf{c} - \mathbf{c}')) = \mathbf{0}$.

By the parameter choices, $\|\mathbf{z} - \mathbf{z}' - \mathbf{S}(\mathbf{c} - \mathbf{c}')\| \le \beta$. In the case that $\mathbf{z} - \mathbf{z}' - \mathbf{S}(\mathbf{c} - \mathbf{c}') = \mathbf{0}$, we use the fact shown in [10] that, with very high probability, there exists $\mathbf{S}'$ such that $\mathbf{T} = \mathbf{AS} = \mathbf{AS}'$ and $\mathbf{z} - \mathbf{z}' - \mathbf{S}'(\mathbf{c} - \mathbf{c}') \ne \mathbf{0}$. Because $\mathcal{B}$ does not have any knowledge about sk, there is a probability of at least $\frac{1}{2}$ such that $\mathsf{sk} = \mathbf{S}'$ and not $\mathbf{S}$ when we choose it, which enables us to construct a non-zero vector that is a solution to $\mathrm{SIS}_{q,n,m,\beta}$. Thus, $\mathsf{Adv}^{\mathrm{SIS}_{q,n,m,\beta}}_{\mathcal{D}} \ge \frac{1}{2} \cdot \mathsf{fork}$. □

Lemma 8. *For any PPT adversary $\mathcal{C}$ against the proposed signature scheme, making at most t_1 (resp. t_2) random oracle queries to RO_1 (resp. RO_2) and t_s distinct signature queries, there exists a PPT adversary $\mathcal{B}$ against the scheme, making at most t_2 random oracle queries to RO_2, such that*

$$\mathsf{Adv}^{\text{euf-ko}}_{\mathsf{Sig},\mathcal{B}}(\lambda) \ge \mathsf{Adv}^{\text{euf-cma}_1}_{\mathsf{Sig},\mathcal{C}}(\lambda) - \frac{(t_1 + t_s)(t_1 + t_2 + t_s)}{q^n}.$$

Proof. Given $\mathcal{C}$, we can construct $\mathcal{B}$ by "simulating" signature queries as shown in Fig. 3. Our goal is, without knowing the trapdoor $\mathbf{R}$ of $\mathbf{A}$, programming the random oracles so that

$$\mathbf{A} \cdot \underbrace{\mathsf{OSign}(\mu)}_{\mathbf{z}} = \mathbf{T} \cdot \underbrace{\mathsf{ORO}_2(\overbrace{\mathsf{ORO}_1(\mathsf{pk}, \mu)}^{\mathbf{r}}, \mu)}_{\mathbf{c}} + \overbrace{\mathsf{ORO}_1(\mathsf{pk}, \mu)}^{\mathbf{r}}.$$

We can see that for a fixed value of $\mathbf{r}$, the value of $\mathbf{c}$ depends on $\mathbf{r}$, and the value of $\mathbf{z}$ then depends on both $\mathbf{r}$ and $\mathbf{c}$. Thus, when $\mathcal{C}$ invokes ORO_1 for $\mathbf{r}$, we need to compute a tuple $(\mathbf{r}, \mathbf{c}, \mathbf{z})$ that satisfies $\mathbf{Az} = \mathbf{Tc} + \mathbf{r}$ and program the random oracles accordingly. The algorithm ORO_1 first samples $(\mathbf{c}, \mathbf{z})$ (if there is no existing value), computes $\mathbf{r} \leftarrow \mathbf{Az} - \mathbf{Tc}$, programs $\mathsf{ORO}_2(\mathbf{r}, \mu) = \mathbf{c}$, and then returns $\mathbf{r}$. The value $(\mathbf{c}, \mathbf{z})$ is stored to answer the signing query. The algorithm ORO_2 returns the value that is programmed by ORO_1 or a result from RO_2 if no value is set, and the algorithm OSign invokes ORO_1 to create a tuple $(\mathbf{r}, \mathbf{c}, \mathbf{z})$ if necessary and then returns $\mathbf{z}$. The simulation works correctly since the equation $\mathbf{Az} = \mathbf{Tc} + \mathbf{r}$ is satisfied and the distribution of $(\mathbf{r}, \mathbf{c}, \mathbf{z})$ is the same as that obtaining from the genuine signer.

$\underline{\mathsf{SV.Init}(1^\lambda)}$

1 : $(\mathbf{A}, \mathbf{R}) \xleftarrow{\$} \mathsf{GenTrap}(1^\lambda)$
2 : $\mathsf{pp} \leftarrow \mathbf{A}$
3 : **return** pp

$\underline{\mathsf{SV.Sample}(\mathsf{sk}_i, \mathsf{pk}_i, \mu)}$

1 : **if** $\mathbf{T}_i \neq \mathbf{AS}_i$ **then return** $\perp$
2 : $\mathbf{r}_i \leftarrow \mathsf{RO}_1(\mathsf{pk}_i, \mu)$
3 : **return** $\mathsf{SampleD}(\mathbf{R}, \mathbf{A}, \mathbf{r}_i, s)$

$\underline{\mathsf{MS.KeyGen}(\mathsf{pp})}$

1 : $\mathbf{S}_i \xleftarrow{\$} \{-d, \ldots, d\}^{m \times k}$
2 : $\mathbf{T}_i \leftarrow \mathbf{AS}_i$
3 : $\mathsf{sk}_i \leftarrow \mathbf{S}_i$
4 : $\mathsf{pk}_i \leftarrow \mathbf{T}_i$
5 : **return** $(\mathsf{sk}_i, \mathsf{pk}_i)$

$\underline{\mathsf{MS.Sign}(\mathsf{sk}_i, \mathsf{PK}, \mu)}$

1 : $\mathbf{r} \leftarrow \sum_{j=1}^{N} \mathsf{RO}_1(\mathsf{pk}_j, \mu)$
2 : $\mathbf{c}_i \leftarrow \mathsf{RO}_2(\mathbf{r}, \mathsf{pk}_i, \mu)$
3 : $\mathbf{y}_i \xleftarrow{\$} \mathsf{SV.Sample}(\mathsf{sk}_i, \mathsf{pk}_i, \mu)$
4 : $\mathbf{z}_i \leftarrow \mathbf{S}_i\mathbf{c}_i + \mathbf{y}_i$
5 : $\sigma_i \leftarrow \mathbf{z}_i$
6 : Go to 3 with probability $\mathsf{Rej}(\mathbf{S}_i\mathbf{c}_i, \mathbf{z}_i)$
7 : **return** σ_i

$\underline{\mathsf{MS.Verify}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{PK}, \mu, \sigma_i)}$

1 : $\mathbf{r}_i \leftarrow \mathsf{RO}_1(\mathsf{pk}_i, \mu)$
2 : $\mathbf{r} \leftarrow \sum_{j=1}^{N} \mathsf{RO}_1(\mathsf{pk}_j, \mu)$
3 : $\mathbf{c}_i \leftarrow \mathsf{RO}_2(\mathbf{r}, \mathsf{pk}_i, \mu)$
4 : $\mathbf{u}_i \leftarrow \mathbf{T}_i\mathbf{c}_i + \mathbf{r}_i$
5 : **if** $\mathbf{Az}_i = \mathbf{u}_i$ **and** $\|\mathbf{z}_i\| \leq 2s\sqrt{m}$ **then**
6 : **return** 1
7 : **else return** 0

$\underline{\mathsf{MS.Agg}(\mathsf{pp}, \mathsf{PK}, \mu, \{\sigma_j\}_{j=1}^{N})}$

1 : **for** $i = 1$ **to** N **do**
2 : **if** $\mathsf{MS.Verify}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{PK}, \mu, \sigma_i) = 0$ **then return** $\perp$
3 : $\mathbf{z} \leftarrow \sum_{j=1}^{N} \mathbf{z}_j$
4 : $\Sigma \leftarrow \mathbf{z}$
5 : **return** Σ

$\underline{\mathsf{MS.MVerify}(\mathsf{pp}, \mathsf{PK}, \mu, \Sigma)}$

1 : $\mathbf{r} \leftarrow \sum_{j=1}^{N} \mathsf{RO}_1(\mathsf{pk}_j, \mu)$
2 : **for** $i = 1$ **to** N **do** $\mathbf{c}_i \leftarrow \mathsf{RO}_2(\mathbf{r}, \mathsf{pk}_i, \mu)$
3 : $\mathbf{u} \leftarrow \sum_{j=1}^{N} \mathbf{T}_j\mathbf{c}_j + \mathbf{r}$
4 : **if** $\mathbf{Az} = \mathbf{u}$ **and** $\|\mathbf{z}\| \leq 2Ns\sqrt{m}$ **then return** 1
5 : **else return** 0

Fig. 4. The proposed multisignature scheme.

The only case that $\mathcal{B}$ fails is when $\mathcal{B}$ aborts in Line 7 of ORO_1. This happens when $T_2[(\mathbf{r}, \mu)]$ is set to a different value by any of the algorithms. We upper-bound the probability that this happens by the probability that $T_2[(\mathbf{r}, \cdot)]$ is set for any message. Since the distribution of $\mathbf{r}$ is close to uniform in $\mathbb{Z}_q^n$, the probability that we encounter the same $\mathbf{r}$ is $\frac{1}{q^n}$. By the union bound, the probability that $T_2[(\mathbf{r}, \cdot)]$ is set in the previous algorithm invocations is at most $(t_1 + t_2 + t_s) \cdot \frac{1}{q^n}$. Because $\mathcal{B}$ can fail when ORO_1 or OSign is called, the probability that $\mathcal{B}$ fails is at most $(t_1 + t_s)(t_1 + t_2 + t_s) \cdot \frac{1}{q^n}$. □

We conclude the security proof as the following theorem.

Theorem 9. *For any PPT adversary $\mathcal{C}$ against the proposed signature scheme, making at most t_1 (resp. t_2) random oracle queries to RO_1 (resp. RO_2) and t_s distinct signature queries, there is a PPT algorithm $\mathcal{D}$ solving $\mathrm{SIS}_{q,n,m,\beta}$ for $\beta = (4s + 2d\kappa)\sqrt{m}$ such that*

$$\mathsf{Adv}_{\mathcal{D}}^{\mathrm{SIS}_{q,n,m,\beta}} \geq \frac{1}{2}\left(\frac{1}{t_2 + 1}\left(\mathsf{Adv}_{\mathsf{Sig},\mathcal{C}}^{\text{euf-cma}_1}(\lambda) - \epsilon\right)^2 - \frac{1}{|H|}\right)$$

where $\epsilon = (t_1 + t_s)(t_1 + t_2 + t_s)/q^n$.

IV. Proposed Multisignature Scheme

A. Description

In this section, we extend our proposed signature scheme in the previous section to the multisignature scheme shown in Fig. 4. The variable $1 \leq i \leq N$ denotes the index of a signer.

In the scheme, all signers use the same $\mathbf{A}$, similar to previous works. However, we cannot give the trapdoor $\mathbf{R}$ to each signer as any signer can forge a valid signature of other signers and thus a valid multisignature. Instead, $\mathbf{R}$ is kept with the honest centralized server (called SV) and each signer then communicates with the server when $\mathsf{SampleD}$ is needed.

Nevertheless, the scheme is not secure if a signer can obtain the preimage of any value. For example, in MS.MVerify, the value of $\mathbf{u}$ can be publicly computed and the scheme is broken if any signer can ask the server for the preimage of $\mathbf{u}$, which is indeed a valid multisignature. To solve this issue, we restrict that each signer can only obtain the preimage of the value of the form $\mathsf{RO}_1(\mathsf{pk}_i, \mu)$. This is done by the signer submitting pk_i and μ and the server confirming that the signer has the corresponding sk_i, e.g., using an identification scheme.

Furthermore, because we now Gaussian sample the preimage of $\mathbf{r}_i$ instead of $\mathbf{z}_i$ as done in the signature scheme, $\mathbf{z}_i = \mathbf{S}_i\mathbf{c}_i + \mathbf{y}_i$ might leak an information about $\mathbf{S}_i\mathbf{c}_i$. Referring to [10], we need to resample $\mathbf{y}_i$ before outputting $\mathbf{z}_i$ with probability $\mathsf{Rej}(\mathbf{S}_i\mathbf{c}_i, \mathbf{z}_i) = 1 - \min\left(\frac{D_s(\mathbf{z}_i)}{M \cdot D_s^{(\mathbf{S}_i\mathbf{c}_i)}(\mathbf{z}_i)}, 1\right)$ where $D_s^{(\mathbf{S}_i\mathbf{c}_i)}$ denotes D_s centered at $\mathbf{S}_i\mathbf{c}_i$ instead of $\mathbf{0}$, and M is a suitable constant. By this restart, it is proved that the distribution of $\mathbf{z}_i$ is close to D_s. Apart from what mentioned, everything is the same as the signature scheme: each signer

$\mathsf{XSample}(\mathsf{sk}_i, \mathsf{pk}_i, \mu)$

1 : **if** $\mathbf{T}_i \neq \mathbf{AS}_i$ **then return** $\perp$
2 : **if** $T_1[(\mathsf{pk}_i, \mu)] = \perp$ **then**
3 : $\quad \mathbf{r}_i \leftarrow \mathsf{XRO}_1(\mathsf{pk}_i, \mu)$
4 : **return** $T_1[(\mathsf{pk}_i, \mu)]$

$\mathsf{XRO}_1(x)$

1 : parse x as (pk_i, μ)
2 : **if** $T_1[(\mathsf{pk}_i, \mu)] = \perp$ **then**
3 : $\quad T_1[(\mathsf{pk}_i, \mu)] \xleftarrow{\$} D_s$
4 : **return** $\mathbf{A} \cdot T_1[(\mathsf{pk}_i, \mu)]$

$\mathcal{B}^{\mathsf{XRO}_1, \mathsf{RO}_2}(1^\lambda, \mathsf{PK}, \{\mathsf{sk}_j\}_{j=2}^N)$

1 : $T_1' \leftarrow [\,]$
2 : $T_2 \leftarrow [\,]$
3 : $(\mu^*, \Sigma^*) \xleftarrow{\$}$
$\quad \mathcal{C}^{\mathsf{ORO}_1, \mathsf{ORO}_2, \mathsf{OSign}}(1^\lambda, \mathsf{PK}, \{\mathsf{sk}_j\}_{j=2}^N)$
4 : **return** (μ^*, Σ^*)

$\mathsf{ORO}_1(x)$

1 : parse x as (pk, μ)
2 : **if** $\mathsf{pk} \neq \mathsf{pk}_1$ **then return** $\mathsf{XRO}_1(x)$
3 : **if** $T_1'[(\mathsf{pk}_1, \mu)] = \perp$ **then**
4 : $\quad \mathbf{c}_1 \xleftarrow{\$} H;\ \mathbf{z}_1 \xleftarrow{\$} D_s$
5 : $\quad T_1'[(\mathsf{pk}_1, \mu)] \leftarrow (\mathbf{c}_1, \mathbf{z}_1)$
6 : $(\mathbf{c}_1, \mathbf{z}_1) \leftarrow T_1'[(\mathsf{pk}_1, \mu)]$
7 : $\mathbf{r}_1 \leftarrow \mathbf{A}\mathbf{z}_1 - \mathbf{T}_1\mathbf{c}_1$
8 : $\mathbf{r} \leftarrow \mathbf{r}_1 + \sum_{j=2}^N \mathsf{ORO}_1(\mathsf{pk}_j, \mu)$
9 : **if** $T_2[(\mathbf{r}, \mathsf{pk}_1, \mu)] \notin \{\perp, \mathbf{c}_1\}$ **then abort**
10 : $T_2[(\mathbf{r}, \mathsf{pk}_1, \mu)] \leftarrow \mathbf{c}_1$
11 : **return** $\mathbf{r}_1$

$\mathsf{ORO}_2(x)$

1 : **if** $T_2[x] = \perp$ **then**
2 : $\quad T_2[x] \leftarrow \mathsf{RO}_2(x)$
3 : **return** $T_2[x]$

$\mathsf{OSign}(\mu)$

1 : $\mathbf{r}_1 \leftarrow \mathsf{ORO}_1(\mathsf{pk}_1, \mu)$
2 : $(\mathbf{c}_1, \mathbf{z}_1) \leftarrow T_1'[(\mathsf{pk}_1, \mu)]$
3 : **return** $\mathbf{z}_1$

Fig. 5. The algorithms simulating multisignature queries.

constructs $\mathbf{z}_i$ where $\mathbf{A}\mathbf{z}_i = \mathbf{T}_i\mathbf{c}_i + \mathbf{r}_i$ and the multisignature is $\Sigma = \mathbf{z} = \sum_{j=1}^N \mathbf{z}_j$ where $\mathbf{A}\mathbf{z} = \sum_{j=1}^N \mathbf{T}_j\mathbf{c}_j + \mathbf{r}_j$.

We point out that, unlike previous works, there is no broadcasting among signers and they do not need to be online at the same time. For MS.Agg, anyone, signer or not, can execute it and each signer only needs to send σ_i to this person, thus no broadcasting is required. This makes the signing process more flexible. Moreover, if any signer needs to resample $\mathbf{y}_i$ because of the rejection, other signers do not have to resample theirs. This is different from all previous works where all signers have to restart the protocol when any signer needs to, the process which implies additional cost. Thus, our multisignature scheme reduces the communication cost in two aspects. The only requirements of the proposed scheme are the honest server and a secure communication channel between the server and each signer.

B. Correctness Proof

From the definition of MS.Sign, $\mathbf{A}\mathbf{z}_i = \mathbf{u}_i$ must hold for all $1 \leq i \leq N$. Considering Lines 1–2 of MS.Agg, the aggregation algorithm returns $\perp$ if any signature does not pass the verification. In the case that $\Sigma \neq \perp$, we know that $\mathbf{A}\mathbf{z} = \mathbf{A}\sum_{j=1}^N \mathbf{z}_j = \sum_{j=1}^N \mathbf{u}_j = \mathbf{u}$ and $\|\mathbf{z}_i\| \leq 2s\sqrt{m}$. Thus, $\|\mathbf{z}\| \leq 2Ns\sqrt{m}$ and Σ is a valid multisignature.

The only case that MS.Agg returns $\perp$ is when there exists $1 \leq i \leq N$ such that $\|\mathbf{z}_i\| > 2s\sqrt{m}$. By Lemma 6 and the union bound, this event happens with probability at most $N2^{-m}$, which is negligible. Therefore, the multisignature from MS.Agg is accepted with probability close to 1.

C. Security Proof

The security proof for the proposed multisignature scheme is more complicated comparing to the signature scheme. This is because, given an instance $\mathbf{A}$ of $\mathrm{SIS}_{q,n,m,\beta}$, we need to provide the results of SV.Sample for key pairs owned by the adversary without having its trapdoor $\mathbf{R}$. We again deal with this using a simulation as shown in Fig. 5.

Lemma 10. *For any PPT adversary $\mathcal{B}$ against the proposed multisignature scheme, making at most t_2 random oracle queries to RO_2, there is a PPT algorithm $\mathcal{D}$ solving $\mathrm{SIS}_{q,n,m,\beta}$ for $\beta = (4Ns + 2d\kappa)\sqrt{m}$ such that*

$$\mathsf{Adv}_{\mathcal{D}}^{\mathrm{SIS}_{q,n,m,\beta}} \geq \frac{1}{2}\left(\frac{1}{t_2+1} \cdot \left(\mathsf{Adv}_{\mathsf{MS},\mathcal{B}}^{\text{euf-ko}}(\lambda)\right)^2 - \frac{1}{|H|}\right).$$

Proof. For an instance $\mathbf{A}$ of $\mathrm{SIS}_{q,n,m,\beta}$ and $1 \leq i \leq N$, we choose $\mathbf{S}_i \xleftarrow{\$} \{-d, \ldots, d\}^{m \times k}$, compute $\mathbf{T}_i \leftarrow \mathbf{A}\mathbf{S}_i$, and send PK and $\{\mathsf{sk}_j\}_{j=2}^N$ to $\mathcal{B}$. We only give $\mathcal{B}$ an access to XSample and XRO_1 shown in Fig. 5. Distinguishing between RO_1 and XRO_1 is similar to the problem of [10, Definition 2.3] which is computationally hard. $\mathcal{B}$ can normally invoke RO_2.

This time we are interested in two runs of the adversary where the value of $\mathbf{c}_1 = \mathsf{RO}_2(\mathbf{r}, \mathsf{pk}_1, \mu)$ are different and $\mathbf{c}_i$ for $2 \leq i \leq N$ are the same. Let the two outputs of the forking algorithm be $(\mu_1^*, \Sigma_1^* = \mathbf{z})$ and $(\mu_2^*, \Sigma_2^* = \mathbf{z}')$ and the outputs of RO_2 be $\mathbf{c}_i$ and $\mathbf{c}_i'$. Since two runs are the same up to, but not include, the computation of $\mathbf{c}_1$ and $\mathbf{c}_1'$, the value of $\mathbf{r}_i$ are the same for all $1 \leq i \leq N$. Thus, $\mathbf{r} = \mathbf{A}\mathbf{z} - \sum_{j=1}^N \mathbf{T}_j\mathbf{c}_j =$

$\mathbf{Az}' - \sum_{j=1}^{N} \mathbf{T}_j \mathbf{c}'_j$. Since $\mathbf{c}_i = \mathbf{c}'_i$ for $2 \leq i \leq N$, we have $\mathbf{A}(\mathbf{z} - \mathbf{z}' - \mathbf{S}_1(\mathbf{c}_1 - \mathbf{c}'_1)) = \mathbf{0}$. The same reasoning as in Lemma 7 is then applied to achieve the desired bound. □

Lemma 11. *For any PPT adversary $\mathcal{C}$ against the proposed multisignature scheme, making at most t_1 (resp. t_2) random oracle queries to RO_1 (resp. RO_2) and t_s distinct signature queries, there exists a PPT adversary $\mathcal{B}$ against the scheme, making at most t_2 random oracle queries to RO_2, such that*

$$\mathsf{Adv}^{\text{euf-ko}}_{\mathsf{MS},\mathcal{B}}(\lambda) \geq \mathsf{Adv}^{\text{euf-cma}_1}_{\mathsf{MS},\mathcal{C}}(\lambda) - \frac{(t_1 + t_s)(t_1 + t_2 + t_s)}{q^n}.$$

Proof. The proof follows the same approach as in Lemma 8 except that $\mathcal{C}$ has an access to $\mathsf{XSample}$, ORO_1 (which calls XRO_1), ORO_2, and OSign. □

We finally obtain the theorem describing the security of our multisignature scheme as follows.

Theorem 12. *For any PPT adversary $\mathcal{C}$ against the proposed multisignature scheme, making at most t_1 (resp. t_2) random oracle queries to RO_1 (resp. RO_2) and t_s distinct signature queries, there is a PPT algorithm $\mathcal{D}$ solving $\mathrm{SIS}_{q,n,m,\beta}$ for $\beta = (4Ns + 2d\kappa)\sqrt{m}$ such that*

$$\mathsf{Adv}^{\mathrm{SIS}_{q,n,m,\beta}}_{\mathcal{D}} \geq \frac{1}{2}\left(\frac{1}{t_2 + 1}\left(\mathsf{Adv}^{\text{euf-cma}_1}_{\mathsf{MS},\mathcal{C}}(\lambda) - \epsilon\right)^2 - \frac{1}{|H|}\right)$$

where $\epsilon = (t_1 + t_s)(t_1 + t_2 + t_s)/q^n$.

Remark. *The proposed multisignature scheme can be modified to produce an aggregated public key, which is a single public key representing the whole group of signers. We refer the interested readers to [3], [6] for more details.*

V. Conclusion

We have presented a lattice-based multisignature scheme that eliminates the need of broadcasting information among signers. This improvement is due to lattice trapdoors. By the result of this work, it is interesting to utilize lattice trapdoors to improve other lattice-based cryptosystems in a similar manner.

Even though the communication cost is reduced, we need an honest server to keep signers from knowing the trapdoor and forging a multisignature. It is not trivial to remove the server in our scheme by lattice primitives that we know and we therefore leave this as an open question.

Our future work is to consider other security models that are stronger than the existentially unforgeable under one-per-message chosen-message attack such as the knowledge of the secret key model, the certified key model, the key verification model, and the plain public-key model. Also, we plan to instantiate all the parameters of the schemes so that we can compare other aspects of our schemes with previous works based on the same security level.

Acknowledgment

The authors would like to thank the reviewers for their constructive comments regarding the security of the proposed schemes. The first author would like to thank Nina Bindel for her advice on the proofs, and Kittiphop Phalakarn for his suggestions on the schemes. The first author is supported by the Ripple Impact Fund through a Ripple Graduate Fellowship. The second author is supported by JSPS Grant-in-Aid for Transformative Research Areas A grant number JP21H05845.

References

[1] K. Itakura and K. Nakamura, "A public-key cryptosystem suitable for digital multisignatures," *NEC Research & Development*, no. 71, pp. 1–8, 1983.

[2] M. Bellare and G. Neven, "Identity-based multi-signatures from RSA," in *Cryptographers' Track at the RSA Conference*, pp. 145–162, Springer, 2007.

[3] D. Boneh, M. Drijvers, and G. Neven, "Compact multi-signatures for smaller blockchains," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 435–464, Springer, 2018.

[4] D. Micciancio and O. Regev, "Lattice-based cryptography," in *Post-quantum cryptography*, pp. 147–191, Springer, 2009.

[5] R. El Bansarkhani and J. Sturm, "An efficient lattice-based multisignature scheme with applications to bitcoins," in *International Conference on Cryptology and Network Security*, pp. 140–155, Springer, 2016.

[6] C. Ma and M. Jiang, "Practical lattice-based multisignature schemes for blockchains," *IEEE Access*, vol. 7, pp. 179765–179778, 2019.

[7] M. Fukumitsu and S. Hasegawa, "A lattice-based provably secure multisignature scheme in quantum random oracle model," in *International Conference on Provable Security*, pp. 45–64, Springer, 2020.

[8] I. Damgård, C. Orlandi, A. Takahashi, and M. Tibouchi, "Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices," in *IACR International Conference on Public-Key Cryptography*, pp. 99–130, Springer, 2021.

[9] V. Lyubashevsky, "Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 598–616, Springer, 2009.

[10] V. Lyubashevsky, "Lattice signatures without trapdoors," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 738–755, Springer, 2012.

[11] T. Güneysu, V. Lyubashevsky, and T. Pöppelmann, "Practical lattice-based cryptography: A signature scheme for embedded systems," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 530–547, Springer, 2012.

[12] Z. Jaroucheh, B. Ghaleb, and W. J. Buchanan, "SklCoin: Toward a scalable proof-of-stake and collective signature based consensus protocol for strong consistency in blockchain," in *2020 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pp. 143–150, IEEE, 2020.

[13] M. Kansal and R. Dutta, "Round optimal secure multisignature schemes from lattice with public key aggregation and signature compression," in *International Conference on Cryptology in Africa*, pp. 281–300, Springer, 2020.

[14] Z.-Y. Liu, Y.-F. Tseng, and R. Tso, "Cryptanalysis of a round optimal lattice-based multisignature scheme.," *IACR Cryptol. ePrint Arch.*, 2020.

[15] D. Micciancio and C. Peikert, "Trapdoors for lattices: Simpler, tighter, faster, smaller," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 700–718, Springer, 2012.

[16] E. Kiltz, V. Lyubashevsky, and C. Schaffner, "A concrete treatment of fiat-shamir signatures in the quantum random-oracle model," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 552–586, Springer, 2018.

[17] N. Gama and P. Q. Nguyen, "Predicting lattice reduction," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 31–51, Springer, 2008.

[18] A. Mittelbach and M. Fischlin, *The Theory of Hash Functions and Random Oracles: An Approach to Modern Cryptography*. Springer Nature, 2021.