

Two-Round Multi-Signature from Okamoto Signature

Kwangsue Lee*

Hyoseung Kim[†]

Abstract

Multi-signature (MS) is a special type of public key signature (PKS) in which multiple signers participate cooperatively to generate a signature for a single message. Recently, applications that use an MS scheme to strengthen the security of blockchain wallets or to strengthen the security of blockchain consensus protocols are attracting a lot of attention. In this paper, we propose an efficient two-round MS scheme based on Okamoto signature rather than Schnorr signature. To this end, we first propose a new PKS scheme by modifying the Okamoto signature scheme, and prove the unforgeability of our PKS scheme under the discrete logarithm assumption in the algebraic group model (AGM) and the non-programmable random oracle model (ROM). Next, we propose a two-round MS scheme based on the new PKS scheme and prove the unforgeability of our MS scheme under the discrete logarithm assumption in the AGM and the non-programmable ROM. Our MS scheme is the first one to prove security among two-round MS based on Okamoto signature.

Keywords: Public-key signature, Multi-signature, Okamoto signature, Key aggregation, Algebraic group model.

*Sejong University, Seoul, Korea. Email: kwangsue@sejong.ac.kr.

[†]Korea University, Seoul, Korea. Email: hyoseung_kim@korea.ac.kr.

1 Introduction

Multi-signature (MS) is a special kind of public-key signature (PKS) in which multiple signers can cooperatively participate to create a signature for a single message and verify the signature by using the public keys of all signers participated in the signature generation. Multi-signature becomes an interesting MS scheme only when the size of the multi-signature is compact regardless of the number of cooperating signers because an MS scheme can be easily built from the existing PKS scheme in a simple way of attaching individual signatures of PKS schemes. Interactive MS schemes can be constructed from the existing Fiat-Shamir based signature schemes [4, 7, 24], and non-interactive MS schemes also can be constructed based on bilinear groups [5, 6, 10, 21]. In recent years, research on multi-signature is attracting heavy attention again because it can be effectively used to enhance the security of a blockchain wallet or to perform secure consensus among multiple nodes in a blockchain.

A popular way to design a PKS scheme is to convert an identification protocol into a PKS scheme by using the Fiat-Shamir transformation. The Schnorr signature is the famous example of this case [28]. A PKS scheme derived from this method has the advantage of being widely implemented and used in various places because it can be proven under standard assumptions such as the discrete logarithm assumption and the scheme is very efficient. One important way to design an MS scheme is to convert a Fiat-Shamir based PKS scheme to an interactive MS scheme. Bellare and Neven [4] have shown that a three-round MS scheme can be constructed from the Schnorr signature scheme in the plain public-key model. Afterwards, a number of two-round MS schemes that improve the rounds required in the signature generation process were proposed [2, 23, 30]. However, Drijvers et al. [9] showed that all of these two-round MS schemes can be attacked by using a parallel signing session attack with the Wagner algorithm, and it is difficult to prove the security of these MS schemes by using the meta-reduction technique. To solve this problem, a number of secure two-round MS schemes based on the Schnorr signature or a trapdoor commitment scheme have been proposed recently [1, 3, 9, 25, 26].

The Okamoto signature scheme is also one of the Fiat-Shamir based PKS schemes [27]. An important feature of Okamoto signature different from Schnorr signature is that Schnorr-based signatures use the zero-knowledge property to handle the signature query of an attacker, but Okamoto signature use the witness indistinguishability to handle the signature query of the attacker. Due to this difference, the simulation of the signature in the security proof of Okamoto signature can be easily processed using the private key selected by a simulator. The Okamoto signature scheme can also be converted to a three-round MS scheme by following the conversion method Bellare and Neven [4]. To reduce the number of rounds further, Ma et al. [23] proposed two-round MS scheme from Okamoto signature, but this scheme is not secure against the parallel signing session attack as shown by Drijvers et al. [9]. Many secure two-round MS schemes based on Schnorr signature or trapdoor commitments [1, 3, 9, 25], but constructing a secure two-round MS scheme based on Okamoto signature is still an unsolved problem.

In this paper, we focus on the problem of constructing a two-round MS scheme based on Okamoto signature. Multi-signature design based on Okamoto signature is an interesting problem because it can present a new design direction for MS schemes different from the existing design of two-round MS schemes. In addition, an MS scheme based on Okamoto signature can use a weaker random oracle model because it uses a non-programmable random oracle model (NPROM) instead of a programmable random oracle model (PROM).

1.1 Our Contributions

Public-Key Signature. We first modify the Okamoto signature scheme and propose a new PKS scheme suitable for multi-signature. The essential part to the modification is to set the commitment element in a message dependent way such as $R = (g^m h)^{r_1} (g_2^m h_2)^{r_2}$ instead of $R = g^{r_1} g_2^{r_2}$. And we prove that the proposed PKS scheme is secure under the discrete logarithm problem is hard in the algebraic group model (AGM) and non-programmable random oracle model (NPROM). Compared to the existing Okamoto signature scheme, our proposed new PKS scheme increases the size of the public key by one group element, and the size of the signature is the same, but requires an additional exponentiation when signing. And when verifying the signature, the Okamoto signature scheme requires 3 exponentiations, but our PKS scheme requires 6 exponentiations. Although our PKS scheme does not provide improved efficiency compared to the Okamoto signature scheme, our PKS scheme is meaningful in that it can be the basis of a secure two-round MS scheme and it provides a simple way to perform the security proof in the AGM.

Two-Round Multi-Signature. Next, we propose a two-round MS scheme that supports the public key aggregation from our PKS scheme devised above. And then we prove that our MS scheme is secure even when an attacker performs parallel signing session queries under the discrete logarithm assumption in the AGM and NPROM. There are two issues to consider in the security proof of the two-round MS scheme: the rogue-key attack and parallel signing session attack. In the rogue-key attack, an attacker forges a multi-signature by manipulating a carefully crafted public key without knowing the corresponding private key since the attacker can select the arbitrary public key of a co-signer in the plain public-key model. Fortunately, by using the public key aggregation method of previous study, this rogue-key attack can be easily prevented in our MS scheme. The parallel signing session attack is a clever attack in which an attacker requests multiple signature query sessions in parallel in a two-round MS scheme since the signing process is an interactive protocol and the attacker manipulates the obtained signatures of parallel sessions to derive a final forged signature. Recall that many two-round MS schemes proposed early can be attacked by using this parallel signing session attack. To prevent this parallel signing session attack, we generate a commitment in the form $R = (g^m h)^{r_1} (g_2^m h_2)^{r_2}$ where m is a message. As the commitment is configured depending on the signature message in this way, if an attacker performs the parallel signing session attack, the commitments for the same message can be changed to a new commitment, but these commitments cannot be converted to a new commitment for a different message. The detailed explanation of the security proof is given in the security analysis part of our paper. The detailed comparison of our MS scheme and other MS schemes is given in Table 1.

1.2 Related Work

Multi-Signature. Multi-signature (MS) is a kind of public key signature in which multiple signers participate to generate a signature for a common message and anyone can verify the signature with the public keys of multiple signers. Early MS schemes were vulnerable to a rogue-key attack, in which an attacker arbitrarily sets the public key of a signer participating in multi-signature to perform a forgery attack. Bellare and Neven [4] introduced the plain public-key model in which an attacker can freely set a signers' public key without proving the knowledge of the secret key, and proposed a three-round MS scheme that is secure against the rogue-key attack by modifying the Schnorr signature scheme. Since then, a number of two-round MS schemes have been proposed to improve the round complexity of Fiat-Shamir based MS schemes [2, 23, 30]. However, Drijvers et al. [9] showed that these two-round MS schemes are vulnerable to parallel signing session attack by using Wagner's algorithm, and proposed a modified MS scheme by modifying the existing BCJ-MS scheme. Maxwell et al. [24] presented the MuSig scheme in which the

Table 1: Comparison of Fiat-Shamir based multi-signature schemes

Scheme	RN, KA	PK Size	Sig Size	Sign	Verify	Security
BN [4]	3, N	$ \mathbb{G} $	$ \mathbb{G} + \mathbb{Z}_p $	1E	$(n + 1)E$	DL, ROM
	3, N	$2 \mathbb{G} $	$2 \mathbb{G} + \mathbb{Z}_p $	2E	$2(n + 1)E$	DDH, ROM
MPSW [24]	3, Y	$ \mathbb{G} $	$ \mathbb{G} + \mathbb{Z}_p $	1E	2E	DL, ROM
mBCJ [9]	2, N	$ \mathbb{G} + 2 \mathbb{Z}_p $	$2 \mathbb{G} + 3 \mathbb{Z}_p $	5E	6E	DL, ROM
MuSig-DN [26]	2, Y	$ \mathbb{G} $	$ \mathbb{G} + \mathbb{Z}_p $	NIZK proof	2E	DL, DDH, ZK, PRF, ROM
MuSig2 [25]	2, Y	$ \mathbb{G} $	$ \mathbb{G} + \mathbb{Z}_p $	7E	2E	OMDL, ROM
	2, Y	$ \mathbb{G} $	$ \mathbb{G} + \mathbb{Z}_p $	3E	2E	OMDL, AGM + ROM
DWMS [1]	2, Y	$ \mathbb{G} $	$ \mathbb{G} + \mathbb{Z}_p $	$(2n + 2)E$	2E	OMDL + 2ES, AGM + ROM
HBMS [3]	2, Y	$ \mathbb{G} $	$ \mathbb{G} + 2 \mathbb{Z}_p $	2E	3E	XIDL, ROM or DL, AGM + ROM
Ours	2, Y	$2 \mathbb{G} $	$3 \mathbb{Z}_p $	4E	6E	DL, AGM + NPROM

Let λ be a security parameter and n be the number of co-signers. We denote RN for the number of rounds and KA for key aggregation. We use $|\mathbb{G}|$ and $|\mathbb{Z}_p|$ for the bit size of a group element in \mathbb{G} and \mathbb{Z}_p respectively. We use symbols E for an exponentiation in \mathbb{G} .

signers' public keys are aggregated into one short public key in the three-round MS scheme and showed that this MS scheme can be used for Bitcoin. Recently, a number of secure two-round MS schemes, MuSig-DN, MuSig2, DWMS, and HBMS, have been proposed [1, 3, 25, 26]. Another way to design an MS scheme is to convert an aggregate signature scheme into a non-interactive MS scheme by setting the message to be the same for each signer. Using this idea, Boneh et al. [7] propose an efficient non-interactive MS scheme from the BLS short signature scheme and proved the security in the plain public-key model. Drijvers et al. [10] propose a non-interactive MS scheme with forward security from a sequential aggregate signature scheme that can be used in blockchain consensus protocols.

Threshold Signature. Threshold signature (TS) is a kind of public key signature such that a threshold number of signers cooperate to generate a signature on a message and the signature can be verified by a single public key. Multi-signature can also be viewed as a special form of threshold signature where the number of threshold is equal to the number of all signers. Since the ECDSA scheme is a standard signature scheme which is widely used in cryptocurrency like Bitcoin, many studies have conducted to convert the ECDSA scheme into an efficient threshold ECDSA scheme [12, 13, 19, 20]. Recently, efficient threshold signature schemes have been proposed by modifying Schnorr signature scheme [18]. An important difference between threshold signature and multi-signature is the key generation process. In multi-signature, signers can generate keys independently of each other. Contrary to this, threshold signature requires to distribute a common secret key to multiple signers, so a rather complicated distributed key generation protocol must be introduced. A distributed key generation (DKG) protocol allows to share a common secret to many signers without a trusted center. If a common secret to be shared is a field element, a DKG protocol can be implemented by using a verifiable secret sharing (VSS) scheme that can privately verify the validity of a

shared secret [14]. If a common secret is a group element, a DKG protocol can be implemented by using a public verifiable secret sharing (PVSS) scheme that can publicly verify the validity of a shared secret [17]. Recently, Groth [16] proposed a PVSS scheme that can support a field element by splitting a common secret into multiple chunks, and constructed a non-interactive DKG scheme by combining the PVSS scheme with a binary tree encryption scheme with forward secrecy.

Aggregate Signature. Aggregate signature (AS) is a special type of public key signature that allows multiple signers to create signatures for different messages and aggregate them into a single signature. The concept of aggregate signatures was introduced by Boneh et al. [8] and they constructed an efficient aggregate signature scheme by modifying the BLS short signature scheme in bilinear groups. Since then, many aggregate signature schemes based on bilinear groups and trapdoor functions have been proposed [6, 15, 21, 22]. The security of aggregate signatures is proven in the knowledge of secret key (KOSK) model which requires the proof of secret key in key registration process, and it is stronger than the plain public-key model of multi-signatures. Aggregate signatures are divided into three types: full aggregation, sequential aggregation, and synchronized aggregation according to the method of aggregation. A fully aggregate signature scheme is the most flexible type of aggregate signatures that allows anyone to non-interactively aggregate individual signatures generated by different signers on different messages into a compact signature [8]. A sequential aggregate signature scheme supports for a signer to sequentially add his signature to the previous aggregate signature received from the previous signer [21, 22]. A synchronized aggregate signature scheme is similar to full aggregate signatures except that all signers have the synchronized information and individual signatures with the same synchronized information can be non-interactively aggregated [15]. As previously described, a pairing-based non-interactive multi-signature scheme can be constructed from an aggregate signature scheme if the same message is used for all signers.

2 Preliminaries

In this section, we describe the algebraic adversary, the discrete logarithm assumption, and the Schwartz-Zippel lemma for security proof our schemes.

Definition 2.1 (Algebraic Algorithm [11]). Let \mathbb{G} be a group with order p . We say that an algorithm \mathcal{A}_{alg} is algebraic if it satisfies the following requirements: whenever \mathcal{A}_{alg} outputs a group element $Z \in \mathbb{G}$, it also outputs a representation $\vec{z} = (z_1, \dots, z_\ell) \in \mathbb{Z}_p^\ell$ such that $Z = \prod_{k=1}^{\ell} V_k^{z_k}$ where V_1, \dots, V_ℓ are group elements that are given to \mathcal{A}_{alg} during its execution.

Assumption 1 (Discrete Logarithm). Let (p, \mathbb{G}) be a description of the group of prime order p . Let g be a generator of \mathbb{G} . The discrete logarithm (DL) assumption is that if the challenge values $D = (p, \mathbb{G}, g, g^x)$ are given, no PPT algorithm \mathcal{B} can compute x with more than a negligible advantage. The advantage of \mathcal{B} is defined as $\text{Adv}_{\mathcal{B}}^{DL}(\lambda) = \Pr[\mathcal{B}(D) = x]$ where the probability is taken over the random choice of $x \in \mathbb{Z}_p$.

Lemma 2.1 (Schwartz-Zippel Lemma [29]). Let $f(x_1, \dots, x_n)$ be a non-zero polynomial of total degree d . Let $S \subseteq \mathbb{F}$ be any finite set. Then if r_1, \dots, r_n are randomly chosen from S , $\Pr[f(r_1, \dots, r_n) = 0] \leq d/|S|$.

3 Public-Key Signature

In this section, we propose a new public key signature scheme by modifying the Okamoto signature scheme, and prove the security in the algebraic group model and random oracle model.

3.1 Definition

The syntax of public-key signature is generally composed of key generation, signing, and verification algorithms. We consider a cryptographic scheme in which multiple users use common public parameters, so we add a setup algorithm to generate public parameters. The detailed syntax of PKS is given as follows.

Definition 3.1 (Public-Key Signature). A public-key signature (PKS) scheme consists of four PPT algorithms **Setup**, **GenKey**, **Sign**, and **Verify**, which are defined as follows:

Setup(1^λ). The setup algorithm takes as input the security parameters λ in unary, and outputs public parameters PP .

GenKey(PP). The key generation algorithm takes as input public parameters PP , and outputs a private key SK and a public key PK .

Sign(SK, M). The signing algorithm takes as input a message M and a private key SK , and outputs a signature σ .

Verify(PK, σ, M). The verification algorithm takes as input a signature σ , a message M , and a public key PK , and outputs 1 if the signature is valid and 0 otherwise.

The correctness requirement is that for PP output by **Setup**(1^λ), any (SK, PK) output by **GenKey**(PP) and any M , we have that **Verify**($PK, \text{Sign}(SK, M), M$) = 1.

The standard security model of public-key signatures is the unforgeability under chosen message attack (UF-CMA). In this model, an attacker is initially given a challenge public key for attack, and can request a signature query for any message and receive a signature. Finally, the attacker outputs a forged signature for a message. The attacker is successful if the forged signature passes the verification algorithm and the message has not been queried before. The detailed security model of PKS is described as follows.

Definition 3.2 (Unforgeability). The security notion of a PKS scheme is unforgeability under chosen message attack (UF-CMA), which is defined in terms of the following experiment between a challenger \mathcal{C} and a PPT adversary \mathcal{A} :

1. **Setup**: \mathcal{C} first generates PP by running **Setup**(1^λ). Next, it obtains a key pair (SK, PK) by running **GenKey**(PP). It gives PK to \mathcal{A} .
2. **Signature Query**: \mathcal{A} adaptively requests a signature on a message M to sign under the challenge public key PK , and it receives a signature σ .
3. **Output**: Finally, \mathcal{A} outputs a forged signature σ^* on a message M^* under the public key PK . \mathcal{C} outputs 1 if the forged signature satisfies the following two conditions, or outputs 0 otherwise: 1) **Verify**(PK, σ^*, M^*) = 1, and 2) The corresponding message M^* must not have been queried by \mathcal{A} to the signing oracle.

The advantage of \mathcal{A} is defined as $\text{Adv}_{\mathcal{A}}^{\text{PKS}}(\lambda) = \Pr[\mathcal{C} = 1]$ where the probability is taken over all the randomness of the experiment. A PKS scheme is UF-CMA secure if all PPT adversaries have at most a negligible advantage in the above experiment.

3.2 Construction

Our PKS scheme is a modification of the PKS scheme of Okamoto [27]. The Okamoto PKS scheme generates a commitment as $R = g^{r_1} g_2^{r_2}$ where r_1 and r_2 are random exponents, but our PKS scheme generates a commitment as $R = (g^m h)^{r_1} (g_2^m h_2)^{r_2}$ to depend on a message m . This modification helps to simplify the security proof of our PKS scheme in the algebraic group model and it enables to construct a secure multi-signature scheme in the next section. The detailed description of our PKS scheme is given as follows:

PKS.Setup(1^λ): It first generates a cyclic group \mathbb{G} of prime order p where the bit size of p is $\Theta(\lambda)$. It generates two random generators $g, h \in \mathbb{G}$. It selects a random exponent $\alpha \in \mathbb{Z}_p$ and sets $g_2 = g^\alpha, h_2 = h^\alpha$. It chooses cryptographic hash functions H_1, H_2 such that $H_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ and $H_2 : \{0, 1\}^* \rightarrow \mathbb{Z}_p$. Finally, it outputs public parameters $PP = (p, \mathbb{G}, g, g_2, h, h_2, H_1, H_2)$.

PKS.GenKey(PP): It selects random $x_1, x_2 \in \mathbb{Z}_p$ and computes $X = g^{x_1} g_2^{x_2}, Y = h^{x_1} h_2^{x_2}$. It outputs a private key $SK = (PP, x_1, x_2)$ and a public key $PK = (PP, X, Y)$.

PKS.Sign(SK, M): Let $SK = (PP, x_1, x_2)$. It first calculates a hash $m = H_1(M)$. Next, it selects random $r_1, r_2 \in \mathbb{Z}_p$ and computes $R = (g^m h)^{r_1} (g_2^m h_2)^{r_2}$. It calculates $c = H_2(R, M)$ and computes $s_1 = r_1 + x_1 c \pmod p, s_2 = r_2 + x_2 c \pmod p$. It outputs a signature $\sigma = (c, s_1, s_2)$.

PKS.Verify(PK, σ, M): Let $\sigma = (c, s_1, s_2)$ and $PK = (PP, X, Y)$. It first calculates $m = H_1(M)$. Next, it computes $R = (g^m h)^{s_1} (g_2^m h_2)^{s_2} / (X^m Y)^c$ and checks that $c \stackrel{?}{=} H_2(R, M)$. If the equation holds, then it outputs 1. Otherwise, it outputs 0.

The correctness of this PKS scheme can be easily verified when $m = H_1(M)$ through the following equation

$$\begin{aligned} (g^m h)^{s_1} (g_2^m h_2)^{s_2} &= (g^m h)^{r_1 + x_1 c} (g_2^m h_2)^{r_2 + x_2 c} \\ &= (g^m h)^{r_1} (g_2^m h_2)^{r_2} (g^m h)^{x_1 c} (g_2^m h_2)^{x_2 c} \\ &= R \left((g^{x_1} g_2^{x_2})^m (h^{x_1} h_2^{x_2}) \right)^c \\ &= R (X^m Y)^c. \end{aligned}$$

3.3 Security Analysis

In order to prove the security of our PKS scheme under the discrete logarithm assumption, it is needed to devise a method to simulate the signature query requested by an adversary and to extract the discrete logarithm from a forged signature submitted by the adversary. One nice feature of Okamoto signature is that the signature simulation is very simple because a simulator chooses a private key itself and generates a signature by using the private key [27]. The signature simulation of our PKS scheme is also handled very simply as the same as that of Okamoto signature. In order to extract the discrete logarithm from the forged signature, we take advantage of the fact that an algebraic adversary additionally submits the representation of a group element when it submits the group element of the forged signature.

In a Schnorr-based signature scheme, a formula for discrete logarithm can be derived using the representation of a group element and the verification equation of the scheme, and the extraction of discrete logarithm is possible because the denominator of the formula is not zero with high probability due to the randomness of the random oracle model. However, unlike the Schnorr-based signature scheme, the Okamoto-based signature scheme additionally includes a signature element submitted by an adversary in the denominator of

the discrete logarithm-related formula, so it is difficult to analyze that the denominator is not zero by simply using the randomness of the random oracle model. To solve this problem, we divided adversaries into three types. In the case of type 1 and type 2 adversaries, the discrete logarithm problem is simply planted to enable the extraction of the discrete logarithm. And in the case of type 3 adversary, we will show that it is difficult for the adversary to submit a valid forged signature due to the restriction of the security model through probability analysis. The detailed security proof of our PKS scheme is given as follows:

Theorem 3.1. *The above PKS scheme is UF-CMA secure in the algebraic group model and the random oracle model if the DL assumption hold. That is, for any PPT algebraic adversary \mathcal{A}_{alg} , there exist a PPT algorithm \mathcal{B} such that $\text{Adv}_{\mathcal{A}_{alg}}^{\text{PKS}}(\lambda) \leq 2\text{Adv}_{\mathcal{B}}^{\text{DL}}(\lambda) + \text{negl}(\lambda)$.*

Proof. Suppose there exists an algebraic adversary \mathcal{A}_{alg} that forges the above PKS scheme with non-negligible advantage ε . A reduction algorithm \mathcal{B} that solves the DL assumption is given as input a challenge tuple $D = (p, \mathbb{G}, g, g^a)$. Then \mathcal{B} that interacts with \mathcal{A}_{alg} is described as follows:

Setup: The algorithm \mathcal{B} first chooses a random bit $b \in \{0, 1\}$ to guess the type of an adversary. If $b = 0$, then it selects a random exponent $h' \in \mathbb{Z}_p$ and sets $g_2 = g^a, h = g^{h'}, h_2 = g_2^{h'}$. Otherwise, it selects a random exponent $\alpha \in \mathbb{Z}_p$ and sets $g_2 = g^\alpha, h = g^a, h_2 = (g^a)^\alpha$. It sets public parameters $PP = (p, \mathbb{G}, g, g_2, h, h_2, H_1, H_2)$ where H_1 and H_2 are two hash functions that are modeled as random oracles. Next, it selects random exponents $x_1, x_2 \in \mathbb{Z}_p$ and computes $X = g^{x_1} g_2^{x_2}, Y = h^{x_1} h_2^{x_2}$. It keeps $SK = (PP, x_1, x_2)$ internally and gives $PK = (PP, X, Y)$ to \mathcal{A}_{alg} .

Hash Query: If \mathcal{A}_{alg} request an H_1 or H_2 hash query, then \mathcal{B} handles this query as follows:

- H_1 hash query for (M) : If $(M, \cdot) \in L_{H_1}$, then it retrieves (M, m) from L_{H_1} . Otherwise, it selects random $m \in \mathbb{Z}_p$ and adds (M, m) to L_{H_1} . It gives m to \mathcal{A}_{alg} .
- H_2 hash query for (R, M) : If $(R, M, \cdot) \in L_{H_2}$, then it retrieves (R, M, c) from L_{H_2} . Otherwise, it selects random $c \in \mathbb{Z}_p$ and adds (R, M, c) to L_{H_2} . It gives c to \mathcal{A}_{alg} .

Signature Query: If \mathcal{A}_{alg} request a signature query for a message M , then \mathcal{B} adds M to Q and generates a signature $\sigma = (c, s_1, s_2)$ by running **PKS.Sign** (SK, M) since it has SK . It gives σ to \mathcal{A}_{alg} . Recall that \mathcal{A}_{alg} is implicitly given a commitment $R = (g^m h)^{s_1} (g_2^m h_2)^{s_2} (X^m Y)^{-c}$ from the signature σ where $m = H_1(M)$.

Note that \mathcal{A}_{alg} is an algebraic adversary that when it requests hash queries with a group element $Z \in \mathbb{G}$, it also submits a representation $\vec{z} = (z_1, \dots, z_\ell)$ for the group element Z such that $Z = \prod_{i=1}^{\ell} V_i^{z_i}$ and $\{V_i\}$ are group elements given to \mathcal{A}_{alg} . For the simplicity of the notation, we do not describe representations for group elements in hash queries. We assume that the representations of group elements submitted by \mathcal{A}_{alg} are implicitly stored in the lists maintained by \mathcal{B} .

Output: Finally, \mathcal{A}_{alg} outputs a forged signature $\sigma^* = (c^*, s_1^*, s_2^*)$ on a message M^* . \mathcal{B} checks that **PKS.Verify** $(PK, \sigma^*, M^*) = 1$ and $M^* \notin Q$.

From the verification algorithm of the PKS scheme, it can derive the commitment group element R^* of σ^* by computing $R^* = (g^{m^*} h)^{s_1^*} (g_2^{m^*} h_2)^{s_2^*} (X^{m^*} Y)^{-c^*}$ where $m^* = H_1(M^*)$. Next, it finds the representation $\vec{z} = (z_1, \dots, z_6, z_{7,1}, \dots, z_{7,q_s})$ for the element R^* such as $R^* = g^{z_1} g_2^{z_2} h^{z_3} h_2^{z_4} X^{z_5} Y^{z_6} \prod_{k=1}^{q_s} (R^{(k)})^{z_{7,k}}$ where $X = g^{x_1} g_2^{x_2}, Y = h^{x_1} h_2^{x_2}$, and $R^{(k)} = (g^{m^{(k)}} h)^{r_1^{(k)}} (g_2^{m^{(k)}} h_2)^{r_2^{(k)}}$ is the commitment of k -th signature query. By

combining above equations, it can derive the following equation

$$\begin{aligned}
1 &= (g^{m^*} h)^{s_1^*} (g_2^{m^*} h_2)^{s_2^*} (X^{m^*} Y)^{-c^*} R^{*-1} \\
&= g^{m^* s_1^*} h^{s_1^*} g_2^{m^* s_2^*} h_2^{s_2^*} \left((g^{x_1} g_2^{x_2})^{m^*} (h^{x_1} h_2^{x_2}) \right)^{-c^*} \\
&\quad \left(g^{z_1} h^{z_3} g_2^{z_2} h_2^{z_4} (g^{x_1} g_2^{x_2})^{z_5} (h^{x_1} h_2^{x_2})^{z_6} \prod_{k=1}^{q_S} (g^{m^{(k)}} h)^{r_1^{(k)} z_{7,k}} (g_2^{m^{(k)}} h_2)^{r_2^{(k)} z_{7,k}} \right)^{-1} \\
&= g^{m^* (s_1^* - x_1 c^*)} h^{(s_1^* - x_1 c^*)} g_2^{m^* (s_2^* - x_2 c^*)} h_2^{(s_2^* - x_2 c^*)} \\
&\quad \left(g^{z_1 + x_1 z_5 + \sum_{k=1}^{q_S} m^{(k)} r_1^{(k)} z_{7,k}} h^{z_3 + x_1 z_6 + \sum_{k=1}^{q_S} r_1^{(k)} z_{7,k}} g_2^{z_2 + x_2 z_5 + \sum_{k=1}^{q_S} m^{(k)} r_2^{(k)} z_{7,k}} h_2^{z_4 + x_2 z_6 + \sum_{k=1}^{q_S} r_2^{(k)} z_{7,k}} \right)^{-1} \\
&= g^{A_1} h^{A_2} g_2^{B_1} h_2^{B_2}
\end{aligned}$$

where A_1, A_2, B_1 , and B_2 are variables defined as

$$\begin{aligned}
A_1 &:= m^* (s_1^* - x_1 c^*) - (z_1 + x_1 z_5 + \sum_{k=1}^{q_S} m^{(k)} r_1^{(k)} z_{7,k}), \\
A_2 &:= (s_1^* - x_1 c^*) - (z_3 + x_1 z_6 + \sum_{k=1}^{q_S} r_1^{(k)} z_{7,k}), \\
B_1 &:= m^* (s_2^* - x_2 c^*) - (z_2 + x_2 z_5 + \sum_{k=1}^{q_S} m^{(k)} r_2^{(k)} z_{7,k}), \\
B_2 &:= (s_2^* - x_2 c^*) - (z_4 + x_2 z_6 + \sum_{k=1}^{q_S} r_2^{(k)} z_{7,k}).
\end{aligned}$$

To solve the discrete logarithm, we classify algebraic adversaries into the following three types depending on the conditions of variables:

- Type-1: An algebraic adversary is Type-1 if $B_1 + \text{dlog}_g(h)B_2 \not\equiv 0 \pmod{p}$.
- Type-2: An algebraic adversary is Type-2 if $B_1 + \text{dlog}_g(h)B_2 \equiv 0 \pmod{p}$ and $B_2 \not\equiv 0 \pmod{p}$.
- Type-3: An algebraic adversary is Type-3 if $B_1 + \text{dlog}_g(h)B_2 \equiv 0 \pmod{p}$ and $B_2 \equiv 0 \pmod{p}$.

Let F be the event that an adversary succeeds to forge a multi-signature and T_i be the event that an adversary is Type- i . Since the random bit b is hidden to the adversary and b is independent to the type of the adversary, we have that $\Pr[b = 0 \wedge F|T_i] = \Pr[b = 1 \wedge F|T_i]$ for each type of the adversary. If the Type-1 adversary is successful to forge and the guess of the reduction algorithm is correct ($b = 0$), then the reduction can compute the discrete logarithm as $\text{dlog}_g(g_2) = -(A_1 + h'A_2)/(B_1 + h'B_2) \pmod{p}$ since $g_2 = g^a$ and $B_1 + h'B_2 \not\equiv 0 \pmod{p}$. That is, $\Pr[b = 0 \wedge F|T_1] \leq \mathbf{Adv}_B^{DL}(\lambda)$. If the Type-2 adversary is successful to forge and the guess of the reduction is correct ($b = 1$), then the reduction can compute the discrete logarithm as $\text{dlog}_{g_2}(h_2) = -B_1/B_2 \pmod{p}$ since $g_2 = g^a, h_2 = (g^a)^\alpha, B_1 + \text{dlog}_g(h)B_2 \equiv 0 \pmod{p}$, and $B_2 \not\equiv 0 \pmod{p}$. That is, $\Pr[b = 1 \wedge F|T_2] \leq \mathbf{Adv}_B^{DL}(\lambda)$. From Lemma 3.3, the probability of the Type-3 adversary to successfully forge is negligible. That is, $\Pr[F|T_3] \leq \text{negl}(\lambda)$. Therefore, we obtain the following

result

$$\begin{aligned}
\mathbf{Adv}_{\mathcal{A}_{alg}}^{UF-CMA}(\lambda) &= \Pr[F \wedge T_1] + \Pr[F \wedge T_2] + \Pr[F \wedge T_3] \\
&= \Pr[T_1] \Pr[F|T_1] + \Pr[T_2] \Pr[F|T_2] + \Pr[T_3] \Pr[F|T_3] \\
&= \Pr[T_1] (\Pr[b = 0 \wedge F|T_1] + \Pr[b = 1 \wedge F|T_1]) + \\
&\quad \Pr[T_2] (\Pr[b = 0 \wedge F|T_2] + \Pr[b = 1 \wedge F|T_2]) + \Pr[T_3] \Pr[F|T_3] \\
&\leq \Pr[T_1] 2\mathbf{Adv}_{\mathcal{B}}^{DL}(\lambda) + \Pr[T_2] 2\mathbf{Adv}_{\mathcal{B}}^{DL}(\lambda) + \Pr[T_3] \text{negl}(\lambda) \\
&\leq \Pr[T_1] 2\mathbf{Adv}_{\mathcal{B}}^{DL}(\lambda) + (1 - \Pr[T_1]) 2\mathbf{Adv}_{\mathcal{B}}^{DL}(\lambda) + \text{negl}(\lambda) \\
&\leq 2\mathbf{Adv}_{\mathcal{B}}^{DL}(\lambda) + \text{negl}(\lambda).
\end{aligned}$$

This completes our proof. \square

Lemma 3.2. *In the above PKS scheme, the private key exponents (x_1, x_2) and random exponents $\{(r_1^{(k)}, r_2^{(k)})\}$ for signature queries are statistically hidden to an algebraic adversary.*

Proof. In order to show that the private key exponents (x_1, x_2) and random exponents $\{(r_1^{(k)}, r_2^{(k)})\}$ selected by the reduction algorithm are statistically hidden from the adversary, we should show that these exponents can be changed to different exponents $(\tilde{x}_1, \tilde{x}_2)$ and $\{(\tilde{r}_1^{(k)}, \tilde{r}_2^{(k)})\}$ while the public key group elements, the commitment group elements, and the signatures given to the adversary are fixed.

Let (X, Y) be the challenge public key. If the private key exponents (x_1, x_2) can be changed to different private key exponents $(\tilde{x}_1, \tilde{x}_2)$, then we obtain the first relation $x_1 + \alpha x_2 \equiv \tilde{x}_1 + \alpha \tilde{x}_2 \pmod p$ from the following equation

$$\begin{aligned}
X &= (g^{x_1} g_2^{x_2}) = g^{x_1 + \alpha x_2} = g^{\tilde{x}_1 + \alpha \tilde{x}_2} = (g^{\tilde{x}_1} g_2^{\tilde{x}_2}), \\
Y &= (h^{x_1} h_2^{x_2}) = h^{x_1 + \alpha x_2} = h^{\tilde{x}_1 + \alpha \tilde{x}_2} = (h^{\tilde{x}_1} h_2^{\tilde{x}_2}).
\end{aligned}$$

Let $R^{(k)}$ be the commitment element of the k -th signature query. If the random exponents $(r_1^{(k)}, r_2^{(k)})$ can be changed to different random exponents $(\tilde{r}_1^{(k)}, \tilde{r}_2^{(k)})$, then we obtain the second relation $r_1^{(k)} + \alpha r_2^{(k)} \equiv \tilde{r}_1^{(k)} + \alpha \tilde{r}_2^{(k)} \pmod p$ from the following equation

$$\begin{aligned}
R^{(k)} &= (g^{m^{(k)}} h)^{r_1^{(k)}} (g_2^{m^{(k)}} h_2)^{r_2^{(k)}} = g^{(m^{(k)} + h')(r_1^{(k)} + \alpha r_2^{(k)})} \\
&= g^{(m^{(k)} + h')(\tilde{r}_1^{(k)} + \alpha \tilde{r}_2^{(k)})} = (g^{m^{(k)}} h)^{\tilde{r}_1^{(k)}} (g_2^{m^{(k)}} h_2)^{\tilde{r}_2^{(k)}}.
\end{aligned}$$

Let $(s_1^{(k)}, s_2^{(k)})$ be the signature of the k -th signature query where $s_1^{(k)} = r_1^{(k)} + x_1 c^{(k)}$ and $s_2^{(k)} = r_2^{(k)} + x_2 c^{(k)}$. If the random exponents (x_1, x_2) and $(r_1^{(k)}, r_2^{(k)})$ can be changed to different random exponents $(\tilde{x}_1, \tilde{x}_2)$ and $(\tilde{r}_1^{(k)}, \tilde{r}_2^{(k)})$, then we obtain the following third and fourth relations

$$\begin{aligned}
r_1^{(k)} + x_1 c^{(k)} &\equiv \tilde{r}_1^{(k)} + \tilde{x}_1 c^{(k)} \pmod p, \\
r_2^{(k)} + x_2 c^{(k)} &\equiv \tilde{r}_2^{(k)} + \tilde{x}_2 c^{(k)} \pmod p.
\end{aligned}$$

Now, we argue that new private key exponents and new random exponents can satisfy the above four relations and these exponents are different with the original exponents. From the above first, second, and

third relations, we set the new exponents as follows

$$\begin{aligned}\tilde{x}_1 &\leftarrow \mathbb{Z}_p^*, \\ \tilde{x}_2 &:= x_2 + (x_1 - \tilde{x}_1)\alpha^{-1} \pmod{p}, \\ \tilde{r}_1^{(k)} &:= r_1^{(k)} + (x_1 - \tilde{x}_1)c^{(k)} \pmod{p}, \\ \tilde{r}_2^{(k)} &:= r_2^{(k)} + (r_1^{(k)} - \tilde{r}_1^{(k)})\alpha^{-1} \pmod{p}.\end{aligned}$$

Next, we show that these new exponents satisfy the fourth relation as follows

$$\begin{aligned}r_2^{(k)} - \tilde{r}_2^{(k)} + x_2c^{(k)} - \tilde{x}_2c^{(k)} \\ \equiv -(r_1^{(k)} - \tilde{r}_1^{(k)})\alpha^{-1} - (x_1 - \tilde{x}_1)\alpha^{-1}c^{(k)} \\ \equiv -((r_1^{(k)} - \tilde{r}_1^{(k)}) + (x_1 - \tilde{x}_1)c^{(k)})\alpha^{-1} \\ \equiv 0 \pmod{p}.\end{aligned}$$

This completes our proof. □

Lemma 3.3. *If the algebraic adversary is Type-3, then the advantage of the adversary in UF-CMA game is negligible.*

Proof. From Theorem 3.1, we have the equation $g^{A_1}h^{A_2}g_2^{B_1}h_2^{B_2} = 1$ where variables B_1 and B_2 are defined as follows

$$\begin{aligned}B_1 &:= m^*(s_2^* - x_2c^*) - (z_2 + x_2z_5 + \sum_{k=1}^{q_S} m^{(k)}r_2^{(k)}z_{7,k}), \\ B_2 &:= (s_2^* - x_2c^*) - (z_4 + x_2z_6 + \sum_{k=1}^{q_S} r_2^{(k)}z_{7,k}).\end{aligned}$$

Now, we analyze the conditions to satisfy $B_2 \equiv 0 \pmod{p}$. From Lemma 3.2, we know that x_2 and $\{r_2^{(k)}\}$ are statistically hidden to the adversary. To satisfy $B_2 \equiv 0 \pmod{p}$, the term x_2c^* of B_2 that is not directly controlled by the adversary should be cancelled out. To analyze this, we consider the following two cases:

- Case 1: Let BAD_1 be an event that x_2c^* is cancelled by $(z_4 + x_2z_6 + \sum_{k=1}^{q_S} r_2^{(k)}z_{7,k})$. Recall that the term $(z_4 + x_2z_6 + \sum_{k=1}^{q_S} r_2^{(k)}z_{7,k})$ is associated with the element R^* . In the signing algorithm, c^* is the output of a hash function H_2 that takes R^* as an input and H_2 is modeled as a random oracle. Thus, c^* is a random value independent of R^* by the property of the random oracle. This means that the probability of BAD_1 is at most $1/p$.
- Case 2: Let BAD_2 be the event that the term x_2c^* is cancelled by s_2^* . Recall that the term s_2^* is the output of the adversary as the forged signature and x_2 is statistically hidden to the adversary. The only way to cancel out this term is for the adversary to construct a forged signature by combining the simulated signatures $\{(s_1^{(k)}, s_2^{(k)})\}$ given from the signature queries since the reduction algorithm simply constructs a signature $s_2^{(k)} = r_2^{(k)} + x_2c^{(k)}$ by using the hidden private key element x_2 . In this case, the term $(s_2^* - x_2c^*)$ additionally contains a statistically hidden random exponent $r_2^{(k)}$ from the commitment $R^{(k)}$ for some k . Thus, there should exist an index $k \in \{1, \dots, q_S\}$ such that $z_{7,k} \neq 0$

mod p since \mathcal{A}_{alg} is an algebraic adversary that submits a group element with a representation of group elements given to the adversary.

From the conditions $B_1 + \text{dlog}_g(h)B_2 \equiv 0 \pmod{p}$ and $B_2 \equiv 0 \pmod{p}$ of the Type-3 adversary, we have that $B_1 \equiv B_2 \equiv 0 \pmod{p}$. By combining B_1 and B_2 , we have the following equation

$$-B_1 + m^*B_2 \equiv \left(z_2 + x_2z_5 - m^*(z_4 + x_2z_6) + \sum_{k=1}^{q_S} (m^{(k)} - m^*)r_2^{(k)}z_{7,k} \right) \equiv 0 \pmod{p}.$$

Since $z_{7,k} \not\equiv 0 \pmod{p}$ for some k and $r_2^{(k)}$ is statistically hidden to the adversary, the above equation can be reshaped as a degree-one polynomial $C_1r_2^{(k)} + C_0 \equiv 0 \pmod{p}$ where a coefficient C_1 is expressed as $C_1 = (m^{(k)} - m^*)z_{7,k}$. By the Schwartz-Zippel lemma, the probability of the above polynomial to be zero is at most $1/p$ if $r_2^{(k)}$ is randomly selected and $C_1 \not\equiv 0 \pmod{p}$. By the restrictions of the security model 3.2, we have $M^* \notin Q$. Thus the probability that $m^{(k)} - m^* \equiv 0 \pmod{p}$ for some k when $M^* \notin Q$ is bounded by q_S/p since H_1 is modeled as a random oracle. This means that the probability of BAD_2 is at most $(q_S + 1)/p$.

The success probability of the adversary is bounded by the probability of all bad events and the probability of all bad events are bounded as

$$\Pr[BAD] \leq \Pr[BAD_1] + \Pr[BAD_2] \leq (q_{S_1} + 2)/p.$$

This completes our proof. □

3.4 Discussion

Multi-User Security. In the security proof, we analyzed the security of our PKS scheme in the single-user setting. In the multi-user setting, many public keys PK_1, \dots, PK_n are given to an adversary where n is bounded by a polynomial, and the adversary forges a signature for one of these public keys. In general, a PKS scheme which provides the single-user security also satisfies the multi-user security, but the security reduction is not tight since it has a loss of a factor n . In the security proof of our PKS scheme, a simulator can freely select the private key of each user. Thus, it is possible to prove the multi-user security our PKS scheme with tight proof.

4 Multi-Signature

In this section, we propose a two-round multi-signature scheme supporting public-key aggregation based on the PKS scheme in the previous section and prove that it is secure in the discrete logarithm assumption in the AGM and ROM.

4.1 Definition

Multi-signature (MS) is a special kind of PKS in which multiple signers participate to generate a multi-signature for a message, and the multi-signature can be verified by using all public keys of the signers participated in the signature generation. We define the syntax of MS that supports the aggregation public keys of the signers into a single public key. The detailed syntax of MS supporting public key aggregation is given as follows.

Definition 4.1 (Multi-Signature). A multi-signature (MS) scheme with key aggregation consists of five PPT algorithms **Setup**, **GenKey**, **AggKey**, **Sign**, and **Verify**, which are defined as follows:

Setup(1^λ). The setup algorithm takes as input the security parameters λ in unary, and outputs public parameters PP .

GenKey(PP). The key generation algorithm takes as input public parameters PP , and outputs a private key SK and a public key PK .

AggKey(LK). The key aggregation algorithm takes as input a list of public keys $LK = (PK_1, \dots, PK_n)$, and outputs an aggregated public key AK .

Sign(SK_i, LK, M). The signing algorithm takes as input a private key SK_i , a list of public keys LK , and a message M , and outputs a multi-signature σ .

Verify(LK, σ, M). The verification algorithm takes as input a list of public keys LK , a signature σ , and a message M , and outputs either 1 or 0 depending on the validity of the signature.

The correctness requirement is that for PP output by **Setup**(1^λ), (SK_i, PK_i) output by **GenKey**(PP), and any M , we have that **Verify**($LK, \text{Sign}(SK_i, LK, M), M$) = 1.

The security model of MS extends the standard security model of PKS to the multi-user setting, which is called the plain public-key model [4]. In this plain public-key model, it is possible for an attacker to freely select the public keys of co-signers except the target public key. Because of this relaxation, the attacker can create a fake public key without knowing the private key of that public key, which is called a rogue-key attack. Additionally, if the signing protocol is composed of multiple rounds, the attacker can request parallel signing queries for multiple signatures when querying the signature. Finally, the attacker succeeds in forgery if the target public key is included in the final multi-signature and a target message not been queried before in the signing queries. The detailed definition of the plain security model of MS is defined as follows.

Definition 4.2 (Unforgeability). The security notion of an MS scheme in the plain public-key model is unforgeability under a chosen message attack (MS-UF-CMA), which is defined in terms of the following experiment between a challenger \mathcal{C} and a PPT adversary \mathcal{A} :

1. **Setup**: \mathcal{C} obtains public parameters PP by running **Setup**(1^λ) and obtains a challenge key pair (SK^*, PK^*) by running **GenKey**(PP). It gives PK^* to \mathcal{A} .
2. **Signature Query**: \mathcal{A} adaptively requests a multi-signature on a message M to sign under the challenge public key PK , and it receives a multi-signature σ .
3. **Output**: Finally, \mathcal{A} outputs a forged multi-signature σ^* on a message M^* under public keys $LK^* = (PK_1, \dots, PK_n)$. \mathcal{C} outputs 1 if the forged multi-signature satisfies the following three conditions, or outputs 0 otherwise: 1) **Verify**(LK^*, σ^*, M^*) = 1, 2) The challenge public key PK^* must exist in LK^* , and 3) The message M^* must not have been queried by \mathcal{A} to the signing oracle.

The advantage of \mathcal{A} is defined as $\text{Adv}_{\mathcal{A}}^{\text{MS}}(\lambda) = \Pr[\mathcal{C} = 1]$ where the probability is taken over all the randomness of the experiment. An MS scheme is MS-UF-CMA secure if all PPT adversaries have at most a negligible advantage in the above experiment.

4.2 Construction

In order to design a secure two-round MS scheme, it is necessary to design a method to be secure against the Wagner algorithm using parallel signing query as shown in previous studies [9]. To do this, we change the random commitment element dependent on the signature message. That is, a commitment element is formed as $R = (g^m h)^{r_1} (g_2^m h_2)^{r_2}$ where m is a message and r_1, r_2 are random exponents. The advantage of message dependent commitment like this is that even if an attacker obtains multiple commitment elements by requesting parallel signing queries, it is difficult for the attacker to derive another commitment for a new message because the commitment elements can be converted only for the same message. To perform such a commitment, the MS scheme needs to include h and h_2 elements in addition to g and g_2 elements in public parameters. Each private key of a user is set to the same x_1 and x_2 field elements as the Okamoto signature, the public key is set to $X = g^{x_1} g_2^{x_2}$ and $Y = h^{x_1} h_2^{x_2}$ due to additional public parameters. Note that the public key of the Okamoto signature scheme consists of one group element X , but our MS scheme consists of two group elements X and Y . The method of supporting the public key aggregation follows the previous method [24], and the aggregated public key consists of two group elements. The detailed description of our MS scheme is given as follows:

MS.Setup(1^λ): It first generates a cyclic group \mathbb{G} of prime order p of bit size $\Theta(\lambda)$. It chooses random generators $g, h \in \mathbb{G}$. It selects a random exponent $\alpha \in \mathbb{Z}_p$ and sets $g_2 = g^\alpha, h_2 = h^\alpha$. Next, it chooses cryptographic hash functions H_1, H_2, H_3 such that $H_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_p, H_2 : \{0, 1\}^* \rightarrow \mathbb{Z}_p,$ and $H_3 : \{0, 1\}^* \rightarrow \mathbb{Z}_p$. It outputs public parameters $PP = (p, \mathbb{G}, g, g_2, h, h_2, H_1, H_2, H_3)$.

MS.GenKey(PP): It chooses random $x_{i,1}, x_{i,2} \in \mathbb{Z}_p$ and computes $X_i = g^{x_{i,1}} g_2^{x_{i,2}}, Y_i = h^{x_{i,1}} h_2^{x_{i,2}}$. It outputs a private key $SK_i = (PP, x_{i,1}, x_{i,2})$ and a public key $PK_i = (PP, X_i, Y_i)$.

MS.AggKey(LK): Let $LK = (PK_1, \dots, PK_n)$ be the list of public keys where $PK_i = (PP, X_i, Y_i)$. It calculates $a_i = H_3(LK, PK_i)$ for all $i \in [n]$. Next, it computes $AX = \prod_{i=1}^n X_i^{a_i}$ and $AY = \prod_{i=1}^n Y_i^{a_i}$. It outputs an aggregated public key $AK = (PP, AX, AY)$.

MS.Sign(SK_i, LK, M): Let $SK_i = (PP, x_{i,1}, x_{i,2})$ and $LK = (PK_1, \dots, PK_n)$ where $PK_i = (PP, X_i, Y_i)$. It obtains AK by running **MS.AggKey**(LK) and calculates $a_i = H_3(LK, PK_i)$.

1. It calculates a hash $m = H_1(M)$. It selects random $r_{i,1}, r_{i,2} \in \mathbb{Z}_p$ and computes $R_i = (g^m h)^{r_{i,1}} (g_2^m h_2)^{r_{i,2}}$. Next, it broadcasts R_i to all co-signers.
2. It receives $\{R_j\}_{1 \leq j \neq i \leq n}$ from the co-signers. It computes $AR = \prod_{i=1}^n R_i$. It calculates $c = H_2(AK, AR, M)$. Next, it computes $s_{i,1} = r_{i,1} + x_{i,1} a_i c \pmod p$ and $s_{i,2} = r_{i,2} + x_{i,2} a_i c \pmod p$. It broadcasts $(s_{i,1}, s_{i,2})$ to all co-signers.
3. It receives $\{(s_{j,1}, s_{j,2})\}_{1 \leq j \neq i \leq n}$ from the co-signers. Next, it sets $s_1 = \sum_{i=1}^n s_{i,1} \pmod p$ and $s_2 = \sum_{i=1}^n s_{i,2} \pmod p$. It outputs a multi-signature $\sigma = (c, s_1, s_2)$.

MS.Verify(LK, σ, M): Let $\sigma = (c, s_1, s_2)$ be a multi-signature on a message M under the list of public keys LK . It obtains $AK = (PP, AX, AY)$ by running **MS.AggKey**(LK). It calculates a hash $m = H_1(M)$. Next, it derives

$$AR = (g^m h)^{s_1} (g_2^m h_2)^{s_2} / (AX^m AY)^c$$

and checks that $c \stackrel{?}{=} H_2(AK, AR, M)$. If the equation holds, then it outputs 1. Otherwise, it outputs 0.

The correctness of this MS scheme can be easily verified when $m = H_1(M)$ through the following equation

$$\begin{aligned}
(g^m h)^{s_1} (g_2^m h_2)^{s_2} &= (g^m h)^{\sum_{i=1}^n (r_{i,1} + x_{i,1} a_i c)} (g_2^m h_2)^{\sum_{i=1}^n (r_{i,2} + x_{i,2} a_i c)} \\
&= (g^m h)^{\sum_{i=1}^n r_{i,1}} (g_2^m h_2)^{\sum_{i=1}^n r_{i,2}} (g^m h)^{\sum_{i=1}^n x_{i,1} a_i c} (g_2^m h_2)^{\sum_{i=1}^n x_{i,2} a_i c} \\
&= \prod_{i=1}^n \left((g^m h)^{r_{i,1}} (g_2^m h_2)^{r_{i,2}} \right) \cdot \prod_{i=1}^n \left((g^m h)^{x_{i,1} a_i} (g_2^m h_2)^{x_{i,2} a_i} \right)^c \\
&= \prod_{i=1}^n R_i \cdot \prod_{i=1}^n \left((g^{x_{i,1}} g_2^{x_{i,2}})^{a_i m} (h^{x_{i,1}} h_2^{x_{i,2}})^{a_i} \right)^c \\
&= \prod_{i=1}^n R_i \cdot \left(\prod_{i=1}^n X_i^{a_i m} \prod_{i=1}^n Y_i^{a_i} \right)^c \\
&= AR \cdot (AX^m AY)^c.
\end{aligned}$$

4.3 Security Analysis

The overall strategy to prove the security of our MS scheme is mostly similar to the strategy to prove the security of our PKS scheme in the previous section. That is, the simulator of security proof processes the signature query of an adversary by using a self-selected private key, and divides the algebraic adversaries into three types to derive discrete logarithms from the forged signature of the adversary. First, in the case of type 1 and type 2 adversaries, if the challenge element of the discrete logarithm assumption is embedded in g_2 and h respectively, it is possible for the simulator to extract the discrete logarithm without difficulty by using the forged signature submitted by the adversary and the representation of a group element in the forged signature. The difficult part of the proof is to show that a type 3 adversary has a negligible probability of succeeding in forgery. To do this, we take advantage of the fact that the private key elements and the random commitment exponents are statistically hidden from the adversary, which is the important characteristic of Okamoto signature. In this case, if we use the condition that the target message m^* is different from a message $m^{(k)}$ queried in the signing query which is the security constraint of the MS-UF-CMA security model. In this case, it can be shown that the probability of successful forgery of the type 3 adversary is negligible. Additionally, in the case of our MS scheme, the type 3 adversary analysis is somewhat complicated because the final commitment element is aggregated from individual commitment elements generated by co-signers. The detailed security proof of our MS scheme is given as follows:

Theorem 4.1. *The above MS scheme is MS-UF-CMA secure in the algebraic group model if the DL assumption holds. That is, for any PPT algebraic adversary \mathcal{A}_{alg} , there exist PPT algorithms \mathcal{B} such that $\text{Adv}_{\mathcal{A}_{alg}}^{MS}(\lambda) \leq 2\text{Adv}_{\mathcal{B}}^{DL}(\lambda) + \text{negl}(\lambda)$.*

Proof. Suppose there exists an algebraic adversary \mathcal{A}_{alg} that forges the above MS scheme with non-negligible advantage ε . A reduction algorithm \mathcal{B} that solves the DL assumption is given as input a challenge tuple $D = (p, \mathbb{G}, g, g^a)$. Then \mathcal{B} that interacts with \mathcal{A}_{alg} is described as follows:

Setup: The algorithm \mathcal{B} first chooses a random bit $b \in \{0, 1\}$ to guess the type of an adversary. If $b = 0$, then it selects a random exponent $h' \in \mathbb{Z}_p$ and sets $g_2 = g^a, h = g^{h'}, h_2 = g_2^{h'}$. Otherwise, it selects a random exponent $\alpha \in \mathbb{Z}_p$ and sets $g_2 = g^\alpha, h = g^a, h_2 = (g^a)^\alpha$. It sets public parameters $PP = (p, \mathbb{G}, g, g_2, h, h_2, H_1, H_2)$ where H_1 and H_2 are hash functions that are modeled as random oracles. Next, it selects random exponents $x_1^*, x_2^* \in \mathbb{Z}_p$ and computes $X^* = g^{x_1^*} g_2^{x_2^*}, Y^* = h^{x_1^*} h_2^{x_2^*}$. It sets a challenge private key $SK^* = (PP, x_1^*, x_2^*)$ and a challenge public key $PK^* = (PP, X^*, Y^*)$. It keeps SK^* internally and gives $PK_1 = PK^*$ to \mathcal{A}_{alg} .

Hash Query: If \mathcal{A}_{alg} request an H_1 , H_2 , or H_3 hash query, then \mathcal{B} handles this query as follows:

- H_1 hash query for (M) : If $(M, \cdot) \in L_{H_1}$, then it retrieves (M, m) from L_{H_1} . Otherwise, it selects random $m \in \mathbb{Z}_p$ and adds (M, m) to L_{H_1} . It gives m to \mathcal{A}_{alg} .
- H_2 hash query for (AK, AR, M) : If $(AK, AR, M, \cdot) \in L_{H_2}$, then it retrieves (AK, AR, M, c) from L_{H_2} . Otherwise, it selects random $c \in \mathbb{Z}_p$ and adds (AK, AR, M, c) to L_{H_2} . It gives c to \mathcal{A}_{alg} .
- H_3 hash query for (LK, PK_i) : If $(LK, PK_i, \cdot) \in L_{H_3}$, then it retrieves (LK, PK_i, a_i) from L_{H_3} . Otherwise, it selects random $a_i \in \mathbb{Z}_p$ and adds (LK, PK_i, a_i) to L_{H_3} . It gives a_i to \mathcal{A}_{alg} .

Signature Query: If \mathcal{A}_{alg} request a first round or second round signature query, then \mathcal{B} handles this query as follows:

- First round signature query for (M) : It adds M to Q and calculates $m = H_1(M)$. It selects random exponents $r_{1,1}, r_{1,2} \in \mathbb{Z}_p$ and computes $R_1 = (g^m h)^{r_{1,1}} (g_2^m h_2)^{r_{1,2}}$. It adds $(M, R_1, r_{1,1}, r_{1,2})$ to L_{S_1} . It gives R_1 to \mathcal{A}_{alg} .
- Second round signature query for $(LK, M, \{R_i\}_{i=1}^n)$ where $LK = (PK_1, \dots, PK_n)$: If $(M, R_1, \cdot, \cdot) \notin L_{S_1}$ or $PK_1 \neq PK^*$, then it returns 0. It retrieves $(M, R_1, r_{1,1}, r_{1,2})$ from L_{S_1} . It computes $AR = \prod_{i=1}^n R_i$. It obtains AK by running **MS.AggKey** (LK) and calculates $a_1 = H_3(LK, PK_1)$. It calculates $c = H_2(AK, AR, M)$. Next, it computes $s_{1,1} = r_{1,1} + x_1^* a_1 c \pmod p$ and $s_{1,2} = r_{1,2} + x_2^* a_1 c \pmod p$. It adds $(LK, M, \{R_i\}_{i=1}^n, AR, c, s_{1,1}, s_{1,2}, r_{1,1}, r_{1,2})$ to L_{S_2} . It gives $(s_{1,1}, s_{1,2})$ to \mathcal{A}_{alg} .

Note that \mathcal{A}_{alg} is an algebraic adversary that when it requests hash and signature queries with a group element $Z \in \mathbb{G}$, it also submits a representation $\vec{z} = (z_1, \dots, z_\ell)$ for the group element Z such that $Z = \prod_{i=1}^\ell V_i^{z_i}$ and $\{V_i\}$ are group elements given to \mathcal{A}_{alg} . For the simplicity of the notation, we do not describe representations for group elements in hash and signature queries. We assume that the representations of group elements submitted by \mathcal{A}_{alg} are implicitly stored in the lists maintained by \mathcal{B} .

Output: Finally, \mathcal{A}_{alg} outputs a forged multi-signature $\sigma^* = (c^*, s_1^*, s_2^*)$ on a message M^* under a list of public keys $LK^* = (PK_1, \dots, PK_n)$. \mathcal{B} checks that **MS.Verify** $(LK^*, \sigma^*, M^*) = 1$, $PK_1 = PK^*$, and $M^* \notin Q$.

From the verification algorithm of the MS scheme, it can derive the following equation

$$\begin{aligned} AR^* &= (g^{m^*} h)^{s_1^*} (g_2^{m^*} h_2)^{s_2^*} (AX^{m^*} AY)^{-c^*} \\ &= g^{m^* s_1^*} h^{s_1^*} g_2^{m^* s_2^*} h_2^{s_2^*} \left(X_1^{a_1} \prod_{i=2}^n X_i^{a_i} \right)^{-m^* c^*} \left(Y_1^{a_1} \prod_{i=2}^n Y_i^{a_i} \right)^{-c^*} \end{aligned}$$

where $X_1 = g^{x_1^*} g_2^{x_2^*}$ and $Y_1 = h^{x_1^*} h_2^{x_2^*}$. Next, it finds representations $\vec{z} = (z_1, \dots, z_{7,1}, \dots, z_{7,q_{S_1}})$ for the group element AR^* , $\vec{u}^{(i)} = (u_1^{(i)}, \dots, u_{7,1}^{(i)}, \dots, u_{7,q_{S_1}}^{(i)})$ for the group element X_i , and $\vec{v}^{(i)} = (v_1^{(i)}, \dots, v_{7,1}^{(i)}, \dots, v_{7,q_{S_1}}^{(i)})$ for the group element Y_i such as

$$\begin{aligned} AR^* &= g^{z_1} g_2^{z_2} h^{z_3} h_2^{z_4} X_1^{z_5} Y_1^{z_6} \prod_{k=1}^{q_{S_1}} (R_1^{(k)})^{z_{7,k}}, \\ X_i &= g^{u_1^{(i)}} g_2^{u_2^{(i)}} h^{u_3^{(i)}} h_2^{u_4^{(i)}} X_1^{u_5^{(i)}} Y_1^{u_6^{(i)}} \prod_{k=1}^{q_{S_1}} (R_1^{(k)})^{u_{7,k}^{(i)}}, \\ Y_i &= g^{v_1^{(i)}} g_2^{v_2^{(i)}} h^{v_3^{(i)}} h_2^{v_4^{(i)}} X_1^{v_5^{(i)}} Y_1^{v_6^{(i)}} \prod_{k=1}^{q_{S_1}} (R_1^{(k)})^{v_{7,k}^{(i)}} \end{aligned}$$

where $R_1^{(k)} = (g^{m^{(k)}} h)^{r_{1,1}^{(k)}} (g_2^{m^{(k)}} h_2)^{r_{2,2}^{(k)}}$ is the commitment of k -th first round signature query. By combining above equations, it can derive the following simplified equation

$$g^{A_1} h^{A_2} g_2^{B_1} h_2^{B_2} = g^{A_1 + \text{dlog}_g(h)A_2} g_2^{B_1 + \text{dlog}_g(h)B_2} = 1$$

where A_1, A_2, B_1, B_2 are variables defined by the forged signature (c^*, s_1^*, s_2^*) , the representations $\vec{z}, \{\vec{u}^{(i)}, \vec{v}^{(i)}\}$, private key elements x_1^*, x_2^* , random exponents $\{r_{1,1}^{(k)}, r_{1,2}^{(k)}\}$, and message hashes $m^*, \{m^{(k)}\}$.

To solve the discrete logarithm, we classify algebraic adversaries into the following three types depending on the conditions of variables:

- Type-1: An algebraic adversary is Type-1 if $B_1 + \text{dlog}_g(h)B_2 \not\equiv 0 \pmod{p}$.
- Type-2: An algebraic adversary is Type-2 if $B_1 + \text{dlog}_g(h)B_2 \equiv 0 \pmod{p}$ and $B_2 \not\equiv 0 \pmod{p}$.
- Type-3: An algebraic adversary is Type-3 if $B_1 + \text{dlog}_g(h)B_2 \equiv 0 \pmod{p}$ and $B_2 \equiv 0 \pmod{p}$.

Let F be the event that an adversary succeeds to forge a multi-signature and T_i be the event that an adversary is Type- i . Since the random bit b is hidden to the adversary and b is independent to the type of the adversary, we have that $\Pr[b = 0 \wedge F|T_i] = \Pr[b = 1 \wedge F|T_i]$ for each type of the adversary. If the Type-1 adversary is successful to forge and the guess of the reduction algorithm is correct ($b = 0$), then the reduction can compute the discrete logarithm as $\text{dlog}_g(g_2) = -(A_1 + h'A_2)/(B_1 + h'B_2) \pmod{p}$ since $g_2 = g^a$ and $B_1 + h'B_2 \not\equiv 0 \pmod{p}$. That is, $\Pr[b = 0 \wedge F|T_1] \leq \mathbf{Adv}_{\mathcal{B}}^{DL}(\lambda)$. If the Type-2 adversary is successful to forge and the guess of the reduction is correct ($b = 1$), then the reduction can compute the discrete logarithm as $\text{dlog}_{g_2}(h_2) = -B_1/B_2 \pmod{p}$ since $g_2 = g^\alpha, h_2 = (g^\alpha)^\alpha, B_1 + \text{dlog}_g(h)B_2 \equiv 0 \pmod{p}$, and $B_2 \not\equiv 0 \pmod{p}$. That is, $\Pr[b = 1 \wedge F|T_2] \leq \mathbf{Adv}_{\mathcal{B}}^{DL}(\lambda)$. From Lemma 4.3, If the probability of the Type-3 adversary to successfully forge is negligible. That is, $\Pr[F|T_3] \leq \text{negl}(\lambda)$. Therefore, we obtain the following result

$$\begin{aligned} \mathbf{Adv}_{\mathcal{A}_{alg}}^{MS-UF-CMA}(\lambda) &= \Pr[F \wedge T_1] + \Pr[F \wedge T_2] + \Pr[F \wedge T_3] \\ &= \Pr[T_1] \Pr[F|T_1] + \Pr[T_2] \Pr[F|T_2] + \Pr[T_3] \Pr[F|T_3] \\ &= \Pr[T_1] (\Pr[b = 0 \wedge F|T_1] + \Pr[b = 1 \wedge F|T_1]) + \\ &\quad \Pr[T_2] (\Pr[b = 0 \wedge F|T_2] + \Pr[b = 1 \wedge F|T_2]) + \Pr[T_3] \Pr[F|T_3] \\ &\leq \Pr[T_1] 2\mathbf{Adv}_{\mathcal{B}}^{DL}(\lambda) + \Pr[T_2] 2\mathbf{Adv}_{\mathcal{B}}^{DL}(\lambda) + \Pr[T_3] \text{negl}(\lambda) \\ &\leq \Pr[T_1] 2\mathbf{Adv}_{\mathcal{B}}^{DL}(\lambda) + (1 - \Pr[T_1]) 2\mathbf{Adv}_{\mathcal{B}}^{DL}(\lambda) + \text{negl}(\lambda) \\ &\leq 2\mathbf{Adv}_{\mathcal{B}}^{DL}(\lambda) + \text{negl}(\lambda). \end{aligned}$$

This completes our proof. \square

Lemma 4.2. *In the above MS scheme, the private key exponents (x_1^*, x_2^*) and random exponents $\{(r_{1,1}^{(k)}, r_{1,2}^{(k)})\}$ for first round signature queries are statistically hidden to an algebraic adversary.*

Proof. In order to show that the private key exponents (x_1^*, x_2^*) and random exponents $\{(r_{1,1}^{(k)}, r_{1,2}^{(k)})\}$ selected by the reduction algorithm are statistically hidden from the adversary, we should show that these exponents can be changed to different exponents $(\tilde{x}_1^*, \tilde{x}_2^*)$ and $\{(\tilde{r}_{1,1}^{(k)}, \tilde{r}_{1,2}^{(k)})\}$ while the public key group elements, the commitment group elements, and the partial signatures given to the adversary are fixed.

Let (X_1, Y_1) be the challenge public key. If the private key exponents (x_1^*, x_2^*) can be changed to different private key exponents $(\tilde{x}_1^*, \tilde{x}_2^*)$, then we obtain the first relation $x_1^* + \alpha x_2^* \equiv \tilde{x}_1^* + \alpha \tilde{x}_2^* \pmod{p}$ from the following equation

$$\begin{aligned} X_1 &= (g^{x_1^*} g_2^{x_2^*}) = g^{x_1^* + \alpha x_2^*} = g^{\tilde{x}_1^* + \alpha \tilde{x}_2^*} = (g^{\tilde{x}_1^*} g_2^{\tilde{x}_2^*}), \\ Y_1 &= (h^{x_1^*} h_2^{x_2^*}) = h^{x_1^* + \alpha x_2^*} = h^{\tilde{x}_1^* + \alpha \tilde{x}_2^*} = (h^{\tilde{x}_1^*} h_2^{\tilde{x}_2^*}). \end{aligned}$$

Let $R_1^{(k)}$ be the commitment element of the k -th signature query. If the random exponents $(r_{1,1}^{(k)}, r_{1,2}^{(k)})$ can be changed to different random exponents $(\tilde{r}_{1,1}^{(k)}, \tilde{r}_{1,2}^{(k)})$, then we obtain the second relation $r_{1,1}^{(k)} + \alpha r_{1,2}^{(k)} \equiv \tilde{r}_{1,1}^{(k)} + \alpha \tilde{r}_{1,2}^{(k)} \pmod{p}$ from the following equation

$$\begin{aligned} R_1^{(k)} &= (g^{m^{(k)}} h)^{r_{1,1}^{(k)}} (g_2^{m^{(k)}} h_2)^{r_{1,2}^{(k)}} = g^{(m^{(k)} + h')(r_{1,1}^{(k)} + \alpha r_{1,2}^{(k)})} \\ &= g^{(m^{(k)} + h')(\tilde{r}_{1,1}^{(k)} + \alpha \tilde{r}_{1,2}^{(k)})} = (g^{m^{(k)}} h)^{\tilde{r}_{1,1}^{(k)}} (g_2^{m^{(k)}} h_2)^{\tilde{r}_{1,2}^{(k)}}. \end{aligned}$$

Let $(s_{1,1}^{(k)}, s_{1,2}^{(k)})$ be the partial signature of the k -th signature query where $s_{1,1}^{(k)} = r_{1,1}^{(k)} + x_1^* a_1 c^{(k)}$ and $s_{1,2}^{(k)} = r_{1,2}^{(k)} + x_2^* a_1 c^{(k)}$. If the random exponents (x_1^*, x_2^*) and $(r_{1,1}^{(k)}, r_{1,2}^{(k)})$ can be changed to different random exponents $(\tilde{x}_1^*, \tilde{x}_2^*)$ and $(\tilde{r}_{1,1}^{(k)}, \tilde{r}_{1,2}^{(k)})$, then we obtain the following third and fourth relations

$$\begin{aligned} r_{1,1}^{(k)} + x_1^* a_1 c^{(k)} &\equiv \tilde{r}_{1,1}^{(k)} + \tilde{x}_1^* a_1 c^{(k)} \pmod{p}, \\ r_{1,2}^{(k)} + x_2^* a_1 c^{(k)} &\equiv \tilde{r}_{1,2}^{(k)} + \tilde{x}_2^* a_1 c^{(k)} \pmod{p}. \end{aligned}$$

Now, we argue that new private key exponents and new random exponents can satisfy the above four relations and these exponents are different with the original exponents. From the above first, second, and third relations, we set the new exponents as follows

$$\begin{aligned} \tilde{x}_1^* &\leftarrow \mathbb{Z}_p^*, \\ \tilde{x}_2^* &:= x_2^* + (x_1^* - \tilde{x}_1^*) \alpha^{-1} \pmod{p}, \\ \tilde{r}_{1,1}^{(k)} &:= r_{1,1}^{(k)} + (x_1^* - \tilde{x}_1^*) a_1 c^{(k)} \pmod{p}, \\ \tilde{r}_{1,2}^{(k)} &:= r_{1,2}^{(k)} + (r_{1,1}^{(k)} - \tilde{r}_{1,1}^{(k)}) \alpha^{-1} \pmod{p}. \end{aligned}$$

Next, we show that these new exponents satisfy the fourth relation as follows

$$\begin{aligned} &r_{1,2}^{(k)} - \tilde{r}_{1,2}^{(k)} + x_2^* a_1 c^{(k)} - \tilde{x}_2^* a_1 c^{(k)} \\ &\equiv -(r_{1,1}^{(k)} - \tilde{r}_{1,1}^{(k)}) \alpha^{-1} - (x_1^* - \tilde{x}_1^*) \alpha^{-1} a_1 c^{(k)} \\ &\equiv -((r_{1,1}^{(k)} - \tilde{r}_{1,1}^{(k)}) + (x_1^* - \tilde{x}_1^*) a_1 c^{(k)}) \alpha^{-1} \\ &\equiv 0 \pmod{p}. \end{aligned}$$

This completes our proof. □

Lemma 4.3. *If the algebraic adversary is Type-3, then the advantage of the adversary in MS-UF-CMA game is negligible.*

Proof. Let AR^* be the group element derived from a forged multi-signature σ^* . From the verification algorithm, the forged signature $\sigma^* = (c^*, s_1^*, s_2^*)$ with the element AR^* satisfies the following equation

$$\begin{aligned}
1 &= (g^{m^*} h)^{s_1^*} (g_2^{m^*} h_2)^{s_2^*} AR^{*-1} (AX^{m^*} AY)^{-c^*} \\
&= g^{m^* s_1^*} h^{s_1^*} g_2^{m^* s_2^*} h_2^{s_2^*} AR^{*-1} \left(\prod_{i=1}^n X_i^{a_i} \right)^{-m^* c^*} \left(\prod_{i=1}^n Y_i^{a_i} \right)^{-c^*} \\
&= g^{m^* s_1^*} h^{s_1^*} g_2^{m^* s_2^*} h_2^{s_2^*} \cdot X_1^{-a_1 m^* c^*} Y_1^{-a_1 c^*} \cdot AR^{*-1} \cdot \prod_{i=2}^n X_i^{-m^* a_i c^*} \prod_{i=2}^n Y_i^{-a_i c^*} \\
&= g^{m^* s_1^*} h^{s_1^*} g_2^{m^* s_2^*} h_2^{s_2^*} \cdot (g^{x_1^*} g_2^{x_2^*})^{-a_1 m^* c^*} (h^{x_1^*} h_2^{x_2^*})^{-a_1 c^*} \cdot AR^{*-1} \cdot \prod_{i=2}^n X_i^{-m^* a_i c^*} \prod_{i=2}^n Y_i^{-a_i c^*} \\
&= g^{m^* s_1^* - m^* x_1^* a_1 c^*} h^{s_1^* - x_1^* a_1 c^*} g_2^{m^* s_2^* - m^* x_2^* a_1 c^*} h_2^{s_2^* - x_2^* a_1 c^*} \cdot AR^{*-1} \cdot \prod_{i=2}^n X_i^{-m^* a_i c^*} \prod_{i=2}^n Y_i^{-a_i c^*}.
\end{aligned}$$

Next, we find the representation $\vec{z} = (z_1, \dots, z_6, z_{7,1}, \dots, z_{7,k})$ from L_{H_2} for the group element AR^* such as

$$\begin{aligned}
AR^* &= g^{z_1} g_2^{z_2} h^{z_3} h_2^{z_4} X_1^{z_5} Y_1^{z_6} \prod_{k=1}^{q_{S_1}} (R_1^{(k)})^{z_{7,k}} \\
&= g^{z_1} g_2^{z_2} h^{z_3} h_2^{z_4} (g^{x_1^*} g_2^{x_2^*})^{z_5} (h^{x_1^*} h_2^{x_2^*})^{z_6} \prod_{k=1}^{q_{S_1}} ((g^{m^{(k)}} h)^{r_{1,1}^{(k)}} (g_2^{m^{(k)}} h_2)^{r_{1,2}^{(k)}})^{z_{7,k}} \\
&= g^{z_1} g_2^{z_2} h^{z_3} h_2^{z_4} g^{x_1^* z_5} g_2^{x_2^* z_5} h^{x_1^* z_6} h_2^{x_2^* z_6} g^{\sum_{k=1}^{q_{S_1}} m^{(k)} r_{1,1}^{(k)} z_{7,k}} h^{\sum_{k=1}^{q_{S_1}} r_{1,1}^{(k)} z_{7,k}} g_2^{\sum_{k=1}^{q_{S_1}} m^{(k)} r_{1,2}^{(k)} z_{7,k}} h_2^{\sum_{k=1}^{q_{S_1}} r_{1,2}^{(k)} z_{7,k}} \\
&= g^{z_1 + x_1^* z_5 + \sum_{k=1}^{q_{S_1}} m^{(k)} r_{1,1}^{(k)} z_{7,k}} h^{z_3 + x_1^* z_6 + \sum_{k=1}^{q_{S_1}} r_{1,1}^{(k)} z_{7,k}} g_2^{z_2 + x_2^* z_5 + \sum_{k=1}^{q_{S_1}} m^{(k)} r_{1,2}^{(k)} z_{7,k}} h_2^{z_4 + x_2^* z_6 + \sum_{k=1}^{q_{S_1}} r_{1,2}^{(k)} z_{7,k}}.
\end{aligned}$$

We can also find the representations $\vec{u}^{(i)} = (u_1^{(i)}, \dots, u_6^{(i)}, u_{7,1}^{(i)}, \dots, u_{7,k}^{(i)})$, $\vec{v}^{(i)} = (v_1^{(i)}, \dots, v_6^{(i)}, v_{7,1}^{(i)}, \dots, v_{7,k}^{(i)})$ from L_{H_2} for the group elements X_i, Y_i respectively such as

$$\begin{aligned}
X_i &= g^{u_1^{(i)}} g_2^{u_2^{(i)}} h^{u_3^{(i)}} h_2^{u_4^{(i)}} X_1^{u_5^{(i)}} Y_1^{u_6^{(i)}} \prod_{k=1}^{q_{S_1}} (R_1^{(k)})^{u_{7,k}^{(i)}} \\
&= g^{u_1^{(i)} + x_1^* u_5^{(i)} + \sum_{k=1}^{q_{S_1}} m^{(k)} r_{1,1}^{(k)} u_{7,k}^{(i)}} h^{u_3^{(i)} + x_1^* u_6^{(i)} + \sum_{k=1}^{q_{S_1}} r_{1,1}^{(k)} u_{7,k}^{(i)}} g_2^{u_2^{(i)} + x_2^* u_5^{(i)} + \sum_{k=1}^{q_{S_1}} m^{(k)} r_{1,2}^{(k)} u_{7,k}^{(i)}} h_2^{u_4^{(i)} + x_2^* u_6^{(i)} + \sum_{k=1}^{q_{S_1}} r_{1,2}^{(k)} u_{7,k}^{(i)}}, \\
Y_i &= g^{v_1^{(i)}} g_2^{v_2^{(i)}} h^{v_3^{(i)}} h_2^{v_4^{(i)}} X_1^{v_5^{(i)}} Y_1^{v_6^{(i)}} \prod_{k=1}^{q_{S_1}} (R_1^{(k)})^{v_{7,k}^{(i)}} \\
&= g^{v_1^{(i)} + x_1^* v_5^{(i)} + \sum_{k=1}^{q_{S_1}} m^{(k)} r_{1,1}^{(k)} v_{7,k}^{(i)}} h^{v_3^{(i)} + x_1^* v_6^{(i)} + \sum_{k=1}^{q_{S_1}} r_{1,1}^{(k)} v_{7,k}^{(i)}} g_2^{v_2^{(i)} + x_2^* v_5^{(i)} + \sum_{k=1}^{q_{S_1}} m^{(k)} r_{1,2}^{(k)} v_{7,k}^{(i)}} h_2^{v_4^{(i)} + x_2^* v_6^{(i)} + \sum_{k=1}^{q_{S_1}} r_{1,2}^{(k)} v_{7,k}^{(i)}}.
\end{aligned}$$

By combining above equations, we can derive the equation $g^{A_1} h^{A_2} g_2^{B_1} h_2^{B_2} = 1$ where variables B_1 and B_2 are

defined as follows

$$\begin{aligned}
B_1 &= m^*(s_2^* - x_2^* a_1 c^*) - \left(z_2 + x_2^* z_5 + \sum_{k=1}^{q_{S_1}} m^{(k)} r_{1,2}^{(k)} z_{7,k} \right) - \\
&\quad \sum_{i=2}^n \left(u_2^{(i)} + x_2^* u_5^{(i)} + \sum_{k=1}^{q_{S_1}} m^{(k)} r_{1,2}^{(k)} u_{7,k}^{(i)} \right) (m^* a_i c^*) - \sum_{i=2}^n \left(v_2^{(i)} + x_2^* v_5^{(i)} + \sum_{k=1}^{q_{S_1}} m^{(k)} r_{1,2}^{(k)} v_{7,k}^{(i)} \right) (a_i c^*), \\
B_2 &= (s_2^* - x_2^* a_1 c^*) - \left(z_4 + x_2^* z_6 + \sum_{k=1}^{q_{S_1}} r_{1,2}^{(k)} z_{7,k} \right) - \\
&\quad \sum_{i=2}^n \left(u_4^{(i)} + x_2^* u_6^{(i)} + \sum_{k=1}^{q_{S_1}} r_{1,2}^{(k)} u_{7,k}^{(i)} \right) (m^* a_i c^*) - \sum_{i=2}^n \left(v_4^{(i)} + x_2^* v_6^{(i)} + \sum_{k=1}^{q_{S_1}} r_{1,2}^{(k)} v_{7,k}^{(i)} \right) (a_i c^*).
\end{aligned}$$

Now, we analyze the conditions to satisfy $B_2 \equiv 0 \pmod{p}$. From Lemma 4.2, we know that x_2^* and $\{r_{1,2}^{(k)}\}$ are statistically hidden to the adversary. To satisfy $B_2 \equiv 0 \pmod{p}$, the term $x_2^* a_1 c^*$ of B_2 that is not directly controlled by the adversary should be cancelled out. To analyze this, we consider the following three cases:

- Case 1: Let BAD_1 be an event that $x_2^* a_1 c^*$ is cancelled by $(z_4 + x_2^* z_6 + \sum_{k=1}^{q_{S_1}} r_{1,2}^{(k)} z_{7,k})$. Recall that the term $(z_4 + x_2^* z_6 + \sum_{k=1}^{q_{S_1}} r_{1,2}^{(k)} z_{7,k})$ is associated with the group element AR^* . In the signing algorithm, c^* is the output of a hash function H_2 that takes AR^* as an input and H_2 is modeled as a random oracle. Thus, c^* is a random value independent of AR^* by the property of the random oracle. This means that the probability of BAD_1 is at most $1/p$.
- Case 2: Let BAD_2 be the event that $x_2^* a_1 c^*$ is cancelled by $\sum_{i=2}^n (u_4^{(i)} + x_2^* u_6^{(i)} + \sum_{k=1}^{q_{S_1}} r_{1,2}^{(k)} u_{7,k}^{(i)}) (m^* a_i c^*) + \sum_{i=2}^n (v_4^{(i)} + x_2^* v_6^{(i)} + \sum_{k=1}^{q_{S_1}} r_{1,2}^{(k)} v_{7,k}^{(i)}) (a_i c^*)$. Recall that the terms $(u_4^{(i)} + x_2^* u_6^{(i)} + \sum_{k=1}^{q_{S_1}} r_{1,2}^{(k)} u_{7,k}^{(i)})$ and $(v_4^{(i)} + x_2^* v_6^{(i)} + \sum_{k=1}^{q_{S_1}} r_{1,2}^{(k)} v_{7,k}^{(i)})$ are associated with the group elements X_i and Y_i respectively. In the key aggregation algorithm, a_1 is the output of a hash function H_2 that takes $LK = (PK_1, \dots, PK_n)$ and $PK_i = (PP, X_i, Y_i)$ as inputs and H_3 is modeled as a random oracle. Thus, a_1 is a random value independent of LK and PK_i by the property of the random oracle. This means that the probability of BAD_2 is at most $1/p$.
- Case 3: Let BAD_3 be the event that the term $x_2^* a_1 c^*$ is cancelled by s_2^* . Recall that the term s_2^* is the output of the adversary as the forged multi-signature and x_2^* is statistically hidden to the adversary. The only way to cancel out this term is for the adversary to construct a forged multi-signature by combining the simulated signatures $\{(s_{1,1}^{(k)}, s_{1,2}^{(k)})\}$ given from the second round signature queries since the reduction algorithm simply constructs a partial signature $s_2^{(k)} = r_{1,2}^{(k)} + x_2^* a_1 c^{(k)}$ by using the hidden private key element x_2^* . In this case, the term $(s_2^* - x_2^* a_1 c^*)$ additionally contains a statistically hidden random exponent $r_{1,2}^{(k)}$ from the commitment $R_1^{(k)}$ for some k . Thus, there should exist an index $k \in \{1, \dots, q_{S_1}\}$ such that $z_{7,k} \not\equiv 0 \pmod{p}$ since \mathcal{A}_{alg} is an algebraic adversary that submits a group element with a representation of group elements given to the adversary.

From the conditions $B_1 + \text{dlog}_g(h)B_2 \equiv 0 \pmod{p}$ and $B_2 \equiv 0 \pmod{p}$ of the Type-3 adversary, we

have that $B_1 \equiv B_2 \equiv 0 \pmod{p}$. By combining B_1 and B_2 , we have the following equation

$$\begin{aligned}
-B_1 + m^* B_2 &\equiv \left(z_2 + x_2^* z_5 - m^* (z_4 + x_2^* z_6) + \sum_{k=1}^{q_{S_1}} (m^{(k)} - m^*) r_{1,2}^{(k)} z_{7,k} \right) + \\
&\quad \sum_{i=2}^n \left(u_2^{(i)} + x_2^* u_5^{(i)} - m^* (u_4^{(i)} + x_2^* u_6^{(i)}) + \sum_{k=1}^{q_{S_1}} (m^{(k)} - m^*) r_{1,2}^{(k)} u_{7,k}^{(i)} \right) (m^* a_i c^*) + \\
&\quad \sum_{i=2}^n \left(v_2^{(i)} + x_2^* v_5^{(i)} - m^* (v_4^{(i)} + x_2^* v_6^{(i)}) + \sum_{k=1}^{q_{S_1}} (m^{(k)} - m^*) r_{1,2}^{(k)} v_{7,k}^{(i)} \right) (a_i c^*) \\
&\equiv 0 \pmod{p}.
\end{aligned}$$

Since $z_{7,k} \not\equiv 0 \pmod{p}$ for some k and $r_{1,2}^{(k)}$ is statistically hidden to the adversary, the above equation can be reshaped as a degree-one polynomial $C_1 r_{1,2}^{(k)} + C_0 \equiv 0 \pmod{p}$ where a coefficient C_1 is expressed as

$$C_1 = (m^{(k)} - m^*) \left(z_{7,k} + \sum_{i=2}^n u_{7,k}^{(i)} (m^* a_i c^*) + \sum_{i=2}^n v_{7,k}^{(i)} (a_i c^*) \right).$$

By the Schwartz-Zippel lemma, the probability of the above polynomial to be zero is at most $1/p$ if $r_{1,2}^{(k)}$ is randomly selected and $C_1 \not\equiv 0 \pmod{p}$. By the restrictions of the security model 4.2, we have $M^* \notin Q$. The probability that $m^{(k)} - m^* \equiv 0 \pmod{p}$ for some k when $M^* \notin Q$ is bounded by q_{S_1}/p since H_1 is modeled as a random oracle. The probability that $z_{7,k} + \sum_{i=2}^n u_{7,k}^{(i)} (m^* a_i c^*) + \sum_{i=2}^n v_{7,k}^{(i)} (a_i c^*) \equiv 0 \pmod{p}$ is bounded by $1/p$ since c^* is the output of H_2 when AR^* , X_i , and Y_i are given as inputs where $z_{7,k}, u_{7,k}^{(i)}, v_{7,k}^{(i)}$ are associated with AR^*, X_i, Y_i respectively. Thus the probability that $C_1 \equiv 0 \pmod{p}$ is bounded by $(q_{S_1} + 1)/p$. This means that the probability of BAD_3 is at most $(q_{S_1} + 2)/p$.

The success probability of the adversary is bounded by the probability of all bad events and the probability of all bad events are bounded as

$$\Pr[BAD] \leq \Pr[BAD_1] + \Pr[BAD_2] + \Pr[BAD_3] \leq (q_{S_1} + 4)/p.$$

This completes our proof. □

4.4 Discussion

Proof-of-Possession Multi-Signature. Our MS scheme requires $2n$ exponentiations to aggregate the public keys of co-signers. As suggested by the previous studies, if a public key additionally includes the proof of possession of a private key, it is possible to simply aggregate public keys by multiplying the public key elements without using expensive exponentiations. At this time, the security model that uses the proof of possession of the private key is a weaker model than the plain public-key model.

Synchronized Multi-Signature. If signers participating in multi-signature share the same information that is synchronized with each other, such as time or session count information, it is possible for co-signers to create a commitment by using the synchronized information instead of a message when creating a commitment. As an example, in the consensus protocol of a blockchain, the information of a previous block can be used as synchronization information. If such synchronized information exists, the signers can compute the commitment in advance and share it before the message to be signed is provided.

5 Conclusion

In this paper, we proposed a new PKS scheme and a two-round MS scheme by modifying the Okamoto signature scheme. And we proved the unforgeability of these PKS and MS schemes under the discrete logarithm assumption in the AGM and the non-programmable ROM. Our proposed MS scheme is the first two-round MS scheme with the security proof derived from Okamoto signature. An interesting open problem is to construct an efficient two-round MS scheme based on Okamoto signature in the standard model rather than in the AGM.

References

- [1] Handan Kiliç Alper and Jeffrey Burdges. Two-round trip Schnorr multi-signatures via delinearized witnesses. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021*, volume 12825 of *Lecture Notes in Computer Science*, pages 157–188. Springer, 2021.
- [2] Ali Bagherzandi, Jung Hee Cheon, and Stanislaw Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM Conference on Computer and Communications Security - CCS 2008*, pages 449–458. ACM, 2008.
- [3] Mihir Bellare and Wei Dai. Chain reductions for multi-signatures and the HBMS scheme. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021*, volume 13093 of *Lecture Notes in Computer Science*, pages 650–678. Springer, 2021.
- [4] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM Conference on Computer and Communications Security - CCS 2006*, pages 390–399. ACM, 2006.
- [5] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In Yvo Desmedt, editor, *Public-Key Cryptography - PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2003.
- [6] Alexandra Boldyreva, Craig Gentry, Adam O’Neill, and Dae Hyun Yum. Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. *Cryptology ePrint Archive*, Report 2007/438, 2010. <http://eprint.iacr.org/2007/438>.
- [7] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018*, volume 11273 of *Lecture Notes in Computer Science*, pages 435–464. Springer, 2018.
- [8] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, 2003.
- [9] Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs. On the security of two-round multi-signatures. In *IEEE Symposium on Security and Privacy, SP 2019*, pages 1084–1101. IEEE, 2019.

- [10] Manu Drijvers, Sergey Gorbunov, Gregory Neven, and Hoeteck Wee. Pixel: Multi-signatures for consensus. In Srđjan Capkun and Franziska Roesner, editors, *USENIX Security 2020*, pages 2093–2110. USENIX Association, 2020.
- [11] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018*, volume 10992 of *Lecture Notes in Computer Science*, pages 33–62. Springer, 2018.
- [12] Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ECDSA with fast trustless setup. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM Conference on Computer and Communications Security - CCS 2018*, pages 1179–1194. ACM, 2018.
- [13] Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve A. Schneider, editors, *Applied Cryptography and Network Security - ACNS 2016*, volume 9696 of *Lecture Notes in Computer Science*, pages 156–174. Springer, 2016.
- [14] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *J. Cryptol.*, 20(1):51–83, 2007.
- [15] Craig Gentry and Zulfikar Ramzan. Identity-based aggregate signatures. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public-Key Cryptography - PKC 2006*, volume 3958 of *Lecture Notes in Computer Science*, pages 257–273. Springer, 2006.
- [16] Jens Groth. Non-interactive distributed key generation and key resharing. Cryptology ePrint Archive, Paper 2021/339, 2021. <https://eprint.iacr.org/2021/339>.
- [17] Kobi Gurkan, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. Aggregatable distributed key generation. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology - EUROCRYPT 2021*, volume 12696 of *Lecture Notes in Computer Science*, pages 147–176. Springer, 2021.
- [18] Chelsea Komlo and Ian Goldberg. FROST: flexible round-optimized Schnorr threshold signatures. In Orr Dunkelman, Michael J. Jacobson Jr., and Colin O’Flynn, editors, *Selected Areas in Cryptography - SAC 2020*, volume 12804 of *Lecture Notes in Computer Science*, pages 34–65. Springer, 2020.
- [19] Yehuda Lindell. Fast secure two-party ECDSA signing. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017*, volume 10402 of *Lecture Notes in Computer Science*, pages 613–644. Springer, 2017.
- [20] Yehuda Lindell and Ariel Nof. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM Conference on Computer and Communications Security - CCS 2018*, pages 1837–1854. ACM, 2018.
- [21] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 465–485. Springer, 2006.

- [22] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. Sequential aggregate signatures from trapdoor permutations. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 74–90. Springer, 2004.
- [23] Changshe Ma, Jian Weng, Yingjiu Li, and Robert H. Deng. Efficient discrete logarithm based multi-signature scheme in the plain public key model. *Des. Codes Cryptogr.*, 54(2):121–133, 2010.
- [24] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple Schnorr multi-signatures with applications to Bitcoin. *Des. Codes Cryptogr.*, 87(9):2139–2164, 2019.
- [25] Jonas Nick, Tim Ruffing, and Yannick Seurin. MuSig2: Simple two-round Schnorr multi-signatures. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021*, volume 12825 of *Lecture Notes in Computer Science*, pages 189–221. Springer, 2021.
- [26] Jonas Nick, Tim Ruffing, Yannick Seurin, and Pieter Wuille. MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM Conference on Computer and Communications Security - CCS '20*, pages 1717–1731. ACM, 2020.
- [27] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53. Springer, 1992.
- [28] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptol.*, 4(3):161–174, 1991.
- [29] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.
- [30] Ewa Syta, Iulia Tamas, Dylan Visher, David Isaac Wolinsky, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ismail Khoffi, and Bryan Ford. Keeping authorities “honest or bust” with decentralized witness cosigning. In *IEEE Symposium on Security and Privacy - SP 2016*, pages 526–545. IEEE Computer Society, 2016.