International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 11, Number 2, pages 319–337, July 2021

A Tightly Secure DDH-based Multisignature with Public-Key Aggregation

Masayuki Fukumitsu
Faculty of Information Media
Hokkaido Information University
Nishi-Nopporo 59–2, Ebetsu, Hokkaido, 069–8585, Japan

Shingo Hasegawa
Graduate School of Information Sciences
Tohoku University
41 Kawauchi, Aoba-ku, Sendai, Miyagi, 980–8576, Japan

Received: February 3, 2021
Revised: April 30, 2021
Accepted: June 14, 2021
Communicated by Yasuyuki Nogami

Abstract

From the birth of the blockchain technology, multisignatures attract much attention as a tool for handling blockchain transactions. Concerning the application to the blockchain, multisignatures with public-key aggregation, which can compress public keys of signers to a single public key, is preferable to the standard multisignature because the public keys and the signature used in a transaction are stored to verify the transaction later. Several multisignature schemes with public key aggregation are proposed, however, there are no known schemes having a tight security reduction.

We propose a first multisignature with public-key aggregation whose security is proven to be tightly secure under the DDH assumption in the random oracle model. Our multisignature is based on the DDH-based multisignature by Le, Yang, and Ghorbani, however, our security proof is different from theirs. The idea of our security proof originates from another DDH-based multisignature by Le, Bonnecaze, and Gabillon whose security proof is tightly one. By tailoring their security proof to a setting which admits the public-key aggregation, we can prove the tight security of our multisignature.

Keywords: Multisignature, Key Aggregation, DDH Assumption, Tight Security, Blockchain

1 Introduction

The multisignature enables multiple signers to sign a common single message in an interactive manner. It attracts much attention as a tool for handling the blockchain transactions because the multisignature can authenticate the participation of multiple signers to the signature generation by a single signature. Since the size of a signature by multisignatures is independent of the number of signers generally, the storage for signatures can be much reduced rather than the case where each signer issues their own signature individually by using ordinary signatures.

Concerning the application to the blockchain, the multisignatures with *public-key aggregation* is preferable to the standard multisignture. The public-key aggregation property can compress the public keys of involving signers to a single *aggregated* public key as well as a signature. On the blockchain transaction, both signers' public keys and a signature are stored in the blockchain to verify the transaction. Thus the multisignature with public-key aggregation can reduce the storage for public keys used in the generation of a signature.

There exist multisignature schemes which support the public-key aggregation. Maxwell, Poelstra, Seurin, and Wuille [18] proposed the first multisignature with public-key aggregation based on the discrete logarithm (DL) assumption. Following their work, several multisignatures with public-key aggregation based on the DL assumption are constructed [7, 6, 13]. Not only the DL-based schemes but also lattice-based multisignatures with public-key aggregation are given [16, 10].

All multisignatures described above are proven to be secure in the random oracle model (ROM) [3] with respect to the plain public-key model [2], which does not require any assumption on the key generation. However, their security proofs are loose ones. Namely, the success probability of attacking a scheme is less than that of breaking the underlying cryptographic assumption with some polynomial factor, whereas the running time of attacking the scheme is almost the same as that of breaking the assumption. Such a polynomial loss factor requires us to set a long or large parameter of the scheme, and hence the size efficiency of the scheme becomes worse, even though a public key and a signature are aggregated. In this sense, the *tight security* is desirable, which implies that both probabilities of attacking the scheme or the assumption are almost the same as well as the running times. Thus a tight security proof makes the parameter of the scheme be compact as that of the underlying cryptographic assumption. Although several multisignature schemes with tight security have been proposed in [1, 19, 8, 9, 21], none of them support the public-key aggregation. Therefore, constructing a multisignature with public-key aggregation having a tight security proof is an important open question.

1.1 Our Contribution

We propose a first multisignature with public-key aggregation whose security is proven to be tightly secure. We show that the security of our scheme reduces to the DDH assumption via a tight security proof in ROM with respect to the plain public-key model. The construction of our multisignature is almost the same as the DDH-based multisignature (LYG multisignature) by [13]. However, our security proof is totally different from theirs. The idea of our proof originates from a multisignature by Le, Bonnecaze, and Gabillon [12] whose security reduces to the DDH assumption tightly. By tailoring their security proof to a setting which admits the public-key aggregation, we can prove the tight security of our multisignature.

The construction of our multisignature scheme is almost the same as that of the LYG multisignature except for the number of hash values transferred in the first round communication, which is used to commit temporary randomness of each signer. Namely, the communication efficiency of ours is a little better than theirs and the size of keys and signatures are identical in both schemes. Our result, therefore, suggests that we give tight security while maintaining the public-key aggregation functionality and the efficiency of the LYG multisignature.

As an application of our multisignature scheme, we give an *interactive aggregate signature* with the public-key aggregation. The interactive aggregate signature enables each signer to choose their own message instead of the common message. Namely, the interactive aggregate signature can be considered as a generalization of the multisignature. We show that our proposed multisignature can be converted into the interactive aggregate signature with a slight modification based on the idea of [17]. Surprisingly, this modification does not affect the security proof at all since we employ the random oracle model. Therefore we can obtain the security proof of our interactive aggregate signature as a corollary of the one of our proposed multisignature.

1.2 Proof Technique

We briefly explain the idea of the security proof of our proposed scheme. Before that, we now recap that of the LYG multisignture. Le, Yang, and Ghorbani proved that the security of their multisignature reduces to the DDH assumption via a loose security reduction in ROM. To break the

Table 1: Summary of multisignatures with public-key aggregation

Scheme	Assumption	Rounds	Security	Model	Reduction
[18]	DL	3	PPK	ROM	loose
[7]	DL	2	PPK	ROM	loose
[6]	coCDH	1	PPK	ROM	loose
[13]	DDH	3	PPK	ROM	loose
[16]	Ring-SIS, DCK, Re-DCK	4	PPK	ROM	loose
[10]	SIS, Re-DCK	1	PPK	ROM	loose
[ours]	DDH	3	PPK	ROM	tight

The column Assumption means the security assumptions required in each scheme. The column Rounds means the number of interactions among signers in the signing protocol. All schemes are proven to be secure with respect to the plain public-key model in the random oracle model. All security reductions are loose reductions except ours.

DDH assumption by utilizing an adversary $\mathcal{A}$ against the LYG multisignature scheme, a reduction $\mathcal{R}$ was constructed to extract information about a given DDH instance from the aggregated public key under which $\mathcal{A}$ forges. This can be done by using the general forking lemma [2], and it is known that the (general) forking lemma induces loose security. Due to the same reason, the security of other multisignatures [18, 7, 6, 16] is also loose.

To avoid the general forking lemma, we aim to employ the lossy proof technique introduced by Katz and Wang [11]. [11] proposed an ordinary signature scheme whose security reduces to the DDH assumption via a tight reduction. They constructed their reduction $\mathcal{R}$ in a way that it decides that a given instance is a yes-instance with the probability which is almost the same as the success probability of $\mathcal{A}$ if $\mathcal{R}$ is indeed given a yes-instance, whereas it does with negligible probability if it is given a no-instance. This technique is also applied to construct the tight security reduction of the LBG multisignature [12]. However, their approach cannot be applied to our case immediately, since we now consider the multisignatures with *public-key aggregation*. In our case, an aggregated public key under which $\mathcal{A}$ forges can be controlled by $\mathcal{A}$, whereas a public key under which it forges is decided by $\mathcal{R}$ in the Katz-Wang signature case. For this difficulty, we find desirable features of an aggregated public key of the LYG multisignature on which our proposed scheme is based. The first one is that such an aggregated public key can be regarded as a single public key of the Katz-Wang signature scheme. The second one is that an aggregated public key which includes an irregular public key, i.e. a no-instance, is also a no-instance with high probability. This enables us to prove the tight security of our proposed scheme by directly applying the lossy proof technique, and hence we no longer use the general forking lemma even in the public-key aggregation setting.

1.3 Related Works

The multisignature by Maxwell, Poelstra, Seurin, and Wuille [18] is the first multisignature which supports the public-key aggregation. Their scheme is based on Schnorr signature [20], and the signing protocol involves 3-round interaction. Its security is proven under the DL assumption in ROM. Drijvers, Edalatnejad, Ford, Kiltz, Loss, Neven, and Stepanovs [7] proposed another DL-based multisignature. Their scheme requires 2-round interaction only in the signing protocol. For the pairing-based multisignature with public-key aggregation, Boneh, Drijvers, and Neven [6] constructed a non-interactive multisignature in the sense that the signing protocol can be done in 1-round. Le, Yang, and Ghorbani [13] proposed a multisignature with public-key aggregation under the DDH assumption. Concerning the schemes except for the DL or DDH assumptions, Ma and Jiang [16] and Kansal and Dutta [10] proposed a lattice-based multisignature with public-key aggregation, respectively. Although their multisignatures are expected as quantum-resistant ones, there exists a disadvantage such that the size of the signature is proportional to the number of signers involved in the signing protocol. We note that the attack to the multisignature scheme [10] was

Table 2: Comparison of component size among multisignatures with public-key aggregation when using a cyclic group of order q and hash functions with ℓ-bit outputs.

Scheme	Public Parameter	Component Size		Communication Size			Signature Size
		pk	sk	1st	2nd	3rd	
[18]	$(\mathbb{G}, q, \mathsf{g})$	$\lvert\mathbb{G}\rvert$	$\lvert\mathbb{Z}_q\rvert$	ℓ	$\lvert\mathbb{G}\rvert$	$\lvert\mathbb{Z}_q\rvert$	$2\lvert\mathbb{Z}_q\rvert$
[7]	$(\mathbb{G}, q, \mathsf{g})$	$\lvert\mathbb{G}\rvert + 2\lvert\mathbb{Z}_q\rvert$	$\lvert\mathbb{Z}_q\rvert$	$2\lvert\mathbb{G}\rvert$	$3\lvert\mathbb{Z}_q\rvert$	-	$2\lvert\mathbb{G}\rvert + 3\lvert\mathbb{Z}_q\rvert$
[6]	$(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, \mathsf{g}_1, \mathsf{g}_2, e)$	$\lvert\mathbb{G}_2\rvert$	$\lvert\mathbb{Z}_q\rvert$	$\lvert\mathbb{G}_1\rvert$	-	-	$\lvert\mathbb{G}_1\rvert$
[13]	$(\mathbb{G}, q, \mathsf{g}, \mathsf{h})$	$2\lvert\mathbb{G}\rvert$	$\lvert\mathbb{Z}_q\rvert$	$2\lvert\mathbb{Z}_q\rvert$	$2\lvert\mathbb{G}\rvert$	$\lvert\mathbb{Z}_q\rvert$	$2\lvert\mathbb{Z}_q\rvert$
[ours]	$(\mathbb{G}, q, \mathsf{g}, \mathsf{h})$	$2\lvert\mathbb{G}\rvert$	$\lvert\mathbb{Z}_q\rvert$	ℓ	$2\lvert\mathbb{G}\rvert$	$\lvert\mathbb{Z}_q\rvert$	$2\lvert\mathbb{Z}_q\rvert$

The column Public Parameter displays a public parameter of each scheme. The public parameter of DL-based scheme [18, 7] consists of the group description $\mathbb{G}$ of a prime order q and its generator g. In the DDH-based scheme [13] and ours, the additional generator h is appended. For the pairing-based scheme [6], three groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of a prime order q are given. g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. e is a non-degenerate bilinear pairing from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$.
The column Component Size displays the storage of each component. $|\mathbb{G}|$ means the size of one element in $\mathbb{G}$. $|\mathbb{G}_1|$, $|\mathbb{G}_2|$ and $|\mathbb{Z}_q|$ are also defined in the same manner. The column Communication Size displays the traffic expended in each round of the signing protocol. The column Signature Size displays the size of the signature.

Table 3: Comparison of computation time among multisignatures with public-key aggregation when using a cyclic group of order q.

Scheme	Signing Time per Signer	Verification Time
[18]	$\mathrm{Exp}_{\mathbb{G}} + S\,\mathrm{Mul}_{\mathbb{G}} + S(\mathrm{Exp}_{\mathbb{G}} + \mathrm{Mul}_{\mathbb{G}})$	$S(\mathrm{Exp}_{\mathbb{G}} + \mathrm{Mul}_{\mathbb{G}}) + 2\,\mathrm{Exp}_{\mathbb{G}} + \mathrm{Mul}_{\mathbb{G}}$
[7]	$5\,\mathrm{Exp}_{\mathbb{G}} + 3(S+1)\,\mathrm{Mul}_{\mathbb{G}}$	$6\,\mathrm{Exp}_{\mathbb{G}} + (S+4)\,\mathrm{Mul}_{\mathbb{G}}$
[6]	$\mathrm{Exp}_{\mathbb{G}_1} + S\,\mathrm{Mul}_{\mathbb{G}_1}$	$S(\mathrm{Exp}_{\mathbb{G}_2} + \mathrm{Mul}_{\mathbb{G}_2}) + 2\,\mathrm{Pair} + \mathrm{Mul}_{\mathbb{G}_T}$
[13]	$2\,\mathrm{Exp}_{\mathbb{G}} + 2S\,\mathrm{Mul}_{\mathbb{G}} + 2S(\mathrm{Exp}_{\mathbb{G}} + \mathrm{Mul}_{\mathbb{G}})$	$2S(\mathrm{Exp}_{\mathbb{G}} + \mathrm{Mul}_{\mathbb{G}}) + 4\,\mathrm{Exp}_{\mathbb{G}} + 2\,\mathrm{Mul}_{\mathbb{G}}$
[ours]	$2\,\mathrm{Exp}_{\mathbb{G}} + 2S\,\mathrm{Mul}_{\mathbb{G}} + 2S(\mathrm{Exp}_{\mathbb{G}} + \mathrm{Mul}_{\mathbb{G}})$	$2S(\mathrm{Exp}_{\mathbb{G}} + \mathrm{Mul}_{\mathbb{G}}) + 4\,\mathrm{Exp}_{\mathbb{G}} + 2\,\mathrm{Mul}_{\mathbb{G}}$

The columns Signing Time per Signer and Verification Time display the computation time of the signing algorithm and that of the verification algorithm, respectively. $\mathrm{Exp}_{\mathbb{G}}$ and $\mathrm{Mul}_{\mathbb{G}}$ means the time for required the exponentiation and the multiplication in the group $\mathbb{G}$, respectively. Pair means the time of pairing computation. S is the number of signers involved in the signing protocol. Note that we omit the operations done over $\mathbb{Z}_q$ from the evaluation because the operations done over the group $\mathbb{G}$ are dominant rather than the ones over $\mathbb{Z}_q$.

pointed out in [14].

The summary of multisignature schemes with public-key aggregation including ours is given in Table 1. All schemes listed in Table 1 are proven to be secure in the PPK model and ROM, however, only our scheme achieves the tight security.

We compare the efficiency of our proposed multisignature scheme with the existing multisignature scheme based on a cyclic group of prime order. The result is given in Table 2 and Table 3. In Table 2, we evaluate the size efficiency. This table shows that the size of components of the DDH-based scheme [13] and ours is twice that of the DL-based scheme [18]. Although the scheme by [7] is also based on the DL assumption, the scheme requires a larger component size to achieve the two-round signing protocol. The pairing-based scheme [6] has the highest size efficiency.

Table 3 represents the comparison of the time efficiency. The DL-based scheme [18] achieves the fastest computation time in both the signing and the verification. The DDH-based scheme [13] and ours take twice the time of [18]. The time efficiency of [6] is worst among the multisiganture schemes in the table although it achieves the best size efficiency. This is because the scheme requires the computation of the bilinear pairing to make the signing protocol non-interactive. This can be considered as a tradeoff.

We finally consider the applicability of our proof technique to other multisignature schemes. The targets are the CDH-based tightly secure multisignature schemes by [12] and [21]. Since our proposed scheme is based on the DDH assumption, it is natural to consider whether or not our technique is applicable to the CDH-based schemes. Both CDH-based multisignature schemes [12, 21] have a 3-round signature generation protocol as well as our DDH-based scheme. However, our technique seems not to employ them. This is because our technique utilizes the property that the aggregated public key becomes *irregular* in a sense that it has no corresponding secret key with an overwhelming probability, if it is generated from a group of public keys which includes at least one irregular public key. In the security proof, we set the challenge public key to be irregular to use this property. On the other hand, on the CDH-based schemes, the challenge public key has to be regular because a CDH-instance is embedded into the challenge public key to extract a solution of the CDH-instance from the forgery.

We also note the security model used in the original proof of the CDH-based scheme by [12]. In the paper [12], the security proof is claimed under the PPK model. However, we find that the proof requires actually the knowledge of the secret key model [4, 15] which is a more restricted security model than the PPK model employed in this paper. This is because their security reduction needs secret keys corresponding to the public keys which the adversary outputs with the forgery to compute a solution of the given CDH-instance. Although the multisignature scheme of [21] achieves the security in the PPK model, it employs the bilinear pairing.

As described above, our technique cannot be applied to the CDH-based multisignature schemes immediately. Thus we need another proof technique to make these schemes support the public key aggregation. It is an important open question.

2 Preliminaries

For any algorithm $\mathcal{A}$, we denote by $y \leftarrow \mathcal{A}(x)$ that $\mathcal{A}$ outputs y on input x. In particular, $\mathcal{A}(x)$ stands for a random variable for $\mathcal{A}$'s output on input x, where the probability is taken over the internal coin flips in $\mathcal{A}$. For a finite set X, $x \in_{\mathrm{U}} X$ means that x is chosen from X uniformly at random.

2.1 Multisignature

We recap the notion of a multisignature MSig. MSig with the public-key aggregation consists of six components $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{KAgg}, \mathsf{KAVer}, \mathsf{Sig}, \mathsf{Ver})$ [13]. We consider the situation where a group of signers $(\mathsf{S}_1, \ldots \mathsf{S}_S)$ signs a message M. After a public parameter pp is generated by the probabilistic polytime algorithm $\mathsf{Setup}(1^\kappa)$ for a security parameter κ, each signer S_i generates a secret and public key pair (sk_i, pk_i) by using the probabilistic polytime key generator $\mathsf{KGen}(pp)$. Then, the set $L = \{pk_1, \ldots, pk_S\}$ of public keys can be aggregated as $apk = \mathsf{KAgg}(L)$ by the deterministic polytime algorithm KAgg, and one can ensure that the aggregated public key apk is given by the set L by executing the deterministic polytime algorithm $\mathsf{KAVer}(apk, L)$. To issue a multisignature σ of M under the public-key set L, the signing protocol is executed by running the probabilistic polytime signing algorithm $\mathsf{Sig}(pp, sk_i, L, M)$ by each signer S_i. Then, the validation of σ can be evaluated by executing the polytime algorithm $\mathsf{Ver}(pp, L, M, \sigma)$.

The security under the *plain public-key (ppk)* model is defined by the ppk game of MSig depicted in Figure 1. Without loss of generality, we suppose that $\mathcal{C}$ plays a roll of the first signer S_1 during the ppk game. For any polynomial T and any function ϵ, we say that MSig is (T, ϵ, Q_S)*-ppk secure* if for any adversary $\mathcal{A}$ which runs in time at most $T(\kappa)$ and makes at most $Q_S(\kappa)$ queries in Sign phase, $\mathcal{A}$ can win the ppk game of MSig with probability $\epsilon(\kappa)$ for sufficiently large κ.

MSig is $(T, \epsilon, Q_S, Q_1, Q_2, Q_3, \ldots)$*-ppk secure in the random oracle model (ROM)* if MSig is (T, ϵ, Q_S)*-ppk* secure where $\mathcal{A}$ makes at most $Q_n(\kappa)$ queries to the random oracle H_n for any n and sufficiently large κ. In the security proof, the random oracle model is generally emulated by the

- Init: A challenger $\mathcal{C}$ generates $pp \leftarrow \mathsf{Setup}(1^\kappa)$ and $(sk^*, pk^*) \leftarrow \mathsf{KGen}(1^\kappa)$, and then sends (pp, pk^*) to an adversary $\mathcal{A}$.
- Sign: Given a public-key set $L^{(t)}$ and a message $M^{(t)}$, $\mathcal{C}$ and $\mathcal{A}$ run the signing protocol in which $\mathcal{C}$ plays a role of the signer S_1 which owns $pk_1^{(t)} = pk^*$, whereas $\mathcal{A}$ does that of the other signers S_i $(2 \leq i \leq S)$.
- Chal: Given (L^*, M^*, σ^*) from $\mathcal{A}$ finally, $\mathcal{A}$ *wins the ppk game of* MSig if the followings hold:

(W.1) $pk_1^* = pk^*$ for $L^* = \{pk_1^*, \ldots, pk_S^*\}$.

(W.2) (L^*, M^*) does not appeared in Sign phase.

(W.3) $\mathsf{Ver}(pp, L^*, M^*, \sigma^*) = 1$.

Figure 1: Plain public-key game of MSig between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$

$\underline{\mathsf{LS}(\mathfrak{L}, \mathfrak{R}, \xi)}$:

- return $(\mathfrak{L}, v)$ if there exists $v \in \mathfrak{R}$ such that $(\xi, v) \in \mathfrak{L}$.
- return $(\mathfrak{L} \cup \{(\xi, v)\}, v)$ for $v \in_{\mathrm{U}} \mathfrak{R}$ otherwise.

Figure 2: Lazy sampling algorithm LS given a list $\mathfrak{L}$ of previous pairs of an input and its output on H, the description of the set $\mathfrak{R}$ and an input $\xi \in \mathfrak{D}$

lazy sampling algorithm. For the function $\mathsf{H} : \mathfrak{D} \to \mathfrak{R}$, the lazy sampling algorithm LS is defined as in Figure 2. Here, we will directly write the set $\mathfrak{R}$ instead of its description.

3 DDH Assumption

In this section, we define the *decisional Diffie-Hellman (DDH) assumption* [5]. We consider a polytime group parameter generator GGen defined in Figure 3. We define the two instance generators $\mathsf{Gen}_{\mathrm{reg}}^{\mathsf{DDH}}$ and $\mathsf{Gen}_{\mathrm{loss}}^{\mathsf{DDH}}$ depicted in Figure 4. For any polynomial T and ϵ, the (T, ϵ)-*DDH assumption* states that for any adversary $\mathcal{D}$ which runs in time at most T, $\mathcal{D}$ can distinguish the distributions between $D_{\mathrm{reg}}(\kappa)$ and $D_{\mathrm{loss}}(\kappa)$, which are defined below, with probability $\epsilon(\kappa)$ for sufficiently large κ.

$\underline{D_{\mathrm{reg}}(\kappa)}$: $(\mathbb{G}, q, \mathsf{g}, \mathsf{h}, y_1, y_2)$ such that

- $(\mathbb{G}, q, \mathsf{g}, \mathsf{h}) \leftarrow \mathsf{GGen}(1^\kappa)$, and
- $(x, (y_1, y_2)) \leftarrow \mathsf{Gen}_{\mathrm{reg}}^{\mathsf{DDH}}(\mathbb{G}, q, \mathsf{g}, \mathsf{h})$,

$\underline{D_{\mathrm{loss}}(\kappa)}$: $(\mathbb{G}, q, \mathsf{g}, \mathsf{h}, y_1, y_2)$ such that

- $(\mathbb{G}, q, \mathsf{g}, \mathsf{h}) \leftarrow \mathsf{GGen}(1^\kappa)$, and
- $((x_1, x_2), (y_1, y_2)) \leftarrow \mathsf{Gen}_{\mathrm{loss}}^{\mathsf{DDH}}(\mathbb{G}, q, \mathsf{g}, \mathsf{h})$.

For pairs $(y_{1,1}, y_{1,2}), (y_{2,1}, y_{2,2}) \in \mathbb{G}^2$, we define the *multiplication* of pairs $(y_{1,1}, y_{1,2}) \times (y_{2,1}, y_{2,2})$ by $(y_{1,1} \times y_{2,1}, y_{1,2} \times y_{2,2}) \in \mathbb{G}^2$, namely the element-wise multiplication. Then, the following lemma holds.

Lemma 1. *For any κ, let $(\mathbb{G}, q, \mathsf{g}, \mathsf{h}) \leftarrow \mathsf{GGen}(1^\kappa)$ and let $((x_{1,1}, x_{1,2}), (y_{1,1}, y_{1,2})) \leftarrow \mathsf{Gen}_{\mathrm{loss}}^{\mathsf{DDH}}(\mathbb{G}, q, \mathsf{g}, \mathsf{h})$. Let S be a polynomial in κ. For any $2 \leq i \leq S$, let $(y_{i,1}, y_{i,2}, a_i) \in \mathbb{G} \times \mathbb{G} \times \mathbb{Z}_q$. If $a_1 \in_{\mathrm{U}} \mathbb{Z}_q$, then with probability $2/q$, there exists $x \in \mathbb{Z}_q$ such that*

$$\prod_{i=1}^{S} (y_{i,1}^{a_i}, y_{i,2}^{a_i}) = (\mathsf{g}^x, \mathsf{h}^x).$$

$\mathsf{GGen}(1^\kappa)$: return $(\mathbb{G}, q, \mathsf{g}, \mathsf{h})$ such that

- $\mathbb{G}$ is the description of the group of prime order q,
- q is polynomial-length in κ, and
- g and h are its generators.

Figure 3: Group parameter generator GGen

$\mathsf{Gen}^{\mathsf{DDH}}_{\mathrm{reg}}(\mathbb{G}, q, \mathsf{g}, \mathsf{h})$:

- $x \in_{\mathrm{U}} \mathbb{Z}_q$.
- $(y_1, y_2) = (\mathsf{g}^x, \mathsf{h}^x)$.
- return $(x, (y_1, y_2))$.

$\mathsf{Gen}^{\mathsf{DDH}}_{\mathrm{loss}}(\mathbb{G}, q, \mathsf{g}, \mathsf{h})$:

- $x_1, x_2 \in_{\mathrm{U}} \mathbb{Z}_q$.
- $(y_1, y_2) = (\mathsf{g}^{x_1}, \mathsf{h}^{x_2})$.
- return $((x_1, x_2), (y_1, y_2))$.

Figure 4: Regular DDH instance generator $\mathsf{Gen}^{\mathsf{DDH}}_{\mathrm{reg}}$ and Lossy DDH instance generator $\mathsf{Gen}^{\mathsf{DDH}}_{\mathrm{loss}}$

Proof. For any $2 \le i \le S$, there exists $(x_{i,1}, x_{i,2}) \in \mathbb{Z}_q^2$ such that $(y_{i,1}, y_{i,2}) = (\mathsf{g}^{x_{i,1}}, \mathsf{h}^{x_{i,2}})$. This is because $y_{i,1}, y_{i,2} \in \mathbb{G}$, and g and h are generators of $\mathbb{G}$. On the other hand, it follows from $\mathsf{Gen}^{\mathsf{DDH}}_{\mathrm{loss}}$ that $(y_{1,1}, y_{1,2}) = (\mathsf{g}^{x_{1,1}}, \mathsf{h}^{x_{1,2}})$. Then, $\prod_{i=1}^{S} \left(y_{i,1}^{a_i}, y_{i,2}^{a_i}\right)$ can be expressed as follows:

$$\begin{aligned} \prod_{i=1}^{S} y_{i,1}^{a_i} &= \prod_{i=1}^{S} (\mathsf{g}^{x_{i,1}})^{a_i} = \prod_{i=1}^{S} \mathsf{g}^{a_i x_{i,1}} = \mathsf{g}^{\sum_{i=1}^{S} a_i x_{i,1}}, \\ \prod_{i=1}^{S} y_{i,2}^{a_i} &= \prod_{i=1}^{S} (\mathsf{h}^{x_{i,2}})^{a_i} = \prod_{i=1}^{S} \mathsf{h}^{a_i x_{i,2}} = \mathsf{h}^{\sum_{i=1}^{S} a_i x_{i,2}}. \end{aligned} \tag{1}$$

We now evaluate the probability that there exists $x \in \mathbb{Z}_q$ such that $\prod_{i=1}^{S} \left(y_{i,1}^{a_i}, y_{i,2}^{a_i}\right) = (\mathsf{g}^x, \mathsf{h}^x)$ under the condition $x_{1,1} \neq x_{1,2}$. By Eq. (1), this means that $\sum_{i=1}^{S} a_i x_{i,1} \equiv x \equiv \sum_{i=1}^{S} a_i x_{i,2} \pmod q$. This can be represented as $a_1(x_{1,1} - x_{1,2}) \equiv \sum_{i=2}^{S} a_i(x_{i,2} - x_{i,1}) \pmod q$. It follows from the condition $x_{1,1} \neq x_{1,2}$ that

$$a_1 \equiv \frac{\sum_{i=2}^{S} a_i(x_{i,2} - x_{i,1})}{x_{1,1} - x_{1,2}} \pmod q. \tag{2}$$

Namely, in order that such x exists, Eq. (2) must holds. Since $a_1 \in_{\mathrm{U}} \mathbb{Z}_q$, the probability of it under the condition $x_{1,1} \neq x_{1,2}$ is $1/q$. On the other hand, the probability that $x_{1,1} = x_{1,2}$ is $1/q$ since $x_{1,1}, x_{1,2} \in_{\mathrm{U}} \mathbb{Z}_q$. Thus, the probability of the existence of such x is bound by the following:

$$\left(1 - \frac{1}{q}\right) \cdot \frac{1}{q} + \frac{1}{q} < \frac{2}{q}.$$

□

We fix $(\mathbb{G}, q, \mathsf{g}, \mathsf{h}) \leftarrow \mathsf{GGen}(1^\kappa)$. For any $z_1, z_2 \in \mathbb{G}$, consider the following set:

$$\mathfrak{S}_{(z_1,z_2)} = \{(\mathsf{g}^r, \mathsf{h}^r) \times (z_1, z_2) \mid r \in \mathbb{Z}_q\} \subseteq \mathbb{G}^2.$$

For the identity e of $\mathbb{G}$, we write $\mathfrak{S} = \mathfrak{S}_{(e,e)}$.

Lemma 2. *Let $r \in_{\mathrm{U}} \mathbb{Z}_q$, and let $y_1, y_2 \in \mathbb{G}$ and $s \in \mathbb{Z}_q$. Then, it holds that*

- *$r + s \bmod q$ is uniformly distributed over $\mathbb{Z}_q$.*

- $(\mathsf{g}^r \cdot y_1, \mathsf{h}^r \cdot y_2)$ *is uniformly distributed over* $\mathfrak{S}_{(y_1,y_2)}$.

- *The cardinality of* $\mathfrak{S}_{(y_1,y_2)}$ *is* q.

Proof. Consider the map $r \in \mathbb{Z}_q$ to $r + s \bmod q$ for fixed s, and the map $r \in \mathbb{Z}_q$ to $(\mathsf{g}^r \cdot y_1, \mathsf{h}^r \cdot y_2) \in \mathfrak{S}_{(y_1,y_2)}$ for fixed (y_1, y_2). It is obvious that both maps are bijective because g and h are generators of $\mathbb{G}$. Since r is chosen uniformly from $\mathbb{Z}_q$, $r + s \bmod q$ is uniformly distributed over $\mathbb{Z}_q$, and $(\mathsf{g}^r \cdot y_1, \mathsf{h}^r \cdot y_2)$ is uniformly distributed over $\mathfrak{S}_{(y_1,y_2)}$, respectively. Moreover, since the map r to $(\mathsf{g}^r \cdot y_1, \mathsf{h}^r \cdot y_2)$ is bijective and the cardinality of $\mathbb{Z}_q$ is q, the cardinality of $\mathfrak{S}_{(y_1,y_2)}$ is also q. □

4 Proposed Multisignature

In this section, we describe the proposed multisignature and then prove the tight security of their multisignature.

4.1 Protocol

The proposed multisignature $\mathsf{MSig}_{\mathrm{ours}}$ involves three hash functions $\mathsf{H}_{\mathsf{agg}} : \{0,1\}^* \to \mathbb{Z}_q$, $\mathsf{H}_{\mathsf{cmt}} : \mathbb{G} \times \mathbb{G} \to \{0,1\}^{\ell}$ and $\mathsf{H}_{\mathsf{sig}} : \{0,1\}^* \to \mathbb{Z}_q$. The construction is depicted in Figure 5.

4.2 Correctness

In a similar manner to [13], the correctness of $\mathsf{MSig}_{\mathrm{ours}}$ can be estimated in the following way. Suppose the situation where a group of signers $(\mathsf{S}_1, \ldots, \mathsf{S}_S)$ signs a message M. Let $pp \leftarrow \mathsf{Setup}(1^{\kappa})$ for a security parameter κ. Each signer S_i generates a secret and public key pair $(sk_i, pk_i) = (x_i, (y_{i,1}, y_{i,2})) \leftarrow \mathsf{KGen}(pp)$, and then runs $\mathsf{Sig}(pp, sk_i, L, M)$ to issue a multisignature $\sigma = (c, z)$ of M under the public-key set $L = \{pk_1, \ldots, pk_S\}$.

$\mathsf{KGen}(pp)$, (R1.1) and (R3.6) imply that for each $1 \le i \le S$,

$$(\mathsf{g}^{z_i}, \mathsf{h}^{z_i}) = \left(\mathsf{g}^{r_i + sk_i \cdot c_i}, \mathsf{h}^{r_i + sk_i \cdot c_i}\right) = \left(w_{i,1} \cdot y_{i,1}^{c_i}, w_{i,2} \cdot y_{i,2}^{c_i}\right), \tag{3}$$

where c_i is set as in (R3.5). Since $apk = (y_1, y_2) = \prod_{i=1}^{S} \left(y_{i,1}^{\mathsf{H}_{\mathsf{agg}}(pk_i, L)}, y_{i,2}^{\mathsf{H}_{\mathsf{agg}}(pk_i, L)}\right)$ as in $\mathsf{KAgg}(L)$ and $c_i = \mathsf{H}_{\mathsf{agg}}(pk_i, L) \cdot c \bmod q$ as in (R3.5) for each $1 \le i \le S$, we have

$$\begin{aligned}
(y_1^c, y_2^c) &= \prod_{i=1}^{S} \left(\left(y_{i,1}^{\mathsf{H}_{\mathsf{agg}}(pk_i, L)}\right)^c, \left(y_{i,2}^{\mathsf{H}_{\mathsf{agg}}(pk_i, L)}\right)^c \right) \\
&= \prod_{i=1}^{S} \left(y_{i,1}^{\mathsf{H}_{\mathsf{agg}}(pk_i, L) \cdot c}, y_{i,2}^{\mathsf{H}_{\mathsf{agg}}(pk_i, L) \cdot c} \right) \\
&= \prod_{i=1}^{S} \left(y_{i,1}^{c_i}, y_{i,2}^{c_i} \right).
\end{aligned}$$

- $\mathsf{Setup}(1^\kappa)$ generates $(\mathbb{G}, q, \mathsf{g}, \mathsf{h}) \leftarrow \mathsf{GGen}(1^\kappa)$ as a public parameter pp.
- $\mathsf{KGen}(pp)$ is run by each signer S_i. The procedure is the same as that of $\mathsf{Gen}^{\mathsf{DDH}}_{\mathrm{reg}}(pp)$. We denote by $(sk_i, pk_i) = (x_i, (y_{i,1}, y_{i,2})) \leftarrow \mathsf{KGen}(pp)$ S_i's secret key and public key pair.
- $\mathsf{KAgg}(L)$ returns the aggregated key $apk = (y_1, y_2)$ of the set $L = \{pk_1, \ldots, pk_S\}$ of public keys of all the signers S_i by $(y_1, y_2) = \prod_{i=1}^{S} \left(y_{i,1}^{\mathsf{H}_{\mathsf{agg}}(pk_i, L)}, y_{i,2}^{\mathsf{H}_{\mathsf{agg}}(pk_i, L)} \right)$.
- $\mathsf{KAVer}(apk, L)$ returns 1 if $apk = \mathsf{KAgg}(L)$.
- $\mathsf{Sig}(pp, sk_i, L, M)$ is run by each signer S_i to issue a multisignature $\sigma = (c, z)$ for the message M as follows:

  R1: S_i broadcasts h_i after the followings:

  (R1.1) $r_i \in_{\mathrm{U}} \mathbb{Z}_q$; $(w_{i,1}, w_{i,2}) = (\mathsf{g}^{r_i}, \mathsf{h}^{r_i})$.
  (R1.2) $h_i = \mathsf{H}_{\mathsf{cmt}}(w_{i,1}, w_{i,2})$.

  R2: S_i receives $\{h_j\}_{j \neq i}$ from the co-signers. Then S_i broadcasts $(w_{i,1}, w_{i,2})$.

  R3: S_i receives $\{(w_{j,1}, w_{j,2})\}_{j \neq i}$ from the co-signers. Then S_i broadcasts z_i after the followings:

  (R3.1) abort if there exists an index $j \neq i$ such that $h_j \neq \mathsf{H}_{\mathsf{cmt}}(w_{j,1}, w_{j,2})$.
  (R3.2) $(w_1, w_2) = \prod_{i=1}^{S} (w_{i,1}, w_{i,2})$.
  (R3.3) $apk \leftarrow \mathsf{KAgg}(L)$.
  (R3.4) $c = \mathsf{H}_{\mathsf{sig}}(apk, L, w_1, w_2, M)$.
  (R3.5) $c_i = \mathsf{H}_{\mathsf{agg}}(pk_i, L) \cdot c \bmod q$.
  (R3.6) $z_i = r_i + sk_i \cdot c_i \bmod q$.

  R4: S_i receives $\{z_j\}_{j \neq i}$ from the co-signers. Then S_i outputs $\sigma = (c, z)$ as a multisignature, where $z = \sum_{i=1}^{S} z_i \bmod q$.
- $\mathsf{Ver}(pp, L, M, \sigma)$ returns 1 if it satisfies that $c = \mathsf{H}_{\mathsf{sig}}(apk, L, w'_1, w'_2, M)$, where
  - $apk = (y_1, y_2) \leftarrow \mathsf{KAgg}(L)$, and
  - $(w'_1, w'_2) = \left(\mathsf{g}^z y_1^{-c}, \mathsf{h}^z y_2^{-c}\right)$.

Figure 5: Proposed multisignature $\mathsf{MSig}_{\mathrm{ours}}$

It follows from (R3.2) and R4 that in $\mathsf{Ver}(pp, L, M, \sigma)$,

$$\begin{aligned}
(w'_1, w'_2) &= \left(\mathsf{g}^z y_1^{-c}, \mathsf{h}^z y_2^{-c}\right) \\
&= \left(\mathsf{g}^{\sum_{i=1}^{S} z_i} \prod_{i=1}^{S} y_{i,1}^{-c_i}, \mathsf{h}^{\sum_{i=1}^{S} z_i} \prod_{i=1}^{S} y_{i,2}^{-c_i} \right) \\
&= \left(\prod_{i=1}^{S} \mathsf{g}^{z_i} \prod_{i=1}^{S} y_{i,1}^{-c_i}, \prod_{i=1}^{S} \mathsf{h}^{z_i} \prod_{i=1}^{S} y_{i,2}^{-c_i} \right) \\
&= \prod_{i=1}^{S} \left(\mathsf{g}^{z_i} y_{i,1}^{-c_i}, \mathsf{h}^{z_i} y_{i,2}^{-c_i} \right) \\
&= \prod_{i=1}^{S} (w_{i,1}, w_{i,2}) \\
&= (w_1, w_2).
\end{aligned}$$

By (R3.4), we have $c = \mathsf{H}_{\mathsf{sig}}(apk, L, w_1, w_2, M) = \mathsf{H}_{\mathsf{sig}}(apk, L, w'_1, w'_2, M)$, and hence Ver always returns 1.

4.3 Tight Security

We prove the tight security of $\mathsf{MSig}_{\text{ours}}$.

Theorem 1. *Let T be a polynomial and let ϵ be a function. Assume that the (T, ϵ)-DDH assumption holds. Then, $\mathsf{MSig}_{\text{ours}}$ is $(T', \epsilon', Q_S, Q_{\mathsf{agg}}, Q_{\mathsf{cmt}}, Q_{\mathsf{sig}})$-ppk secure in ROM, where*

$$T' = T - O(Q_{\mathsf{agg}} + Q_{\mathsf{cmt}} + Q_{\mathsf{sig}} + Q_S \cdot S),$$
$$\epsilon' \leq \epsilon + \frac{(Q_{\mathsf{cmt}} + Q_S)^2}{q} + \frac{1}{2^\ell} + \frac{(Q_{\mathsf{sig}} + Q_S)^2}{q} + \frac{3}{q},$$

where Q_S, Q_{agg}, Q_{cmt} and Q_{sig} are some polynomials which represent the numbers of queries to Sign phase, $\mathsf{H}_{\mathsf{agg}}$, $\mathsf{H}_{\mathsf{cmt}}$ and $\mathsf{H}_{\mathsf{sig}}$, respectively.

Proof. We prove the statement by the hybrid argument. For each $0 \leq n \leq 5$, let Win_n be the probability that $\mathcal{A}$ wins $\mathbf{Game}_n$. The description of each game is depicted in Figures 6 to 11.

The role and the overview of each game are as follows. $\mathbf{Game}_0$ is just the original ppk game of $\mathsf{MSig}_{\text{ours}}$. $\mathbf{Game}_1$ excludes the possibility that $\mathcal{A}$ finds a collision on $\mathsf{H}_{\mathsf{cmt}}$. Namely we can ensure that all hash values h_i computed in (R1.2) of Sig are distinct. In $\mathbf{Game}_2$, the timing and the way that $\mathcal{C}$ computes answers for signing queries is changed. $\mathcal{C}$ computes signatures for signing queries by using the list of hash queries for $\mathsf{H}_{\mathsf{cmt}}$. $\mathbf{Game}_3$ excludes the possibility that $\mathcal{A}$ predetermines the challenge value c. This change enables $\mathcal{C}$ to program the hash value of $\mathsf{H}_{\mathsf{sig}}$. In $\mathbf{Game}_4$, Sign phase is changed to the one so that $\mathcal{C}$ can compute signatures without the secret key. In other words, Sign phase is replaced with the simulated one. Moreover, we can show this simulation is perfect. In $\mathbf{Game}_5$, the public key is replaced with the one which is generated by the lossy DDH instance generator. This change is justified by the DDH assumption.

We show that the changes of games can be done in a way that $\mathcal{A}$ cannot distinguish the changes. We also show that the winning probability of $\mathcal{A}$ in $\mathbf{Game}_5$ is bounded by some negligible probability.

$\mathbf{Game}_0$ This game is identical to the ppk game of $\mathsf{MSig}_{\text{ours}}$. Namely, this is described as in Figure 6. Since any adversary $\mathcal{A}$ of running time T' wins the ppk game of $\mathsf{MSig}_{\text{ours}}$ with probability ϵ', we have

$$\mathsf{Win}_0 = \epsilon'. \tag{4}$$

$\mathbf{Game}_1$ This game excludes the possibility that $\mathcal{A}$ finds a collision on $\mathsf{H}_{\mathsf{cmt}}$. Namely, this is the same as $\mathbf{Game}_0$ except that the abort condition depicted in Figure 7 is added in $\mathsf{H}_{\mathsf{cmt}}$ phase. The difference between $\mathbf{Game}_0$ and $\mathbf{Game}_1$ is whether or not $\mathcal{C}$ aborts when a collision on $\mathsf{H}_{\mathsf{cmt}}$ is found. For each t-th query to $\mathsf{H}_{\mathsf{cmt}}$ phase, this probability is evaluated by $(t-1)/q$. This is because LS samples any hash value uniformly at random from the range $\mathbb{Z}_q$ of $\mathsf{H}_{\mathsf{cmt}}$. Since $\mathcal{A}$ and $\mathcal{C}$ make at most Q_{cmt} and Q_S queries to $\mathsf{H}_{\mathsf{cmt}}$ phase, respectively, the abort probability in this game is $\sum_{t=1}^{Q_{\mathsf{cmt}}+Q_S} (t-1)/q \leq (Q_{\mathsf{cmt}} + Q_S)^2/q$. Thus, we have

$$|\mathsf{Win}_1 - \mathsf{Win}_0| \leq \frac{(Q_{\mathsf{cmt}} + Q_S)^2}{q}. \tag{5}$$

$\mathbf{Game}_2$ In this game, the timing and the way that $\mathcal{C}$ computes answers for signing queries is changed. More precicely, this game is the same as $\mathbf{Game}_1$ except that R2 and R3 of Sign phase are replaced as in Figure 8. Namely, $\left(w_1^{(t)}, w_2^{(t)}\right)$ is generated at R2 instead of R3 by finding $\left\{\left(w_{j,1}^{(t)}, w_{j,2}^{(t)}\right)\right\}$ from the hash list $\mathfrak{L}_{\mathsf{cmt}}$.

- Init: $\mathcal{C}$ sets $\mathfrak{L}_{\mathsf{agg}} = \mathfrak{L}_{\mathsf{cmt}} = \mathfrak{L}_{\mathsf{sig}} = \emptyset$, generates $pp = (\mathbb{G}, q, \mathsf{g}, \mathsf{h}) \leftarrow \mathsf{Setup}(1^\kappa)$ and $(sk^*, pk^*) \leftarrow \mathsf{KGen}(pp)$, and then sends (pp, pk^*).
- $\mathsf{H}_{\mathsf{agg}}$: Given a pair $\xi^{(t)} = \big(pk^{(t)}, L^{(t)}\big)$, return $a^{(t)}$ for $\big(\mathfrak{L}_{\mathsf{agg}}, a^{(t)}\big) \leftarrow \mathsf{LS}\big(\mathfrak{L}_{\mathsf{agg}}, \mathbb{Z}_q, \xi^{(t)}\big)$.
- $\mathsf{H}_{\mathsf{cmt}}$: Given a pair $\xi^{(t)} = \Big(w_1^{(t)}, w_2^{(t)}\Big)$, return $h^{(t)}$ for $\big(\mathfrak{L}_{\mathsf{cmt}}, h^{(t)}\big) \leftarrow \mathsf{LS}\Big(\mathfrak{L}_{\mathsf{cmt}}, \{0,1\}^\ell, \xi^{(t)}\Big)$.
- $\mathsf{H}_{\mathsf{sig}}$: Given a tuple $\xi^{(t)} = \Big(apk^{(t)}, L^{(t)}, w_1^{(t)}, w_2^{(t)}, M^{(t)}\Big)$, return $c^{(t)}$ for $\big(\mathfrak{L}_{\mathsf{sig}}, c^{(t)}\big) \leftarrow \mathsf{LS}\big(\mathfrak{L}_{\mathsf{sig}}, \mathbb{Z}_q, \xi^{(t)}\big)$.
- Sign: Given a public-key set $L^{(t)} = \Big\{pk_1^{(t)}, \ldots, pk_S^{(t)}\Big\}$ and a message $M^{(t)}$, $\mathcal{C}$ and $\mathcal{A}$ run the signing protocol in which $\mathcal{C}$ plays a role of the signer S_1 which owns $pk_1^{(t)} = pk^*$. Namely, $\mathcal{C}$ executes $\mathsf{Sig}(pp, sk^*, L^{(t)}, M^{(t)})$ as follows:

  R1: $\mathcal{C}$ broadcasts $h_1^{(t)}$ after the followings:

  (R1.1) $r_1^{(t)} \in_{\mathrm{U}} \mathbb{Z}_q$; $\Big(w_{1,1}^{(t)}, w_{1,2}^{(t)}\Big) = \Big(\mathsf{g}^{r_1^{(t)}}, \mathsf{h}^{r_1^{(t)}}\Big)$.

  (R1.2) $h_1^{(t)} = \mathsf{H}_{\mathsf{cmt}}(w_{1,1}^{(t)}, w_{1,2}^{(t)})$.

  R2: Receiving $\Big\{h_j^{(t)}\Big\}_{j=2}^{S}$ from $\mathcal{A}$, $\mathcal{C}$ broadcasts $\Big(w_{1,1}^{(t)}, w_{1,2}^{(t)}\Big)$.

  R3: Receiving $\Big\{\Big(w_{j,1}^{(t)}, w_{j,2}^{(t)}\Big)\Big\}_{j=2}^{S}$ from $\mathcal{A}$, $\mathcal{C}$ broadcasts $z_1^{(t)}$ after the followings:

  (R3.1) abort if there exists an index $2 \le j \le S$ such that $h_j^{(t)} \ne \mathsf{H}_{\mathsf{cmt}}(w_{j,1}^{(t)}, w_{j,2}^{(t)})$.

  (R3.2) $\Big(w_1^{(t)}, w_2^{(t)}\Big) = \prod_{i=1}^{S} \Big(w_{i,1}^{(t)}, w_{i,2}^{(t)}\Big)$.

  (R3.3) $apk^{(t)} \leftarrow \mathsf{KAgg}(L^{(t)})$.

  (R3.4) $c^{(t)} = \mathsf{H}_{\mathsf{sig}}(apk^{(t)}, L^{(t)}, w_1^{(t)}, w_2^{(t)}, M^{(t)})$.

  (R3.5) $c_1^{(t)} = \mathsf{H}_{\mathsf{agg}}(pk^*_, L^{(t)}) \cdot c^{(t)} \bmod q$.

  (R3.6) $z_1^{(t)} = r_1^{(t)} + sk^* \cdot c_1^{(t)} \bmod q$.

  R4: Receiving $\Big\{z_j^{(t)}\Big\}_{j=2}^{S}$ from $\mathcal{A}$, $\mathcal{C}$ returns $\sigma^{(t)} = \big(c^{(t)}, z^{(t)}\big)$, where $z^{(t)} = \sum_{i=1}^{S} z_i^{(t)} \bmod q$.
- Chal: Given (L^*, M^*, σ^*) from $\mathcal{A}$ finally, $\mathcal{A}$ wins $\mathbf{Game}_0$ if the followings hold:

(W.1) $pk_1^* = pk^*$ for $L^* = \{pk_1^*, \ldots, pk_S^*\}$.

(W.2) (L^*, M^*) does not appeared in Sign phase.

(W.3) $\mathsf{Ver}(pp, L^*, M^*, \sigma^*) = 1$, namely it satisfies that $c^* = \mathsf{H}_{\mathsf{sig}}(apk^*, L^*, w_1^*, w_2^*, M^*)$, where

  – $\sigma^* = (c^*, z^*)$,

  – $apk^* = (y_1^*, y_2^*) \leftarrow \mathsf{H}_{\mathsf{agg}}(L^*)$, and

  – $(w_1^*, w_2^*) = \Big(\mathsf{g}^{z^*}(y_1^*)^{-c^*}, \mathsf{h}^{z^*}(y_2^*)^{-c^*}\Big)$.

Figure 6: Description of $\mathbf{Game}_0$

In $\mathsf{H}_{\mathsf{cmt}}$ phase, the following abort condition is added: $\mathcal{C}$ aborts if a *collision* is found, namely LS samples $h^{(t)} \in_{\mathrm{U}} \{0,1\}^\ell$ which is already contained in $\mathfrak{L}_{\mathsf{cmt}}$ for some input ξ.

Figure 7: Changed processes in $\mathbf{Game}_1$ from $\mathbf{Game}_0$

$\mathcal{C}$ generates $\left(w_1^{(t)}, w_2^{(t)}\right)$ in R2 instead of R3. The R2 and R3 are replaced as follows:

R2: Receiving $\left\{h_j^{(t)}\right\}_{j=2}^{S}$ from $\mathcal{A}$, $\mathcal{C}$ broadcasts $\left(w_{1,1}^{(t)}, w_{1,2}^{(t)}\right)$ after the followings:

(R2.1) For each $2 \le j \le S$, find $\left(\left(g_{j,1}^{(t)}, g_{j,2}^{(t)}\right), h_j^{(t)}\right)$ from $\mathfrak{L}_{\mathsf{cmt}}$. It aborts if there is no such pair.

(R2.2) $\left(w_1^{(t)}, w_2^{(t)}\right) = \left(w_{1,1}^{(t)}, w_{1,2}^{(t)}\right) \cdot \prod_{j=2}^{S} \left(g_{j,1}^{(t)}, g_{j,2}^{(t)}\right)$.

(R2.3) $apk^{(t)} \leftarrow \mathsf{KAgg}(L^{(t)})$.

(R2.4) $c^{(t)} = \mathsf{H}_{\mathsf{sig}}(apk^{(t)}, L^{(t)}, w_1^{(t)}, w_2^{(t)}, M^{(t)})$.

(R2.5) $c_1^{(t)} = \mathsf{H}_{\mathsf{agg}}(pk^*, L^{(t)}) \cdot c^{(t)} \bmod q$.

(R2.6) $z_1^{(t)} = r_1^{(t)} + sk^* \cdot c_1^{(t)} \bmod q$.

R3: Receiving $\left\{\left(w_{j,1}^{(t)}, w_{j,2}^{(t)}\right)\right\}_{j=2}^{S}$ from $\mathcal{A}$, $\mathcal{C}$ broadcasts $z_1^{(t)}$ after the following:

(R3.1) abort if there exists an index $2 \le j \le S$ such that $\left(w_{j,1}^{(t)}, w_{j,2}^{(t)}\right) \neq \left(g_{j,1}^{(t)}, g_{j,2}^{(t)}\right)$.

Figure 8: Changed processes in **Game**$_2$ from the previous games

(R2.4) is replaced with the following processes: $c^{(t)} \in_{\mathrm{U}} \mathbb{Z}_q$, and then abort if the "failure" event is happened, namely the hash value of $\left(apk^{(t)}, L^{(t)}, w_1^{(t)}, w_2^{(t)}, M^{(t)}\right)$ is already determined, or $\mathfrak{L}_{\mathsf{sig}} = \mathfrak{L}_{\mathsf{sig}} \cup \left\{\left(apk^{(t)}, L^{(t)}, w_1^{(t)}, w_2^{(t)}, M^{(t)}\right), c^{(t)}\right\}$ otherwise.

Figure 9: Changed processes in **Game**$_3$ from the previous games

We first show that the condition $\left(w_{j,1}^{(t)}, w_{j,2}^{(t)}\right) \neq \left(g_{j,1}^{(t)}, g_{j,2}^{(t)}\right)$ in (R3.1) of **Game**$_2$ is equivalent to $h_j^{(t)} \neq \mathsf{H}_{\mathsf{cmt}}(w_{j,1}^{(t)}, w_{j,2}^{(t)})$ which is the abort condition in (R3.1) of **Game**$_1$. We fix an index $1 \le t \le Q_{\mathsf{sig}}$, and assume that for each $2 \le j \le S$, there exists $\left(g_{j,1}^{(t)}, g_{j,2}^{(t)}\right)$ such that $\left(\left(g_{j,1}^{(t)}, g_{j,2}^{(t)}\right), h_j^{(t)}\right)$ is in the list $\mathfrak{L}_{\mathsf{cmt}}$ in (R2.1). Since the non-abortion in $\mathsf{H}_{\mathsf{cmt}}$ phase guarantees that there is no collision in $\mathsf{H}_{\mathsf{cmt}}$, $\left(w_{j,1}^{(t)}, w_{j,2}^{(t)}\right) \neq \left(g_{j,1}^{(t)}, g_{j,2}^{(t)}\right)$ holds if and only if $h_j^{(t)} \neq \mathsf{H}_{\mathsf{cmt}}(w_{j,1}^{(t)}, w_{j,2}^{(t)})$ holds.

We next evaluate the abort probability in (R2.1), which does not exist in **Game**$_1$, for some t-th query to Sign phase. The non-existence of $\left(\left(g_{j,1}^{(t)}, g_{j,2}^{(t)}\right), h_j^{(t)}\right)$ means that $\mathcal{A}$ sends $h_j^{(t)}$ which is not obtained from $\mathsf{H}_{\mathsf{cmt}}$ in R1. On the other hand, $\mathcal{A}$ is required to broadcast $\left(w_{j,1}^{(t)}, w_{j,2}^{(t)}\right)$ such that $h_j^{(t)} = \mathsf{H}_{\mathsf{cmt}}(w_{j,1}^{(t)}, w_{j,2}^{(t)})$ in R2, and hence $\mathcal{A}$ needs to guess such a pair. Since the hash value of $\left(g_{j,1}^{(t)}, g_{j,2}^{(t)}\right)$ is determined by LS, $\mathcal{A}$ can get such a pair with probability $1/2^{\ell}$. Thus, we have

$$|\mathsf{Win}_2 - \mathsf{Win}_1| \le \frac{1}{2^{\ell}}. \tag{6}$$

Game$_3$ This game excludes the possibility that $\mathcal{A}$ predetermines the challenge value c. Namely, this is the same as **Game**$_2$ except that the hash value $c^{(t)}$ is directly sampled from $\mathbb{Z}_q$ instead of using LS as in Figure 9. **Game**$_3$ behaves like **Game**$_2$ unless the "failure" event is happened: for some t-th query to Sign phase, the tuple $\left(apk^{(t)}, L^{(t)}, w_1^{(t)}, w_2^{(t)}, M^{(t)}\right)$, which is determined before running (R2.4), is already contained in $\mathfrak{L}_{\mathsf{sig}}$ as some input. We now focus on the pair $\left(w_1^{(t)}, w_2^{(t)}\right)$. As

R1 and R2 are replaced with the followings:

R1: $\mathcal{C}$ broadcasts $h_1^{(t)}$ after the followings:

(R1.1) $c^{(t)} \in_{\mathrm{U}} \mathbb{Z}_q$; $z_1^{(t)} \in_{\mathrm{U}} \mathbb{Z}_q$.

(R1.2) $c_1^{(t)} = \mathsf{H}_{\mathsf{agg}}(pk^*, L^{(t)}) \cdot c^{(t)} \bmod q$.

(R1.3) $\left(w_{1,1}^{(t)}, w_{1,2}^{(t)}\right) = \left(\mathsf{g}^{z_1^{(t)}}\left(y_{1,1}^*\right)^{-c_1^{(t)}}, \mathsf{h}^{z_1^{(t)}}\left(y_{1,2}^*\right)^{-c_1^{(t)}}\right)$, where $\left(y_{1,1}^*, y_{1,2}^*\right) = pk^*$.

(R1.4) $h_1^{(t)} = \mathsf{H}_{\mathsf{cmt}}(w_{1,1}^{(t)}, w_{1,2}^{(t)})$.

R2: Receiving $\left\{h_j^{(t)}\right\}_{j=2}^{S}$ from $\mathcal{A}$, $\mathcal{C}$ broadcasts $\left(w_{1,1}^{(t)}, w_{1,2}^{(t)}\right)$ after the following:

(R2.1) For each $2 \leq j \leq S$, find $\left(\left(g_{j,1}^{(t)}, g_{j,2}^{(t)}\right), h_j^{(t)}\right)$ from $\mathfrak{L}_{\mathsf{cmt}}$. It aborts if there is no such pair.

(R2.2) $\left(w_1^{(t)}, w_2^{(t)}\right) = \left(w_{1,1}^{(t)}, w_{1,2}^{(t)}\right) \cdot \prod_{j=2}^{S}\left(g_{j,1}^{(t)}, g_{j,2}^{(t)}\right)$.

(R2.3) $apk^{(t)} \leftarrow \mathsf{KAgg}(L^{(t)})$.

(R2.4) $\mathfrak{L}_{\mathsf{sig}} = \mathfrak{L}_{\mathsf{sig}} \cup \left\{\left(apk^{(t)}, L^{(t)}, w_1^{(t)}, w_2^{(t)}, M^{(t)}\right), c^{(t)}\right\}$.

Figure 10: Changed processes in **Game**$_4$ from the previous games

in (R1.1), the pair $\left(w_{1,1}^{(t)}, w_{1,2}^{(t)}\right)$ of $\left(w_1^{(t)}, w_2^{(t)}\right) = \left(w_{1,1}^{(t)}, w_{1,2}^{(t)}\right) \cdot \prod_{j=2}^{S}\left(g_{j,1}^{(t)}, g_{j,2}^{(t)}\right)$ is set as $\left(\mathsf{g}^{r_1^{(t)}}, \mathsf{h}^{r_1^{(t)}}\right)$ for $r_1^{(t)} \in_{\mathrm{U}} \mathbb{Z}_q$. Lemma 2 implies that $\left(w_1^{(t)}, w_2^{(t)}\right)$ is uniformly distributed over $\mathfrak{G}_{\prod_{j=2}^{S}\left(g_{j,1}^{(t)}, g_{j,2}^{(t)}\right)}$, and $\left(w_1^{(t)}, w_2^{(t)}\right)$ is uniquely determined by $r_1^{(t)}$ only. Therefore, the probability of the failure event is evaluated by $(t-1)/q$ for each t-th query in the similar manner to the evaluation of Win_1. Since $\mathcal{A}$ makes at most Q_{sig} queries to $\mathsf{H}_{\mathsf{sig}}$ and $\mathcal{C}$ appends pairs to $\mathfrak{L}_{\mathsf{sig}}$ in (R2.4) at most Q_S times, the total probability of failure in this game is evaluated by $\sum_{t=1}^{Q_{\mathsf{sig}}+Q_S}(t-1)/q = \left(Q_{\mathsf{sig}} + Q_S\right)^2/q$. Thus we have

$$|\mathsf{Win}_3 - \mathsf{Win}_2| \leq \frac{(Q_{\mathsf{sig}} + Q_S)^2}{q}. \tag{7}$$

Game$_4$ In this game, Sign phase is changed to the one so that $\mathcal{C}$ can computes signatures without the secret key. This is the same as **Game**$_3$ except that the generation process of $\left(w_{1,1}^{(t)}, w_{1,2}^{(t)}\right)$ and that of $z^{(t)}$ are replaced as in Figure 10. We now show that these replacements do not affect the winning probability.

We first show that $\left(\left(w_1^{(t)}, w_2^{(t)}\right), c_1^{(t)}, z_1^{(t)}\right)$ induces the acceptance on Ver. As shown in Eq. (3), $\left(\mathsf{g}^{z_1^{(t)}}, \mathsf{h}^{z_1^{(t)}}\right) = \left(w_{1,1}^{(t)}\left(y_{1,1}^*\right)^{c_1^{(t)}}, w_{1,2}^{(t)}\left(y_{1,2}^*\right)^{c_1^{(t)}}\right)$ for $pk^* = \left(y_{1,1}^*, y_{1,2}^*\right)$ is required. The process (R1.3) of **Game**$_4$ implies that the same condition holds. Hence $\left(\left(w_1^{(t)}, w_2^{(t)}\right), c_1^{(t)}, z_1^{(t)}\right)$ in **Game**$_4$ is accepted by Ver.

We next estimate the distributions of the broadcasted values $\left(h_1^{(t)}, \left(w_{1,1}^{(t)}, w_{1,2}^{(t)}\right), z_1^{(t)}\right)$ by $\mathcal{C}$ in **Game**$_3$ and **Game**$_4$. $\left(w_{1,1}^{(t)}, w_{1,2}^{(t)}\right)$ is set as $\left(\mathsf{g}^{r_1^{(t)}}, \mathsf{h}^{r_1^{(t)}}\right)$ for $r_1^{(t)} \in_{\mathrm{U}} \mathbb{Z}_q$ in **Game**$_3$. Lemma 2 implies that this is uniformly distributed over $\mathfrak{G}$. On the other hand, it is set as $\left(w_{1,1}^{(t)}, w_{1,2}^{(t)}\right) = \left(\mathsf{g}^{z_1^{(t)}}\left(y_{1,1}^*\right)^{-c_1^{(t)}}, \mathsf{h}^{z_1^{(t)}}\left(y_{1,2}^*\right)^{-c_1^{(t)}}\right)$ in **Game**$_4$. Since $\left(y_{1,1}^*, y_{1,2}^*\right)$ is generated as $\left(\mathsf{g}^{sk^*}, \mathsf{h}^{sk^*}\right) \leftarrow \mathsf{KGen}(pp)$

Game$_5$: In Init phase, $\mathcal{C}$ generates pk^* from $\mathsf{Gen}_{\mathrm{loss}}^{\mathsf{DDH}}(pp)$ instead of $\mathsf{KGen}(pp)$.

Figure 11: Changed processes in **Game**$_5$ from the previous games

as described in Init phase, we have $\left(w_{1,1}^{(t)}, w_{1,2}^{(t)}\right) = \left(\mathsf{g}^{z_1^{(t)} - sk^* \cdot c_1^{(t)}}, \mathsf{h}^{z_1^{(t)} - sk^* \cdot c_1^{(t)}}\right)$. It follows from $z_1^{(t)} \in_{\mathrm{U}} \mathbb{Z}_q$ and Lemma 2 that $z_1^{(t)} - sk^* \cdot c_1^{(t)} \bmod q$ is uniformly distributed over $\mathbb{Z}_q$, and hence $\left(w_{1,1}^{(t)}, w_{1,2}^{(t)}\right)$ is also uniformly distributed over $\mathfrak{G}$. The distributions of $h_1^{(t)}$ are the same in both games, because $h_1^{(t)}$ is sampled uniformly from $\{0,1\}^\ell$ by LS in both games. In (R2.6) of **Game**$_3$, $z_1^{(t)}$ is set as $r_1^{(t)} + sk^* \cdot c_1^{(t)} \bmod q$ for $r_1^{(t)} \in_{\mathrm{U}} \mathbb{Z}_q$, whereas it is chosen uniformly at random from $\mathbb{Z}_q$ in (R1.1) of **Game**$_4$. Lemma 2 implies that $z_1^{(t)}$ of **Game**$_3$ is also uniformly distributed over $\mathbb{Z}_q$. Therefore, the distributions of $\left(h_1^{(t)}, \left(w_1^{(t)}, w_2^{(t)}\right), z_1^{(t)}\right)$ in **Game**$_3$ and **Game**$_4$ are identical. Thus, we have

$$\mathsf{Win}_4 = \mathsf{Win}_3. \tag{8}$$

Game$_5$ In this game, the public key is replaced with the one which is generated by the lossy DDH instance generator. This game is the same as **Game**$_4$ except that pk^* is generated from $\mathsf{Gen}_{\mathrm{loss}}^{\mathsf{DDH}}$ instead of $\mathsf{KGen} = \mathsf{Gen}_{\mathrm{reg}}^{\mathsf{DDH}}$. Then, the difference between Win_4 and Win_5 can be evaluated by constructing the DDH adversary $\mathcal{D}$ in a way that $\mathcal{D}$ proceeds in the same way as the challenger $\mathcal{C}$ of **Game**$_4$ (and **Game**$_5$) except that $\mathcal{D}$ sets (pp, pk) to the pair $((\mathbb{G}, q, \mathsf{g}, \mathsf{h}), (y_1, y_2))$ given by the DDH challenger. The formal description is given in Figure 12. Observe that the process of $\mathcal{D}$ coincides with **Game**$_4$ when the given pair $((\mathbb{G}, q, \mathsf{g}, \mathsf{h}), (y_1, y_2))$ is chosen according to the distribution $D_{\mathrm{reg}}(\kappa)$, whereas this does with **Game**$_5$ when the given pair $((\mathbb{G}, q, \mathsf{g}, \mathsf{h}), (y_1, y_2))$ is chosen according to the distribution $D_{\mathrm{loss}}(\kappa)$. Since $\mathcal{A}$ makes queries to $\mathsf{H}_{\mathsf{agg}}$, $\mathsf{H}_{\mathsf{cmt}}$, $\mathsf{H}_{\mathsf{sig}}$, and Sign phases at most Q_{agg}, Q_{cmt}, Q_{sig}, and Q_S times, respectively, and Sign phase is run in time $O(S)$ for each query from $\mathcal{A}$, $\mathcal{D}$ runs in time $T' + O(Q_{\mathsf{agg}} + Q_{\mathsf{cmt}} + Q_{\mathsf{sig}} + Q_S \cdot S) = T$, the (T, ϵ)-DDH assumption implies that it can distinguish these distributions D_{reg} and D_{loss} with probability at most ϵ. Thus, we have

$$|\mathsf{Win}_5 - \mathsf{Win}_4| \leq \epsilon. \tag{9}$$

The upper bound of Win_5 Let $(M^*, L^*, (c^*, z^*))$ be a final output by $\mathcal{A}$ in **Game**$_5$ and let $(w_1^*, w_2^*) = \left(\mathsf{g}^{z^*}(y_1^*)^{-c^*}, \mathsf{h}^{z^*}(y_2^*)^{-c^*}\right)$, where $L^* = \{pk_1^*, pk_2^*, \ldots, pk_S^*\}$ and $apk^* = (y_1^*, y_2^*) \leftarrow \mathsf{KAgg}(L^*)$.

For $c^* \in \mathbb{Z}_q$ of the signature, we now estimate the probability of the existence of a hash value $c^* \in \mathbb{Z}_q$ which induces the acceptance of Ver concerning the fixed (w_1^*, w_2^*). Assume that there exist distinct hash values $c^* \neq c'$ such that both c^* and c' induce the acceptance. Then, it holds that for some $z^*, z' \in \mathbb{Z}_q$,

$$\begin{aligned} w_1^* &= \mathsf{g}^{z^*}(y_1^*)^{-c^*} = \mathsf{g}^{z'}(y_1^*)^{-c'}, \\ w_2^* &= \mathsf{h}^{z^*}(y_2^*)^{-c^*} = \mathsf{h}^{z'}(y_2^*)^{-c'}. \end{aligned}$$

This implies that $y_1^* = \mathsf{g}^{(z^*-z')/(c^*-c')}$ and $y_2^* = \mathsf{h}^{(z^*-z')/(c^*-c')}$. It follows from the definition of KAgg that $y_1^* = \prod_{i=1}^S \left(y_{i,1}^*\right)^{a_i^*}$ and $y_2^* = \prod_{i=1}^S \left(y_{i,2}^*\right)^{a_i^*}$, where $pk_i^* = \left(y_{i,1}^*, y_{i,2}^*\right)$ and $a_i^* = \mathsf{H}_{\mathsf{agg}}(pk_i^*, L^*)$ for any $1 \leq i \leq S$. $pk_1^* = pk^*$ is generated by $\mathsf{Gen}_{\mathrm{loss}}^{\mathsf{DDH}}$. Then, Lemma 1 implies that the probability that distinct c^* and c' exist is $2/q$.

On the other hand, the probability that $\mathcal{A}$ get an accepting hash value c^* under the condition that there is at most one accepting hash value is $1/q$ since any hash value is chosen uniformly at random from $\mathbb{Z}_q$. Thus, we have

$$\mathsf{Win}_5 \leq \frac{2}{q} + \left(1 - \frac{2}{q}\right) \cdot \frac{1}{q} < \frac{3}{q}. \tag{10}$$

Eventually, we can evaluate from Eqs. (4)–(10) that

$$\epsilon' \leq \epsilon + \frac{(Q_{\mathsf{cmt}} + Q_S)^2}{q} + \frac{1}{2^\ell} + \frac{(Q_{\mathsf{sig}} + Q_S)^2}{q} + \frac{3}{q}.$$

□

5 Application to Interactive Aggregate Signature

We consider the *interactive aggregate signature* with the public-key aggregation which is a generalization of the multisignature in this section. As opposed to the multisignature case, the interactive aggregate signature enables each signer to choose their own message instead of the common message.

We show that our proposed multisignature can be converted into the interactive aggregate signature with a slight modification. The conversion is based on the idea of [17] in which the conversion of the multisignature by [2] is considered. In the interactive aggregate signature, each signer can choose their own message. Thus one can consider that the interactive aggregate signature can be obtained from the multisignature by regarding the set of pairs of each signer's public key and message as the common message. However, as discussed in [17], the observation above is insufficient. This is because the interactive aggregate signature has to authenticate the correspondence between signers and messages correctly. In order to solve the problem, we change the signing algorithm slightly so that the signing algorithm takes the signer's index in the *ordered* set of public key/message pairs as input. More precisely, we add the signer's index to the input for the hash function. Surprisingly, this modification does not affect the security proof at all since we employ the random oracle model. Therefore we can apply Theorem 1 in the interactive aggregate signature case.

We start with the definition of the interactive aggregate signature with the public-key aggregation. Based on the syntax by [17], the interactive aggregate signature IAS with the public-key aggregation is defined by the six algorithms $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{KAgg}, \mathsf{KAVer}, \mathsf{Sig}, \mathsf{Ver})$. In a similar manner to the multisignature case, we consider the group of signers $(\mathsf{S}_1, \ldots \mathsf{S}_S)$. The setup algorithm Setup generates a public parameter pp for input security parameter κ. Each signer S_i computes a secret and public key (sk_i, pk_i) by the key generation algorithm KGen from pp. Let $L = \{pk_1, \ldots, pk_S\}$ be an *ordered* set of public keys of signers, namely, the signer S_i has the public key pk_i. The deterministic key aggregation algorithm KAgg computes the aggregated public key $apk = \mathsf{KAgg}(L)$ and the validation of apk is verified by the deterministic algorithm KAVer for (apk, L). The signers issue an aggregated signature σ on the ordered set of public key and message pairs $K = \{(pk_1, m_1), \ldots, (pk_S, m_S)\}$. In the signing protocol, the signer S_i executes the signing algorithm $\mathsf{Sig}(pp, sk_i, K, i)$. The index i represents that S_i corresponds the key and message pair (pk_i, m_i). The deterministic verification algorithm Ver checks whether or not the signature σ is valid on (pp, S, σ).

The security of the interactive aggregate signature is defined in the same way as the multisignature case. The security game in the ppk model is depicted in Figure 13. For any polynomial T and any function ϵ, we say that IAS is (T, ϵ, Q_S)-*ppk secure* if for any adversary $\mathcal{A}$ which runs in time at most $T(\kappa)$ and makes at most $Q_S(\kappa)$ queries in Sign phase, $\mathcal{A}$ can win the ppk game of IAS with probability $\epsilon(\kappa)$ for sufficiently large κ. We also say that IAS is $(T, \epsilon, Q_S, Q_1, Q_2, Q_3, \ldots)$-*ppk secure in ROM* if IAS is (T, ϵ, Q_S)-*ppk* secure where $\mathcal{A}$ makes at most $Q_n(\kappa)$ queries to the random oracle H_n for any n and sufficiently large κ.

The description of our interactive aggregate signature is given in Figure 14. The security IAS is proven in the same way as Theorem 1. As described above, the security proof is just all the same as the one of Theorem 1, thus we omit the proof.

Theorem 2. *Let T be a polynomial and let ϵ be a function. Assume that the (T, ϵ)-DDH assumption holds. Then,* IAS *is $(T', \epsilon', Q_S, Q_{\mathsf{agg}}, Q_{\mathsf{cmt}}, Q_{\mathsf{sig}})$-ppk secure in ROM, where*

$$T' = T - O(Q_{\mathsf{agg}} + Q_{\mathsf{cmt}} + Q_{\mathsf{sig}} + Q_S \cdot S),$$

$$\epsilon' \leq \epsilon + \frac{(Q_{\mathsf{cmt}} + Q_S)^2}{q} + \frac{1}{2^\ell} + \frac{(Q_{\mathsf{sig}} + Q_S)^2}{q} + \frac{3}{q},$$

where Q_S*,* Q_{agg}*,* Q_{cmt} *and* Q_{sig} *are some polynomials which represent the numbers of queries to Sign phase,* $\mathsf{H}_{\mathsf{agg}}$*,* $\mathsf{H}_{\mathsf{cmt}}$ *and* $\mathsf{H}_{\mathsf{sig}}$*, respectively.*

6 Concluding Remarks

We have proposed a first multisignature with public-key aggregation whose security is proven to be tightly secure under the DDH assumption in ROM with respect to the plain public-key model. Although our construction $\mathsf{MSig}_{\mathsf{ours}}$ is almost the same as the DDH-based multisignature by [13], our security proof is totally different from theirs. Namely, we have given tight security while maintaining the public-key aggregation functionality and the efficiency of the LYG multisignature.

As an application of our multisignature scheme, we have shown that our proposed multisignature can be converted into the interactive aggregate signature with a slight modification. This modification does not affect the security proof at all due to the random oracle model. Then we can obtain the security proof of our interactive aggregate signature as a corollary of the one of our proposed multisignature.

Acknowledgment

We would like to thank the anonymous reviewers for their valuable comments and suggestions. This work was supported in part by JSPS KAKENHI Grant Numbers JP18K11288 and JP19K20272.

References

[1] Ali Bagherzandi and Stanisław Jarecki. Multisignatures using proofs of secret key possession, as secure as the Diffie-Hellman problem. In Rafail Ostrovsky, Roberto De Prisco, and Ivan Visconti, editors, *Security and Cryptography for Networks*, pages 218–235, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[2] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 390–399, New York, NY, USA, 2006. ACM.

[3] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, pages 62–73, New York, NY, USA, 1993. ACM.

[4] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the Gap-Diffie-Hellman-group signature scheme. In Yvo G. Desmedt, editor, *Public Key Cryptography — PKC 2003*, pages 31–46, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[5] Dan Boneh. The decision Diffie-Hellman problem. In Joe P. Buhler, editor, *Algorithmic Number Theory*, pages 48–63, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[6] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018*, pages 435–464, Cham, 2018. Springer International Publishing.

[7] Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs. On the security of two-round multi-signatures. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1084–1101, 2019.

[8] Masayuki Fukumitsu and Shingo Hasegawa. A tightly-secure lattice-based multisignature. In *Proceedings of the 6th on ASIA Public-Key Cryptography Workshop*, APKC '19, pages 3–11, New York, NY, USA, 2019. ACM.

[9] Masayuki Fukumitsu and Shingo Hasegawa. Linear lossy identification scheme derives tightly-secure multisignature. In *2020 15th Asia Joint Conference on Information Security (AsiaJCIS)*, pages 24–31, 2020.

[10] Meenakshi Kansal and Ratna Dutta. Round optimal secure multisignature schemes from lattice with public key aggregation and signature compression. In Abderrahmane Nitaj and Amr Youssef, editors, *Progress in Cryptology - AFRICACRYPT 2020*, pages 281–300, Cham, 2020. Springer International Publishing.

[11] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, CCS '03, pages 155–164, New York, NY, USA, 2003. ACM.

[12] Duc-Phong Le, Alexis Bonnecaze, and Alban Gabillon. Multisignatures as secure as the Diffie-Hellman problem in the plain public-key model. In Hovav Shacham and Brent Waters, editors, *Pairing-Based Cryptography – Pairing 2009*, pages 35–51, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[13] Duc-Phong Le, Guomin Yang, and Ali Ghorbani. A new multisignature scheme with public key aggregation for blockchain. In *2019 17th International Conference on Privacy, Security and Trust (PST)*, pages 1–7, 2019.

[14] Zi-Yuan Liu, Yi-Fan Tseng, and Raylin Tso. Cryptanalysis of a round optimal lattice-based multisignature scheme. Cryptology ePrint Archive, Report 2020/1172, 2020. `https://eprint.iacr.org/2020/1172`.

[15] 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*, pages 465–485, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[16] Changshe Ma and Mei Jiang. Practical lattice-based multisignature schemes for blockchains. *IEEE Access*, 7:179765–179778, 2019.

[17] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple Schnorr multi-signatures with applications to bitcoin. Cryptology ePrint Archive, Report 2018/068, 2018. `https://eprint.iacr.org/2018/068`.

[18] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple Schnorr multi-signatures with applications to bitcoin. *Designs, Codes and Cryptography*, 87(9):2139–2164, 2019.

[19] Jong Hwan Park and Young-Ho Park. A tightly-secure multisignature scheme with improved verification. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E99.A(2):579–589, 2016.

[20] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO' 89 Proceedings*, pages 239–252, New York, NY, 1990. Springer New York.

[21] Naoto Yanai. Meeting tight security for multisignatures in the plain public key model. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E101.A(9):1484–1493, 2018.

Given a tuple $pp = (\mathbb{G}, q, \mathsf{g}, \mathsf{h})$ and (y_1, y_2), proceeds as follows:

- Init: $\mathcal{D}$ sets $\mathfrak{L}_{\mathsf{agg}} = \mathfrak{L}_{\mathsf{cmt}} = \mathfrak{L}_{\mathsf{sig}} = \emptyset$, and then sends $(pp, pk^* = (y_1, y_2))$ to $\mathcal{A}$.
- $\mathsf{H}_{\mathsf{agg}}$: Given a pair $\xi^{(t)} = \left(pk^{(t)}, L^{(t)}\right)$, return $a^{(t)}$ for $\left(\mathfrak{L}_{\mathsf{agg}}, a^{(t)}\right) \leftarrow \mathsf{LS}\left(\mathfrak{L}_{\mathsf{agg}}, \mathbb{Z}_q, \xi^{(t)}\right)$.
- $\mathsf{H}_{\mathsf{cmt}}$: Given a pair $\xi^{(t)} = \left(w_1^{(t)}, w_2^{(t)}\right)$, return $h^{(t)}$ for $\left(\mathfrak{L}_{\mathsf{cmt}}, h^{(t)}\right) \leftarrow \mathsf{LS}\left(\mathfrak{L}_{\mathsf{cmt}}, \{0,1\}^{\ell}, \xi^{(t)}\right)$. $\mathcal{D}$ aborts if a *collision* is found, namely LS samples $h^{(t)} \in_{\mathrm{U}} \{0,1\}^{\ell}$ which is already contained in $\mathfrak{L}_{\mathsf{cmt}}$ for some input ξ.
- $\mathsf{H}_{\mathsf{sig}}$: Given a tuple $\xi^{(t)} = \left(apk^{(t)}, L^{(t)}, w_1^{(t)}, w_2^{(t)}, M^{(t)}\right)$, return $h^{(t)}$ for $\left(\mathfrak{L}_{\mathsf{sig}}, h^{(t)}\right) \leftarrow \mathsf{LS}\left(\mathfrak{L}_{\mathsf{sig}}, \mathbb{Z}_q, \xi^{(t)}\right)$.
- Sign: Given a public-key set $L^{(t)} = \left\{pk_1^{(t)}, \ldots, pk_S^{(t)}\right\}$ and a message $M^{(t)}$, $\mathcal{D}$ and $\mathcal{A}$ run the signing protocol in which $\mathcal{D}$ plays a role of the signer S_1 which owns $pk_1^{(t)} = pk^*$. Namely, $\mathcal{D}$ executes $\mathsf{Sig}(pp, sk^*, L^{(t)}, M^{(t)})$ as follows:

  R1: $\mathcal{D}$ broadcasts $h_1^{(t)}$ after the followings:

  (R1.1) $c^{(t)} \in_{\mathrm{U}} \mathbb{Z}_q$; $z_1^{(t)} \in_{\mathrm{U}} \mathbb{Z}_q$.

  (R1.2) $c_1^{(t)} = \mathsf{H}_{\mathsf{agg}}(pk^*, L^{(t)}) \cdot c^{(t)} \bmod q$.

  (R1.3) $\left(w_{1,1}^{(t)}, w_{1,2}^{(t)}\right) = \left(\mathsf{g}^{z_1^{(t)}} \left(y_{1,1}^*\right)^{-c_1^{(t)}}, \mathsf{h}^{z_1^{(t)}} \left(y_{1,2}^*\right)^{-c_1^{(t)}}\right)$, where $\left(y_{1,1}^*, y_{1,2}^*\right) = pk^*$.

  (R1.4) $h_1^{(t)} = \mathsf{H}_{\mathsf{cmt}}(w_{1,1}^{(t)}, w_{1,2}^{(t)})$.

  R2: Receiving $\left\{h_j^{(t)}\right\}_{j=2}^{S}$ from $\mathcal{A}$, $\mathcal{D}$ broadcasts $\left(w_{1,1}^{(t)}, w_{1,2}^{(t)}\right)$ after the following:

  (R2.1) For each $2 \leq j \leq S$, find $\left(\left(g_{j,1}^{(t)}, g_{j,2}^{(t)}\right), h_j^{(t)}\right)$ from $\mathfrak{L}_{\mathsf{cmt}}$. It aborts if there is no such pair.

  (R2.2) $\left(w_1^{(t)}, w_2^{(t)}\right) = \left(w_{1,1}^{(t)}, w_{1,2}^{(t)}\right) \cdot \prod_{j=2}^{S} \left(g_{j,1}^{(t)}, g_{j,2}^{(t)}\right)$.

  (R2.3) $apk^{(t)} \leftarrow \mathsf{KAgg}(L^{(t)})$.

  (R2.4) $\mathfrak{L}_{\mathsf{sig}} = \mathfrak{L}_{\mathsf{sig}} \cup \left\{\left(apk^{(t)}, L^{(t)}, w_1^{(t)}, w_2^{(t)}, M^{(t)}\right), c^{(t)}\right\}$.

  R3: Receiving $\left\{\left(w_{j,1}^{(t)}, w_{j,2}^{(t)}\right)\right\}_{j=2}^{S}$ from $\mathcal{A}$, $\mathcal{D}$ broadcasts $z_1^{(t)}$ after the following:

  (R3.1) abort if there exists an index $2 \leq j \leq S$ such that $\left(w_{j,1}^{(t)}, w_{j,2}^{(t)}\right) \neq \left(g_{j,1}^{(t)}, g_{j,2}^{(t)}\right)$.

  R4: Receiving $\left\{z_j^{(t)}\right\}_{j=2}^{S}$ from $\mathcal{A}$, $\mathcal{D}$ returns $\sigma^{(t)} = \left(c^{(t)}, z^{(t)}\right)$, where $z^{(t)} = \sum_{i=1}^{S} z_i^{(t)} \bmod q$.

- Chal: Given (L^*, M^*, σ^*) from $\mathcal{A}$ finally, $\mathcal{D}$ returns 1 if the followings hold:

(W.1) $pk_1^* = pk^*$ for $L^* = \{pk_1^*, \ldots, pk_S^*\}$.

(W.2) (L^*, M^*) does not appeared in Sign phase.

(W.3) $\mathsf{Ver}(pp, L^*, M^*, \sigma^*) = 1$, namely it satisfies that $c^* = \mathsf{H}_{\mathsf{sig}}(apk^*, L^*, w_1^*, w_2^*, M^*)$, where

- $\sigma^* = (c^*, z^*)$,
- $apk^* = (y_1^*, y_2^*) \leftarrow \mathsf{H}_{\mathsf{agg}}(L^*)$, and
- $(w_1^*, w_2^*) = \left(\mathsf{g}^{z^*} (y_1^*)^{-c^*}, \mathsf{h}^{z^*} (y_2^*)^{-c^*}\right)$.

Figure 12: Description of $\mathcal{D}$

- Init: A challenger $\mathcal{C}$ generates $pp \leftarrow \mathsf{Setup}(1^\kappa)$ and $(sk^*, pk^*) \leftarrow \mathsf{KGen}(1^\kappa)$, and then sends (pp, pk^*) to an adversary $\mathcal{A}$.
- Sign: Given an ordered set of public key and message pairs $K^{(t)}$, $\mathcal{C}$ and $\mathcal{A}$ run the signing protocol in which $\mathcal{C}$ plays a role of the signer S_1 which owns $pk_1^{(t)} = pk^*$, whereas $\mathcal{A}$ does that of the other signers S_i $(2 \le i \le S)$.
- Chal: Given (K^*, σ^*) from $\mathcal{A}$ finally, $\mathcal{A}$ *wins the ppk game of* IAS if the followings hold:

(W.1) $pk_1^* = pk^*$ for $K^* = \{(pk_1^*, m_1^*), \dots, (pk_S^*, m_S^*)\}$.
(W.2) K^* does not appeared in Sign phase.
(W.3) $\mathsf{Ver}(pp, K^*, \sigma^*) = 1$.

Figure 13: Plain public-key game of IAS between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$

- $\mathsf{Setup}(1^\kappa)$ generates $(\mathbb{G}, q, \mathsf{g}, \mathsf{h}) \leftarrow \mathsf{GGen}(1^\kappa)$ as a public parameter pp.
- $\mathsf{KGen}(pp)$ is run by each signer S_i. The procedure is the same as that of $\mathsf{Gen}_{\mathrm{reg}}^{\mathsf{DDH}}(pp)$. We denote by $(sk_i, pk_i) = (x_i, (y_{i,1}, y_{i,2})) \leftarrow \mathsf{KGen}(pp)$ S_i's secret key and public key pair.
- $\mathsf{KAgg}(L)$ returns the aggregated key $apk = (y_1, y_2)$ of the ordered set $L = \{pk_1, \dots, pk_S\}$ of public keys of all the signers S_i by $(y_1, y_2) = \prod_{i=1}^{S} \left(y_{i,1}^{\mathsf{H}_{\mathsf{agg}}(pk_i, L, i)}, y_{i,2}^{\mathsf{H}_{\mathsf{agg}}(pk_i, L, i)} \right)$.
- $\mathsf{KAVer}(apk, L)$ returns 1 if $apk = \mathsf{KAgg}(L)$.
- $\mathsf{Sig}(pp, sk_i, S, i)$ is run by each signer S_i to issue an aggregate signature $\sigma = (c, z)$ for the ordered set $K = \{(pk_1, m_1), \dots, (pk_S, m_S)\}$ as follows:

  R1: S_i broadcasts h_i after the followings:
  (R1.1) $r_i \in_{\mathrm{U}} \mathbb{Z}_q$; $(w_{i,1}, w_{i,2}) = (\mathsf{g}^{r_i}, \mathsf{h}^{r_i})$.
  (R1.2) $h_i = \mathsf{H}_{\mathsf{cmt}}(w_{i,1}, w_{i,2})$.

  R2: S_i receives $\{h_j\}_{j \ne i}$ from the co-signers. Then S_i broadcasts $(w_{i,1}, w_{i,2})$.

  R3: S_i receives $\{(w_{j,1}, w_{j,2})\}_{j \ne i}$ from the co-signers. Then S_i broadcasts z_i after the followings:
  (R3.1) abort if there exists an index $j \ne i$ such that $h_j \ne \mathsf{H}_{\mathsf{cmt}}(w_{j,1}, w_{j,2})$.
  (R3.2) $(w_1, w_2) = \prod_{i=1}^{S} (w_{i,1}, w_{i,2})$.
  (R3.3) $apk \leftarrow \mathsf{KAgg}(L)$.
  (R3.4) $c = \mathsf{H}_{\mathsf{sig}}(apk, K, w_1, w_2)$.
  (R3.5) $c_i = \mathsf{H}_{\mathsf{agg}}(pk_i, L, i) \cdot c \bmod q$.
  (R3.6) $z_i = r_i + sk_i \cdot c_i \bmod q$.

  R4: S_i receives $\{z_j\}_{j \ne i}$ from the co-signers. Then S_i outputs $\sigma = (c, z)$ as a multisignature, where $z = \sum_{i=1}^{S} z_i \bmod q$.
- $\mathsf{Ver}(pp, S, \sigma)$ returns 1 if it satisfies that $c = \mathsf{H}_{\mathsf{sig}}(apk, S, w_1', w_2')$, where
  - $apk = (y_1, y_2) \leftarrow \mathsf{KAgg}(L)$, and
  - $(w_1', w_2') = \left(\mathsf{g}^z y_1^{-c}, \mathsf{h}^z y_2^{-c}\right)$.

Figure 14: Proposed interactive aggregate signature IAS