








Achieving Pairing-Free Aggregate Signatures using Pre-Communication between Signers

Kaoru Takemure^{1,2}(✉) , Yusuke Sakai² , Bagus Santoso¹ ,
Goichiro Hanaoka² , and Kazuo Ohta^{1,2} 

¹ The University of Electro-Communications, Tokyo, Japan

² National Institute of Advanced Industrial Science and Technology (AIST),
Tokyo, Japan

Abstract. Most aggregate signature schemes are relying on pairings, but high computational and storage costs of pairings limit the feasibility of those schemes in practice. Zhao proposed the first pairing-free aggregate signature scheme (AsiaCCS 2019). However, the security of Zhao's scheme is based on the hardness of a newly introduced non-standard computational problem. The recent impossibility results of Drijvers et al. (IEEE S&P 2019) on two-round pairing-free multi-signature schemes whose security based on the standard discrete logarithm (DL) problem has strengthened the view that constructing a pairing-free aggregate signature scheme which is proven secure based on standard problems such as DL problem is indeed a challenging open problem.

In this paper, we offer a novel solution to this open problem. We introduce a new paradigm of aggregate signatures, i.e., aggregate signatures with an additional *pre-communication* stage. In the pre-communication stage, each signer interacts with the aggregator to agree on a specific random value *before deciding messages to be signed*. We also discover that the impossibility results of Drijvers et al. apply if the adversary can decide the whole randomness part of any individual signature. Based on the new paradigm and our discovery of the applicability of the impossibility result, we propose a pairing-free aggregate signature scheme such that any individual signature includes a random nonce which can be freely generated by the signer. We prove the security of our scheme based on the hardness of the *standard DL problem*. As a trade-off, in contrast to the plain public-key model, which Zhao's scheme uses, we employ a more restricted key setup model, i.e., the knowledge of secret-key model.

Keywords: Aggregate Signatures · Pre-Communication · Knowledge of Secret Key Model · Rogue-Key Attack

1 Introduction

Boneh et al. [8] introduced the concept of *aggregate signatures*, in which individual signatures on different messages generated by n signers are combined by

any party acting as an *aggregator* into a single signature with the length shorter than the total length of n individual signatures. The aggregate signature scheme proposed in [8] requires bilinear map computations using pairings in the verification step, and the security of the scheme is based on the hardness of the pairing-based Diffie-Hellman assumption.

However, from the perspective of practical implementation and security guarantee, it is much preferable if we can avoid the pairings completely. First, the pairing computation is still relatively quite costly. Since most pairing-based schemes require pairing computations in verification, for the situation where the verifiers are lightweight devices, such schemes might not be suitable. In addition, recently, large cryptanalytic effort, such as [15, 19], revealed a new weakness of pairing based problems, and in a subsequent paper by Guillevic [15], it was shown that we need to make the field size of the group used for pairing $\approx 75\%$ larger than the initial recommendation of the parameter for 128-bit security.

Recently, Zhao proposed an aggregate signature scheme based on the sigma protocol which does not require pairing computation at all [32]. However, the security of his scheme is based on the hardness of a non-standard computational problem, i.e., the *non-malleable discrete logarithm (NMDL)* assumption, which is newly introduced by Zhao in the same paper.

Therefore, constructing an aggregate signature scheme with the following properties is a very important open problem from the practical and theoretical points of view: (1) *pairing-free*, i.e., the scheme does not rely on pairing computations or pairing-based assumption, and (2) provably secure based on well-established standard assumptions, e.g., standard discrete logarithm problem. The aim of this paper is to propose a solution to this open problem. For simplicity, we will focus only on pairing-free schemes here afterward.

1.1 Properties of Aggregate Signatures and Multi-signatures

Another cryptographic primitive which is closely related to aggregate signatures is *multi-signatures*. In a multi-signature scheme, the combined signature must be the combination of signatures on the *same* message, while in an aggregate signature scheme, the combined signature can be the combination of signatures on *different* messages. We will show below several properties related to the signature generating procedure and the security, which most aggregate signatures and multi-signatures have in common.

Stages in Combining Signatures. Here, we unify the representation of signature generating procedures in most (pairing-free) multi-signatures and aggregate signatures into a sequence of three stages.¹

- *Stage I (Offline Stage).* In this stage, each signer performs the necessary interactive communication with other signers before deciding the message to be signed.

¹ It should be noted that an aggregate signature schemes or a multi-signature do not have to have all the three stages.

Table 1. Comparison among Pairing-Free Multi-signature and Aggregate Signature Schemes

	Multi signatures			Aggregate signatures	
	BN(-IAS) [5]	CoSi [10, 29]	mBCJ [2, 10]	Zhao [32]	PCAS
#rounds in Stage I	0	0	0	0	1
Non-interactive message decision in Stage II	No	No	No	Yes	Yes
#rounds in Stage III	3	2	2	1	1
#allowed concurrent signing queries	poly(n)	log(n)	poly(n)	poly(n)	poly(n)
Security Assumption	DL	OMDL	DL	NMDL	DL
Key-Setup Model	plain PK	KOSK	KV	plain PK	KOSK

* PCAS is our proposed scheme. The first row and the third row indicate the number of interactive communication between signers in Stage I and Stage III respectively. The second row indicates whether the message decision in Stage II is carried *without* any interaction between signers. The fourth row indicates the maximum number of concurrent signing queries which is allowed without breaking the security of the scheme. Here, n indicates the number of signers. DL, OMDL, NMDL indicate the standard discrete logarithm problem, one-more discrete logarithm problem, and non-malleable discrete logarithm problem [32], respectively. We describe the notions of the key-setup model mentioned at the final row in the paragraph *Attacks and Key-Setup Model* in Sect. 1.1.

- *Stage II (Message Decision Stage)*. In this stage, signers decide the message they will individually sign and eventually include in the final combined signature. In the case of multi-signatures, since all signatures to combine have to be signatures on one single same message, it is almost natural that the signers communicate to each other *interactively* to decide the message to be signed in this stage. In the case of aggregate signatures, generally, a signer does not need to share the message with other signers.
- *Stage III (Online Stage)*. In this stage, signers share specific values related to the messages decided in Stage II to others via interactive communication.

Research Question. We compare several pairing-free multi-signature and aggregate signature schemes in Table 1. Notice that most pairing-free multi-signature schemes require more than one communication round in the online stage (Stage III), while they achieve provable security based on the hardness of standard computational problems [2, 5, 29]. On the other hand, the pairing-free aggregate signature scheme, Zhao’s scheme only requires a single communication round in the online stage, while it achieves provable security using the hardness of newly introduced non-standard computational problems. Our question here is as follows.

“Is it possible to construct a new scheme which achieves the best of the two worlds: (1) one communication round in Stage III, and (2) provable security based on the hardness of standard computational problems ?”

Aggregate Signatures based on Multi-signatures. In a multi-signature scheme, signatures on the same message are combined into the final signature. However, one can easily tweak the scheme such that the combined signature will be a signature on multiple different messages decided by different signers. In Stage II, via an interactive message decision process, each signer can send an individual

message to all other signers and then combine all different individual messages into one single message by simple concatenation. This single message will be the message to be signed which is agreed by all signers. In [5], Bellare and Neven introduced this concept as *Interactive Aggregate Signatures (IAS)*.

Attacks and Key-Setup Model. In both multi-signatures and aggregate signatures, one should consider an attack scenario which is called the *rogue-key attack*. In a rogue-key attack, an attacker generates public keys dishonestly and tries to forge a combined signature involving such dishonest keys. In general, we can guarantee the security of the scheme against the rogue-key attacks using the following two basic strategies. The first is (i) *to prove directly that there exists no rogue-key attack*, and the second is (ii) *to exclude rogue-key attacks by a specific key registration protocol*. These two approaches are formally modeled by (i) *the plain public-key (PK) model* [5] and (ii) *the knowledge of secret keys (KOSK) model* [6, 21], respectively.

- (i) *The plain PK model* is the model without any assumption in the key setup. In the security model, an adversary can freely choose all cosigners' public keys excluding at least one honest signer's key.
- (ii) *The KOSK model* is the model where all signers need to prove the validity of their public key. In the security model, an adversary can freely pick all cosigners' public keys, but it must output the secret keys corresponding to these public keys. In practice, the KOSK model can be implemented using one of the following models: (1) a *trusted setup* model [25], in which a dedicated key registration protocol is needed to be executed by each signer, (2) the *key verification (KV)* model [2], and (3) the *proof-of-possession (PoP)* model [27], where each signer submits a certificate to prove possession of a secret key.

1.2 Our Contributions

In this paper, we propose a new paradigm for constructing aggregate signature which we call *aggregate signatures with pre-communication (AS with PreCom)*. We propose an aggregate signature scheme based on the new paradigm, which we name PCAS, and proved its security based on the standard discrete logarithm (DL) assumption in the KOSK model using a random oracle. We show the comparison of PCAS with other *pairing-free* multi-signature and aggregate signature schemes in Table 1 (We also show the performance comparison among aggregate signature scheme and related schemes in Table 2 in Sect. 4).

Most aggregate signature schemes (either with pairings or without pairings) do not have any interactive round between signers in Stage I. In contrast, an aggregate signature scheme with pre-communication, have one interactive round in Stage I before the message deciding stage (Stage II). We believe that this drawback only has minor effects on the practical use. As shown in Table 1, PCAS still keeps the most important feature of aggregate signatures, i.e., any signer is allowed to choose their individual message to be signed without interacting with

other signers in Stage II. Moreover, one should notice that the total number of interactive rounds in PCAS, i.e., two, is the lowest number of interactive rounds that the multi-signature schemes (either with pairing or without pairing) can ever achieve in theory and these multi-signature schemes are being used widely in real-world practice today [11].

Comparison to Zhao’s Aggregate Signature Scheme. As opposed to Zhao’s aggregate signature scheme [32] which is proven based on a non-standard computational problem NMDL, our proposed aggregate signature scheme PCAS is proven secure under the standard discrete-logarithm (DL) assumption. To prove the security of our scheme, we assume the KOSK model as the key-setup model.² Although the KOSK model is more costly compared to the plain PK model, there are several practical methods for implementing the KOSK model as mentioned in the previous section.

PCAS achieves a smaller signature size than the signature size in Zhao’s scheme [32]. Concretely, let n be the number of individual signatures to combine. For λ -bit security, in Zhao’s scheme the combined signature includes n group elements whose total size is about $2\lambda n$ bits, while in PCAS the combined signature includes a random string with the total size of λn bits.

Circumventing Impossibility Results of Drijvers et al. [10]. In [10] Drijvers et al. showed attacks against several two-round multi-signature schemes and also show the impossibility of proving the security of those schemes. Since our proposed aggregate signature scheme bears a resemblance to CoSi scheme [29], one of the multi-signature schemes covered in [10], one may wonder whether the impossibility results of Drijvers et al. are applicable to our proposed scheme. However, as shown in a more detailed explanation at Sect. 5, the random value t which is freshly chosen by the signer in every signature query, is actually sufficient for our proposed scheme to avoid the impossibility results. Concretely, the root of the impossibility results is the adversary’s ability to force the honest signer to use a specific hash value c of the adversary’s choice in the response to a signature query. This ability is eliminated by the random value t , which makes the adversary unable to predict the challenge c that the honest signer will use since c is computed depending on the value of t in our scheme. For more detail, see the full version of this paper.

1.3 Difficulty and Our Techniques

The Schnorr digital signature scheme [28] built from the Schnorr identification by the Fiat-Shamir transform [12] is used in many applications as well as ours because of the small computational complexity and well-established security. Let q be a prime integer, g be a generator of a cyclic group G with order q ,

² The KOSK model is essential because there is a *sub-exponential* attack against this scheme in the plain PK model by using k -sum algorithm as in [10]. For more detail of this attack, see the full version of this paper.

X be a public key, m be a message, (R, s) is a signature on m , and H be a hash function $H : \{0, 1\}^* \rightarrow Z_q$. The verification formula of the Schnorr signature scheme is $R = g^s X^{-c}$ where $c = H(R, X, m)$.³ We can aggregate the formula because of the linearity. More specifically, for all $i = 1, \dots, n$, when each signer \mathcal{S}_i (with public key X_i) submits a signature (R_i, s_i) on a message m_i , one can compress all signatures into (\tilde{R}, \tilde{s}) where $\tilde{R} = \prod_{i=1}^n R_i$ and $\tilde{s} = \sum_{i=1}^n s_i \pmod q$. Then the verification formula is $\tilde{R} = g^{\tilde{s}} \prod_{i=1}^n X_i^{c_i}$ where $c_i = H(R_i, X_i, m_i)$.⁴

However, there are three difficulties in extending the Schnorr digital signature scheme to multi-signatures or aggregate signatures by the above compression.

First, (I) *all signers need to share \tilde{R} before generating a signature*. On the Schnorr signature, a signer inputs R_i to the hash function to generate c_i . If an aggregator compresses all signers' R_i into \tilde{R} , a verifier cannot know R_i and cannot compute c_i . Thus we need to replace R_i with \tilde{R} in the input of the hash function, but in that case, then all signers require \tilde{R} for generating signatures.

Second, (II) *by sharing \tilde{R} , a reduction fails to simulate the honest signer in the security proof*. In the Schnorr signature scheme, the reduction simulates the signing oracle by the honest-verifier zero-knowledge property of the sigma protocol and the random oracle. In detail, the reduction chooses s and c at uniformly random from Z_q , computes $R \leftarrow g^s X^{-c}$, sets $H(R, X, m) \leftarrow c$ in the random oracle table, and return (R, s) as a signature. If $H(R, X, m)$ is predefined by hash queries, the reduction cannot set $H(R, X, m) \leftarrow c$ and cannot complete this simulation. In the case that the input R of the hash function is changed to \tilde{R} , the reduction can compute \tilde{R} only after an adversary outputs all cosigners' R_i . Thus an adversary can know \tilde{R} before the reduction knows it, and can prevent the reduction from setting $H(\tilde{R}, X, m) \leftarrow c$ in the random oracle table by making a hash query (\tilde{R}, X, m) .

Third, (III) *it is hard to compute the solution of the DL problem from forgeries because of the term related to cosigners*. Recall that the verification formula is $\tilde{R} = g^{\tilde{s}} \prod_i X_i^{-c_i}$. Let \tilde{X} be an instance of the DL problem the reduction tries to solve. For simplicity, we assume the restricted case where the k -th signer is the honest signer ($X_k = \tilde{X}$) and a forger assigns distinct group elements to cosigners' key.⁵ The reduction uses the rewinding technique and obtains the two formulae $\tilde{R} = g^{\tilde{s}} \prod_i X_i^{-c_i}$ and $\tilde{R}' = g^{\tilde{s}'} \prod_i X_i^{-c'_i}$ where $\tilde{R} = \tilde{R}'$, and $c_k \neq c'_k$. When the reduction tries to extract the discrete logarithm of \tilde{X} by dividing the above two formulae, it can obtain

$$\tilde{X}^{c_k - c'_k} = g^{\tilde{s} - \tilde{s}'} \prod_{i \neq k} X_i^{-c_i + c'_i}. \quad (1)$$

³ For the convenience of considering multiple users, we added the public key to the input of the hash function.

⁴ If we set $m_1 = m_2 = \dots = m_n$, then we can see it as multi-signatures.

⁵ In [5], Bellare and Neven consider the case where there are several public keys with the same values.

However, notice here that the term $\prod_{i \neq k} X_i^{-c_i + c'_i}$ related to cosigners becomes the barrier for the reduction to extract the discrete logarithm of \tilde{X} .

Next, we show how Bellare-Neven multi-signature scheme [5] circumvents the above difficulties. First, note that in a multi-signature scheme, all signers who participate will generate signatures on the same message and are allowed to interact with each other in the signing procedure. For (I), all signers share $\{R_i\}_i$ in the signing protocol and compute \tilde{R} . For (II), each signer generates a commitment to R_i by using a hash function and sends the commitment to all other signers, before it sends R_i . By this, in the security proof, when the reduction receives all cosigners' commitments to $\{R_i\}_i$, it can compute \tilde{R} by searching all cosigners' R_i in the random oracle table simulating the hash function before the adversary knows \tilde{R} . For (III), the reduction programs the random oracle carefully as follows. For the hash query (\tilde{R}, X_k, L, m) where L is the list of the signers' public keys, the reduction fixes $H(\tilde{R}, X_i, L, m)$ to the random value for all $i \neq k$ before it defines $H(\tilde{R}, X_k, L, m)$. By this careful programming of a random oracle, the reduction can make the situation that $c_i = c'_i$ holds for $i \neq k$ in Eq. (1) and can cancel out the term $\prod_{i \neq k} X_i^{-c_i + c'_i}$ related to cosigners. Then it can extract the solution to the problem as $(\tilde{s} - \tilde{s}') / (c_k - c'_k) \pmod q$ without cosigners' secret key. Thus, this scheme can be proved secure in the *plain PK model*.

Unfortunately, these techniques to circumvent the three difficulties (I)-(III) are effective only for multi-signatures, not for aggregate signatures. Recall that the techniques to circumvent (I) and (II) require the communication in the signing phase. Applying them to aggregate signatures will automatically destroy the advantage of aggregate signatures over multi-signatures, i.e., the freedom of the signers to sign their own chosen message individually without sharing it with other signers beforehand. And the technique to circumvent (III) is simply impossible to apply on aggregate signatures. This technique works *only if* all messages in the signatures to be combined are fixed *before* the rewinding point in the security proof. However, in aggregate signatures, the cosigners controlled by the adversary always have the freedom to change the messages in the signatures to be combined any time, *even after* the rewinding point. Therefore, we need to explore other approaches to overcome the above three difficulties in aggregate signatures.

We overcome the three difficulties as follows. For (I), noticing that \tilde{R} is pre-communicable, we introduce the pre-communication and exclude the communication in the signing protocol. For (II), we resolve the difficulty by adding the random value t_i generated by the signer in the signing phase to the input of the hash function to produce c_i . In more details, each c_i is computed as $c_i \leftarrow H(\tilde{R}, X_i, t_i, m_i)$ and a set of t_i is included in an aggregate signature as $(\tilde{R}, \tilde{s}, \{t_i\}_i)$. Consequently, thanks to t_i , the reduction can succeed in simulating the honest signer no matter how cleverly the adversary behaves, because the adversary should guess the random value t_i . For (III), we use the KOSK model. By this, the reduction can obtain cosigners' secret keys x_i for $i \neq k$ and com-

pute the discrete logarithm of $\prod_{i \neq k} X_i^{-c_i + c'_i}$ in Eq. (1). Therefore it can extract a solution to the DL problem as $(\tilde{s} - \tilde{s}' - \sum_{i \neq k} x_i(c_i - c'_i))/(c_k - c'_k) \pmod q$.⁶

1.4 Related Work

Boneh et al. suggested the idea of aggregate signatures and proposed the first aggregate signature scheme using pairing [8]. Bellare et al. showed that the aggregate signature scheme [8] is secure even if the restriction of different pairs of a public key and a message between all signers is eliminated [4]. There are many pairing-based aggregate signature schemes [1, 7, 16, 17, 21, 23, 26].

Lysyanskaya et al. introduced a notion of sequential aggregate signatures, where signers sequentially generate a signature on his message by using previous signers' messages and signatures and provided the first sequential aggregate signature scheme built from the RSA assumption [22]. After that, pairing-based sequential aggregate signature schemes [13, 20, 21] and pairing-free sequential aggregate signature schemes [3, 9, 26] were proposed.

Gentry and Ramzan proposed the first aggregate signature in the synchronized setting [14], and Ahn et al. formalized the synchronized aggregate signatures, in which signatures generated in the same period can be compressed into an aggregate signature. Hohenberger and Waters provided an RSA-based synchronized aggregate signature scheme [18]. We can implement an AS with PreCom scheme using a synchronized aggregate signature scheme as follows. In a PreCom phase, the signers can agree on the time period by pre-communication. A restriction of this approach is that the number of the signatures the signers can issue is bounded at the setup time. Our proposed scheme does not have such a restriction.

Identity-based aggregate signatures [7, 14, 17, 30] are the aggregate signatures in which each signer is assigned an ID and creates a signature by using a secret key that a private key generator generates by the master secret key and the signer's ID. Bellare and Neven proposed a DL-based multi-signature scheme and mentioned the applicability of multi-signatures to (interactive) aggregates signature [5]. This application presupposes that signers can share messages.

Zhao proposed an aggregate signature scheme for blockchain applications [32]. This scheme is asynchronous and constructed from general elliptic curves. He stated that the proposed scheme is more applicable to blockchain applications than pairing-based aggregate signatures for the system complexity and the verification speed. His scheme is an extension of the Γ -signature [31] to aggregate signatures. Though the signature size linearly depends on the number of signers, this scheme is proved secure in the plain PK model and requires no communication between signers for signing. The security of this scheme is based on the non-malleable discrete logarithm (NMDL) assumption. This assumption

⁶ Here, we implicitly assumed the same restriction as we assumed in Sect. 1.3 for discussing Bellare-Neven's approach to the difficulty (III). However, this restriction can be removed in the actual proof of this proposed scheme. For detail, see the security model in Sect. 3.1.

is only justified in the generic group model [24] with random oracles, where an adversary is allowed to query both of the random oracle and the generic group oracle.

2 Preliminaries

2.1 Notation

For a prime integer q , we denote the ring of integers modulo q by Z_q and the multiplicative group of Z_q by Z_q^* . Let G be a cyclic group of order q and let g be a generator of G . For a set A , we write $a \stackrel{\$}{\leftarrow} A$ to mean that a is chosen at uniformly random from A . For a probabilistic algorithm B , we write $b \leftarrow \mathcal{B}(\beta_1, \dots; \rho)$ to mean that B on inputs β_1, \dots and random tape ρ outputs b , and $b \stackrel{\$}{\leftarrow} \mathcal{B}(\beta_1, \dots)$ to mean that ρ is chosen at uniformly random and let $b \leftarrow \mathcal{B}(\beta_1, \dots; \rho)$.

2.2 Hardness Assumption

We now recall the definition of the discrete logarithm assumption.

Definition 1 (Discrete Logarithm Assumption). For (G, g, q) , let \mathcal{E} be a PPT algorithm that is given y chosen at uniformly random from G . We say that \mathcal{E} (t, ε) -breaks DL if \mathcal{E} runs in time at most t and outputs x such that $y = g^x$ with probability at least ε .

3 Aggregate Signatures with Pre-Communication

3.1 Definition

In this paper, particularly, we introduce a model where, before signing, each signer communicates with the aggregator and shares information in advance, which we hereafter call helper information. Note that communication in this model is the one-to-one communication between a signer and the aggregator. We now describe the definition of aggregate signature (AS) with pre-communication (PreCom) below. We illustrate pre-communication and aggregation in Fig. 1.

Definition 2 (AS with PreCom). An AS with PreCom consists of the following five algorithms and one protocol. Let n be the number of signers and let i be the index of a signer.

Setup $(1^\lambda) \rightarrow pp$. The public parameter generation algorithm takes as input a security parameter 1^λ , then outputs a public parameter pp .

KeyGen $(pp) \rightarrow (pk, sk)$. The key generation algorithm takes as input a public parameter pp , then outputs a public key pk and a secret key sk .

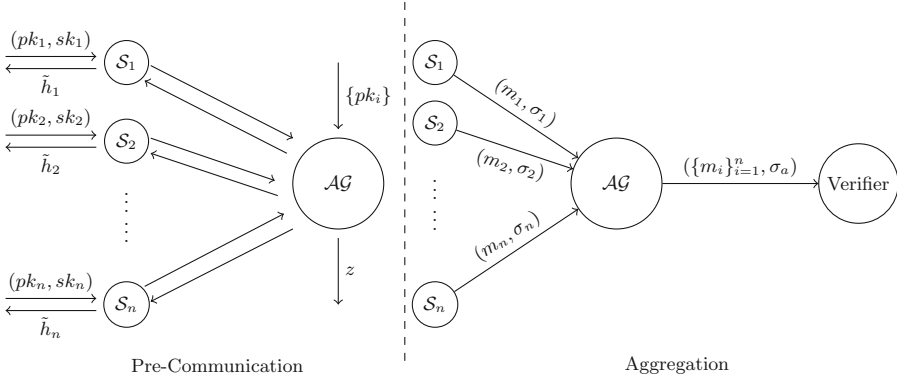


Fig. 1. Aggregate Signature with Pre-Communication: The arrows that denote the communication are simplified to one round communication as in the proposed scheme in this paper. In our model, we do not restrict the number of rounds to one.

PreCom $\langle \mathcal{S}_1(pk_1, sk_1), \dots, \mathcal{S}_n(pk_n, sk_n), \mathcal{AG}(\{pk_i\}_{i=1}^n) \rangle \rightarrow (\tilde{h}_1, \dots, \tilde{h}_n, z)$. The pre-communication protocol is executed between each signer \mathcal{S}_i with input a public key pk_i and a secret key sk_i and an aggregator \mathcal{AG} with input all the signers' public keys $\{pk_i\}_{i=1}^n$. After the protocol terminates, each \mathcal{S}_i and \mathcal{AG} obtain \tilde{h}_i and z as helper information, respectively.

Sign $(pp, pk, sk, \tilde{h}, m) \rightarrow \sigma$. The signing algorithm takes as input a public parameter pp , a public key pk , a secret key sk , helper information \tilde{h} , and a message m , then outputs a signature σ .

Agg $(pp, z, \{(pk_i, m_i, \sigma_i)\}_{i=1}^n) \rightarrow \sigma_a$. The aggregation algorithm takes as input a public parameter pp , helper information z , and a set of all signers' public keys, messages, and signatures $\{(pk_i, m_i, \sigma_i)\}_{i=1}^n$, then outputs an aggregate signature σ_a .

AggVer $(pp, \{(pk_i, m_i)\}_{i=1}^n, \sigma_a) \rightarrow \{0, 1\}$. The aggregate signature verification algorithm takes as input a public parameter pp , a set of all signers' public keys and messages $\{(pk_i, m_i)\}_{i=1}^n$, and an aggregate signature σ_a , then outputs 0 (REJECT) or 1 (ACCEPT).

For any set of messages $\{m_i\}_{i=1}^n$, if all signers and an aggregator behave honestly, then $\Pr[\mathbf{AggVer}(pp, \{(pk_i, m_i)\}_{i=1}^n, \sigma_a) = 1] = 1$ holds.

Security Model of AS with PreCom. Below, we show the definition of existential unforgeability under the chosen-message attack to AS with PreCom in the random oracle model and knowledge of secret key (KOSK) model [6, 21]. This security definition requires that it be infeasible to forge aggregate signatures involving at least one honest signer. In the security model here, as a forger \mathcal{F} , we consider aggregators who corrupt signers except for one honest signer. Also

the forger \mathcal{F} can execute the pre-communication and the aggregation protocols with an honest signer several times, and after that, it tries to output a forgery. Then the forger \mathcal{F} can arbitrarily choose the corrupted cosigners' public keys even though it must output secret keys corresponding to these public keys. This restriction is called the KOSK model.

Formally, the security model here is defined by the three-phase game of the following.

Setup. The challenger chooses the parameter $pp \stackrel{\$}{\leftarrow} \mathbf{Setup}(1^\lambda)$ and the key pair $(pk, sk) \stackrel{\$}{\leftarrow} \mathbf{KeyGen}(pp)$. It runs a forger \mathcal{F} on input pk and pp .

Signing Queries. The challenger receives (j, i_j) as a *PreCom signing query*. The challenger and \mathcal{F} execute the pre-communication protocol $\mathbf{PreCom} \langle \mathcal{S}_1(pk_1, sk_1), \dots, \mathcal{S}_n(pk_n, sk_n), \mathcal{AG}(\{pk_i\}_{i=1}^n) \rangle \rightarrow (\tilde{h}_1, \dots, \tilde{h}_n, z)$ where the challenger behaves as $\mathcal{S}_{i_j}(pk, sk)$ and all the other parties are controlled by \mathcal{F} . Then, the challenger obtains the helper information \tilde{h}_{i_j} and stores this information with the PreCom signing query. The challenger receives (j, m') as a *message signing query*. It reads out \tilde{h}_{i_j} and computes $\sigma'_j \leftarrow \mathbf{Sign}(pp, pk, sk, \tilde{h}_{i_j}, m')$. The challenger returns σ'_j to \mathcal{F} . \mathcal{F} is allowed to concurrently make any number of above queries where it is allowed to make only one message signing query per one PreCom signing query.⁷

Output. After \mathcal{F} terminates, it outputs n key pairs $\{(pk_i, sk_i)\}_{i=1}^n$, a set of messages $\{m_i^*\}_{i=1}^n$, and a forgery σ_a^* where the following holds.

- $\{pk_i\}_{i=1}^n$ is distinct to each other.
- $pk \in \{pk_i\}_{i=1}^n$.
- sk_k is \perp where k is such that $pk_k = pk$.

If $\mathbf{AggVer}(pp, \{(pk_i, m_i^*)\}_{i=1}^n, \sigma_a^*) = 1$ is true and m_i^* has never been queried where i is such that $pk_i = pk$, then \mathcal{F} is said to succeed in forgery.

Definition 3 (Unforgeability in KOSK Model for AS with PreCom).

Let N be a maximum number of cosigners being involved in the forgery. We say that \mathcal{F} $(t, q_S, q_H, N, \varepsilon)$ -break AS with PreCom if \mathcal{F} runs in at most t time, makes at most q_S signing queries and at most q_H random oracle queries, and succeeds in forgery in the above game with probability at least ε . For an AS scheme with PreCom, if there are no \mathcal{F} that $(t, q_S, q_H, N, \varepsilon)$ -breaks it, we say the scheme is $(t, q_S, q_H, N, \varepsilon)$ -secure.

3.2 Our AS Scheme with PreCom (PCAS)

In this section, we propose the AS scheme with PreCom PCAS based on the discrete logarithm assumption in the random oracle model and the KOSK model. This scheme is an extension of the Schnorr signature scheme to aggregate signatures. We introduce the pre-communication to solve the difficulty (I) in Sect. 1.3 without the communication in the signing phase.

⁷ This restriction is essential. If this restriction is omitted, there is an attack against our proposed scheme. See Remark 1 for more detail.

The Algorithms and Protocol of PCAS. Below, we now show the algorithms and protocol of PCAS.

Setup(1^λ) $\rightarrow pp$. It chooses (G, q, g) , a hash function $H : \{0, 1\}^* \rightarrow Z_q$, and a parameter κ , then outputs $pp = (G, q, g, H, \kappa)$.

KeyGen(pp) $\rightarrow (pk, sk)$. It computes $x \xleftarrow{\$} Z_q$ and $X \leftarrow g^x$, then outputs the public key $pk = X$ and the secret key $sk = x$.

PreCom($\langle \mathcal{S}_1(pk_1, sk_1), \dots, \mathcal{S}_n(pk_n, sk_n), \mathcal{AG}(\{pk_i\}_{i=1}^n) \rangle$) $\rightarrow (\tilde{h}_1, \dots, \tilde{h}_n, z)$. For all $i \in [1, n]$, firstly, each signer \mathcal{S}_i computes $r_i \xleftarrow{\$} Z_q$ and $R_i \leftarrow g^{r_i}$ and sends R_i to the aggregator. The aggregator generates $\tilde{R} \leftarrow \prod_{i=1}^n R_i$ from given $\{R_i\}_{i=1}^n$, and returns \tilde{R} to all the signers. Each signer \mathcal{S}_i and the aggregator store $\tilde{h}_i = (r_i, \tilde{R})$ and $z = \tilde{R}$ as the helper information, respectively.

Sign(pp, pk, sk, \tilde{h}, m) $\rightarrow \sigma$. It chooses a value $t \xleftarrow{\$} \{0, 1\}^\kappa$ at uniformly random, computes $c \leftarrow H(\tilde{R}, X, t, m)$ and $s \leftarrow cx + r \pmod q$, then outputs $\sigma = (s, t)$ as a signature.

Agg($pp, z, \{(pk_i, m_i, \sigma_i)\}_{i=1}^n$) $\rightarrow \sigma_a$. It computes $\tilde{s} \leftarrow \sum_{i=1}^n s_i \pmod q$, then outputs the aggregate signature $\sigma_a = (\tilde{s}, \{t_i\}_{i=1}^n, \tilde{R})$.

AggVer($pp, \{(pk_i, m_i)\}_{i=1}^n, \sigma_a$) $\rightarrow \{0, 1\}$. If $\{pk_i\}_{i=1}^n$ are not distinct to each other, it outputs 0. For all $i \in [1, n]$, it computes $c_i \leftarrow H(\tilde{R}, X_i, t_i, m_i)$. If $\tilde{R} = g^{\tilde{s}} \prod_{i=1}^n X_i^{-c_i}$ holds, then outputs 1. Otherwise outputs 0.

For the verification formula, it holds that $g^{\tilde{s}} \prod_{i=1}^n X_i^{-c_i} = g^{\sum x_i c_i + r_i} g^{\sum -x_i c_i} = g^{\sum r_i} = \tilde{R}$. Thus, an aggregate signature is accepted with probability 1 when it is generated honestly.

Remark 1. Note that already used helper information cannot be reused because the adversary can obtain two distinct signatures generated from the same helper information and extract a secret key by exploiting the special soundness property. Moreover, in the aggregation phase, if several signers fail to participate in this phase, the protocol terminates, and it should be restarted from pre-communication.

The Security of PCAS. We should overcome two difficulty (II) and (III) in Sect. 1.3 to prove PCAS secure. (For more detail, see Sect. 1.3).

To overcome the difficulty (II), we add the random value t to the input of a hash function which produces c . We explain how this t enables us to simulate the signing oracle. Towards this end, let us review how to simulate the honest signer for the Schnorr signature. Firstly, the reduction receives m' from a forger as a signing query, randomly chooses (c, s) and computes $R \leftarrow g^s X^{-c}$ where X is the honest signer's public key. After that, it sets $H(R, X, m') \leftarrow c$ in the random oracle table and return (R, s) as a valid signature to a forger. In this case, $H(R, X, m')$ is not predefined with overwhelming probability because R is a fresh random value generated by the reduction. For PCAS, the reduction needs to set $H(\tilde{R}, X, t, m') \leftarrow c$ in the random oracle table. Although a forger can decide \tilde{R} and m' , it cannot obtain t until the reduction return (s, t) . Therefore,

the reduction can set $H(\tilde{R}, X, t, m') \leftarrow c$ between receiving \tilde{R} and m' from a forger and returning (s, t) .

To overcome the difficulty (III), we consider the KOSK model. By this, the reduction can make use of the cosigners' secret keys to extract the solution to the DL problem. Moreover, the KOSK model is essential for PCAS because there is a rogue-key attack in the plain PK model. We describe this attack in the full version of this paper.

The following theorem states that PCAS is secure under the discrete logarithm assumption in the random oracle model and the KOSK model.

Theorem 1. *If there is a forger \mathcal{F} that $(t, q_S, q_H, N, \varepsilon)$ -breaks PCAS, then there is an algorithm \mathcal{B} that (t', ε') -breaks DL such that*

$$\varepsilon' \geq \frac{\varepsilon^2}{q_H + 1} - \frac{2q_S(2q_H + q_S - 1)}{(q_H + 1)2^{\kappa+1}} - \frac{1}{q}, \quad t' \leq 2t + 2q_S t_{exp} + O(q_H + q_S + 1),$$

where t_{exp} is the time for an exponentiation in G and we assume that $\kappa = \lambda$.

Proof. We first show the construction of the algorithm \mathcal{B} which can solve the DL problem using the forger \mathcal{F} . \mathcal{B} is given an instance of the DL problem Y and a parameter (G, q, g) .

To construct \mathcal{B} , let \mathcal{A} be the algorithm as follows. On inputs (G, q, g, Y) , $h_1, \dots, h_{q_H+1} \in Z_q$, and a random tape ρ , \mathcal{A} runs \mathcal{F} on inputs (G, q, g) and Y as an honest signer's public key. It initializes counters $ctr_1 = 1$, $ctr_2 = 0$ and tables $T[\cdot]$, $L[\cdot]$ to be empty, where $T[\cdot]$ is a random oracle table and $L[\cdot]$ is a table that stores helper information of PreCom for signing queries. It responds to \mathcal{F} 's hash queries and signing queries as follows.

Hash Query $H(Q)$. A query Q is parsed as $Q = (\tilde{R}, X, t, m)$. In the case that $X = Y$, \mathcal{A} lets $T[Q] = h_{ctr_1}$ and $ctr_1 \leftarrow ctr_1 + 1$ if $T[Q]$ is undefined. In the case that $X \neq Y$, \mathcal{A} lets $c \xleftarrow{\$} Z_q$, $T[Q] \leftarrow c$ if $T[Q]$ is undefined. It returns $T[Q]$.

Signing Query. Firstly, when \mathcal{A} receives the signal to start PreCom, it sets $ctr_2 \leftarrow ctr_2 + 1$, chooses $s', c' \xleftarrow{\$} Z_q$, computes $R' \leftarrow g^{s'} X_k^{-c'}$, and sends R' to \mathcal{F} . After that, when \mathcal{A} is given \tilde{R}' from \mathcal{F} , \mathcal{A} assigns $L[ctr_2] \leftarrow (s', c', \tilde{R}')$. When receiving a query (m', J) , \mathcal{A} sets $M' \leftarrow M' \cup \{m'\}$ and reads $L[J]$. It returns \perp to \mathcal{F} if $L[J]$ is empty. \mathcal{A} chooses $t' \xleftarrow{\$} \{0, 1\}^\kappa$ and sets $Q' = (\tilde{R}', Y, t', m')$. It sets $bad \leftarrow true$ and halts with output \perp if $T[Q']$ is already defined. Otherwise it assigns $T[Q'] \leftarrow c'$, empties $L[J]$ and returns (s', t') to \mathcal{F} .

Finally, \mathcal{F} outputs $\{X_i^*\}_{i=1}^{n^*}$ which is the set of public keys including Y , $\{x_i^*\}_{i \in [1, n^*] \setminus \{k\}}$ which is the set of secret keys corresponding to the public keys except X_k such that $Y = X_k$, the set of messages $\{m_i^*\}_{i=1}^{n^*}$, and a forgery $(\tilde{s}^*, \{t_i^*\}_{i=1}^{n^*}, \tilde{R}^*)$. \mathcal{A} checks whether $m_k^* \notin M'$ and $\mathbf{AggVer}(pp, \{(X_i^*, m_i)\}_{i=1}^{n^*}, (\tilde{s}^*, \{t_i^*\}_{i=1}^{n^*}, \tilde{R}^*)) = 1$ holds, and it outputs \perp if not. Otherwise \mathcal{A} outputs

$(I, \{(X_i, x_i)\}_{i \in [1, n] \setminus \{k\}}, (\tilde{s}^*, \{c_i^*\}_{i=1}^{n^*}, \tilde{R}^*))$ where $c_i^* = T[\tilde{R}^*, X_i, t_i^*, m_i^*]$ and I is the index such that $h_I = T[\tilde{R}^*, Y, t_k^*, m_k^*]$.

\mathcal{B} obtains the following two sequences by rewinding \mathcal{A} according to the Bellare-Neven general forking Lemma [5].

$$\begin{aligned} & (I^{(1)}, \{(X_i^{(1)}, x_i^{(1)})\}_{i \in [1, n^{(1)}] \setminus \{k^{(1)}\}}, (\tilde{s}^{(1)}, \{c_i^{(1)}\}_{i=1}^{n^{(1)}}, \tilde{R}^{(1)})) \\ & (I^{(2)}, \{(X_i^{(2)}, x_i^{(2)})\}_{i \in [1, n^{(2)}] \setminus \{k^{(2)}\}}, (\tilde{s}^{(2)}, \{c_i^{(2)}\}_{i=1}^{n^{(2)}}, \tilde{R}^{(2)})) \\ & \text{s.t. } \tilde{R}^{(1)} = \tilde{R}^{(2)} \wedge I^{(1)} = I^{(2)} \wedge c_{k^{(1)}}^{(1)} \neq c_{k^{(2)}}^{(2)} \end{aligned}$$

Since the above sequences satisfy the verification formula, we have

$$\tilde{R}^{(1)} = g^{\tilde{s}^{(1)}} \prod_{i=1}^{n^{(1)}} X_i^{(1)-c_i^{(1)}} \quad \text{and} \quad \tilde{R}^{(2)} = g^{\tilde{s}^{(2)}} \prod_{i=1}^{n^{(2)}} X_i^{(2)-c_i^{(2)}}.$$

By $\tilde{R}^{(1)} = \tilde{R}^{(2)}$, dividing the above two equations gives

$$Y^{c_1^{(1)} - c_1^{(2)}} = g^{\tilde{s}^{(1)} - \tilde{s}^{(2)}} \prod_{i \in [1, n^{(1)}] \setminus \{k^{(1)}\}} X_i^{(1)-c_i^{(1)}} \prod_{i \in [1, n^{(2)}] \setminus \{k^{(2)}\}} X_i^{(2)c_i^{(2)}}.$$

Therefore, finally \mathcal{B} outputs the following as the solution to the instance Y of the DL problem.

$$y \leftarrow \frac{\tilde{s}^{(1)} - \tilde{s}^{(2)} - \sum_{i \in [1, n^{(1)}] \setminus \{k^{(1)}\}} x_i^{(1)} c_i^{(1)} + \sum_{i \in [1, n^{(2)}] \setminus \{k^{(2)}\}} x_i^{(2)} c_i^{(2)}}{c_{k^{(1)}}^{(1)} - c_{k^{(2)}}^{(2)}} \pmod{q} \quad (2)$$

\mathcal{B} succeeds in outputting y if and only if it succeeds in forking \mathcal{A} . Let frk be the probability of succeeding in forking \mathcal{A} , and then the success probability ε' of \mathcal{B} is equal to frk . Let acc be the probability that \mathcal{A} outputs the sequence. We have

$$acc = \Pr[bad \neq true \wedge \mathcal{F} \text{ succeed}] \geq \Pr[\mathcal{F} \text{ succeed}] - \Pr[bad = true].$$

The event $bad = true$ happens when \mathcal{A} cannot set $H(\tilde{R}', X, t', m') \leftarrow c'$ in the random oracle table due to a predefined $H(\tilde{R}', X, t', m')$. \mathcal{F} can cause this event by guessing t' which is the part of a signature that the signing oracle returns. How \mathcal{F} maximizes the probability of causing this event is as follows. Firstly, for a hash query $Q_k = (\tilde{R}', X, t', m')$, \mathcal{F} fixes \tilde{R}', X , and m' and queries q_H times with t' different from each other. After that, \mathcal{F} makes q_S signing queries by using \tilde{R}', X , and m' . Let Hit_f be the event that $bad = true$ is happened in the f th time signing query. Note that one new row in the random oracle table is created every time \mathcal{F} makes signing query. Then $\Pr[bad = true]$ is bounded as follows.

$$\begin{aligned} \Pr[bad = true] &= \Pr[\text{Hit}_1 \vee \text{Hit}_2 \vee \dots \vee \text{Hit}_{q_S}] \\ &\leq \Pr[\text{Hit}_1] + \Pr[\text{Hit}_2] + \dots + \Pr[\text{Hit}_{q_S}] \\ &\leq \frac{q_H}{2^\kappa} + \frac{q_H + 1}{2^\kappa} + \dots + \frac{q_H + q_S - 1}{2^\kappa} = \frac{q_S(2q_H + q_S - 1)}{2^{\kappa+1}}. \end{aligned}$$

Thus we obtain

$$acc \geq \varepsilon - \frac{q_S(2q_H + q_S - 1)}{2^{\kappa+1}}.$$

By the Bellare-Neven general forking lemma [5], we have

$$\begin{aligned} \varepsilon' = frk &\geq acc \left(\frac{acc}{q_H + 1} - \frac{1}{q} \right) \geq \frac{acc^2}{q_H + 1} - \frac{1}{q} \\ &= \frac{1}{q_H + 1} \left(\varepsilon - \frac{q_S(2q_H + q_S - 1)}{2^{\kappa+1}} \right)^2 - \frac{1}{q} \\ &\geq \frac{\varepsilon^2}{q_H + 1} - \frac{2q_S(2q_H + q_S - 1)}{(q_H + 1)2^{\kappa+1}} - \frac{1}{q}. \end{aligned}$$

The running time t' of \mathcal{B} is twice as the running time t of \mathcal{F} plus $O(q_H + q_S + 1)$ time needed to answer hash queries plus $2q_S t_{exp}$ time because each signing query involves two exponentiation in G . \square

The Restriction on the Public Keys. For PCAS, all signers' public keys need to be distinct from each other. The reason is as follows: in the security proof, if several cosigners are having the same public key as an honest signer, the denominator of Eq. (2) is $\sum_{i \in [1, n^{(1)}] \text{ s.t. } Y = X_i^{(1)}} c_i^{(1)} - \sum_{i \in [1, n^{(2)}] \text{ s.t. } Y = X_i^{(2)}} c_i^{(2)}$. In this situation, we cannot know whether this denominator is not equal to 0 only from condition $c_1^{(1)} \neq c_1^{(2)}$.

On the KOSK Model and Its Implementation. We used the KOSK model in the security proof for simplicity and necessity. In practice, a possible way to implement the KOSK model is to use a proof-of-possession (PoP). The security of this implementation depends on the security of PoP. For example, we may consider the case of using the Schnorr signature [28] as PoP. More specifically, if a signer is required to include the PoP signed by his secret key in his public key, then, in the security game, a forger outputs the PoP signed by the secret keys behind the cosigners' public keys, not the secret keys. Since the set of the cosigners' secret keys is necessary for the proof of Theorem 1, proving the security of PCAS with this PoP is not trivial. A possible way to prove such a scheme secure is applying Bagherzandi-Cheon-Jarecki generalized forking lemma [2].

4 Performance Comparison among Aggregate Signature Scheme and Related Schemes

In this section, we compare the proposed aggregate signature scheme with pre-communication PCAS with the Zhao's aggregate signature scheme [32] and the Bellare-Neven interactive aggregate signature scheme BN-IAS [5]. These schemes are constructed based on the Schnorr signature scheme [28]. Note that we suppose the situation that these schemes are used for the same purpose of compressing signatures on different messages into a compact signature. Then, we focus on

Table 2. Performance Comparison among Aggregate Signature Scheme and Related Schemes

Scheme	BN-IAS [5]	Zhao [32]	PCAS
Type	IAS	standard AS	AS with PreCom
No sharing Messages	No	Yes	Yes
Communication Complexity	$(M + l_0 + G + Z_q) \times n(n-1)$	$2n Z_q $	$n(2 G + Z_q + \kappa)$
Signature Size	$ Z_q + G $	$ Z_q + n G $	$ Z_q + G + n\kappa$
Assumption	DL	NMDL	DL
Key Setup	plain PK	plain PK	KOSK
Restriction in Aggregation	No Restriction	Distinct (pk, m)	Distinct pk
Withdrawal	No	Yes	No

* The row 1 and 2 indicate pairing-free aggregate signature (AS) scheme and related schemes. In row 4 and 5, $|M|$, $|Z_q|$ and $|G|$ indicate the size of a element in $|M|$, Z_q , and G . Also, n denotes the number of signers, and l_0 and κ are specific parameters on each scheme. Especially, the bit-length of κ is as same as the security parameter in general. The row 6 and 7 show that the assumption and the key-setup model (cf., the notion of models in Sect. 1.1) in which each scheme is proved secure, where DL and NMDL indicate the discrete logarithm assumption and non-malleable DL assumption [32]. The row 8 shows the restriction of all signers' public keys and/or messages to be accepted in the verification. The final row shows the possibility of a continuation of the procedure in the case where signers disappear before the aggregation phase.

sharing messages, communication complexity, withdrawal, the key setup model, assumptions, and the size of the aggregate signature for the comparison. Table 2 summarizes this comparison.

Necessity of Sharing Messages. On the above purpose, BN-IAS is the multi-signature scheme used as an aggregate signature scheme. More detail, this scheme generates a combined signature on different messages by seeing a set of signers' messages as one message. Then, all players need to execute the interactive protocols in Stage II in Sect. 1.1.

PCAS and Zhao's scheme need not share messages between all signers. Especially, PCAS requires interactive protocol as PreCom, however, this interaction is executed in Stage I. Thus, it achieves no sharing messages. Zhao's scheme has the standard construction of the aggregate signature, so it has no interaction protocol.

Communication Complexity. Firstly, let n be the number of signers, and $|M|$ be the size of a message M . Moreover, we consider the communication complexity including the cost of one-shot communication from signers to an aggregator for submitting a signature.

For BN-IAS, all signers need to share messages before the signing phase. Also, the signing protocol requires three-round communication between every two signers. Therefore, this scheme requires $n(n-1)/2$ channels, and the total communication complexity per channel is $2|M| + 2l_0 + 2|G| + 2|Z_q|$ where l_0 is the bit-length of the range of the hash function to produce the commitment to commitment element on the Schnorr signature scheme. The total communication complexity in aggregation protocol is $n(n-1)(|M| + l_0 + |G| + |Z_q|)$.

PCAS needs one bidirectional communication to share helper information between signers and the aggregator in the pre-communication phase. Hence this scheme requires n channels, and the total communication complexity per channel is $2|G| + |Z_q| + \kappa$. Then the total communication complexity is $n(2|G| + |Z_q| + \kappa)$.

Zhao’s scheme has no interactive protocol, so there are only communications for submitting signatures. Then this scheme requires n channels, and the total communication complexity is $2n|Z_q|$.

The Size of an Aggregate Signature. The size of a signature of BN-IAS is $|Z_q| + |G|$, and hence it is independent of n . The signature size of PCAS is $|Z_q| + |G| + n\kappa$ and the signature size of Zhao’s scheme is $|Z_q| + n|G|$. Notably, both sizes depend on n . We can pick κ to be equal to the security parameter λ because we should consider only target collisions for the hash function in the proof of Theorem 1. Also, We can have $\kappa \leq |G|$ because the order of G is about $2^{2\lambda}$ in general. Therefore PCAS can achieve a smaller signature size than Zhao’s scheme.

Key Setup Model, Assumptions, and Acceptable Condition. We proved PCAS secure under the DL assumption in the KOSK model and the random oracle model. This scheme needs to use a PoP to prove the correct generation of a public key in practical due to the KOSK model. BN-IAS was proved secure under the DL assumption in the random oracle model and the plain PK model. Zhao’s scheme was proved secure under the NMDL assumption in the random oracle model and the plain PK model. Zhao also showed the hardness of the NMDL problem in the generic group model [24] and the random oracle model.

BN-IAS has no restrictions on public keys and messages, namely, we may include duplicate public keys and messages in aggregation. In Zhao’s scheme, an aggregate signature is not accepted when all signers’ pairs of a public key and a message are not distinct to each other. In PCAS, the aggregate signature can be accepted at least every public keys should be distinct.

Withdrawal. BN-IAS must halt and restart a signing protocol when some signers disappear in the signing phase. Also, PCAS must halt and restart a signing protocol when some signers fail to participate in an aggregation phase. On the other hand, Zhao’s scheme can continue the process in such a situation because each signer generates a signature without any communication.

5 How to Avoid Drijvers et al.’s Impossibility

Drijvers et al. showed that several multi-signature schemes claimed to be secure are in fact insecure by both concrete attacks and meta-reductions demonstrating the impossibility of proving their security [10]. Because all of such schemes are based on the Schnorr identification or extensions thereof, one may wonder if their attacks or meta-reduction arguments are applicable or not to our schemes and, if not, may want to know the reason for the inapplicability. In particular, the CoSi scheme [29], which is one of the targets of these attacks and meta-reductions is

quite similar to our PCAS scheme and is essentially the PCAS scheme without t . Therefore, it seems to be reasonable that our PCAS scheme is a target of (a natural extension of) these attacks and meta-reductions.

We discover that the meta-reductions of Drijvers et al. apply to any multi-signature or aggregate signature scheme based on sigma protocol, e.g., Schnorr identification scheme, with the following properties: (1) the challenge of each signer depends solely on the message to be signed and the combined commitment, and (2) a malicious aggregator can control the values of the combined commitment. Exploiting the above properties, a meta-reduction algorithm can somehow rewind any reduction algorithm which simulates an honest signer and *force* the honest signer to use different challenges of the malicious aggregator's choice. Thus, if the reduction simulates the honest signer perfectly, the meta-reduction algorithm obtains two distinct individual signatures based on two different combined commitments but the same fixed individual commitment from the honest signer. The special soundness property of sigma protocol enables the meta-reduction algorithm to break the hardness of underlying computational problem and thus the impossibility holds.

For the PCAS scheme, this structure is eliminated by introducing a random value of t . The point is that t is chosen by the honest signer *after an aggregator broadcasting the combined commitment*. Due to this t , a malicious aggregator cannot force the honest signer to use the challenge of the malicious aggregator's choice.

In the full version of this paper, we elaborate more on the above-outlined weakness of the PCAS scheme without t from the viewpoints of both concrete attacks and meta-reduction arguments. Furthermore, we explain how this weakness was overcome by the introduction of the random value t .

6 Conclusion

In this paper, we propose a new paradigm *pre-communication* and the PCAS scheme which is constructed based on this new paradigm and proved secure under the standard DL assumption and the KOSK model. By presenting the concrete rogue-key attack, we state that the KOSK model is essential for PCAS. Moreover, we explain that we avoided Drijvers et al.'s attacks and impossibility results.

In practice, PCAS need proof-of-possession (PoP) because of their security in the KOSK model. Therefore to analyze the security of schemes equipped with a concrete PoP is an important open question.

Acknowledgments. This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO). This work was supported by JST CREST Grant Number JPMJCR19F6, Japan. This work was supported by JSPS KAKENHI Grant Numbers JP18H01438, JP18H03238, JP18H05289, JP18K11292, JP18K11293, JP18K18055, JP19H01109. We are grateful to an anonymous reviewer, who pointed out subtleties in the security definition of aggregate signatures with pre-communication.

References

1. Ahn, J.H., Green, M., Hohenberger, S.: Synchronized aggregate signatures: new definitions, constructions and applications. In: CCS 2010, pp. 473–484 (2010)
2. Bagherzandi, A., Cheon, J.H., Jarecki, S.: Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In: CCS 2008, pp. 449–458 (2008)
3. El Bansarkhani, R., Mohamed, M.S.E., Petzoldt, A.: MQSAS - a multivariate sequential aggregate signature scheme. In: Bishop, M., Nascimento, A.C.A. (eds.) ISC 2016. LNCS, vol. 9866, pp. 426–439. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45871-7_25
4. Bellare, M., Namprempre, C., Neven, G.: Unrestricted aggregate signatures. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 411–422. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73420-8_37
5. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: CCS 2006, pp. 390–399 (2006)
6. Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36288-6_3
7. Boldyreva, A., Gentry, C., O’Neill, A., Yum, D.H.: Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In: CCS 2007, pp. 276–285 (2007)
8. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-39200-9_26
9. Brogle, K., Goldberg, S., Reyzin, L.: Sequential aggregate signatures with lazy verification from trapdoor permutations. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 644–662. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_39
10. Drijvers, M., et al.: On the security of two-round multi-signatures. In: IEEE S&P 2019, pp. 1084–1101 (2019)
11. Drijvers, M., Gorbunov, S., Neven, G., Wee, H.: Pixel: multi-signatures for consensus. In: IACR Cryptology ePrint Archive 2019, p. 514 (2019)
12. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: CRYPTO 1986, pp. 186–194 (1986)
13. Fischlin, M., Lehmann, A., Schröder, D.: History-free sequential aggregate signatures. In: Visconti, I., De Prisco, R. (eds.) SCN 2012. LNCS, vol. 7485, pp. 113–130. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32928-9_7
14. Gentry, C., Ramzan, Z.: Identity-based aggregate signatures. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 257–273. Springer, Heidelberg (2006). https://doi.org/10.1007/11745853_17
15. Guillevic, A.: A short-list of pairing-friendly curves resistant to special TNFS at the 128-bit security level. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. LNCS, vol. 12111, pp. 535–564. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45388-6_19
16. Hohenberger, S., Koppula, V., Waters, B.: Universal signature aggregators. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 3–34. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_1

17. Hohenberger, S., Sahai, A., Waters, B.: Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 494–512. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_27
18. Hohenberger, S., Waters, B.: Synchronized aggregate signatures from the RSA assumption. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10821, pp. 197–229. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78375-8_7
19. Kim, T., Barbulescu, R.: Extended tower number field sieve: a new complexity for the medium prime case. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9814, pp. 543–571. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_20
20. Lee, K., Lee, D.H., Yung, M.: Sequential aggregate signatures made shorter. In: Jacobson, M., Locasto, M., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 202–217. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38980-1_13
21. Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 465–485. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_28
22. Lysyanskaya, A., Micali, S., Reyzin, L., Shacham, H.: Sequential aggregate signatures from trapdoor permutations. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 74–90. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_5
23. Ma, D., Tsudik, G.: Extended abstract: forward-secure sequential aggregate authentication. In: S&P 2007, pp. 86–91 (2007)
24. Maurer, U.M.: Abstract models of computation in cryptography. In: IMA 2005, pp. 1–12 (2005)
25. Micali, S., Ohta, K., Reyzin, L.: Accountable-subgroup multisignatures: extended abstract. In: CCS 2001, pp. 245–254 (2001)
26. Neven, G.: Efficient sequential aggregate signed data. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 52–69. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_4
27. Ristenpart, T., Yilek, S.: The power of proofs-of-possession: securing multiparty signatures against rogue-key attacks. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 228–245. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72540-4_13
28. Schnorr, C.: Efficient identification and signatures for smart cards. In: CRYPTO 1989, pp. 239–252 (1989)
29. Syta, E., et al.: Keeping authorities “honest or bust” with decentralized witness cosigning. In: S&P 2016, pp. 526–545 (2016)
30. Xu, J., Zhang, Z., Feng, D.: ID-based aggregate signatures from bilinear pairings. In: Desmedt, Y.G., Wang, H., Mu, Y., Li, Y. (eds.) CANS 2005. LNCS, vol. 3810, pp. 110–119. Springer, Heidelberg (2005). https://doi.org/10.1007/11599371_10
31. Yao, A.C., Zhao, Y.: Online/offline signatures for low-power devices. *IEEE Trans. Inf. Forensics Secur.* 8(2), 283–294 (2013)
32. Zhao, Y.: Practical aggregate signature from general elliptic curves, and applications to blockchain. In: AsiaCCS, 2019, pp. 529–538 (2019)