Hindawi
Security and Communication Networks
Volume 2019, Article ID 3424890, 14 pages
https://doi.org/10.1155/2019/3424890

Research Article

A Short Server-Aided Certificateless Aggregate Multisignature Scheme in the Standard Model

Viet Cuong Trinh

Hong Duc University, Thanh Hoa, Vietnam

Correspondence should be addressed to Viet Cuong Trinh; trinhvietcuong@hdu.edu.vn

Received 5 November 2018; Accepted 20 February 2019; Published 18 March 2019

Academic Editor: Bela Genge

Aggregate signature scheme allows each signer to sign a different message and then all those signatures are aggregated into a single short signature. In contrast, multisignature scheme allows multisigners to jointly sign only one message. Aggregate multisignature scheme is a combination of both aforementioned signature schemes, where signers can choose to generate either a multisignature or an aggregate signature. This combination scheme has many concrete application scenarios such as Bitcoin blockchain, Healthcare, Multicast Acknowledgment Aggregation, and so on. On the other hand, to deal with the problems of expensive certificates in certified public key cryptography and key escrow in identity-based cryptography, the notion of *certificateless* public key cryptography has been introduced by Riyami and Paterson at Asiacrypt'03. In this paper, we propose the first certificateless aggregate multisignature scheme that achieves the constant-size of signature and is secure in the standard model under a generalization of the Diffie-Hellman exponent assumption. In our scheme, however, the signature is generated with the help of the authority.

1. Introduction

Certificateless Cryptography. In public key cryptography, a public key is just a random number, to certify that a public key belongs to a specific user we need to provide for this public key a certificate. Public Key Infrastructure (PKI) was introduced for such purpose; however implementing such PKI in the real world requires using a lot of resources since we need to provide, maintain, and revoke a large amount of certificates. Shamir [1] came up with an idea that if a public key is an identity of a specific user (official email address, for example), there is no need to certify this public key; this leads to the introducing of the notion of identity-based cryptography. However, since each identity is mapped to an arbitrary given fixed number, the public key is an arbitrary given fixed number. In traditional public key cryptosystems such as RSA or ElGamal, given such public key, user cannot generate the corresponding secret key. In all identity-based cryptosystems, user's secret key is in fact generated by a private key generator (PKG) who knows the master key of the system; this leads to the fact that PKG knows all secret keys of users in the system; this problem is so-called key escrow problem. To deal with the problems of providing, maintaining, and revoking a large amount of certificates in traditional PKI and key escrow in identity-based cryptography, the notion of *certificateless* public key cryptography has been introduced by Riyami and Paterson at Asiacrypt'03 [2]. In a certificateless cryptosystem, the user's full secret key includes two parts: the first part is a partial secret key generated by PKG from master key and user's identity; the second part is a secret value chosen by user himself/herself. Due to the fact that user's public key is still associated with user's identity and the user's full secret key includes a secret value chosen by user himself/herself, there is no need to use certificate to certify user's public key and PKG does not know the user's full secret key.

Certificateless signature scheme was first introduced in [2] and then has been deeply studied in [6–15] to name a few. Regarding the certificateless signature schemes secured in the standard model, there are currently two approaches to construct. First, using the Waters' hash function [8, 9] or Yum-Lee generic transformations [6, 7], the advantage of this approach is that the resulting schemes can be secure under standard assumptions. However, due to using Waters' hash function or Yum-Lee generic transformations, these schemes suffer relatively large public parameters and heavy computing

Table 1: n' is the size of the aggregating set of signers; $n \geq n'$ is fixed at the setup. P denotes pairing computation, exp denotes exponentiation, and H denotes hash operation. Note that only our scheme is described in type 3 Pairings.

	Signature	Secret-key	Public-key	Signing	Verifying
[3]	$1\|\mathbb{G}\|$	$1\|\mathbb{G}\| + 1\|p\|$	$2\|\mathbb{G}\|$	$n'exp + n'H + 3n'P$	$3P + 1H$
[4]	$1\|\mathbb{G}\|+ 1\|p\|$	$2\|p\|$	$2\|\mathbb{G}\|$	$2n'exp + 2n'H$	$(n'+ 3)exp + 2H$
[5]	$(n'+ 1)\|\mathbb{G}\|$	$1\|\mathbb{G}\|+ 1\|p\|$	$2\|\mathbb{G}\|$	$4n'exp + 3n'H$	$2n'exp + 3P\ (3n'+ 1)H$
Ours	$3\|\mathbb{G}\|+ 1\|\widetilde{\mathbb{G}}\|$	$3\|\mathbb{G}\|+ 1\|p\|$	$3\|\mathbb{G}\|+ (n + 3)\|\widetilde{\mathbb{G}}\|$	$(2n'+ 9)exp + (2n'+ 1)H$	$(n'+ 1)exp + 3P+(2n'+ 1)H$

time. Second, using direct approach [14, 15], which leads to quite efficient resulted schemes (constant-size of both public parameters and signature, as well as efficient computing time); however these schemes are only secure under strong assumptions (generalization of the Diffie-Hellman exponent assumption, GDDHE assumption), although GDDHE assumption introduced by Boneh and Boyen at Eurocrypt'05 [16] now has been accepted widely by researchers [17–22].

Aggregate Signature. Aggregate signature was first introduced by Boneh *et al.* at Eurocrypt'03 [23]. In this scheme, each signer in an aggregating set signs a different message and then all those signatures are aggregated into a single short signature, which is called aggregate signature. As shown in [5, 23–25], aggregate signature can be applied well to several practical applications such as Bitcoin blockchain, Secure BGP protocol (SBGP) [26], Healthcare, and so on. Certificateless aggregate signature was first proposed in [27], since then it has been studied in numerous of papers such as [5, 28–35], to name a few. However, all of these schemes either are insecure [29, 30, 35] or suffer a drawback that the signature size is linear in the number of signers in the aggregating set [5, 28, 31–34]. Moreover, all of these schemes need to use random oracle to prove the security. Very recently, the authors in [5] proposed a new certificateless aggregate scheme with short public parameters and achieving the highest level of security according to the classification given by Huang *el al.* [13] under a standard assumption. However, their scheme still suffers two drawbacks: the signature size is linear in the number of signers in the aggregating set and their scheme needs to use random oracle to prove the security. To our knowledge, the task of designing a certificateless aggregate signature scheme with short signature size and secured in the standard model is still open.

Multisignature. In contrast to aggregate signature, multisignature scheme allows multisigners to jointly sign only one message. This scheme was first introduced in [36], and then it has been the topic of many other works such as [25, 37–43], to name a few. At ACM CCS'01, Micali *el al.* [37] first formalized the security model for a multisignature scheme; they also proposed a multisignature scheme based on Schnorr-signature secured in this model. In [25], the authors defined a multisignature scheme with public key aggregation, for which all public keys of signers in the aggregating set are aggregated into a short aggregate public key through a new additional Key Aggregation algorithm. The advantage of this scheme is that the verifier can only take a constant-size of input (multisignature and aggregate public key) to verify the multisignature, which were showed in [25, 43] that this type of scheme can be applied well to the Bitcoin blockchain application. However, the downside of this scheme is that each aggregating set of signers needs to publish in advance its aggregate public key. Very recently, Boneh *el al.* [43] proposed a new such compact multisignature scheme with public key aggregation; their scheme however is secure in the random oracle model. Regarding the multisignature scheme in the certificateless setting, several schemes were proposed [3, 4, 44, 45]. The authors in [3] addressed the problem of fast verification but they did not give a formal security proof of their schemes. The authors in [4] proposed a certificateless multisignature scheme without using Pairings; they also gave a formal security proof for their scheme under standard assumption. All of these schemes achieve constant-size of signature; however they did not address the problem of public key aggregation and still need to use random oracle to prove the security.

1.1. Our Contribution and Organization of the Paper. In this paper, we extend the work in [14] to consider the combination of an aggregate signature scheme and a multisignature scheme, and in the context of certificateless. More precisely, in our server-aided certificateless aggregate multisignature scheme (SCL-AMS for short), an aggregating set of signers can choose to generate either an aggregate signature or a multisignature with the help of private key generator (PKG); for simplicity such signature is called an aggregate multisignature. If the aggregating set of signers contains only one signer, the resulting signature is a usual signature and the signer does not need the help from PKG.

More precisely, our SCL-AMS scheme has following properties:

(i) the first certificateless aggregate multisignature scheme;

(ii) the signature which contains four elements in all cases;

(iii) being secure against strong Type I and strong Type II adversaries (according to the classification given by Huang *el al.* [13]) in the standard model under GDDHE assumptions;

(iv) server-aided scheme;

(v) support public key aggregation;

(vi) public key size, signing time, and verifying time which depend on the maximum number of signers for one aggregating set, which is fixed at the setup. More details can be found in the Tables 1 and 2.

Table 2: ROM denotes random oracle model, SM denotes standard model, and PKA denotes supporting public key aggregation.

	Multi/Aggre	non-interactive	Security	PKA	Server-Aid
[3]	*yes/no*	*no*	not known	*no*	*no*
[4]	*yes/no*	*no*	ROM+Strong Type I,II+ECDLP assumption	*no* *no*	*no* *no*
[5]	*no/yes*	*yes*	ROM+Super Type I,II +CDH problem	*no* *no*	*no* *no*
Ours	*yes/yes*	*no*	SM+Strong Type I,II+GDDHE assumption	*yes*	*yes*

Although our SCL-AMS scheme requires server-aid and is secure in a strong assumption (GDDHE assumption), our scheme might be considered as a partial answer for the question of designing a certificateless aggregate signature scheme which has short signature size and is secure in the standard model. In addition, our scheme is not only certificateless aggregate signature but also certificateless multisignature with public key aggregation. This type of scheme, recently showed in [25, 43], can be applied well to the Bitcoin blockchain application. We also note that the server-aided model is not new; this model has been used in many other contexts. Regarding the context of signature, we can find several other signature schemes in the literature which use the server-aid model [46–49]. Regarding the GDDHE assumption, it is introduced by Boneh and Boyen at Eurocrypt'05 [16] and now has been accepted and used widely by researchers with a lot of papers based on it [50–52].

We give in Tables 1 and 2 the detailed comparison between our scheme and some latest relevant schemes in the literature.

In the Table 2, regarding *noninteractive*, to generate a multisignature, in [3] the first signer in the aggregating set signs the message and then sends the resulting signature to the second signer, the second signer based on this signature and his/her secret key to generate a new signature and sends this new signature to the third signer, and this process continues to the last signer. That means it needs n' rounds to generate a multisignature, n' is the size of the aggregating set. In [4], each signer needs to broadcast a value to $(n' - 1)$ other signers and receives $(n' - 1)$ values from $(n' - 1)$ other signers before generating multisignature as in usual *noninteractive* scheme [40–42]. In our scheme, in the first round, PKG relies on the description of the aggregating set and messages to compute four elements and sends them to n' signers, all signers then based on these values to generate aggregate multisignature as in usual *noninteractive* scheme [40–42].

Regarding security model, we only consider the *selective* model where adversary has to declare the identity of target user at the beginning of security game; the challenger thus does not need to guess the identity of target user at the beginning of security game. In scheme [4, 5], the authors considered a more practical model, the *adaptive* model where adversary does not have to declare the identity of target user at the beginning of security game; however in their security proof, the challenger needs to guess in advance the identity of the target user at the beginning of security game; their schemes therefore lose a factor in the reduction to the corresponding assumptions. We also note that all of these schemes need to use random oracle to prove the security.

Paper organization. In the Preliminaries section, we present definition and security model for our SCL-AMS scheme, and then some useful tools for our construction. We next describe our SCL-AMS scheme and its security analysis in Section 3. Section 4 is devoted for the conclusion.

2. Preliminaries

In this section, based on security models in [4, 5, 13] we first define the security model for our SCL-AMS scheme; we then give some useful tools that we need to construct our scheme such as Bilinear Groups, Target Collision Resistant Hash Function, and Assumptions.

2.1. Server-Aided Certificateless Aggregate Multisignature Scheme

2.1.1. Definition. There are three entities involved in our server-aided certificateless aggregate multisignature scheme:

(1) a designated authority acting as a Private Key Generator PKG;

(2) user (or signer) who combines with PKG and all other users in an aggregating set to generate an aggregate multisignature (either an aggregate signature or a multisignature). Note that if the aggregating set of users contains just one user, user relies solely on his/her full secret key (without the help of PKG) to generate the aggregate multisignature. In this case, the aggregate multisignature is exactly the usual user's signature;

(3) verifier who is able to check the validity of an aggregate multisignature.

We also note that, in a certificateless system, a user's full secret key SK_{ID} includes two parts: the first part is a partial secret key P_{ID} generated by the PKG from master key and user's identity; the second part is a secret value x_{ID} chosen by user himself/herself.

More formally, our SCL-AMS scheme consists of the following nine probabilistic algorithms.

(i) Setup: it takes as input a security parameter λ; this algorithm generates system parameters param

implicitly including the maximum number of users in an aggregating set n and a master secret key msk.

(ii) Partial-Secret-Key-Generation: it takes as input param, msk, and a user's identity ID; this algorithm generates the partial secret key P_{ID}.

(iii) User-Secret-Value-Generation: it takes as input the security parameter λ and a user's identity ID; this algorithm generates the user's secret value x_{ID}.

(iv) Public-Key-Generation: it takes as input x_{ID}; this algorithm generates the user's public key PK_{ID}.

(v) Full-Secret-Key-Generation: it takes as input P_{ID} and x_{ID}; the algorithm generates the user's full secret key SK_{ID}.

(vi) Aggregation: suppose that the aggregating set of users is $S, |S| \leq n$. There are two cases: if S contains just one user with identity ID, the algorithm takes as input a message M, a user's full secret key SK_{ID}, and param; it generates a signature σ; note that in this case the algorithm works as a usual signing algorithm.

If S contains more than one user, the algorithm now is run by PKG and all users in S, takes as input messages $\{M_i\}_{i\in S}$ and the description of S, and outputs an aggregate multisignature σ on messages $\{M_i\}_{i\in S}$. Note that the signature is called multisignature as usual if all $\{M_i\}_{i\in S}$ are the same.

(vii) Aggregation Verification: it takes as input an aggregate multisignature σ, the description of aggregating set of users S, and messages $\{M_i\}_{i\in S}$ and outputs either 1 indicating that σ is a valid signature or 0 indicating that σ is not a valid signature.

2.1.2. Adversary's Oracles. We have two types of adversary in our SCL-AMS scheme, the first one represents a third party adversary (so-called Type I adversary) and the second one represents the malicious PKG (so-called Type II adversary), and those adversary can ask the following oracles:

(i) PartialSK(ID): when adversary requests a partial secret key corresponding to identity ID, he/she should get the user's partial secret key P_{ID}.

(ii) UserSecret(ID): when adversary requests a user's secret value corresponding to identity ID, he/she should get the user's secret value x_{ID}. Note that if adversary asks both PartialSK and UserSecret oracles on input ID, he/she will get the full secret key of user associated with identity ID.

(iii) RequestPK(ID): when adversary requests a public key corresponding to identity ID, he/she should get the user's public key PK_{ID}.

(iv) ReplacePK($ID, x_{ID'}, \mathsf{PK}_{ID'}$): adversary can choose ID and replace the original public key PK_{ID} to a new public key PK'_{ID} by making a query $(ID, x_{ID'}, \mathsf{PK}_{ID'})$ to challenger.

(v) RequestSig($\{M_i, ID_i\}_{i\in S}$): when adversary requests an aggregate multisignature on messages $\{M_i\}_{i\in S}$ corresponding to users with identities $\{ID_i\}_{i\in S}$, he/she should get a valid aggregate multisignature σ corresponding to $\{M_i, ID_i\}_{i\in S}$. In addition, in case $|S| > 1$, since in the Aggregation algorithm the aggregate multisignature is generated by PKG and all users in S via public channel; adversary may also get the output of PKG as well as the outputs of some users in S.

Informally, a third party adversary (an outsider attacker) cannot know the master key; however he/she can replace any public key with a secret value of his/her choice. To exclude the trivial attack, he/she obviously cannot know the partial secret key of the target user. Regarding malicious PKG (an insider attacker), this adversary knows the master key; however to exclude the trivial attack he/she cannot know the secret value or replace the public key of the target user.

On the other hand, according to the classification given by Huang *el al.* [13], there are three levels of adversary:

(i) *normal* adversary: this adversary cannot ask RequestSig($\{M_i, ID_i\}_{i\in S}$) oracle if at least one public key of a user in S has been replaced;

(ii) *strong* adversary: this adversary still can ask RequestSig($\{M_i, ID_i\}_{i\in S}$) oracle even if all public keys of users in S have been replaced;

(iii) *super* adversary: this adversary is not required to supply the secret value $x_{ID'}$ when asking ReplacePK oracle, and he/she still can ask RequestSig($\{M_i, ID_i\}_{i\in S}$) oracle even if all public keys of users in S have been replaced;

In this paper, we consider the *strong* adversary for both type I and type II.

2.1.3. Security Model. We define two selective security games corresponding to two types of adversary, Game 1 corresponds to third party adversary (denote $\mathscr{A}_1$-strong Type I adversary), and Game 2 corresponds to malicious PKG (denote $\mathscr{A}_2$-strong Type II adversary).

Our SCL-AMS scheme is secure if it resists both adversaries $\mathscr{A}_1$ and $\mathscr{A}_2$.

Game 1. The first game is between $\mathscr{A}_1$ and the challenger $\mathscr{C}$.

Initialization: $\mathscr{C}$ runs the Setup algorithm to generate a master secret key msk and public system parameters param. $\mathscr{C}$ keeps msk secret and gives param to $\mathscr{A}_1$. $\mathscr{A}_1$ declares the target user by sending the target identity ID_ℓ to $\mathscr{C}$.

Queries: $\mathscr{A}_1$ may adaptively ask the following oracles with $\mathscr{C}$: PartialSK(ID), UserSecret(ID), RequestPK(ID), ReplacePK($ID, x_{ID'}, \mathsf{PK}_{ID'}$), and RequestSig($\{M_i, ID_i\}_{i\in S}$). Note that $|S| \leq n$.

Output: Finally, $\mathscr{A}_1$ outputs $(\{M_i^*, ID_i\}_{i\in S^*}, \sigma^*)$, where $|S^*| \leq n, \ell \in S^*$, and σ^* is an aggregate multisignature on messages $\{M_i^*\}_{i\in S^*}$. We say that $\mathscr{A}_1$ wins the game if the following conditions are verified:

(1) PartialSK(ID_ℓ) and RequestSig($\{M_i^*, ID_i\}_{i\in S^*}$) have never been queried.

(2) Aggragation Verification which outputs σ^* is a valid aggregate multisignature on messages $\{M_i^*\}_{i\in S^*}$ under public keys corresponding to $\{ID_i\}_{i\in S^*}$.

Denote $Succ_{\mathcal{A}_1}$ as the success probability that adversary $\mathcal{A}_1$ wins the above game.

Game 2. The second game is between $\mathcal{A}_2$ and the challenger $\mathcal{C}$.

Initialization: $\mathcal{C}$ runs the Setup algorithm to generate a master secret key msk and public system parameters param. $\mathcal{C}$ gives both public param and msk to $\mathcal{A}_2$. $\mathcal{A}_2$ declares the target user by sending the target identity ID_ℓ to $\mathcal{C}$.

Queries: $\mathcal{A}_2$ may adaptively ask the following oracles with $\mathcal{C}$: UserSecret(ID), RequestPK(ID), ReplacePK($ID, x_{ID'}, \mathsf{PK}_{ID'}$), and RequestSig($\{M_i, ID_i\}_{i\in S}$). Note that $|S| \le n$.

Output: Finally, $\mathcal{A}_2$ outputs $(\{M_i^*, ID_i\}_{i\in S^*}, \sigma^*)$, where $|S^*| \le n, \ell \in S^*$, and σ^* is an aggregate multisignature on messages $\{M_i^*\}_{i\in S^*}$. We say that $\mathcal{A}_1$ wins the game if the following conditions are verified:

(1) UserSecret(ID_ℓ), ReplacePK($ID_\ell, x_{ID'}, \mathsf{PK}_{ID'}$), and RequestSig($\{M_i^*, ID_i\}_{i\in S^*}$) have never been queried.

(2) Aggragation Verification which outputs σ^* is a valid aggregate multisignature on messages $\{M_i^*\}_{i\in S^*}$ under public keys corresponding to $\{ID_i\}_{i\in S^*}$.

Denote $Succ_{\mathcal{A}_2}$ the success probability that adversary $\mathcal{A}_2$ wins the above game.

Definition 3. We say that our server-aided certificateless aggregate multisignature scheme is selectively existentially unforgeable against polynomially bounded adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ if the success probabilities $Succ_{\mathcal{A}_1}(\lambda)$ and $Succ_{\mathcal{A}_2}(\lambda)$ are negligible, where λ is the security parameter.

2.2. Bilinear Groups. In this section, we recall the definition of bilinear map and three types of Pairings. Let λ is the security parameter, $p > 2^\lambda$, and $\mathbb{G}$, $\widetilde{\mathbb{G}}$, and $\mathbb{G}_T$ are three finite multiplicative abelian groups of order p. Suppose that $g, \tilde{g}$ are generators of $\mathbb{G}$ and $\widetilde{\mathbb{G}}$, respectively. Denote $e : \mathbb{G} \times \widetilde{\mathbb{G}} \longrightarrow \mathbb{G}_T$ as an admissible asymmetric bilinear map, where for all $a, b \in \mathbb{Z}_p$

(1) $e(g^a, \tilde{g}^b) = e(g, \tilde{g})^{ab}$;

(2) for $g \neq 1_{\mathbb{G}}$ and $\tilde{g} \neq 1_{\widetilde{\mathbb{G}}}$, $e(g, \tilde{g}) \neq 1_{\mathbb{G}_T}$;

(3) $e(g, \tilde{g})$ is efficiently computable.

The tuple $(p, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T, g, \tilde{g}, e)$ is then called a bilinear map group system and is in

(1) Type 1 Pairings if $\mathbb{G} = \widetilde{\mathbb{G}}$

(2) Type 2 Pairings if $\mathbb{G} \neq \widetilde{\mathbb{G}}$ but there is an efficiently computable homomorphism $\phi : \widetilde{\mathbb{G}} \longrightarrow \mathbb{G}$

(3) Type 3 Pairings if $\mathbb{G} \neq \widetilde{\mathbb{G}}$ but there is no efficiently computable homomorphism between $\widetilde{\mathbb{G}}$ and $\mathbb{G}$

In [16], the authors introduced the generalization of the Diffie-Hellman exponent assumption in Type 3 Pairings bilinear map group system.

Assume $g_T = e(g, \tilde{g}) \in \mathbb{G}_T, s, n \in \mathbb{Z}_p^*$, and $P, Q, R \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ are three s-tuples of n-variate polynomials over $\mathbb{F}_p$. Thus, P, Q, and R are just three lists containing s multivariate polynomials each. We write $P = (p_1, p_2, \ldots, p_s)$, $Q = (q_1, q_2, \ldots, q_s)$ $R = (r_1, r_2, \ldots, r_s)$ and impose that $p_1 = q_1 = r_1 = 1$. For any function $h : \mathbb{F}_p \longrightarrow \Omega$ and vector $(x_1, \ldots, x_n) \in \mathbb{F}_p^n$, $h(P(x_1, \ldots, x_n))$ stands for $(h(p_1(x_1, \ldots, x_n)), \ldots, h(p_s(x_1, \ldots, x_n))) \in \Omega^s$. We use a similar notation for the s-tuples Q and R. Let $f \in \mathbb{F}_p[X_1, \ldots, X_n]$. It is said that f depends on (P, Q, R), which denotes $f \in \langle P, Q, R\rangle$, when there exists a linear decomposition

$$f = \sum_{1\le i,j\le s} a_{i,j} \cdot p_i \cdot q_j + \sum_{1\le i\le s} c_i \cdot r_i, \quad a_{i,j}, c_i \in \mathbb{Z}_p. \tag{1}$$

Let P, Q, R be as above and $f \in \mathbb{F}_p[X_1, \ldots, X_n]$. The (P, Q, R, f) − GDDHE problem is defined as follows.

Definition 4 ((P, Q, f) − GDHE [16]). Given the tuple $H(x_1, \ldots, x_n) = (g^{P(x_1,\ldots,x_n)}, \tilde{g}^{Q(x_1,\ldots,x_n)}, g_T^{R(x_1,\ldots,x_n)}) \in \mathbb{G}^s \times \widetilde{\mathbb{G}}^s \times \mathbb{G}_T^s$ compute $g_T^{f(x_1,\ldots,x_n)}$.

Definition 5 ((P, Q, R, f) − GDDHE [16]). Given $H(x_1, \ldots, x_n) = (g^{P(x_1,\ldots,x_n)}, \tilde{g}^{Q(x_1,\ldots,x_n)}, g_T^{R(x_1,\ldots,x_n)}) \in \mathbb{G}^s \times \widetilde{\mathbb{G}}^s \times \mathbb{G}_T^s$ as above and $T \in \mathbb{G}_T$ decide whether $T = g_T^{f(x_1,\ldots,x_n)}$.

2.3. Target Collision Resistant Hash Function

Definition 6. A target collision resistant hash function $\mathcal{H}$ guarantees that given a random element x which is from the valid domain of $\mathcal{H}$, a PPT adversary $\mathcal{A}$ cannot find $y \neq x$ such that $\mathcal{H}(x) = \mathcal{H}(y)$. We let $\mathbf{Adv}_{\mathcal{H},\mathcal{A}}^{TCR} = \Pr[(x, y) \longrightarrow \mathcal{A}(1^k) : \mathcal{H}(x) = \mathcal{H}(y), x \neq y, x, y \in DH]$ be the advantage of $\mathcal{A}$ in successfully finding collisions from a target collision resistant hash function $\mathcal{H}$, where DH is the valid input domain of $\mathcal{H}$; k is the security parameter. If a hash function is chosen from a target collision resistant hash function family, $\mathbf{Adv}_{\mathcal{H},\mathcal{A}}^{TCR}$ is negligible.

In this paper, for simplicity we call such target collision resistant hash function a hash function.

2.4. Assumptions. In this section, we recall one assumption and introduce a new assumption which are used to prove the security of our SCL-AMS scheme. Those assumptions are in the similar style of the PS assumption 1 in [52].

Definition 7.

Assumption 1. Let $(p, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T, g, \tilde{g}, e)$ be a bilinear group setting of type 3. Choose $u, v \xleftarrow{\$} \mathbb{Z}_p^*$, we define the oracle $\mathcal{O}$ on input $m \in \mathbb{Z}_p^*$ that chooses a random $r \in \mathbb{Z}_p^*$ and outputs the tuple $(g^{(u+mv)r}, g^r, \tilde{g}^{v/r}, \tilde{g}^{uv/r})$.

Given $(g, \tilde{g}, \tilde{g}^u, \tilde{g}^v, g^v)$ and unlimited access to this oracle, no adversary can have nonnegligible success probability to generate a pair $(g^{(u+m^*v)r^*}, g^{r^*})$, with $g^{r^*} \neq 1_{\mathbb{G}}$, for a new scalar m^* not asked to $\mathcal{O}$.

Intuitively, when comparing to PS assumption 1 in [52], we set $h = g^r$ and the oracle $\mathcal{O}$ outputs just two more elements in the group $\widetilde{\mathbb{G}}$ which offers no help for the adversary. It is thus obvious that our Assumption 1 holds and we refer the reader to [52] for details. We also note that this assumption was first introduced in [14].

Definition 8.

Assumption 2. Let $(p, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T, e)$ be a bilinear group setting of type 3, with g (resp., $\tilde{g}$) being a generator of $\mathbb{G}$ (resp., $\widetilde{\mathbb{G}}$). Choose $x, y, s, n \xleftarrow{\$} \mathbb{Z}_p^*$; we define two oracles: the oracle $\mathcal{O}_1$ which is on input $\mathsf{ID} \in \mathbb{Z}_p^*$ outputs the triplet$(g^{x/(s+\mathrm{ID})}, g^{y/(s+\mathrm{ID})}, g^{1/(s+\mathrm{ID})})$; the oracle $\mathcal{O}_2$ which is on input $(m, \{\mathsf{ID}_i \in \mathbb{Z}_p^*\}_{i\in S})$ for some set S chooses a random $r \in \mathbb{Z}_p^*$ and outputs the tuple $(g^{(x+my)r/\prod_{i\in S}(s+\mathrm{ID}_i)}, g^{r/\prod_{i\in S}(s+\mathrm{ID}_i)}, g^r, \tilde{g}^{y/r})$.

Given $(g, \tilde{g}, \{\tilde{g}^{s^i}\}_{i=1,\ldots,n}, \tilde{g}^x, \tilde{g}^y, g^x, g^y)$ and unlimited access to both oracles $\mathcal{O}_1$ and $\mathcal{O}_2$, no adversary can have nonnegligible success probability to generate a tuple $(g^{(x+m^*y)r^*/\prod_{i\in S^*}(s+\mathrm{ID}_i)}, g^{r^*/\prod_{i\in S^*}(s+\mathrm{ID}_i)}, g^{r^*}, \tilde{g}^{y/r^*})$, with $g^{r^*} \neq 1_{\mathbb{G}}$, and there exist at least one $\mathsf{ID}_\ell, \ell \in S^*$, not asked to $\mathcal{O}_1$, and input $(m^*, \{\mathsf{ID}_i \in \mathbb{Z}_p^*\}_{i\in S^*})$ not asked to $\mathcal{O}_2$.

For completeness, we prove that our Assumption 2 holds in Bilinear Generic Group in Appendix A.

3. Our SCL-AMS Scheme

In this section, we first gives the detailed construction of our SCL-AMS scheme, then in the next section we give the security analysis of our SCL-AMS scheme.

3.1. Construction. The construction of our SCL-AMS scheme is detailed as follows.

Setup: it takes as input the security parameter λ; the algorithm generates a bilinear map group system $(p, \mathbb{G}, \widetilde{\mathbb{G}}, \mathbb{G}_T, g, \tilde{g}, e)$. Next, it chooses random scalars $s, x, y \xleftarrow{\$} \mathbb{Z}_p^*$, and $n \in \mathbb{Z}_p^*$ is the maximum number of users in an aggregating set, as well as two hash functions $\mathcal{H}_1, \mathcal{H}_2$ where

$$\mathcal{H}_2 : \{0,1\}^* \longrightarrow \mathbb{Z}_p^*, \qquad \mathcal{H}_1 : \mathbb{Z}_p^{*2n} \longrightarrow \mathbb{Z}_p^* \tag{2}$$

Finally, it sets the public parameter

$$\mathsf{param} = \left(g, \tilde{g}, \left\{\tilde{g}^{s^i}\right\}_{i=1,\ldots,n}, \widetilde{X} = \tilde{g}^x, \widetilde{Y} = \tilde{g}^y, X = g^x, Y = g^y, \mathcal{H}_1, \mathcal{H}_2\right) \tag{3}$$

and the master secret key $\mathsf{msk} = s$.

Partial-Secret-Key-Generation: it takes as input param, $\mathsf{msk} = s$, and the user i's identity ID_i; let $\mathsf{ID}_i = \mathcal{H}_2(ID_i)$; the algorithm generates a partial secret key

$$\mathsf{P}_{ID_i} = (\mathsf{P}_{i,1}, \mathsf{P}_{i,2}, \mathsf{P}_{i,3}) = \left(g^{x/(s+\mathrm{ID}_i)}, g^{y/(s+\mathrm{ID}_i)}, g^{1/(s+\mathrm{ID}_i)}\right) \tag{4}$$

for user i.

User-Secret-Value-Generation: it takes as input user i's identity ID_i; the algorithm chooses $b_i \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $x_{ID_i} = b_i$ as user i's secret value.

Public-Key-Generation: it takes as input param and x_{ID_i}; the algorithm sets $PK_{ID_i} = \tilde{g}^{b_i}$ as the public key of user i.

Full-Secret-Key-Generation: it takes as input x_{ID_i} and P_{ID_i}; the algorithm outputs $\mathsf{SK}_i = (x_{ID_i}, \mathsf{P}_{ID_i})$ as the full secret key of user i.

Aggregation: denote S as the aggregating set of users, $|S| \leq n$. There are two cases, if $|S| = 1$, the algorithm takes as input param, ID_i, SK_i, and a message $M_i, i \in S$. The algorithm first chooses $r \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $m = \mathcal{H}_2(M_i), \mathsf{ID}_i = \mathcal{H}_2(ID_i)$ and then outputs signature $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$, where

$$\begin{aligned} \sigma_1 &= (\mathsf{P}_{i,1})^r (\mathsf{P}_{i,2})^{mr} (\mathsf{P}_{i,3})^{b_i r} = g^{(x+b_i+my)r/(s+\mathrm{ID}_i)}, \\ \sigma_2 &= (\mathsf{P}_{i,3})^r = g^{r/(s+\mathrm{ID}_i)}, \\ \sigma_3 &= g^r, \\ \sigma_4 &= \widetilde{Y}^{b_i/r} = \tilde{g}^{b_i y/r} \end{aligned} \tag{5}$$

If $|S| > 1$, the algorithm now is run by PKG and all users in S. More precisely, the algorithm consists of two separate following algorithms: note that $\{\mathsf{ID}_i = \mathcal{H}_2(ID_i)\}_{i\in S}$:

(1) First algorithm: PKG takes as input msk, the description of S, $\{M_i \in \{0,1\}^*\}_{i\in S}$ as well as param; note that if all messages $\{M_i \in \{0,1\}^*\}_{i\in S}$ are the same, the signature is multisignatures.

PKG follows strictly the order of pair $(M_i, ID_i)_{i\in S}$ to compute m as follows:

$$m = \mathcal{H}_1\left(\left\{\mathcal{H}_2(M_i), \mathcal{H}_2(ID_i)\right\}_{i\in S}, \{0\}_{i=1,\ldots,2n-2|S|}\right) \tag{6}$$

Next, it chooses $k \xleftarrow{\$} \mathbb{Z}_p^*$ and computes

$$P_{sig} = \left(g^{(x+my)k/\prod_{i\in S}(s+\mathrm{ID}_i)}, g^{k/\prod_{i\in S}(s+\mathrm{ID}_i)}, g^k, \tilde{g}^{y/k}\right) \tag{7}$$

P_{sig} are then sent to all users in S via public channel or PKG simply publishes it to the public domain. PKG then is not involved in the second algorithm.

(2) Second algorithm: it takes as input P_{sig} and runs by all users in S. First, each user $i \in S$ takes as input $x_{ID_i} = b_i, P_{sig}$ and computes the pair:

$$\left(g^{b_i k/\prod_{i\in S}(s+\mathrm{ID}_i)}, \tilde{g}^{b_i y/k}\right) \tag{8}$$

Finally, select one user in S as designated combiner, $|S| - 1$, other remaining users in S send their above outputs to this designated combiner via public channel, and the aggregate multisignature is then built as follows: designated combiner takes as input

$$\left(P_{sig}, \left\{g^{b_i k/\prod_{i\in S}(s+\mathrm{ID}_i)}, \tilde{g}^{b_i y/k}\right\}_{i\in S}\right) \tag{9}$$

and chooses $r' \xleftarrow{\$} \mathbb{Z}_p^*$ and outputs the aggregate multisignature $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$, where

$$\begin{aligned}\sigma_1 &= g^{(x+b+my)r/\prod_{i\in S}(s+\mathrm{ID}_i)},\\ \sigma_2 &= g^{r/\prod_{i\in S}(s+\mathrm{ID}_i)},\\ \sigma_3 &= g^{r},\\ \sigma_4 &= \widetilde{g}^{by/r}\end{aligned} \tag{10}$$

and $b = \sum_{i\in S} b_i$, $r = r' \cdot k$.

Note that the designated combiner computes

$$\begin{aligned}\sigma_1 &= g^{((x+my)k\cdot r')/\prod_{i\in S}(s+\mathrm{ID}_i)} \cdot \prod_{i\in S} g^{(b_i\cdot k\cdot r')/\prod_{i\in S}(s+\mathrm{ID}_i)};\\ \sigma_4 &= \prod_{i\in S} \widetilde{g}^{b_i y/(k\cdot r')}\end{aligned} \tag{11}$$

Aggregate Verification: it take as input aggregate multisignature $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$, the description of aggregating set of users S, messages $\{M_i \in \{0,1\}^*\}_{i\in S}$, and param, the algorithm computes $m = \mathscr{H}_1(\{\mathscr{H}_2(M_i), \mathscr{H}_2(ID_i)\}_{i\in S}, \{0\}_{i=1,\ldots,2n-2|S|})$, and

$$\begin{aligned}\mathsf{PK}_S &= \prod_{i\in S} \mathsf{PK}_{ID_i} = \widetilde{g}^{b},\\ T &= \widetilde{g}^{\prod_{i\in S}(s+\mathrm{ID}_i)},\\ T' &= \widetilde{X} \cdot \mathsf{PK}_S \cdot \widetilde{Y}^{m} \cdot \widetilde{g} = \widetilde{g}^{x+b+my+1};\end{aligned} \tag{12}$$

note that he/she can compute T from $\{\widetilde{g}^{s^i}\}_{i=1,\ldots,n}$ in param.

Finally, the aggregate multisignature is accepted (output 1) if the following equation holds:

$$\begin{aligned}&e(\sigma_1 \cdot \sigma_2, T) \cdot e(\sigma_3, \sigma_4) = e(\sigma_3, T') \cdot e(Y, \mathsf{PK}_S) \Longleftrightarrow\\ &e(\sigma_1 \cdot \sigma_2, T) \cdot e\left(\sigma_3, \frac{\sigma_4}{T'}\right) = e(Y, \mathsf{PK}_S)\end{aligned} \tag{13}$$

Correctness. It is easy to show that

$$\begin{aligned}&e(\sigma_1 \cdot \sigma_2, T) \cdot e(\sigma_3, \sigma_4)\\ &= e\left(g^{(x+b+my)r/\prod_{i\in S}(s+\mathrm{ID}_i)} \cdot g^{r/\prod_{i\in S}(s+\mathrm{ID}_i)}, \widetilde{g}^{\prod_{i\in S}(s+\mathrm{ID}_i)}\right)\\ &\quad \cdot e\left(g^{r}, \widetilde{g}^{by/r}\right) = e\left(g^{(x+b+my+1)r}, \widetilde{g}\right) \cdot e\left(g, \widetilde{g}^{by}\right)\\ &= e\left(g^{r}, \widetilde{g}^{x+b+my+1}\right) \cdot e\left(g^{y}, \widetilde{g}^{b}\right)\\ &= e\left(\sigma_3, T'\right) \cdot e(Y, \mathsf{PK}_S)\end{aligned} \tag{14}$$

Multisignature Scheme with Public Key Aggregation. In [25], the authors defined a multisignature scheme with public key aggregation, for which the public keys of users in the aggregating set S are aggregated into a short aggregate public key in advance through an additional Key Aggregation algorithm. The advantage of this scheme is that the verifier can only take a constant-size of input (multisignature and aggregate public key) to verify the multisignature, which were showed in [25, 43] that this type of scheme can be applied well to the Bitcoin blockchain application.

It is clear that our scheme is a multisignature scheme with public key aggregation as defined in [25], since we can add a new Key Aggregation algorithm which computes in advance the pair (PK_S, T) as an aggregate public key of the aggregating set S. The verifier therefore just needs to take a constant-size of input to verify the validity of the signature. Note that we can use a short length of bits to describe the aggregating set S.

Remark 9. We also note that, in our scheme, for an aggregate signature, we do not need to require that all messages $\{M_i \in \{0,1\}^*\}_{i\in S}$ are different.

3.2. Efficiency. Regarding the signature size, as in the construction and detailed in Table 1, the signature in our SCL-AMS scheme just contains four elements, three in the group $\mathbb{G}$ and one in the group $\widetilde{\mathbb{G}}$. If we decide to use NIST's figures [53], then to achieve 80 bits or 128 bits of security (that means the adversary needs at least 2^{80} or 2^{128} operations to break the security of our scheme), each element in the group $\mathbb{G}$ is about 160 bits or 256 bits and in the group $\widetilde{\mathbb{G}}$ is about 1024 bits or 3072 bits. This leads to the fact that the signature size of our scheme is 1504 bits or 3840 bits corresponding to the security parameter 80 bits or 128 bits, respectively.

Similarly, the user's full secret key size of our scheme is 640 bits or 1024 bits corresponding to the security parameter 80 bits or 128 bits, respectively.

Regarding the signing phase, there are two cases:

(1) $|S| = 1$, the signing phase requires five exponentiations in $\mathbb{G}$, one exponentiation in $\widetilde{\mathbb{G}}$, and two hash operations.

(2) $|S| > 1$, the signing phase requires $n' + 7$ exponentiations in $\mathbb{G}$, $n' + 2$ exponentiations in $\widetilde{\mathbb{G}}$, and $2n' + 1$ hash operations, where n' is the size of the aggregating set of signers.

Regarding the verification phase, there are also two cases:

(1) $|S| = 1$, the verification phase requires two exponentiations in $\widetilde{\mathbb{G}}$, three Pairings, and two hash operations.

(2) $|S| > 1$, the verification phase requires $n' + 1$ exponentiations in $\widetilde{\mathbb{G}}$, three Pairings, and $2n' + 1$ hash operations, where n' is the size of the aggregating set of signers.

Overall, for both storage and computational complexity, our scheme can meet the requirements for a lightweight device, that means our scheme is suitable for resource-constrained applications such as Bitcoin blockchain, Healthcare, Multicast Acknowledgment Aggregation, vehicular ad hoc networks, wireless sensor networks, Internet of Things, and so on.

3.3. Security. We give in this section two theorems which show that our SCL-AMS scheme is secure.

3.3.1. Secure against Strong Type I Adversary. We first prove that our SCL-AMS scheme is secure against strong Type I adversary.

Theorem 10. *If there exists a strong Type I adversary $\mathcal{A}_1$ defined in Game 1 (see Section 2.1.3), who breaks the security of our* SCL-AMS *scheme with probability ε, then there exists an adversary $\mathcal{C}$ who can break the security of the Assumption 2 with the same probability ε.*

Proof. Suppose that $\mathcal{C}$ is an adversary against our Assumption 2; we let $\mathcal{C}$ act as a challenger in Game 1 (see Section 2.1.3), that means $\mathcal{C}$ simulates $\mathcal{A}_1$ and then uses the output of $\mathcal{A}_1$ to break the security of the Assumption 2.

At the beginning, $\mathcal{C}$ has an instance of Assumption 2: $(g, \tilde{g}, \{\tilde{g}^{s^i}\}_{i=1,\ldots,n}, \tilde{g}^x, \tilde{g}^y, g^x, g^y)$, $x, y, s, n \in \mathbb{Z}_p^*$ as well as the right to access two oracles $\mathcal{O}_1$ and $\mathcal{O}_2$. Note that, for oracle $\mathcal{O}_1$, on input $\mathsf{ID} \in \mathbb{Z}_p^*$, output the triplet $(g^{x/(s+\mathrm{ID})}, g^{y/(s+\mathrm{ID})}, g^{1/(s+\mathrm{ID})})$. For oracle $\mathcal{O}_2$, on input $(m, \{\mathsf{ID}_i \in \mathbb{Z}_p^*\}_{i\in S})$, choose a random $r \in \mathbb{Z}_p$ and output $(g^{(x+my)r/\prod_{i\in S}(s+\mathrm{ID}_i)}, g^{r/\prod_{i\in S}(s+\mathrm{ID}_i)}, g^r, \tilde{g}^{y/r})$.

Note that $\mathcal{C}$ does not know the values s, x, and y.

To simulate $\mathcal{A}_1$, $\mathcal{C}$ first receives from $\mathcal{A}_1$ the target identity ID_ℓ for which he/she intends to attack.

$\mathcal{C}$ chooses two hash functions $\mathcal{H}_1, \mathcal{H}_2$ where

$$\begin{aligned} \mathcal{H}_2 &: \{0,1\}^* \longrightarrow \mathbb{Z}_p^*, \\ \mathcal{H}_1 &: \mathbb{Z}_p^{*2n} \longrightarrow \mathbb{Z}_p^* \end{aligned} \tag{15}$$

and then gives $(g, \tilde{g}, \{\tilde{g}^{s^i}\}_{i=1,\ldots,n}, \tilde{g}^x, \tilde{g}^y, g^x, g^y, \mathcal{H}_1, \mathcal{H}_2)$ to $\mathcal{A}_1$ as public parameters.

To simulate $\mathcal{A}_1$, $\mathcal{C}$ must manage to answer oracles as follows.

PartialSK(ID_i). $\mathcal{C}$ simply requests oracle $\mathcal{O}_1$ on input ID_i and directly forwards the result to $\mathcal{A}_1$. Note that, following Game 1 (see Section 2.1.3), $\mathcal{A}_1$ cannot make PartialSK(ID_i) query on $i = \ell$.

RequestPK(ID_i). $\mathcal{C}$ chooses $b_i \xleftarrow{\$} \mathbb{Z}_p^*$ and gives $PK_{ID_i} = \tilde{g}^{b_i}$ to $\mathcal{A}_1$.

UserSecret(ID_i). $\mathcal{C}$ first checks whether the RequestPK(ID_i) has been queried before; if it is not the case $\mathcal{C}$ first make this query on ID_i itself, using the above descriptions. In both cases, $\mathcal{C}$ knows x_{ID_i}; thus $\mathcal{C}$ gives x_{ID_i} to $\mathcal{A}_1$.

ReplacePK($ID_i, x'_{ID_i}, \mathsf{PK}'_{D_i}$). $\mathcal{C}$ knows x'_{ID_i} and then simply sets $\mathsf{PK}_{ID_i} = \mathsf{PK}'_{D_i} = \tilde{g}^{x'_{ID_i}}$.

RequestSig($\{M_i, ID_i\}_{i\in S}$). $\mathcal{C}$ computes $m = \mathcal{H}_1(\{\mathcal{H}_2(M_i), \mathcal{H}_2(ID_i)\}_{i\in S}, \{0\}_{i=1,\ldots,2n-2|S|})$, and $\{\mathsf{ID}_i = \mathcal{H}_2(ID_i)\}_{i\in S}$, then $\mathcal{C}$ requests oracle $\mathcal{O}_2$ on input $(m, \{\mathsf{ID}_i\}_{i\in S})$ to get the tuple:

$$\left(g^{(x+my)r/\prod_{i\in S}(s+\mathrm{ID}_i)}, g^{r/\prod_{i\in S}(s+\mathrm{ID}_i)}, g^r, \tilde{g}^{y/r}\right) \tag{16}$$

Since $\mathcal{C}$ knows $\{x_{ID_i} = b_i\}_{i\in S}$ (even if PK_{ID_i} has been replaced), he/she can easily compute the aggregate multisignature σ as the Aggregation algorithm and returns σ to $\mathcal{A}_1$.

We also note here that, in our Aggregation algorithm, the output of PKG and the outputs of users who are not designated combiner are sent via public channel (or made public), that means $\mathcal{A}_1$ can request to know these outputs.

Note that the output of PKG is

$$\left(g^{(x+my)k/\prod_{i\in S}(s+\mathrm{ID}_i)}, g^{k/\prod_{i\in S}(s+\mathrm{ID}_i)}, g^k, \tilde{g}^{y/k}\right) \tag{17}$$

and the output of each user in S is

$$\left(g^{b_i k/\prod_{i\in S}(s+\mathrm{ID}_i)}, \tilde{g}^{b_i y/k}\right) \tag{18}$$

for some $k \in \mathbb{Z}_p^*$. However, since $\mathcal{C}$ knows

$$\left(g^{(x+my)r/\prod_{i\in S}(s+\mathrm{ID}_i)}, g^{r/\prod_{i\in S}(s+\mathrm{ID}_i)}, g^r, \tilde{g}^{y/r}\right) \tag{19}$$

and all $\{x_{ID_i} = b_i\}_{i\in S}$, $\mathcal{C}$ can easily answer $\mathcal{A}_1$ by choosing $r' \xleftarrow{\$} \mathbb{Z}_p^*$ and implicitly sets $k = r' \cdot r$.

Finally, $\mathcal{A}_1$ outputs a tuple $(S^*, \{M_i^*\}_{i\in S^*}, \sigma^*)$ for which RequestSig($\{M_i^*, ID_i\}_{i\in S^*}$) has never been queried; note that $\ell \in S^*$.

From $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*)$, where

$$\begin{aligned} \sigma_1^* &= g^{(x+b^*+m^*y)r^*/\prod_{i\in S^*}(s+\mathrm{ID}_i)}, \\ \sigma_2^* &= g^{r^*/\prod_{i\in S^*}(s+\mathrm{ID}_i)}, \\ \sigma_3^* &= g^{r^*}, \\ \sigma_4^* &= \tilde{g}^{b^* y/r^*}, \end{aligned} \tag{20}$$

$b^* = \sum_{i\in S^*} b_i$, m^* is computed as in Aggregation algorithm, and $\mathcal{C}$ knows all $\{x_{ID_i} = b_i\}_{i\in S^*}$; it is easy for $\mathcal{C}$ to recover the tuple

$$T = \left(g^{(x+m^*y)r^*/\prod_{i\in S^*}(s+\mathrm{ID}_i)}, g^{r^*/\prod_{i\in S^*}(s+\mathrm{ID}_i)}, g^{r^*}, \tilde{g}^{y/r^*}\right) \tag{21}$$

Since RequestSig($\{M^*{}_i, ID_i\}_{i\in S^*}$) has never been queried, $(m^*, \{\mathsf{ID}_i\}_{i\in S^*})$ has never been asked to $\mathcal{O}_2$ and ID_ℓ also has never been asked to $\mathcal{O}_1$. This leads to the fact that T is a valid tuple to break the security of the Assumption 2.

Since the simulation is perfect, this leads to the fact that if there exists an adversary $\mathcal{A}_1$ who can break the security of our scheme with success probability ε, then there exists an attacker who can solve the Assumption 2 with the same success probability ε, which concludes our proof. □

3.3.2. Secure against Strong Type II Adversary. In this section, we prove that our SCL-AMS scheme is secure against strong Type II adversary.

Theorem 11. *If there exists an adversary $\mathcal{A}_2$ defined in Game 2 (see Section 2.1.3), who breaks the security of our* SCL-AMS *scheme with probability ε, then there exists an adversary $\mathcal{C}$ who can break the security of the Assumption 1 with the same probability ε.*

Proof. Suppose that $\mathscr{C}$ is an adversary against Assumption 1; we let $\mathscr{C}$ act as a challenger in Game 2 (see Section 2.1.3), that means $\mathscr{C}$ simulates $\mathscr{A}_2$ and then uses the output of $\mathscr{A}_2$ to break the security of the Assumption 1.

At the beginning, $\mathscr{C}$ has the instance of assumption: $(g, \tilde{g}, \tilde{g}^u, \tilde{g}^v, g^v)$ as well as the right to access oracle $\mathcal{O}$. Note that for oracle $\mathcal{O}$, on input $m \in \mathbb{Z}_p^*$, output the tuple $(g^{(u+mv)r}, g^r, , \tilde{g}^{v/r}, \tilde{g}^{uv/r})$.

Note that $\mathscr{C}$ does not know the values u and v.

To simulate $\mathscr{A}_2$, $\mathscr{C}$ first receives from $\mathscr{A}_2$ the target identity ID_ℓ for which he/she intends to attack. $\mathscr{C}$ chooses $x, s, n \xleftarrow{\$} \mathbb{Z}_p^*$ and two hash functions $\mathscr{H}_1, \mathscr{H}_2$ where

$$\begin{aligned} \mathscr{H}_2 &: \{0,1\}^* \longrightarrow \mathbb{Z}_p^*, \\ \mathscr{H}_1 &: \mathbb{Z}_p^{*2n} \longrightarrow \mathbb{Z}_p^* \end{aligned} \tag{22}$$

Next, $\mathscr{C}$ implicitly sets $y = v$ and then gives $(g, \tilde{g}, \{\tilde{g}^{s^i}\}_{i=1,\ldots,n}, \tilde{g}^x, \tilde{g}^y, g^x, g^y, \mathscr{H}_1, \mathscr{H}_2)$ to $\mathscr{A}_2$ as public parameters.

To simulate $\mathscr{A}_2$, $\mathscr{C}$ must manage to answer oracles as follows.

RequestPK(ID_i). If $i \neq \ell$, $\mathscr{C}$ chooses $x_{ID_i} = b_i \xleftarrow{\$} \mathbb{Z}_p^*$ and returns $\mathsf{PK}_{ID_i} = \tilde{g}^{b_i}$ to $\mathscr{A}_2$. If $i = \ell$, $\mathscr{C}$ implicitly sets $b_i = u$ and returns $\mathsf{PK}_{ID_i} = \tilde{g}^u$ to $\mathscr{A}_2$. Note that, in this case, $\mathscr{C}$ does not know x_{ID_i}.

UserSecret(ID_i). $\mathscr{C}$ checks whether RequestPK(ID_i) has been queried before, if it is not the case $\mathscr{C}$ first makes this query on ID_i itself, using the above description. In both cases, $\mathscr{C}$ knows x_{ID_i} and return x_{ID_i} to $\mathscr{A}_2$. Note that follow the definition of security model, $\mathscr{A}_2$ cannot make query on the case $i = \ell$.

ReplacePK($ID_i, x'_{ID_i}, \mathsf{PK}'_{ID_i}$). $\mathscr{C}$ knows x'_{ID_i} and then simply sets $\mathsf{PK}_{ID_i} = \mathsf{PK}'_{ID_i} = \tilde{g}^{x'_{ID_i}}$. Note that, following the definition of security model, $\mathscr{A}_2$ cannot make query on the case $i = \ell$.

RequestSig($\{M_i, ID_i\}_{i\in S}$). $\mathscr{C}$ computes $m = \mathscr{H}_1(\{\mathscr{H}_2(M_i), \mathscr{H}_2(ID_i)\}_{i\in S}, \{0\}_{i=1,\ldots,2n-2|S|})$, and $\{\mathsf{ID}_i = \mathscr{H}_2(ID_i)\}_{i\in S}$.

There are two cases, if $\ell \notin S$ then $\mathscr{C}$ knows all information; thus he/she chooses $r \xleftarrow{\$} \mathbb{Z}_p^*$ and easily computes the aggregate multisignature $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$, where

$$\begin{aligned} \sigma_1 &= g^{(x+b+my)r/\prod_{i\in S}(s+\mathrm{ID}_i)}, \\ \sigma_2 &= g^{r/\prod_{i\in S}(s+\mathrm{ID}_i)}, \\ \sigma_3 &= g^r, \\ \sigma_4 &= \tilde{g}^{by/r} \end{aligned} \tag{23}$$

and $b = \sum_{i\in S} b_i$. Finally, $\mathscr{C}$ returns σ to $\mathscr{A}_2$.

In case $\ell \in S$, $\mathscr{C}$ does not knows b_ℓ; that means $\mathscr{C}$ cannot know b; he/she therefore cannot compute σ_1, σ_4. In this case, $\mathscr{C}$ requests oracle $\mathcal{O}$ on input m to receive the tuple (note that $b_\ell = u$, $y = v$):

$$\left(g^{(u+mv)r}, g^r, \tilde{g}^{v/r}, \tilde{g}^{uv/r}\right) = \left(g^{(b_\ell+my)r}, g^r, \tilde{g}^{y/r}, \tilde{g}^{b_\ell y/r}\right) \tag{24}$$

Since $\mathscr{C}$ knows $\{x_{ID_i} = b_i\}_{\substack{i\in S \\ i\neq\ell}}$ (even if corresponding public keys have been replaced) and x, s, he/she can easily compute

$$\begin{aligned} \sigma_1 &= g^{(x+b_\ell+my)r/\prod_{i\in S}(s+\mathrm{ID}_i)} \cdot g^{(r\cdot b')/\prod_{i\in S}(s+\mathrm{ID}_i)} \\ &= g^{(x+b+my)r/\prod_{i\in S}(s+\mathrm{ID}_i)} \end{aligned} \tag{25}$$

and

$$\begin{aligned} \sigma_2 &= g^{r/\prod_{i\in S}(s+\mathrm{ID}_i)}, \\ \sigma_3 &= g^r, \\ \sigma_4 &= \tilde{g}^{b_\ell y/r} \cdot \tilde{g}^{b' y/r} = \tilde{g}^{by/r} \end{aligned} \tag{26}$$

where $b' = \sum_{\substack{i\in S \\ i\neq\ell}} b_i$. Finally, $\mathscr{C}$ returns σ to $\mathscr{A}_2$.

We also note here that, in our Aggregation algorithm, PKG only runs the first algorithm and does not participate in the second algorithm; that means $\mathscr{A}_2$ knows the output of PKG, which is

$$\left(g^{(x+my)k/\prod_{i\in S}(s+\mathrm{ID}_i)}, g^{k/\prod_{i\in S}(s+\mathrm{ID}_i)}, g^k, \tilde{g}^{y/k}\right) \tag{27}$$

for some $k \in \mathbb{Z}_p^*$. On the other hand, since in Aggregation algorithm, users who are not designated combiner send their outputs to designated combiner via public channel, that means $\mathscr{A}_2$ can know these outputs, which are $|S| - 1$ pairs:

$$\left(g^{b_i k/\prod_{i\in S}(s+\mathrm{ID}_i)}, \tilde{g}^{b_i y/k}\right) \tag{28}$$

To answer $\mathscr{A}_2$, $\mathscr{C}$ first implicitly sets user with identity ID_ℓ as designated combiner. Since $\mathscr{C}$ knows $x, s, g^y, \tilde{g}^y, \{x_{ID_i} = b_i\}_{\substack{i\in S \\ i\neq\ell}}$, he/she can choose $k \xleftarrow{\$} \mathbb{Z}_p^*$ and easily answer $\mathscr{A}_2$.

Finally, $\mathscr{A}_2$ outputs a tuple $(S^*, \{M_i^*\}_{i\in S^*}, \sigma^*)$ for which RequestSig($\{M_i^*, ID_i\}_{i\in S^*}$) has never been queried; note that $\ell \in S^*$.

From $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*)$, where

$$\begin{aligned} \sigma_1^* &= g^{(x+b^*+m^* y)r^*/\prod_{i\in S^*}(s+\mathrm{ID}_i)}, \\ \sigma_2^* &= g^{r^*/\prod_{i\in S^*}(s+\mathrm{ID}_i)}, \\ \sigma_3^* &= g^{r^*}, \\ \sigma_4^* &= \tilde{g}^{b^* y/r^*}, \end{aligned} \tag{29}$$

$b^* = \sum_{i\in S^*} b_i$, and m^* is computed as in Aggregation algorithm.

Since $\mathscr{C}$ knows x, s and all $\{x_{ID_i} = b_i\}_{\substack{i\in S^* \\ i\neq\ell}}$, he/she computes

$$\sigma_1' = g^{(x+b')r^*/\prod_{i\in S^*}(s+\mathrm{ID}_i)} \tag{30}$$

where $b' = \sum_{\substack{i \in S^* \\ i \neq \ell}} b_i$; next he/she computes

$$\left(\frac{\sigma_1^*}{\sigma_1'}\right)^{\prod_{i \in S^*}(s+\mathrm{ID}_i)} = \left(g^{(x+b^*+m^*y-x-b')r^*/\prod_{i \in S^*}(s+\mathrm{ID}_i)}\right)^{\prod_{i \in S^*}(s+\mathrm{ID}_i)} = g^{(b_\ell + m^* y)r^*} = g^{(u+m^* v)r^*} \tag{31}$$

Finally, $\mathscr{C}$ can have the pair

$$T = \left(g^{(u+m^* v)r^*}, g^{r^*}\right) \tag{32}$$

Since $\mathsf{RequestSig}(\{M^*_i, ID_i\}_{i \in S^*})$ has never been queried, m^* has never been asked to $\mathscr{O}$; this leads to the fact that T is a valid pair to break the security of the Assumption 1. Note that $m^* = \mathscr{H}_1(\{\mathscr{H}_2(M_i^*), \mathscr{H}_2(ID_i)\}_{i \in S^*}, \{0\}_{i=1,\ldots,2n-2|S^*|})$.

Since the simulation is perfect, this leads to the fact that if there exists an adversary $\mathscr{A}_2$ who can break the security of our scheme with success probability ε, then there exists an attacker who can solve the Assumption 1 with the same success probability ε, which concludes our proof. □

4. Conclusion

Aggregate signature and multisignature have many practical applications such as Bitcoin blockchain, Healthcare, Multicast Acknowledgment Aggregation, and so on, while certificateless is an interesting technique to avoid the use of expensive certificate in PKI and key escrow problem in identity-based cryptography. In this paper, we consider the combination of an aggregate signature scheme and a multisignature scheme in the certificateless setting. We concretely propose the first server-aided certificateless aggregate multisignature scheme which achieves the following properties:

(i) the signature contains four elements in all cases. When using NIST's figures [53], the signature size of our scheme is 1504 bits or 3840 bits corresponding to the security parameter 80 bits or 128 bits, respectively. The user's full secret key size is 640 bits or 1024 bits corresponding to the security parameter 80 bits or 128 bits, respectively;

(ii) it is secure against strong Type I and strong Type II adversaries in the standard model under GDDHE assumptions;

(iii) support public key aggregation;

(iv) public key size, signing time, and verifying time depend on the maximum number of signers for one aggregating set, which is fixed at the setup.

Appendix

A. Proof of Assumption 2 in Bilinear Generic Group

Let $q \in \mathbb{Z}_p^*$ be the maximum number of queries that the adversary $\mathscr{A}$ can ask each oracle $\mathscr{O}_1$ and oracle $\mathscr{O}_2$. The input for which $\mathscr{A}$ can have comes from asking oracles $\mathscr{O}_1$, $\mathscr{O}_2$, and param. We can write that input as follows.

Regarding the input in the group $\widetilde{\mathbb{G}}$, $\mathscr{A}$ has

$$P = \left(1, x, y, \left\{s^i\right\}_{i \in [n]}, \left\{\frac{y}{r_{i,j}}\right\}_{i,j \in [\sqrt{q}]}\right) \tag{A.1}$$

Regarding the input in the group $\mathbb{G}$, $\mathscr{A}$ has $Q =$

$$\left(1, x, y, \left\{\frac{x}{s+\mathrm{ID}_i}, \frac{y}{s+\mathrm{ID}_i}, \frac{1}{s+\mathrm{ID}_i}\right\}_{\substack{i \in [q] \\ i \neq \ell}}, \left\{\frac{(x+m_i.y).r_{i,j}}{\prod_{k \in S_j}(s+\mathrm{ID}_k)}, \frac{r_{i,j}}{\prod_{k \in S_j}(s+\mathrm{ID}_k)}, r_{i,j}\right\}_{\substack{(i,j) \in [\sqrt{q}] \times [\sqrt{q}] \\ (i,j) \neq (t,t)}}\right) \tag{A.2}$$

where $S^* = S_t, \ell \in S_t, m^* = m_t$.

To prove the security of the assumption, we will show that, simultaneously from P, $\mathscr{A}$ cannot lead to $y/r_{t,t}$ and from Q, $\mathscr{A}$ cannot lead to the triplet

$$T = (T_1, T_2, T_3) = \left(\frac{(x+m_t.y).r_{t,t}}{\prod_{k \in S_t}(s+\mathrm{ID}_k)}, \frac{r_{t,t}}{\prod_{k \in S_t}(s+\mathrm{ID}_k)}, r_{t,t}\right) \tag{A.3}$$

We assume that $\mathscr{A}$ find linear combinations of elements in Q which lead to the triplet T denote these linear combinations (A_1, A_2, A_3) which lead to (T_1, T_2, T_3), respectively. It is easy to see that

$$A_1 = (x + m_t.y).A_2 \tag{A.4}$$

$$A_3 = \left(\prod_{k \in S_t}(s+\mathrm{ID}_k)\right).A_2 \tag{A.5}$$

Regarding the first equation, since in A_1, the highest degree of variables x, y is 1 and $r_{i,j}$ are unknown random constants; in A_2 we cannot have elements

$$\left(x, y, \left\{\frac{x}{s+\mathrm{ID}_i}, \frac{y}{s+\mathrm{ID}_i}\right\}_{\substack{i \in [q] \\ i \neq \ell}}, \left\{\frac{(x+m_i.y).r_{i,j}}{\prod_{k \in S_j}(s+\mathrm{ID}_k)}, r_{i,j}\right\}_{\substack{(i,j) \in [\sqrt{q}] \times [\sqrt{q}] \\ (i,j) \neq (t,t)}}\right) \tag{A.6}$$

Regarding the second equation, since in A_3, the degree of variables s is 0 and $r_{i,j}$ are unknown random constants; in A_2, we cannot have elements

$$1, x, y, \left\{r_{i,j}\right\}_{\substack{(i,j) \in [\sqrt{q}] \times [\sqrt{q}] \\ (i,j) \neq (t,t)}} \tag{A.7}$$

Overall, in A_2, we only can have elements

$$\left(\left\{ \frac{1}{s+\mathrm{ID}_i} \right\}_{\substack{i\in[q]\\ i\neq\ell}}, \left\{ \frac{r_{i,j}}{\prod_{k\in S_j}(s+\mathrm{ID}_k)} \right\}_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ (i,j)\neq(t,t)}} \right); \tag{A.8}$$

that means $\mathscr{A}$ has to find constants $\{c_i\}_{\substack{i\in[q]\\ i\neq\ell}}$, $\{d_{i,j}\}_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ (i,j)\neq(t,t)}}$ to produce A_2:

$$A_2 = \sum_{\substack{i\in[q]\\ i\neq\ell}} \frac{c_i}{s+\mathrm{ID}_i} + \sum_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ (i,j)\neq(t,t)}} \frac{d_{i,j}.r_{i,j}}{\prod_{k\in S_j}(s+\mathrm{ID}_k)} \tag{A.9}$$

Similarly, suppose that $\mathscr{A}$ finds the linear combination of elements in P (denote B) which leads to $y/r_{t,t}$; we have

$$\begin{aligned}
&\Longleftrightarrow B = \frac{y}{r_{t,t}} \\
&\Longleftrightarrow y = B\cdot A_3 \\
&\Longleftrightarrow y = B\cdot \prod_{k\in S_t}(s+\mathrm{ID}_k)\cdot A_2 \\
&\Longleftrightarrow y = B\cdot \prod_{k\in S_t}(s+\mathrm{ID}_k)\left(\sum_{\substack{i\in[q]\\ i\neq\ell}} \frac{c_i}{s+\mathrm{ID}_i} \right. \\
&\qquad \left. + \sum_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ (i,j)\neq(t,t)}} \frac{d_{i,j}.r_{i,j}}{\prod_{k\in S_j}(s+\mathrm{ID}_k)} \right)
\end{aligned} \tag{A.10}$$

It is obvious that B cannot contain the elements $(1, x, \{s^i\}_{i\in[n]})$; that means $\mathscr{A}$ needs to find the constants $a, \{b_{i,j}\}_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ (i,j)\neq(t,t)}}$ such that the following equation holds:

$$\begin{aligned}
\Longleftrightarrow y &= B\cdot \prod_{k\in S_t}(s+\mathrm{ID}_k)\left(\sum_{\substack{i\in[q]\\ i\neq\ell}} \frac{c_i}{s+\mathrm{ID}_i} \right. \\
&\quad \left. + \sum_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ (i,j)\neq(t,t)}} \frac{d_{i,j}.r_{i,j}}{\prod_{k\in S_j}(s+\mathrm{ID}_k)} \right)
\end{aligned} \tag{A.11}$$

$$\begin{aligned}
\Longleftrightarrow y &= \left(a\cdot y + \sum_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ (i,j)\neq(t,t)}} \frac{b_{i,j}\cdot y}{r_{i,j}} \right) \\
&\quad \cdot \prod_{k\in S_t}(s+\mathrm{ID}_k)\left(\sum_{\substack{i\in[q]\\ i\neq\ell}} \frac{c_i}{s+\mathrm{ID}_i} \right. \\
&\quad \left. + \sum_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ (i,j)\neq(t,t)}} \frac{d_{i,j}.r_{i,j}}{\prod_{k\in S_j}(s+\mathrm{ID}_k)} \right)
\end{aligned} \tag{A.12}$$

$$\begin{aligned}
\Longleftrightarrow 1 &= \left(a + \sum_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ (i,j)\neq(t,t)}} \frac{b_{i,j}}{r_{i,j}} \right)\cdot \prod_{k\in S_t}(s+\mathrm{ID}_k) \\
&\quad \cdot \left(\sum_{\substack{i\in[q]\\ i\neq\ell}} \frac{c_i}{s+\mathrm{ID}_i} \right. \\
&\quad \left. + \sum_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ (i,j)\neq(t,t)}} \frac{d_{i,j}.r_{i,j}}{\prod_{k\in S_j}(s+\mathrm{ID}_k)} \right)
\end{aligned} \tag{A.13}$$

From above equation, since $\{r_{i,j}\}_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ (i,j)\neq(t,t)}}$ are unknown, there are two cases:

(1) if there exists at least one $b_{i,j}\neq 0$, to cancel out $r_{i,j}$, it is obvious that $a = 0, \{c_i = 0\}_{\substack{i\in[q]\\ i\neq\ell}}$. We thus can rewrite the above equation as

$$\begin{aligned}
\Longleftrightarrow 1 &= \left(\sum_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ (i,j)\neq(t,t)}} \frac{b_{i,j}}{r_{i,j}} \right)\cdot \prod_{k\in S_t}(s+\mathrm{ID}_k) \\
&\quad \cdot \left(\sum_{\substack{(i,j)\in[\sqrt{q}]\times[\sqrt{q}]\\ (i,j)\neq(t,t)}} \frac{d_{i,j}.r_{i,j}}{\prod_{k\in S_j}(s+\mathrm{ID}_k)} \right)
\end{aligned} \tag{A.14}$$

Since $(i,j)\in[\sqrt{q}]\times[\sqrt{q}], (i,j)\neq(t,t)$ and unknown $r_{i,j}$, it is obvious to deduce that $\mathscr{A}$ cannot find the constants $b_{i,j}, d_{i,j}$ to make the above equation hold.

(2) if $b_{i,j} = 0$ for all i, j, then to cancel out $r_{i,j}$ we have all $d_{i,j} = 0$. The equation now is rewritten as

$$\Longleftrightarrow 1 = a\cdot \prod_{k\in S_t}(s+\mathrm{ID}_k)\sum_{\substack{i\in[q]\\ i\neq\ell}} \frac{c_i}{s+\mathrm{ID}_i} \tag{A.15}$$

It is obvious that, to cancel out s, $|S_t| = 1$. On the other hand, since $\ell \in S_t$, that means S_t contains only ID_ℓ; we can again rewrite the equation as

$$\Longleftrightarrow 1 = a \cdot (s + \mathrm{ID}_\ell) \sum_{\substack{i \in [q] \\ i \neq \ell}} \frac{c_i}{s + \mathrm{ID}_i} \tag{A.16}$$

Since all identities are different, it is obvious that $\mathcal{A}$ cannot find the constants a, c_i to make the above equation hold.

The above two cases conclude our proof.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This paper was written while the author was visiting the Vietnam Institute of Mathematics for Advanced Study in Mathematics (VIASM). He is grateful for the VIASM for its support and hospitality. This work was conducted within the context of the Vietnamese Nafosted Project "Some Advanced Encryption Schemes for Lightweight Devices".

References

[1] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in Cryptology: Proceedings of (CRYPTO '84)*, G. R. Blakley and D. Chaum, Eds., vol. 196 of *Lecture Notes in Computer Science*, pp. 47–53, Springer, Santa Barbara, CA, USA, Berlin, Germany, 1985.

[2] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Advances in Cryptology – ASIACRYPT*, C.-S. Laih, Ed., vol. 2894 of *Lecture Notes in Computer Science*, pp. 452–473, Springer, Taipei, Taiwan, Heidelberg, Germany, 2003.

[3] S. K. H. Islam and G. P. Biswas, "Certificateless short sequential and broadcast multisignature schemes using elliptic curve bilinear pairings," *Journal of King Saud University - Computer and Information Sciences*, vol. 26, no. 1, pp. 89–97, 2014.

[4] S. Hafizul Islam, M. S. Farash, G. P. Biswas, M. K. Khan, and M. S. Obaidat, "A pairing-free certificateless digital multisignature scheme using elliptic curve cryptography," *International Journal of Computer Mathematics*, vol. 94, no. 1, pp. 39–55, 2017.

[5] L. Wu, Z. Xu, D. He, and X. Wang, "New certificateless aggregate signature scheme for healthcare multimedia social network on cloud environment," *Journal of Security and Communication Networks*, vol. 2018, Article ID 2595273, 13 pages, 2018.

[6] D. H. Yum and P. J. Lee, "Generic construction of certificateless signature," in *Proceedings of the ACISP 04: 9th Australasian Conference on Information Security and Privacy*, J. Pieprzyk . In H. Wang, Ed., vol. 3108 of *Lecture Notes in Computer Science*, pp. 200–211, Springer, Sydney, NSW, Australia , Heidelberg, Germany, 2004.

[7] B. Hu, D. Wong, Z. Zhang, and X. Deng, "Key replacement attack against a generic construction of certificateless signature," in *Proceedings of the ACISP 06: 11th Australasian Conference on Information Security and Privacy*, L. M. Batten and R. Safavi-Naini, Eds., vol. 4058 of *Lecture Notes in Computer Science*, pp. 235–246, Springer, Melbourne, Australia, Heidelberg, Germany, 2006.

[8] J. Liu, M. Au, and W. Susilo, "Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standardmodel," in *Proceedings of the 2007 ACM Symp. Information*, vol. 4450, pp. 476–489, Singapore, 2007.

[9] A. W. Dent, B. t. Libert, and K. . Paterson, "Certificateless encryption schemes strongly secure in the standard model," in *Proceedings of the PKC 2008: 11th International Workshop on Theory and Practice in Public Key Cryptography*, R. Cramer, Ed., vol. 4939 of *Lecture Notes in Computer Science*, pp. 344–359, Springer, Barcelona, Spain, Heidelberg, Germany.

[10] Y. Yuan, D. Li, L. Tian, and H. Zhu, "Certificateless signature scheme without random oracles," in *Proceedings of the ISA 2009*, vol. 5576 of *LNCS*, pp. 31–40, 2009.

[11] Y. Yu, Y. Mu, G. Wang, Q. Xia, and B. Yang, "Improved certificateless signature scheme provably secure in the standard model," *IET Information Security*, vol. 6, no. 2, pp. 102–110, 2012.

[12] Q. Xia, C. Xu, and Y. Yu, "Key replacement attack on two certificateless signature schemes without random oracles," *Key Engineering Materials*, vol. 439-440, pp. 1606–1611, 2010.

[13] X. Huang, Y. Mu, W. Susilo, D. S. Wong, and W. Wu, "Certificateless signatures: new schemes and security models," *The Computer Journal*, vol. 55, no. 4, pp. 457–474, 2012.

[14] S. Canard and V. C. Trinh, "An efficient certificateless signature scheme in the standard model," in *Proceedings of the nternational Conference on Information Systems Security, ICISS 201*, I. Ray, M. Gaur, M. Conti, D. Sanghi, and V. Kamakoti, Eds., vol. 10063 of *Lecture Notes in Computer Science*, pp. 175–192, Springer International Publishing.

[15] A. Karati, S. K. H. Islam, and M. Karuppiah, "Provably secure and lightweight certificateless signature scheme for IIoT environments," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3701–3711, 2018.

[16] D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical identity based encryption with constant size ciphertext," in *Advances in Cryptology – EUROCRYPT 2005*, editor R. Cramer, Ed., vol. 3494 of *Lecture Notes in Computer Science*, pp. 440–456, Springer, Aarhus, Denmark, Heidelberg, Germany, 2005.

[17] P. Guillot, A. Nimour, D. H. Phan, and V. C. Trinh, "Optimal public key traitor tracing scheme in non-black box model," in *Proceedings of the AfricaCrypt 2013: 6th International Conference on Cryptology in Africa*, A. Youssef, A. Nitaj, Hassanien, and A. E., Eds., vol. 7918 of *Lecture Notes in Computer Science*, pp. 140–155, Cairo, Egypt, 2013.

[18] S. Canard and V. C. Trinh, "Constant-size ciphertext attribute-based encryption from multi-channel broadcast encryption," in *Proceedings of the International Conference on Information Systems Security, ICISS 2016*, I. Ray, M. Gaur, M. Conti, D. Sanghi, and V. Kamakoti, Eds., vol. 10063 of *Lecture Notes in Computer Science*, pp. 193–211, Springer International Publishing, 2016.

[19] D. H. Phan, D. Pointcheval, and V. C. Trinh, "Multi-channel broadcast encryption," in *Proceedings of the ASIACCS 13: 8th ACM Symposium on Information, Computer and Communications Security*, K. Chen, Q. Xie, W. Qiu, N. Li, and W.-G. Tzeng, Eds., pp. 277–286, ACM Press, Hangzhou, China, 2013.

[20] S. Canard, D. Phan, and V. C. Trinh, "Attribute-based broadcast encryption scheme for lightweight devices," *IET Information Security*, vol. 12, no. 1, pp. 52–59, 2018.

[21] Q. M. Malluhi, A. Shikfa, and V. C. Trinh, "A ciphertext-policy attribute-based encryption scheme with optimized ciphertext size and fast decryption," in *Proceedings of the ASIACCS 17: 12th ACM Symposium on Information, Computer and Communications Security*, pp. 230–240, ACM Press, 2017.

[22] S. Canard, D. H. Phan, D. Pointcheval, and V. . Trinh, "A new technique for compacting ciphertext in multi-channel broadcast encryption and attribute-based encryption," *Theoretical Computer Science*, vol. 723, pp. 51–72, 2018.

[23] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Advances in Cryptology – EUROCRYPT*, E. Biham, Ed., vol. 2656 of *Lecture Notes in Computer Science*, pp. 416–432, Springer, Warsaw, Poland, Springer, Heidelberg, Germany, 2003.

[24] K. Brogle, S. Goldberg, and L. Reyzin, "Sequential aggregate signatures with lazy verification from trapdoor permutations - (extended abstract)," in *Advances in Cryptology – ASIACRYPT 2012*, X. Wang and K. Sako, Eds., vol. 7658 of *Lecture Notes in Computer Science*, pp. 644––662, Springer, Heidelberg, Germany, Beijing, China, 2012.

[25] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille, "Simple schnorr multi-signatures with applications to bitcoin," in *Cryptology ePrint Archive*, 2018, https://eprint.iacr.org/2018/068/20180118:124757.

[26] S. Kent, C. Lynn, and K. Seo, "Secure border gateway protocol (Secure-BGP)," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 4, p. 58292, 2000.

[27] G. Zheng, L. Yu, H. Xuan, and C. Kefei, "Two certificateless aggregate signatures from bilinear maps," in *Proceedings of the SNPD 2007: 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, pp. 188–193, 2007.

[28] L. Zhang and F. Zhang, "A new certificateless aggregate signature scheme," *Computer Communications*, vol. 32, no. 6, pp. 1079–1085, 2009.

[29] H. Xiong, Z. Guan, Z. Chen, and F. Li, "An efficient certificateless aggregate signature with constant pairing computations," *Information Sciences*, vol. 219, pp. 225–235, 2013.

[30] H. Liu, S. Wang, M. Liang, and Y. Chen, "New construction of efficient certificateless aggregate signatures," *International Journal of Security and Its Applications*, vol. 8, no. 1, pp. 411–422, 2014.

[31] D. He, M. Tian, and J. Chen, "Insecurity of an efficient certificateless aggregate signature with constant pairing computations," *Information Sciences*, vol. 268, pp. 458–462, 2014.

[32] H. Tu, D. He, and B. Huang, "Reattack of a certificateless aggregate signature scheme with constant pairing computations," *The Scientific World Journal*, vol. 2014, Article ID 343715, 10 pages, 2014.

[33] L. Cheng, Q. Wen, Z. Jin, H. Zhang, and L. Zhou, "Cryptanalysis and improvement of a certificateless aggregate signature scheme," *Information Sciences*, vol. 295, pp. 337–346, 2015.

[34] Y. Zhang and C. Wang, "Comment on new construction of efficient certificateless aggregate signatures," *International Journal of Security and Its Applications*, vol. 9, no. 1, pp. 147–154, 2015.

[35] P. Kumar, S. Kumari, V. Sharma, A. K. Sangaiah, J. Wei, and X. Li, "A certificateless aggregate signature scheme for healthcare wireless sensor network," *Sustainable Computing*, 2017.

[36] K. Itakura and K. Nakamura, "A public key cryptosystem suitable for digital multisignatures," *NEC Research and Development*, vol. 71, pp. 1–8, 1983.

[37] S. Micali, K. Ohta, and L. Reyzin, "Accountable-subgroup multisignatures: Extended abstract," in *Proceedings of the ACM CCS 01: 8th Conference on Computer and Communications Security*, pp. 245–254, ACM Press, Philadelphia, PA, USA, 2001.

[38] A. Lysyanskaya, "Unique signatures and verifiable random functions from the DH-DDH separation," in *Advances in Cryptology – CRYPTO 2002*, M. Yung, Ed., vol. 2442 of *Lecture Notes in Computer Science*, pp. 597–612, Springer, Berlin, Heidelberg, Germany, 2002.

[39] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme," in *Proceedings of the KC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, Y. Desmedt, Ed., vol. 2567 of *Lecture Notes in Computer Science*, pp. 31–46, Springer, Miami, USA, Heidelberg, Germany, 2003.

[40] A. Bagherzandi, J. Cheon, and S. Jarecki, "Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma," in *Proceedings of the ACM CCS 08: 15th Conference on Computer and Communications Security*, P. Ning, P. F. Syverson, and S. Jha, Eds., pp. 449–458, ACM Press, Alexandria, Virginia, USA, 2008.

[41] R. El Bansarkhani and J. Sturm, "An efficient lattice-based multisignature scheme with applications to bitcoins," in *Proceedings of the CANS 16: 15th International Conference on Cryptology and Network Security*, Lecture Notes in Computer Science, pp. 140–155, Springer, Heidelberg, Germany, 2016.

[42] C. Gentry, A. O'Neill, and L. Reyzin, "A unified framework for trapdoor-permutation-based sequential aggregate signatures," in *Proceedings of the PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part II*, M. Abdalla and R. Dahab, Eds., vol. 10770 of *Lecture Notes in Computer Science*, pp. 34–57, Springer, Heidelberg, Germany, Rio de Janeiro, Brazil, 2018.

[43] D. Boneh, M. Drijvers, and G. Neven, "Compact Multi-signatures for Smaller Blockchains," Cryptology ePrint Archive: Report 2018/483, https://eprint.iacr.org/2018/483.

[44] H. Du and Q. Wen, "Certificateless proxy multi-signature," *Information Sciences*, vol. 276, pp. 21–30, 2014.

[45] Z. Jin, Q. Wang, and Z. Li, "A formal construction of certificateless proxy multi-signature scheme," *International Journal of Security and Networks*, vol. 11, no. 3, pp. 126–139, 2016.

[46] W. Wu, Y. Mu, W. Susilo, and X. Huang, "Server-aided verification signatures: definitions and new constructions," in *Proceedings of the 2nd International Conference on Provable Security Proceeding ProvSec'08*, vol. 5324 of *Lecture Notes in Computer Science*, pp. 141–155, Springer Berlin Heidelberg, Shanghai, China, 2008.

[47] H. Wu, C. Xu, and J. Deng, "Server-aided aggregate verification signature: security definition and construction," *International Journal of Information and Communication Technology*, vol. 7, no. 2/3, p. 278, 2015.

[48] H. Wu, C. Xu, and J. Deng, "A Server-Aided Aggregate Verification Signature Scheme from Bilinear Pairing," in *Proceedings of the 5th International Conference on Intelligent Networking and Collaborative Systems (INCoS)*, 2013.

[49] S. S. M. Chow, M. H. Au, and W. Susilo, "Server-aided signatures verification secure against collusion attack," in *Proceedings of the 6th ACM Symposium on Information, Computer and*

Communications Security Proceeding ASIACCS '11, pp. 401–405, Hong Kong, China, 2011.

[50] D. Boneh, C. Gentry, and B. Waters, "Collusion resistant broadcast encryption with short ciphertexts and private keys," in *Advances in Cryptology – CRYPTO 2005*, V. Shoup, Ed., vol. 3621 of *Lecture Notes in Computer Science*, pp. 258–275, Springer, Heidelberg, Germany.

[51] Y. Rouselakis and B. Waters, "Practical constructions and new proof methods for large universe attribute-based encryption," in *Proceedings of the ACM CCS 13: 20th Conference on Computer and Communications Security*, A.-R. Sadeghi, V. D. Gligor, and M. Yung, Eds., pp. 463–474, ACM Press, Berlin, Germany, 2013.

[52] D. Pointcheval and O. Sanders, "Short randomizable signatures," in *Proceedings of the Topics in Cryptology - CT-RSA 2016 Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference*, vol. 9610 of *Lecture Notes in Computer Science*, pp. 111–126, Springer, San Francisco, CA, USA, 2016.

[53] NIST, *Recommendation for Key Management Part 1: General, NIST Special Publication 800-57*, 2005, http://csrc.nist.gov/publications/nistpubs/800-57/SP800-57-Part1.pdf.