Pairing-based Accountable Subgroup Multi-signatures with Verifiable Group Setup

Ahmet Ramazan Ağırtaş

Oğuz Yayla

Institute of Applied Mathematics,
Middle East Technical University,
06800, Çankaya, Ankara, Turkey
{*agirtas.ramazan, oguz*}*@metu.edu.tr*

January 6, 2022

Abstract

An accountable subgroup multi-signature is a kind of multi-signature scheme in which any subgroup $\mathcal{S}$ of the group $\mathcal{G}$ of potential signers jointly sign a message m, ensuring that each member of $\mathcal{S}$ is accountable for the resulting signature. In this paper we propose three novel pairing-based accountable subgroup multi-signature (ASM) schemes. In the first one, we use Feldman's verifiable secret sharing scheme as an implicit authentication and proof-of-possession for setting up the group $\mathcal{G}$. In the second one, the members participating in authentication is decided by the subgroup itself. In the third one, we consider a designated combiner managing the authentication process. All schemes that we propose here require fewer computations in signature generation, signature aggregation and verification phases than the pairing-based ASM scheme proposed by Boneh, Drijvers and Neven. Moreover, our first and the third ones solve the open problem of constructing an ASM scheme in which the subgroup $\mathcal{S}$ of signers is not known before the signature generation. Besides, we give a method of eliminating the combiner in case of knowing the subgroup of signers $\mathcal{S}$ in advance. Further we extend our proposed schemes to aggregated versions. For n accountable subgroup multi-signatures, aggregated versions of our proposed schemes output an aggregated signature with size of a single group element and require $n+1$ pairings in aggregated signature verification, whereas the partial aggregated ASM scheme of Boneh, Drijvers and Neven gives an aggregated signature with size of $n+1$ group elements and requires $2n+1$ pairings in aggregated signature verification.

1 Introduction

Digital signature schemes play key role in modern cryptographic protocols for verifying the authenticity of any message, such as official documents, financial transactions, e-mails, etc. In particular, increasing popularity of cryptocurrencies [5, 15, 16] in the last decade make the digital signature schemes more significant than before. The structures of signature schemes differ according to the using area, system requirements, and users' needs.

A *multi-signature* [1, 12, 17, 18, 19, 20] is a kind of digital signature in which a group of signers sign the same message jointly. In literature, there are some notions related to multi-signatures for different scenarios. A *group signature* [6, 7] includes a group $\mathcal{G}$ of potential signers, where any member i can anonymously sign a message on behalf of the entire group. In fact, it is not exactly anonymous signature. Because there is a group manager in $\mathcal{G}$ who knows the identities of all of the signers. Besides, it does not support subgroup of signers with more than one member [14]. A *threshold signature* [1, 8, 10, 11] is a variant of multi-signature schemes in which t signers are needed among $n = |\mathcal{G}|$ potential signers in order to create a legitimate signature on behalf of the entire group $\mathcal{G}$. It ensures anonymity, i.e. signers are unknown to any verifier. There is a more general notion called *aggregate signature* [3] which provides an aggregation for different signatures on different messages into a single signature.

Neither group signatures nor threshold signatures and aggregate signatures provide sufficient flexibility and accountability at the same time. An *accountable multi-signature (ASM)* [2, 14] is a multi-signature scheme in which any subgroup $\mathcal{S} \subseteq \mathcal{G}$ jointly sign a message m, ensuring that each member of $\mathcal{S}$ is accountable for the resulting signature. This notion was firstly defined by Micali et al. in [14] by proposing the first ASM scheme. In a more recent paper [2], Boneh, Drijvers and Neven proposed another ASM scheme which is based on BLS signature [4] and solves the open problem in [14], i.e. constructing an ASM scheme in which the subgroup $\mathcal{S} \subseteq \mathcal{G}$ is not determined before the signature generation.

In this paper we focus on accountable subgroup multi-signatures (ASM). We propose three novel accountable subgroup multi-signature schemes. The first one is vASM (verifiable ASM) scheme which is indeed a modified BLS signature. We give a method of generating a membership key via VSS protocol [9], which transforms BLS signature scheme into an ASM scheme. The proposed vASM scheme, which also solves the open problem in [14], requires fewer group operations and bilinear pairings than the ASM schemes proposed in [2]. The second one is ASMwSA (ASM with Subgroup Authentication) in which the subgroup of the signers is known before the protocol starts, and so the members participating in authentication is decided by the subgroup itself. The third one is ASMwCA (ASM with Combiner Authentication) which also provides a solution to the open problem in [14]. The ASMwSA and ASMwCA schemes also require fewer group operations and bilinear pairings than the schemes in [2]. Moreover, we give a method of consecutive and cumulative signing which eliminates the designated combiner in the case that the subgroup $\mathcal{S}$ is known before the signature generation. Further we discuss the aggregated versions of vASM, ASMwSA and ASMwCA schemes. The aggregated versions of our schemes, i.e. AvASM, AASMwSA and AASMwCA output aggregated signatures with size of a single group element, and require $n+1$ pairings for aggregated signature verification, in comparison with the partial aggregated AASM scheme proposed in [2] with the signature size of $n+1$ group elements and verification with $2n+1$ pairings.

The outline of the paper is as follows. In Section 2 we give a brief background information, including definitions of bilinear pairings, Computational co-DHP/ψ-co-DHP, Feldman's Verifiable Secret Sharing (VSS) protocol and BLS signature scheme. In Section 3 we summarize the ASM scheme given in [2]. Then we give our vASM scheme which is infact a modified BLS signature in Section 4. Moreover, in Section 5 we propose ASMwSA and ASMwCA schemes which are based on subgroup authentication instead of global authentication. In Section 6, we summarize the partial aggregated ASM (AASM) scheme given in [2], and discuss the aggregated versions of our proposed schemes. Finally, in Section 7, we compare our new schemes with ASM and AASM schemes in terms of the number of operations required in the phases of the schemes and costs of transmission, broadcasting and storage.

2 Background

In order to provide sufficient background information for readers, we give the definitions of notions which we mainly use throughout this paper. Namely, we give the definitions of bilinear pairings, Feldman's VSS protocol, and BLS signature scheme.

2.1 Bilinear Pairings

Let $\mathbb{G}_1, \mathbb{G}_2$ be two cyclic additive groups of prime order q and $\mathbb{G}_T$ be cyclic multiplicative groups with the same order.

Definition 2.1. A pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ which satisfies the bilinearity and non-degeneracy properties:

- Bilinearity: $e(A^\alpha, B^\beta) = e(A, B)^{\alpha\beta}$ for all $\alpha, \beta \in \mathbb{Z}$, $A \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$.
- Non-degeneracy: $e \neq 1$.

The definitions of underlying hard problems of the schemes in this paper, i.e. computational co-Diffie-Hellman and computational ψ-co-Diffie-Hellman problems are given below.

Definition 2.2 (Computational co-Diffie-Hellman Problem [2])**.** For groups $\mathbb{G}_1 = \langle g_1 \rangle$ and $\mathbb{G}_2 = \langle g_2 \rangle$ of prime order q, define $Adv^{co-CDH}_{\mathbb{G}_1,\mathbb{G}_2}$ of an adversary $\mathcal{A}$ as

$$Pr\left[y = g_1^{\alpha\beta} : (\alpha, \beta) \xleftarrow{\$} \mathbb{Z}_q^2, y \leftarrow \mathcal{A}(g_1^\alpha, g_1^\beta, g_2^\beta)\right],$$

where the probability is taken over the random choices of $\mathcal{A}$ and the random selection of (α, β). $\mathcal{A}$ (τ, ϵ)-breaks the co-CDH problem if it runs in time at most τ and has $Adv^{co-CDH}_{\mathbb{G}_1,\mathbb{G}_2} \geq \epsilon$. co-CDH is (τ, ϵ)-hard if no such adversary exists.

Definition 2.3 (Computational ψ-co-Diffie-Hellman Problem [2])**.** For groups $\mathbb{G}_1 = \langle g_1 \rangle$ and $\mathbb{G}_2 = \langle g_2 \rangle$ of prime order q, let $\mathcal{O}^\psi(.)$ be an oracle that on input $g_2^x \in \mathbb{G}_2$ returns $g_1^x \in \mathbb{G}_1$. Define $Adv^{\psi-co-CDH}_{\mathbb{G}_1,\mathbb{G}_2}$ of an adversary $\mathcal{A}$ as

$$Pr\left[y = g_1^{\alpha\beta} : (\alpha, \beta) \xleftarrow{\$} \mathbb{Z}_q^2, y \leftarrow \mathcal{A}^{\mathcal{O}^\psi(.)}(g_1^\alpha, g_1^\beta, g_2^\beta)\right],$$

where the probability is taken over the random choices of $\mathcal{A}$ and the random selection of (α, β). $\mathcal{A}$ (τ, ϵ)-breaks the ψ-co-CDH problem if it runs in time at most τ and has $Adv^{\psi-co-CDH}_{\mathbb{G}_1,\mathbb{G}_2} \geq \epsilon$. ψ-co-CDH is (τ, ϵ)-hard if no such adversary exists.

2.2 Feldman's VSS Protocol

Feldman's verifiable secret sharing (VSS) scheme [9] is a protocol which is used for sharing a secret among some predetermined players in a verifiable fashion, where Shamir's secret sharing scheme [21] was directly used to share and reconstruct the secret. In addition to Shamir's scheme, the shares can be checked for consistency in Feldman's scheme. To this end, dealer computes commitments with the coefficients of the secret polynomial. By this way, users can verify that they receive the consistent shares from the dealer.

Assume that we have n players. Let $\mathbb{F}_q$ be a finite field with prime order q and g be a primitive element in $\mathbb{F}_q$. The dealer shares a secret as follows:

- Chooses a polynomial of degree $t-1$ ($< q$),

$$f(x) = \alpha_{t-1}x^{t-1} + \ldots + \alpha_1 x + \alpha_0$$

  with distinct and nonzero $\alpha_k \in \mathbb{F}_q^*$ for $k = 0, \ldots, t-1$, where α_0 is the secret to be shared.

- Computes a set of commitments $\text{COM} = \{C_k : C_k = g^{\alpha_k}, k = 0, 1, \ldots, t-1\}$.

- Sends $f(i)$ and COM to the i-th player for $i = 1, 2, \ldots, n$.

After receiving a share and the set of commitments, the i-th player checks

$$g^{f(i)} \stackrel{?}{=} \prod_{k=0}^{t-1} C_k^{i^k}. \tag{2.1}$$

The received share is consistent with the shared secret only if (2.1) is satisfied. If at least any t or more players perform Lagrange interpolation with their shares, they can uniquely determine the secret polynomial and $f(0)$ will yield the secret.

In Section 4 we use this protocol as an implicit authentication and the proof of possession method. We use only the sharing, committing and verifying phases of this protocol.

2.3 BLS Signature Scheme

Let e be an efficient, non-degenerate bilinear map, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$, in groups ($\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_t$) and (g_1, g_2) are generators of the group pair $(\mathbb{G}_1, \mathbb{G}_2)$ respectively. Let $H : \{0,1\}^* \to \mathbb{G}_1$ be a function which maps any arbitrary binary string onto the group $\mathbb{G}_1$. BLS signature scheme has three phases, which we give below shortly.

1. Key Generation:

   Pick a random secret key $sk \xleftarrow{\$} \mathbb{Z}_q$, and compute the public key $pk \leftarrow g_2^{sk}$.

2. Signature Generation:

   Compute the signature $\sigma = H(m)^{sk}$, where m is the message.

3. Verification:

   Accept if and only if $e(H(m), pk) = e(\sigma, g_2)$ holds.

The BLS signature was proved to be secure against existential forgery under adaptive chosen message attacks in the random oracle model in [4]. But it is not safe to use it as a multi-signature scheme directly, because of the "rogue-key" attacks [1, 2]. In order to avoid these attacks, there are some standard defenses, such as either using *proof-of-possession (PoP)* or ensuring that the messages are distinct. Both methods have some advantages and disadvantages. Signing distinct messages hinders users to perform efficient verification [2], and using PoP is not fully compatible with the applications in cryptocurrencies [13]. In order to eliminate these disadvantages, Boneh, Drijvers and Neven proposed in [2] a multi-signature scheme, called MSP, which is indeed a modified version of BLS scheme. Moreover, they proposed an accountable subgroup multi-signatures (ASM) scheme which is a composition of the schemes BLS and MSP.

3 Boneh-Drijvers-Neven ASM Scheme

Throughout this section we follow the notation given in both [2] and the previous sections. Let $\mathcal{PK} := \{pk_1, \cdots, pk_n\}$ be the set of public keys of the group members in $\mathcal{G}$, and let $H_0, H_2 : \{0,1\}^* \to \mathbb{G}_1$ and $H_1 : \{0,1\}^* \to \mathbb{Z}_q$ be the hash functions. The BLS based ASM scheme given in [2] can be stated as follows.

1. Key Generation: Each user $i \in \mathcal{G}$ picks a secret key $sk_i \xleftarrow{\$} \mathbb{Z}_q$, and computes the corresponding public key $pk_i \leftarrow g_2^{sk_i}$, where g_2 is a generator of $\mathbb{G}_2$.

2. Group Setup: Each member $i \in \mathcal{G}$ performs group setup by participating in 1-round interactive protocol for $i = 1, 2, \ldots, n$.

   - Computes aggregated public key apk of the group as $apk = \prod_{i=1}^{n} pk_i^{a_i}$, where $a_i = H_1(pk_i, \mathcal{PK})$.
   - Sends $\mu_{ij} = H_2(apk, j)^{a_i sk_i}$ to j-th user for $j = 1, 2, \ldots, n$ and $j \neq i$.
   - After receiving μ_{ji}, computes $\mu_{ii} = H_2(apk, i)^{a_i sk_i}$.
   - The membership key of user i is $mk_i = \prod_{j=1}^{n} \mu_{ji}$.

3. Signature Generation: A signer $i \in \mathcal{G}$ computes his/her individual signature on the message m

$$s_i = H_0(apk, m)^{sk_i} \cdot mk_i \tag{3.1}$$

   and sends s_i to the combiner.

4. Signature Aggregation: After receiving the individual signatures of the signers, the combiner first forms the set of signers $\mathcal{S} \subseteq \mathcal{G}$. Then, she computes the aggregated subgroup multi-signature $\sigma = (s, pk)$, where $s = \prod_{i \in \mathcal{S}} s_i$ and $pk = \prod_{i \in \mathcal{S}} pk_i$.

5. Verification: Any verifier who is given $\{par, apk, \mathcal{S}, m, \sigma\}$ can verify the signature $\sigma = (s, pk)$ by checking

$$e\Big(H_0(apk, m), pk\Big) \cdot e\Big(\prod_{j \in \mathcal{S}} H_2(apk, j), apk\Big) \stackrel{?}{=} e(s, g_2). \tag{3.2}$$

ASM scheme in [2] is nothing but a composition of a BLS signature and a group-specific membership key mk_i of the signer $i \in \mathcal{G}$. Namely, the first part $H_0(apk, m)^{sk_i}$ of the signature equation (3.1) is a BLS signature on (apk, m) by $|\mathcal{S}|$ signers, on the other hand the second part mk_i is a MSP signature on (apk, i) by all the members $j \in \mathcal{G}$ for $i = 1, 2, \ldots, n$.

The *proof of possession* (PoP) of the secret keys is also discussed in [2]. The ASM scheme with PoP includes each user's signature on their own public keys. The i-th user first chooses a secret key $sk_i \in \mathbb{Z}_q$, then computes $y_i = g_2^{sk_i}$, and constructs the PoP by $\pi_i = H_3(y_i)^{sk_i}$, where $H_3 : \{0,1\}^* \to \mathbb{G}_1$ for $i = 1, 2, \ldots, n$. Then each user has a secret key sk_i and the public key pair (y_i, π_i). In order to compute the aggregated public key (apk) of the group, they first check $e(H_3(y_i), y_i) \stackrel{?}{=} e(\pi_i, g_2)$, then they compute $Y = \prod_{i \in \mathcal{G}} pk_i$ and $h = H_4(\mathcal{PK})$, where $H_4 : \{0,1\}^* \to \mathbb{Z}_q$ is another hash function. And then the aggregated public key is $apk = (Y, h)$. The signature generation, aggregation and verification phases are same with the original ASM scheme.

It is known that using PoP brings additional costs such as the growth in public key size and extra checks. In the PoP variant of ASM scheme [2], each user's public key consists of 2 group elements and each user computes extra 2 pairings before computing apk .

4 vASM: An ASM scheme with VSS based group setup

In this section, we set a special signing key and its public companion for a multi-signature. First of all, each user generates his secret and public key pair independently. Then all users jointly perform a group setup in which they participate in a VSS protocol. At the end of this procedure, each user obtains his membership key, which satisfies a common public commitment generated in the group setup phase. We give the steps of the vASM scheme below.

1. Key Generation: Each user $i \in \mathcal{G}$ picks a secret key $sk_i \xleftarrow{\$} \mathbb{Z}_q$, and computes the public key $pk_i \leftarrow g_2^{sk_i}$, where g_2 is a generator of $\mathbb{G}_2$.

2. Group Setup: Each user $i \in \mathcal{G}$ proceeds as follows:

   - Chooses a polynomial $f_i(x) = \alpha_{n-1}^{(i)} x^{n-1} + \ldots + \alpha_1^{(i)} x + \alpha_0^{(i)} \in \mathbb{Z}_q[x]$, where $\alpha_0^{(i)} = sk_i$ and $\alpha_k^{(i)}$'s are all nonzero and distinct, for $k = 1, \ldots, n-1$.
   - Computes the set of commitments $\text{COM}_i := \{C_k^{(i)} = g_2^{\alpha_k^{(i)}} | k = 0, \ldots, n-1\}$.
   - Sends $(f_i(j), \text{COM}_i)$ to j-th user in $\mathcal{G}$, for $j = 1, \ldots, n$.
   - After receiving $(f_j(i), \text{COM}_j)$,
     - computes the membership key $mk_i = \sum_{j \in \mathcal{G}} f_j(i)$.
     - computes $\text{COM} := \{C_k = \prod_{j \in \mathcal{G}} C_k^{(j)} | k = 0, \ldots, n-1\}$.
   - Checks:

     (a) $C_0 \stackrel{?}{=} \prod_{i \in \mathcal{G}} pk_i$

     (b) $g_2^{mk_i} \stackrel{?}{=} \prod_{k=0}^{n-1} C_k^{i^k}$
   - If either (a) or (b) fails, then she aborts. Else, she makes COM public.

3. Signature Generation: A signer $i \in \mathcal{G}$ computes his/her individual signature $s_i = H_0(m)^{mk_i}$ on the message m and sends s_i to the combiner.

4. Signature Aggregation: After receiving the individual signatures of the signers, the combiner first forms the set of signers $\mathcal{S} \subseteq \mathcal{G}$. Then, she computes the aggregated subgroup multi-signature $\sigma = \prod_{i \in \mathcal{S}} s_i$.

5. Verification: Anyone, who is given $\{par, \text{COM}, \mathcal{S}, m, \sigma\}$, can verify the signature σ by checking

$$e\big(H_0(m), \prod_{k=0}^{n-1} C_k^{\sum_{i \in \mathcal{S}} i^k}\big) \stackrel{?}{=} e(\sigma, g_2).$$

4.1 Remarks on vASM

1. Unlike the threshold multi-signatures [10], ASM schemes [2, 14] provide accountability. Further, in ASM schemes any subgroup $\mathcal{S} \subseteq \mathcal{G}$ can sign a message on behalf of the whole group $\mathcal{G}$, whereas in threshold schemes only subgroups with a sufficient cardinality can sign. Moreover, one can easily transform an ASM scheme into a threshold scheme by setting the threshold as $|\mathcal{S}|$ [14].

2. Since the membership key mk_i of each group member consists of the shares $f_j(i)$ of the secret key of all group members for $i, j = 1, 2, \ldots, n$, the signature σ authenticates the subgroup $\mathcal{S} \subseteq \mathcal{G}$ and shows that each member of $\mathcal{S}$ is authenticated by the other members of $\mathcal{G}$. Hence, the signature is created by $\mathcal{S}$ on behalf of the whole group $\mathcal{G}$.

3. Notice that $C_0 \stackrel{?}{=} \prod_{i \in \mathcal{G}} pk_i$ can be satisfied only if the users know their secret keys. Therefore the consistency checks (a) and (b) in the group setup phase provide proof of possession for each user and force all users to be honest. Hence, in vASM scheme, no user can set a special rogue key.

4. If the members in the group $\mathcal{G}$ change, the group setup phase must be reset with new random polynomials f_i for $i \in \mathcal{G}$. Otherwise, any n corrupted users can obtain any users secret key since the secret polynomials are of degree $n-1$ (see Section 2.2). In order to avoid this vulnerability, the polynomials f_i in the group setup phase of the vASM scheme can be set of degree $r \geq n-1$. In this way, at most $r-n+1$ new comers can be registered to the group $\mathcal{G}$ without resetting the group setup phase. On the other hand, this costs extra computational complexity at the group setup phase.

5. The membership key mk_i in vASM scheme is used to sign the message m instead of the secret key sk_i by each member $i \in \mathcal{G}$ so that one can easily check the accountability of the signer at the verification step. For example, consider the case that Bob has two distinct identities, i.e. his individual identity "*Bob*", and his corporate identity "*CFO of Company X*". Assume that sk_B and mk_B are Bob's secret and the membership keys, respectively. In this case, Bob uses sk_B for spending his own money, besides he signs by mk_B for spending on behalf of the Company X. As this example shows, user i uses his secret key sk_i to sign messages and to participate in any multi-signatures, on the other hand, he participates in the vASM scheme with his membership key mk_i.

4.2 Security of vASM

An accountable subgroup multi-signature (ASM) scheme is defined to be secure if no legitimate signature on any message makes an honest user accountable without participating in signature generation.

For vASM scheme, we assume a dishonest majority, which means that an adversary can corrupt at most $(n-1)$ users in a group of n users at any phase of the scheme. In order to conduct the rogue-key attack, he can choose the corrupted users' keys as he wants.

We use Boldyreva's reduction method [1]. We will show that the security of the proposed vASM scheme given in Section 4 depends on the security of BLS scheme [4].

Theorem 4.1. *Our vASM scheme is a secure accountable multi-signature scheme in the random oracle model.*

Proof. Let $\mathcal{A}$ be a PPT adversary with the advantage $Adv^{vASM}_{par,(\mathbb{G}_1,\mathbb{G}_2)}$, that is the probability of $\mathcal{A}$ to output a legitimate $(m, \sigma, \mathcal{S})$ triple. We construct a PPT adversary $\mathcal{B}$ against BLS signature scheme with the $Adv^{BLS}_{par,(\mathbb{G}_1,\mathbb{G}_2)}$, i.e. the probability of $\mathcal{B}$ to output a valid (m, σ) pair. Moreover $\mathcal{B}$ has access to the hash oracle and the signing oracle. Therefore, it suffices show that $Adv^{vASM}_{par,(\mathbb{G}_1,\mathbb{G}_2)} = Adv^{BLS}_{par,(\mathbb{G}_1,\mathbb{G}_2)}$, which is equivalent to saying if BLS signature scheme is secure then our vASM scheme is secure.

Without loss of generality, we say that the honest user's indice is 1. Adversary $\mathcal{B}$ sends $g_2^{mk_1}$ to the adversary $\mathcal{A}$. Then $\mathcal{A}$ outputs $(n-1)$ pairs of $(mk_j, g_2^{mk_j})$ for $j = 2, \ldots, n$. Adversary $\mathcal{A}$ starts to send the hash and signing queries. Adversary $\mathcal{B}$ answers all queries of adversary $\mathcal{A}$. Eventually $\mathcal{A}$ outputs a valid signature σ_L on the message m with the subgroup $\mathcal{S}$ and $1 \in L$, where L is the set of indices of the

users in the subgroup $\mathcal{S}$. Then adversary $\mathcal{B}$ computes

$$\begin{aligned}\sigma &= \sigma_L . \prod_{j \in L \setminus 1} H_0(m)^{-mk_j} \\ &= H_0(m)^{mk_1}.\end{aligned}$$

It is clear that the signature σ is nothing but a BLS signature on the message m. □

5 Accountable Subgroup Multi-signature Scenarios with Subgroup Authentication

In some cases, a signer $i \in \mathcal{S}$ wants to know other signers $\mathcal{S} \subseteq \mathcal{G}$ in advance. In the ASM scheme in Section 3, any subgroup $\mathcal{S} \subseteq \mathcal{G}$ of signers are authorized to sign any message on behalf of the whole group. Consider that 2 subgroups make 2 opposite decisions. Since either of the subgroups signs on behalf of the entire group $\mathcal{G}$, this causes a conflict. In order to avoid such a case, the legal entities could presume a unique authorized signer (CEO, CFO etc.) in $\mathcal{S}$.

In this section, we consider to replace sk_i with $a_i sk_i$ and mk_i with smk_i in the equation (3.1) for an identifier a_i and a subgroup specific membership key smk_i of the signer $i \in \mathcal{S} \subseteq \mathcal{G}$. Then, we can combine two pairings on the left hand side of the verification equation (3.2). In the following we describe two scenarios.

5.1 ASMwSA: Accountable Subgroup Multi-signature with Subgroup Authentication

In ASMwSA, we consider the case that the subgroup $\mathcal{S}$ is known before the protocol starts. We discard the interactive protocol in the group setup phase and make simple modifications on the ASM scheme given in Section 3. The ASMwSA is as follows:

1. Key Generation: Identical to the Key Generation in Section 3.

2. Group Setup: Each group member $i \in \mathcal{G}$ computes

   - Aggregated public key $apk = \prod_{i \in \mathcal{G}} pk_i^{a_i}$, where $a_i = H_1(pk_i, \mathcal{PK})$.
   - Components of the membership keys $\mu_{ij} = H_2(apk, j)^{a_i sk_i}$ for $j = 1, \ldots, n$, and stores them.

3. Signature Generation: A signer $i \in \mathcal{G}$ computes his/her subgroup-specific membership key $smk_i = \prod_{j \in \mathcal{S}} \mu_{ij}$ and individual signature $s_i = H_0(apk, m)^{a_i sk_i} smk_i$ on the message m and sends s_i to the combiner.

4. Signature Aggregation: After receiving the individual signatures, the combiner computes the aggregated subgroup multi-signature $\sigma = \prod_{i \in \mathcal{S}} s_i$ and $spk = \prod_{i \in \mathcal{S}} pk_i^{a_i}$.

5. Verification: Anyone who is given $\{par, apk, spk, \mathcal{S}, m, \sigma\}$ can verify the signature σ by checking

$$e\left(H_0(apk, m) . \prod_{j \in \mathcal{S}} H_2(apk, j), spk\right) \stackrel{?}{=} e(\sigma, g_2).$$

5.1.1 Remarks on ASMwSA

1. In the group setup phase, the signers only compute μ_{ij} and store them, but they do not send those μ_{ij} to anyone. In ASMwSA, the i-th signer multiplies her signature $s_i = H_0(apk, m)^{a_i sk_i}$ with $smk_i = \prod_{j \in \mathcal{S}} \mu_{ij}$, instead of mk_i as in Section 3, which also results in a legitimate aggregated signature. We note that this eliminates 1 round transmission cost.

2. The signature of the subgroup can be written as below

$$\begin{aligned}\sigma &= H_0(apk, m)^{\sum_{i \in \mathcal{S}} a_i sk_i} \prod_{i \in \mathcal{S}} smk_i \\ &= \left(H_0(apk, m) \cdot \prod_{j \in \mathcal{S}} H_2(apk, j) \right)^{\sum_{i \in \mathcal{S}} a_i sk_i}\end{aligned}$$

   which is indeed a MSP signature on $\left(H_0(apk, m) \cdot \prod_{j \in \mathcal{S}} H_2(apk, j) \right)$. Therefore, the security of the ASMwSA follows from the security of MSP scheme.

5.2 ASMwCA: Accountable Subgroup Multi-signature with Combiner Authentication

In ASMwCA, each user $i \in \mathcal{G}$ sends the membership key components, i.e. μ_{ij} for $j \neq i$, to the combiner. Hence, the signers perform fewer computation, and main work load passes to the combiner. The ASMwCA is as follows:

1. Key Generation: Identical to the Key Generation in Section 3.

2. Group Setup: Each group member $i \in \mathcal{G}$ computes:

   - The aggregated public key apk of $\mathcal{G}$ as $apk = \prod_{i \in \mathcal{G}} pk_i^{a_i}$, where $a_i = H_1(pk_i, \mathcal{PK})$.
   - $\mu_{ii} = H_2(apk, i)^{a_i sk_i}$ and stores it.
   - $\mu_{ij} = H_2(apk, j)^{a_i sk_i}$ for $j = 1, \ldots, n$ and $j \neq i$, and sends them to the combiner.

3. Signature Generation: A signer $i \in \mathcal{G}$ computes individual signature $s_i = H_0(apk, m)^{a_i sk_i} \mu_{ii}$ on the message m and sends s_i to the combiner.

4. Signature Aggregation: After receiving the individual signatures of the signers, the combiner first forms $\mathcal{S} \subseteq \mathcal{G}$. Then, she computes the aggregated subgroup multi-signature $\sigma = \prod_{i \in \mathcal{S}} s_i . \prod_{i \in \mathcal{S}} \prod_{j \in \mathcal{S}} \mu_{ij}$ for $j \neq i$, and aggregated public key spk of the subgroup $\mathcal{S}$ as $spk = \prod_{i \in \mathcal{S}} pk_i^{a_i}$.

5. Verification: Anyone who is given $\{par, apk, spk, \mathcal{S}, m, \sigma\}$ can verify the signature σ by checking

$$e\left(H_0(apk, m) . \prod_{j \in \mathcal{S}} H_2(apk, j), spk \right) \stackrel{?}{=} e(\sigma, g_2).$$

5.2.1 Remarks on ASMwCA

1. The subgroup $\mathcal{S} \in \mathcal{G}$ of the signers is determined by the combiner from the set of the received individual signatures. Hence, no signer knows her co-signers.

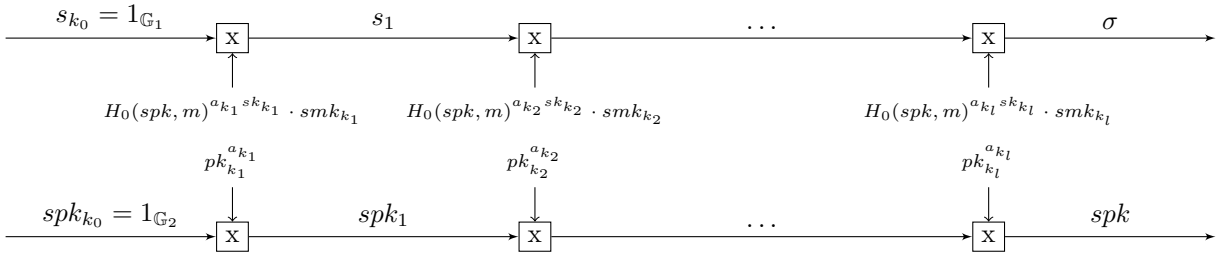

Figure 1: ASM scheme without a combiner

2. In the signature generation phase, $H_0(apk, m)$ may be replaced by $H_0(spk, m)$ so that the signer could set the subgroup $\mathcal{S}$ in advance. This also eliminates combiner's corruption at the signature aggregation phase. Namely, the combiner can not discard the signature s_i of any user $i \in \mathcal{S}$ from σ. But in this case computing spk brings additional cost, i.e. l multi-exponentiations for each signer.

3. Each user $i \in \mathcal{G}$ sends μ_{ij} to the combiner for $i, j = 1, 2, \ldots, n$ and $j \neq i$. Unlike to scenarios in Sections 3 and 5.1, any signer $i \in \mathcal{G}$ does not compute the membership key, but uses only μ_{ii} for the signature generation. The other components, μ_{ij} for $i, j = 1, 2, \ldots, n$ and $j \neq i$, are taken into account by the combiner as she forms the subgroup $\mathcal{S} \subseteq \mathcal{G}$. Therefore, each user's computational cost is reduced, but the work load of the combiner is increased by the number of signers.

4. We note that the user $i \in \mathcal{G}$ may compute her individual signature as

$$s_i = \Big(H_0(apk, m) H_2(apk, i) \Big)^{a_i sk_i}$$

instead of computing and storing μ_{ii} in the group setup phase. However, this costs extra computation of the hash H_2 at the signature generation of each message.

5. Similar to the discussion in Subsection 5.1.1, the security of the ASMwCA can be reduced to the security of MSP scheme.

5.3 Eliminating the combiner

Consider the case that the subgroup $\mathcal{S} := \{k_i : i = 1, 2, \ldots, l\}$ is determined before the signature protocol starts. In order to eliminate the designated combiner, the signer k_i for $i = 1, 2, \ldots, l$ proceeds as follows:

- Computes the i-th aggregated signature $s_{k_i} = s_{k_{i-1}} \cdot \big(H_0(spk, m)^{a_{k_i} sk_{k_i}} \cdot smk_{k_i} \big)$, where $s_{k_{i-1}}$ is the $(i-1)$-th aggregated signature computed by the signer $k_{i-1} \in \mathcal{S}$ and $s_{k_0} = 1_{\mathbb{G}_1}$.
- Computes the i-th aggregated public key $spk_{k_i} = spk_{k_{i-1}} \cdot pk_{k_i}^{a_{k_i}}$, where $spk_{k_{i-1}}$ is the $(i-1)$-th aggregated public key computed by the signer $k_{i-1} \in \mathcal{S}$ and $spk_{k_0} = 1_{\mathbb{G}_2}$.
- Sends (s_{k_i}, spk_{k_i}) to the signer k_{i+1} for $i < l$.
- Finally the last signer k_l outputs the pair s_{k_l}, spk_{k_l}. Then, the aggregated signature and the public key pair for $\mathcal{S}$ will be $\sigma := s_{k_l}$ and $spk := spk_{k_l}$. This process is shown in Figure 1.

In ASMwSA and ASMwCA, each signer sends her individual signature to the combiner. On the other hand, in this scenario, each signer sends the signature to another signer. Hence, this reduces the cost of network traffic and makes the channel traffic intermittent. However, in this case, each user sends two group elements instead of one group element, that is, the transmission size is doubled.

5.4 Advantages of subgroup-specific membership key

1. Group-specific membership keys in Boneh-Drijvers-Neven ASM scheme given in Section 3 are generated by 1 round of interactive protocol in which all members participate. But in case of using subgroup-specific ones given in Sections 5.1 and 5.2, no interactive protocol is needed.

2. Even if one user registers/unregisters to/from the group, the apk and the membership key mk need to be recomputed. This means a new interactive protocol to be held by all the group members again, which is a big deal for large $|\mathcal{G}|$. Consider using $H_2(pk_j)$ instead of $H_2(apk, j)$ for computing the components of membership keys. In the case of using subgroup-specific membership key smk, registering/unregistering a member to/from the group $\mathcal{G}$ does not change anything.

3. Users in $\mathcal{S} \subseteq \mathcal{G}$ do group operations to get smk; but all group members in $\mathcal{G}$ need to do group operations to get mk. If $|\mathcal{S}|$ is very small with respect to $|\mathcal{G}|$, then the computation of smk is easier. Similarly, spk can be calculated by less number of multi exponentiation than apk.

6 Aggregated versions of ASM schemes

The ASM scheme is extended to partial aggregated version AASM scheme in [2] by the help of the key aggregation technique given in [13]. Remember that the ASM signature scheme described in Section 3 outputs a signature $\sigma = (s, pk)$. Consider n-ASM signatures $(apk_1, \mathcal{S}_1, m_1, \sigma_1), \ldots, (apk_n, \mathcal{S}_n, m_n, \sigma_n)$. The AASM scheme outputs the partial aggregated signature $\Sigma = (pk_1, \ldots, pk_n, s)$, where s is the aggregation of $s_1, \ldots, s_n$. In Section 6.1, we describe the AASM scheme that is given in [2].

6.1 Partially Aggregated ASM Scheme (AASM)

The aggregation of several ASM signatures is discussed in [2], in which they aggregate only the second component of the signature. The first components are used in the verification phase. On the other hand, it is noted that the ASM scheme cannot be partially aggregated directly because of the irrelevancy between membership keys and messages [2]. In addition to the phases of ASM scheme, two more phases defined in [2]. The signature aggregation and aggregated signature verification phases are as follows. Let $H_5 : \{0,1\}^* \to \mathbb{Z}_q$ be another hash function.

- Signature Aggregation: Given $(par, \{(apk_i, \mathcal{S}_i, m_i, \sigma_i)_{i=1}^n\})$.
  - Parse σ_i as (s_i, pk_i).
  - For $i = 1, \ldots, n$, compute $b_i \leftarrow H_5\big((apk_i, \mathcal{S}_i, m_i, pk_i), \{(apk_j, \mathcal{S}_j, m_j, pk_j)_{j=1}^n\}\big)$.
  - compute $s \leftarrow \prod_{i=1}^{n} s_i^{b_i}$ and output $\Sigma \leftarrow (pk_1, \ldots, pk_n, s)$.
- Aggregate Signature Verification: Given $(par, \{(apk_i, \mathcal{S}_i, m_i)_{i=1}^n\}, \Sigma)$.
  - Parse Σ as $(pk_1, \ldots, pk_n, s)$.
  - For $i = 1, \ldots, n$, compute $b_i \leftarrow H_5\big((apk_i, \mathcal{S}_i, m_i, pk_i), \{(apk_j, \mathcal{S}_j, m_j, pk_j)_{j=1}^n\}\big)$.
  - Accept if and only if

$$\prod_{i=1}^{n} \Big(e\big(H_0(apk_i, m_i), pk_i^{b_i}\big) \cdot e\big(\prod_{j \in \mathcal{S}_i} H_2(apk_i, j), apk_i^{b_i}\big) \Big) \stackrel{?}{=} e(s, g_2). \tag{6.1}$$

Since the AASM scheme cannot be fully aggregated, the size of the resulting partial aggregated signature grows linearly by the number of ASM signatures. Besides, the verification (6.1) of the AASM scheme requires $(2n+1)$ pairings.

6.2 Aggregated versions of proposed schemes

Our schemes can also be further extended as aggregated versions. Unlike the partial aggregation in AASM scheme, the schemes that we propose in Sections 4 and 5 can be fully aggregated, resulting in a single signature size.

6.2.1 Aggregated vASM Scheme (AvASM)

The vASM scheme described in Section 4 can be extended to aggregated version AvASM as in Section 6.1.

- Signature Aggregation: Given $(par, \{(\mathrm{COM}_i, \mathcal{S}_i, m_i, \sigma_i)_{i=1}^n\})$, compute $\Sigma \leftarrow \prod_{i=1}^{n} \sigma_i$.
- Aggregate Signature Verification: Given $(par, \{(\mathrm{COM}_i, \mathcal{S}_i, m_i)_{i=1}^n\}, \Sigma)$. Accept if and only if

$$\prod_{i=1}^{n} e\left(H_0(m_i), \prod_{k=0}^{|COM_i|} C_{i_k}^{\sum\limits_{j\in\mathcal{S}_i} j^k}\right) = e(\Sigma, g_2). \tag{6.2}$$

  where C_{i_k} is a commitment in COM_i for $k = 0, \ldots, |\mathrm{COM}_i|$.

6.2.2 Aggregated versions of ASMwSA/ASMwCA Schemes (AASMwSA/AASMwCA)

The ASMwSA and ASMwCA schemes described in Section 5 can be extended to aggregated versions AASMwSA and AASMwCA as in Section 6.1. In addition to the phases of the described schemes in Section 5, signature aggregation and aggregate signature verification phases are given below:

- Signature Aggregation: Given $(par, \{(apk_i, spk_i, \mathcal{S}_i, m_i, \sigma_i)_{i=1}^n\})$, compute $\Sigma \leftarrow \prod_{i=1}^{n} \sigma_i$.
- Aggregate Signature Verification: Given $(par, \{(apk_i, spk_i, \mathcal{S}_i, m_i)_{i=1}^n\}, \Sigma)$, accept if and only if

$$\prod_{i=1}^{n} e\left(H_0(apk_i, m_i).\prod_{j\in\mathcal{S}_i} H_2(apk_i, j), spk_i\right) = e(\Sigma, g_2). \tag{6.3}$$

As seen in (6.2) and (6.3), the aggregated versions of the schemes vASM, ASMwSA and ASMwCA provide more practical verifications than AASM [2]. In other words, AvASM, AASMwSA and AASMwCA schemes require $(n+1)$ pairings for verification whereas AASM requires $(2n+1)$.

7 Comparison

In Table 1, we compare ASM, ASM-PoP and the schemes that we propose in this paper. Our comparison contains the number of operations required in each phases of those schemes. For example, consider a comparison of group setup phases of the schemes ASM and vASM. It is seen in Table 1 that the ASM scheme costs of n H_2 hashes , n scalar multiplications and $(n-1)$ group operations in the group setup phase. On the other hand, the vASM scheme has $2n$ scalar multiplications and (n^2+n-2) group operations in the group setup.

It is easily seen from the Table 1 that the vASM scheme requires fewer operations than ASM and ASM-PoP in the verification phase. On the other hand, since the set of commitments COM is public, the broadcasted data size in vASM grows with the size of the group $\mathcal{G}$.

The ASMwSA scheme provides efficient group setup, signature aggregation and verification phases; however, its signature generation requires more group operations and extra storage for $(n-1)$ group elements.

The ASMwCA scheme has efficient group setup and verification phases, but it requires more computations in signature aggregation phase. On the other hand, it requires extra storage for (n^2-n-1) group elements.

8 Conclusion

In this work we propose a novel BLS based ASM scheme (vASM) which is more efficient than ones in [2] in terms of signature generation, signature aggregation and verification. On the other hand, our vASM scheme requires a one-time group setup with more group operations. We propose two more accountable subgroup multi-signature schemes with subgroup authentication (ASMwSA) and combiner authentication (ASMwCA) which are also more efficient, but they have storage and transmission disadvantages in comparison with the ones in [2]. According to the requirements of the system involving an ASM scheme, our schemes could be pretty good alternatives, especially when a faster verification is desired. It would be a good future work to add accountability to known multi-signature schemes by the methods given in this paper.

References

[1] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. pages 31–46, 01 2003.

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

[3] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *Advances in Cryptology — EUROCRYPT 2003*, pages 416–432, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[4] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *J. Cryptol.*, 17(4):297–319, September 2004.

[5] Vitalik Buterin. Ethereum: A next-generation smart contract and decentralized application platform. `https://github.com/ethereum/wiki/wiki/White-Paper`, 2013.

[6] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups. In Burton S. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, pages 410–424, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[7] David Chaum and Eugène van Heyst. Group signatures. In *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April*

Table 1: Comparison of the schemes

Phases	ASM [2]	ASM with PoP [2]	vASM	ASMwSA	ASMwCA
Key Generation	1 S.Mul.	For pk: 1 S.Mul. For proof: 1 Hash (H_3) 1 S.Mul.	1 S.Mul.	1 S.Mul.	1 S.Mul.
Key Aggregation	n Hash (H_1) n S.Mul. $(n-1)$ G.Ops.	For proof: n Hash (H_3) 2n pairings For apk: 1 Hash (H_4) $(n-1)$ G.Ops.		n Hash (H_1) n S.Mul. $(n-1)$ G.Ops.	n Hash (H_1) n S.Mul. $(n-1)$ G.Ops.
Group Setup	n Hash (H_2) n S.Mul. $(n-1)$ G.Ops.	n Hash (H_2) n S.Mul. $(n-1)$ G.Ops.	$2n$ S.Mul. (n^2+n-2) G.Ops.	n Hash (H_2) n S.Mul.	n Hash (H_2) n S.Mul.
Signature Generation	1 Hash (H_0) 1 S.Mul. 1 G.Ops.	1 Hash (H_0) 1 S.Mul. 1 G.Ops.	1 Hash (H_0) 1 S.Mul.	1 Hash (H_0) 1 S.Mul. l G.Ops.	1 Hash (H_0) 1 S.Mul. 1 G.Ops.
Signature Aggregation	$(2l-2)$ G.Ops.	$(2l-2)$ G.Ops.	$(l-1)$ G.Ops.	$(l-1)$ G.Ops.	(l^2-l) G.Ops.
Verification	For $apk^{(c)}$: n Hash (H_1) n S.Mul. $(n-1)$ G.Ops. For verification: 1 Hash (H_0) l Hash (H_2) $(l-1)$ G.Ops. 3 Pairings	For proof: n Hash (H_3) n S.Mul. $2n$ pairings For $apk^{(c)}$: 1 Hash (H_4) $(n-1)$ G.Ops. For verification: 1 Hash (H_0) l Hash (H_2) $(l-1)$ G.Ops. 3 Pairings	1 Hash (H_0) n S.Mul. $(n-1)$ G.Ops. 2 Pairings	For $apk/spk^{(a)}$: n Hash (H_1) n S.Mul. $(n-1)$ G.Ops. For verification: 1 Hash H_0 l Hash (H_2) l G.Ops. 2 Pairings	For $apk/spk^{(a)}$: n Hash (H_1) n S.Mul. $(n-1)$ G.Ops. For verification: 1 Hash (H_0) l Hash (H_2) l G.Ops. 2 Pairings
Transmission	For group setup: $(n-1)$ G.Elt. For signature: 2 G.Elt.	For group setup: $(n-1)$ G.Elt. For signature: 2 G.Elt.	For signature: 1 G.Elt.	For signature: 1 G.Elt.	For group setup: $(n-1)$ G.Elt. For signature: 1 G.Elt.
Broadcasting	For pk: 1 G.Elt.	For pk: 1 G.Elt. For Proof: 1 G.Elt.	For pk: 1 G.Elt. For COM_i: n G.Elt. For COM$^{(b)}$: n G.Elt.	For pk: 1 G.Elt.	For pk: 1 G.Elt.
Storage	For sk: 1 integer For mk: 1 G.Elt.	For sk: 1 integer For mk: 1 G.Elt.	For mk: 1 integer	For sk: 1 integer For mk: n G.Elt.	For sk: 1 integer For mk: 1 G.Elt. Combiner: (n^2-n) G.Elt.

(a) As apk contains the components of spk, no extra cost is needed for computing spk.

(b) Since all the users compute the same commitment set COM, it is enough to be broadcast by only one user.

(c) apk is computed by the verifier. But for avoiding the extra cost, it would be given to the combiner.

S.Mul. Scalar Multiplication

G.Ops. Group Operations

G.Elt. Group element

H_1, H_2, H_3, H_4, H_5 Hash functions are as in the schemes.

8-11, 1991, Proceedings, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer, 1991.

[8] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 307–315, New York, NY, 1990. Springer New York.

[9] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 427–438, Oct 1987.

[10] Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Robust threshold dss signatures. In Ueli Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, pages 354–371, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[11] L. Harn. Group-oriented (t, n) threshold digital signature scheme and digital multisignature. *IEE Proceedings - Computers and Digital Techniques*, 141:307–313(6), September 1994.

[12] K. Itakura. A public-key cryptosystem suitable for digital multisignatures. 1983.

[13] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple schnorr multisignatures with applications to bitcoin. *Designs, Codes and Cryptography*, 87, 09 2019.

[14] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. Accountable-subgroup multisignatures: Extended abstract. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, CCS '01, page 245–254, New York, NY, USA, 2001. Association for Computing Machinery.

[15] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009.

[16] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, and Steven Goldfeder. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, USA, 2016.

[17] K. Ohta and T. Okamoto. Multi-signature schemes secure against active insider attacks (special section on cryptography and information security). *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 82:21–31, 1999.

[18] Kazuo Ohta and Tatsuaki Okamoto. A digital multisignature scheme based on the fiat-shamir scheme. In Hideki Imai, Ronald L. Rivest, and Tsutomu Matsumoto, editors, *Advances in Cryptology — ASIACRYPT '91*, pages 139–148, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[19] David Pointcheval and Jacques Stern. Provably secure blind signature schemes. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology — ASIACRYPT '96*, pages 252–265, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[20] C. P Schnorr. *Journal of Cryptology*, 4:161–174, 1991.

[21] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979.