

VERIFIABLE ACCOUNTABLE SUBGROUP MULTI-SIGNATURES

A THESIS SUBMITTED TO  
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS  
OF  
MIDDLE EAST TECHNICAL UNIVERSITY

BY

AHMET RAMAZAN AĞIRTAŞ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR  
THE DEGREE OF DOCTOR OF PHILOSOPHY  
IN  
CRYPTOGRAPHY

FEBRUARY 2024



Approval of the thesis:

**VERIFIABLE ACCOUNTABLE SUBGROUP MULTI-SIGNATURES**

submitted by **AHMET RAMAZAN AĞIRTAŞ** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Cryptography Department, Middle East Technical University** by,

Prof. Dr. A. Sevtap Kestel  
Dean, Graduate School of **Applied Mathematics**

\_\_\_\_\_

Assoc. Prof. Dr. Oğuz Yayla  
Head of Department, **Cryptography**

\_\_\_\_\_

Assoc. Prof. Dr. Oğuz Yayla  
Supervisor, **Department of Cryptography, METU**

\_\_\_\_\_

**Examining Committee Members:**

Prof. Dr. Ali Aydın Selçuk  
Department of Computer Engineering, TOBB ETU

\_\_\_\_\_

Assoc. Prof. Dr. Oğuz Yayla  
Department of Cryptography, METU

\_\_\_\_\_

Prof. Dr. Zülfükar Saygı  
Department of Mathematics, TOBB ETU

\_\_\_\_\_

Assoc. Prof. Dr. Ahmet Sınak  
Department of Mathematics and Computer Science,  
Necmettin Erbakan University

\_\_\_\_\_

Assist. Prof. Dr. Buket Özkaya  
Department of Cryptography, METU

\_\_\_\_\_

**Date:**

\_\_\_\_\_



**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

Name, Last Name: AHMET RAMAZAN AĞIRTAŞ

Signature :



# ABSTRACT

## VERIFIABLE ACCOUNTABLE SUBGROUP MULTI-SIGNATURES

Ağırtaş, Ahmet Ramazan

Ph.D., Department of Cryptography

Supervisor : Assoc. Prof. Dr. Oğuz Yayla

February 2024, 97 pages

In this thesis, we introduce an accountable subgroup multi-signature (ASM) framework. The framework comprises three novel pairing-based ASM schemes, i.e.,  $vASM$ ,  $ASMwSA$  and  $ASMwCA$ , each designed to be secure against chosen-message attacks and based on the computational co-Diffie-Hellman/ $\psi$ -co-Diffie-Hellman assumption. We address an open problem by proposing novel ASM schemes where the subgroup of signers is unknown before signature generation. Our schemes outperform existing methods in terms of computational efficiency in signature generation, aggregation, and verification. Additionally, we propose novel methods for compartment-based and hierarchical threshold delegation of signing power of the verifiable accountable subgroup multi-signature scheme. We demonstrate that the scheme can function as a proxy signature, allowing an authorized user to delegate signing rights to an unauthorized user or group. We present four constructions, employing the recursive application of  $vASM$ , Shamir's secret sharing scheme, nested secret sharing, and hierarchical threshold secret sharing, comparing their efficiency and security. Moreover, we propose a novel lattice-based ASM scheme ( $vMS_2$ ) by combining the group setup method of  $vASM$  with Damgård et al.'s lattice-based  $MS_2$  multi-signature scheme. We showcase the equivalence of key generation, signature generation, and verification phases with the  $MS_2$  scheme. Our  $vMS_2$  scheme achieves accountability through a joint verifiable secret sharing scheme during group setup, with a cost of slightly higher than the underlying  $MS_2$  scheme.

Keywords: multi-signatures, accountable subgroup multi-signatures, pairing-based cryptography, lattice-based cryptography

# ÖZ

## DOĞRULANABİLİR HESAP VERİLEBİLİR ALTGRUP ÇOKLU İMZALARI

Ağırtaş, Ahmet Ramazan

Doktora, Kriptografi Bölümü

Tez Yöneticisi : Doç. Dr. Oğuz Yayla

Şubat 2024, 97 sayfa

Bu tezde, hesap verilebilir bir altgrup çoklu imza (ASM) çerçevesi tanıtıyoruz. Çerçeve, her biri seçilmiş mesaj saldırılarına karşı güvenli ve hesaplamalı co-Diffie-Hellman/ $\psi$ -co-Diffie-Hellman varsayımına dayanan üç yeni eşleme tabanlı ASM şemasını, yani vASM, ASMwSA ve ASMwCA şemalarını içermektedir. İmza üretimi sırasında imzacı alt grubunun bilinmediği yeni ASM şemaları önererek bir açık problemi ele alıyoruz. Şemalarımız, imza üretimi, birleştirme ve doğrulama aşamalarında hesaplama verimliliği açısından mevcut yöntemleri geride bırakmaktadır. Ek olarak, vASM imza yetkisinin kısmi ve hiyerarşik eşik delegasyonu için yeni metotlar öneriyoruz. Şemanın bir vekil imza olarak işlev görebileceğini, bir yetkili kullanıcının imza haklarını yetkisiz bir kullanıcıya veya gruba devretmesine izin verebileceğini gösteriyoruz. Özyinelemeli vASM uygulaması, Shamir'in sır paylaşım şeması, iç içe sır paylaşımı ve hiyerarşik eşik sır paylaşımı kullanarak dört yeni yapı sunuyor ve bunların verimlilik ve güvenlik açısından karşılaştırmasını yapıyoruz. Ayrıca, vASM'nin grup kurulum yöntemi ile Damgård ve diğerlerinin kafes tabanlı  $MS_2$  çoklu imza şemasını birleştirerek yeni bir kafes tabanlı ASM şeması ( $vMS_2$ ) öneriyoruz. Anahtar üretim, imza üretim ve doğrulama aşamalarının  $MS_2$  şemasıyla eşdeğer olduğunu gösteriyoruz. Önerdiğimiz  $vMS_2$  şeması, grup kurulumu sırasında müşterek doğrulanabilir gizli paylaşım şeması kullanarak hesap verilebilirliği altta yatan  $MS_2$  şemasının biraz üzerinde bir maliyetle gerçekleştirilmektedir.

Anahtar Kelimeler: çoklu imzalar, hesap verilebilir çoklu imzalar, eşleme tabanlı kriptografi, kafes tabanlı kriptografi

*I humbly dedicate this thesis to every member of my beloved nation, to whom I owe my educational journey.*



## **ACKNOWLEDGMENTS**

First and foremost, I would like to thank my supervisor Assoc. Prof. Dr. Oğuz Yayla. Words cannot express my gratitude to him for his guidance, encouragement, and valuable advice during the development of this thesis.

I could not have undertaken this journey without my thesis monitoring/examination committee, which generously provided knowledge and constructive criticism.

I sincerely thank all faculty members for the fantastic classes and discussions.

I would be remiss in not mentioning my family, my spouse Gamze, my sons Ahmet Giray and Mehmet Altay, and my friends. Their belief in me has always kept my spirits and motivation high.



# TABLE OF CONTENTS

ABSTRACT . . . . .	vii
ÖZ . . . . .	ix
ACKNOWLEDGMENTS . . . . .	xiii
TABLE OF CONTENTS . . . . .	xv
LIST OF TABLES . . . . .	xix
LIST OF FIGURES . . . . .	xxi
CHAPTERS	
1 INTRODUCTION . . . . .	1
1.1 Pairing-based Accountable Subgroup Multi-signatures . . . . .	2
1.2 Compartment-based and Hierarchical Threshold Delegated Verifiable Accountable Subgroup Multi-signatures . . . . .	3
1.3 Lattice-based Accountable Subgroup Multi-signature . . . . .	5
2 PRELIMINARIES . . . . .	7
2.1 Preliminaries for pairing-based constructions . . . . .	7
2.1.1 Bilinear pairings and computational hard problems . . . . .	7
2.1.2 Digital Signatures, Multi-signatures and Account- able Subgroup Multi-signatures . . . . .	8

2.1.3	Generalized forking lemma . . . . .	11
2.1.4	BLS Signature Scheme . . . . .	13
2.1.5	Boneh-Drijvers-Neven ASM Scheme . . . . .	14
2.2	Preliminaries for compartment-based and hierarchical threshold delegated verifiable accountable subgroup multi-signatures	16
2.2.1	Shamir’s Secret Sharing (SSS) Scheme . . . . .	17
2.2.2	Feldman’s Verifiable Secret Sharing (VSS) Scheme	18
2.2.3	Hierarchical Threshold Secret Sharing Scheme . .	19
2.2.4	Security of the secret sharing schemes . . . . .	21
2.3	Preliminary for lattice-based constructions . . . . .	22
2.3.1	Assumptions and the notation . . . . .	22
2.3.2	Dilithium-G . . . . .	24
2.3.3	MS <sub>2</sub> Scheme . . . . .	26
3	<b>PAIRING-BASED ACCOUNTABLE SUBGROUP MULTI-SIGNATURES WITH VERIFIABLE GROUP SETUP . . . . .</b>	<b>29</b>
3.1	<b>vASM: An ASM scheme with VSS based group setup . . . . .</b>	<b>30</b>
3.1.1	Security . . . . .	33
3.2	<b>Accountable Subgroup Multi-signature Scenarios with Subgroup Authentication . . . . .</b>	<b>37</b>
3.2.1	<b>ASMwSA: Accountable Subgroup Multi-signature with Subgroup Authentication . . . . .</b>	<b>37</b>
3.2.1.1	Security . . . . .	39
3.2.2	<b>ASMwCA: Accountable Subgroup Multi-signature with Combiner Authentication . . . . .</b>	<b>43</b>

	3.2.2.1	Security . . . . .	45
	3.2.3	Eliminating the combiner . . . . .	45
	3.2.4	Advantages of subgroup-specific membership key .	46
3.3		Aggregated versions of ASM schemes . . . . .	47
	3.3.1	Partially Aggregated ASM Scheme (AASM) . . .	47
	3.3.2	Aggregated versions of proposed schemes . . . . .	48
	3.3.2.1	Aggregated vASM Scheme (AvASM)	48
	3.3.2.2	Aggregated versions of ASMwSA / ASMwCA Schemes . . . . .	50
3.4		Comparison . . . . .	52
4		COMPARTMENT-BASED AND HIERARCHICAL THRESHOLD DELEGATED VERIFIABLE ACCOUNTABLE SUBGROUP MULTI- SIGNATURES . . . . .	55
4.1		Compartment-based and Hierarchical Threshold Delegation of vASM Authority . . . . .	56
	4.1.1	Recursive vASM as a proxy signature . . . . .	57
	4.1.2	Shamir’s Secret Sharing Scheme Based Delegation	61
	4.1.3	Trusted User Based Delegation . . . . .	63
	4.1.4	Hierarchical Threshold Secret Sharing Scheme Based Delegation . . . . .	66
4.2		Comparison . . . . .	68
	4.2.1	Comparison of the proposed constructions . . . . .	68
	4.2.2	Comparison of vASM scheme with the existing proxy signature schemes . . . . .	69

4.2.2.1	vASM signature scheme can serve as a proxy signature ( $n = l = 1$ ) . . . . .	71
4.2.2.2	vASM signature scheme can serve as a proxy multi-signature ( $n > 1$ and $l = 1$ ) . . . . .	71
4.2.2.3	vASM signature scheme can serve as a multi-proxy signature ( $n = 1$ and $l > 1$ ) . . . . .	72
4.2.2.4	vASM signature scheme can serve as a multi-proxy multi-signature ( $n > 1$ and $l > 1$ ) . . . . .	73
5	A LATTICE-BASED ACCOUNTABLE SUBGROUP MULTI-SIGNATURE SCHEME . . . . .	77
5.1	vMS <sub>2</sub> : A lattice-based ASM scheme with verifiable group setup . . . . .	77
5.2	Security Analysis . . . . .	82
5.3	Computational Analysis . . . . .	83
6	CONCLUSION . . . . .	87
	REFERENCES . . . . .	91
	CURRICULUM VITAE . . . . .	97

## LIST OF TABLES

Table 3.1	Comparison of our pairing-based ASM schemes . . . . .	54
Table 4.1	Comparison of the delegation methods . . . . .	69
Table 4.2	Comparison with the existing proxy signature schemes . . . . .	72
Table 4.3	Comparison with the existing proxy multi-signature schemes . . . . .	73
Table 4.4	Comparison with the existing multi-proxy signature schemes . . . . .	74
Table 4.5	Comparison with the existing multi-proxy multi-signature schemes . . . . .	75
Table 5.1	Parameters for $MS_2$ [20] and $vMS_2$ schemes . . . . .	83
Table 5.2	Comparison of the $MS_2$ and $vMS_2$ schemes . . . . .	85



## LIST OF FIGURES

Figure 3.1	ASM scheme without a combiner . . . . .	46
Figure 4.1	Membership key generation of recursive vASM . . . . .	58
Figure 4.2	Shamir's Secret Sharing Scheme Based Delegation . . . . .	61
Figure 4.3	Trusted User Based Delegation . . . . .	64
Figure 4.4	Hierarchical Threshold Secret Sharing Scheme Based Delegation . . . . .	66



# CHAPTER 1

## INTRODUCTION

Digital signature schemes play a crucial role in modern cryptographic protocols for verifying the authenticity of any message, such as official documents, financial transactions, e-mails, etc. In particular, the increasing popularity of cryptocurrencies [16, 47, 48] in the last decade made the digital signature schemes more significant than before. The structures of signature schemes differ according to many reasons, such as use area, system requirements, and users' needs. A *multi-signature* [10, 31, 49, 50, 52, 56] is a kind of digital signature in which a group of signers signs the same message jointly. In literature, there are some notions related to multi-signatures for different scenarios, such as group signatures [17, 19], threshold signatures [10, 21, 28, 29], aggregate signatures [13], ring signatures (also threshold, linkable and traceable variants) [54, 15, 40, 27], accountable subgroup multi-signatures [12, 46], etc. None of them provide sufficient flexibility regarding the number of signers and accountability of the signers at the same time, except for accountable subgroup multi-signatures. An *accountable subgroup multi-signature (ASM)* [12, 46] 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 [46] by proposing the first ASM scheme. In a more recent paper [12], Boneh, Drijvers, and Neven proposed another ASM scheme which is based on BLS signature [14], and solves the open problem in [46], i.e., constructing an ASM scheme in which the subgroup of signers  $\mathcal{S} \subseteq \mathcal{G}$  is not determined before the signature generation.

In this thesis, we contribute to the literature in the following three directions, i.e.,

- we propose three pairing-based ASM schemes (vASM, ASMwSA, and ASMwCA) and their aggregated versions, i.e., (AvASM, AASMwSA, and AASMwCA) in [3],
- we extend our work by proposing four compartment-based and hierarchical threshold delegation methods for vASM scheme in [4] and [5],
- we apply the same idea with lattice assumptions and proposed a lattice-based (vMS<sub>2</sub>) ASM scheme in [6].

### 1.1 Pairing-based Accountable Subgroup Multi-signatures

In literature, the only pairing-based accountable subgroup multi-signature scheme was proposed in [12]. In this ASM scheme, all potential signers in a group  $\mathcal{G}$  participate in a group setup in which they use another multi-signature scheme, i.e., MSP scheme [12], to authenticate each other and to obtain a special key (membership key) for signing on behalf of the group  $\mathcal{G}$ . This way, they authorize each other to sign on behalf of themselves. In this scheme, the verification requires three pairings as two separate keys, i.e., private key (scalar) and membership key (group element), per signer are used for signing. To reduce the number of pairings in the verification, we figure out a method to generate a single scalar membership key by running joint Feldman's Verifiable Secret Sharing (VSS) Scheme in parallel by the potential signers in  $\mathcal{G}$ . Another method for reducing the number of pairing is to use a subgroup-specific membership keys instead of a group-specific ones. We discuss all the methods and propose our pairing-based ASM schemes in Chapter 3.

In Section 2.1, we give a brief background information, including definitions of bilinear pairings, co-CDH/ $\psi$ -co-CDH problems, digital signatures, multi-signatures and accountable subgroup multi-signatures, generalized forking lemma, and BLS signature scheme. In Section 2.1.5, we summarize the ASM scheme given in [12]. Then, in Section 3.1, we give our vASM scheme and prove its security in the random oracle model (ROM). Moreover, in Section 3.2, we give our ASMwSA and ASMwCA schemes which are based on subgroup authentication instead of global authentication. In the same section we prove their security in the random oracle model using gener-

alized forking lemma. In Section 3.3, we summarize the partial aggregated ASM (AASM) scheme given in [12], and discuss the aggregated versions of our proposed schemes and their security. Finally, in Section 3.4, we compare our new schemes with ASM and AASM schemes of [12] in terms of the number of operations required in the phases of the schemes and costs of transmission, broadcasting and storage.

## **1.2 Compartment-based and Hierarchical Threshold Delegated Verifiable Accountable Subgroup Multi-signatures**

Signatory authorities carry out their transactions by assigning a deputy to use this power of signing authority for periods when they cannot be present at their organization. While they can appoint a single deputy among their subordinates, there may also be cases in which they authorize different proxies on different issues. In the literature, several notions define solutions for delegating signing capability to unauthorized users by authorized ones in an organization. *Proxy signature*, which allows an unauthorized user (proxy signer) to sign on behalf of an authorized user (original signer), was first defined by Mambo et al. [44] in 1996. After its first proposal, proxy signatures have been studied in [1, 11, 33, 34, 35, 43, 51, 62, 67, 69]. Moreover the proxy multi-signatures [68], the multi-proxy signatures [61], and the multi-proxy multi-signatures [60] were proposed according to the number of proxies and original signers in the constructions. Among these variants, *proxy multi-signature* is a notion related to the delegation of signing authority, and it was first proposed by Yi et al. [68] in 2000. It is a multi-signature scheme in which a designated proxy signer generates a signature on a common message on behalf of several original signers. Hwang and Shi [61] proposed the concept of the multi-proxy signature scheme in which an original signer can authorize a group of proxy signers as a proxy agent, and a valid multi-proxy signature can be generated only with the participation of all proxy signers. Another proxy signature concept is *multi-proxy multi-signatures* proposed by Hwang and Chen [60] in 2004. In a multi-proxy multi-signature scheme, only the cooperation of a group of original signers designates a group of proxy signers. Then, only the cooperation of all members of the proxy group can generate a valid signature on behalf of the group of original signers. Several proxy signatures also exist,

such as threshold, blind, and ring variants [30, 37, 39, 70]. Most of the proposals in the literature either show that the previous ones were insecure or propose a modified one because of the lack of a standard security model. The security properties of an ideal proxy signature are defined in [44] and extended in [35]. These properties are as follows:

- *Verifiability*: A proxy signature should convince any verifier that the original signer agrees with it.
- *Strong unforgeability*: Except for the designated proxy signer, no one can create a valid proxy signature.
- *Strong identifiability*: From a proxy signature, any verifier should identify the identity of the proxy signer.
- *Strong undeniability*: A proxy signature should have non-repudiation property.
- *Prevention of misuse*: Ensuring that a proxy signing key can be used only proxy signing process, no other purposes.

Boldyreva et al. [11] stated that the above security properties were defined informally and needed to be formalized. For this reason, they first defined a formal security model for the proxy signature schemes and defined the functionalities a proxy signature should have. In this thesis, we show that the verifiable accountable subgroup multi-signature (vASM) scheme [3] can also be used as a proxy signature scheme. The vASM scheme also supports one or more proxies and original signers. In addition, we also show that the signing power of vASM authority can be delegated via appropriate threshold secret sharing schemes [25, 32, 58, 63, 64].

In Section 2.2, we give preliminary information, including definitions of proxy signature schemes, Shamir's secret sharing scheme (SSS) [58], Feldman's verifiable secret sharing (VSS) protocol [25], hierarchical threshold secret sharing scheme (HTSS) [32, 63, 64], security discussion of threshold secret sharing schemes. In Chapter 4, we give our proposed constructions regarding Compartment-based and Hierarchical Threshold Delegated Verifiable Accountable Subgroup Multi-signatures. In Section 4.2, we compare our constructions in terms of the number of operations required

in the phases of our proposed constructions. Then we compare the vASM scheme with the existing proxy signatures, proxy multi-signatures, multi-proxy signatures and multi-proxy multi-signatures in terms of their efficiency.

### 1.3 Lattice-based Accountable Subgroup Multi-signature

The common structure of the existing pairing-based ASM schemes is as follows. Assume that  $\mathcal{G}$  is a group of  $n$  users, and  $\mathcal{S} \subseteq \mathcal{G}$  is a subgroup of  $\tau$  signers among  $n$ . Users in  $\mathcal{G}$  generate their secret and public key pairs individually. Then they participate in an interactive one-time group setup phase. After the group setup phase each user obtains her membership key. Then the users in  $\mathcal{S}$  sign a common message  $m$  with her secret and/or membership keys. A combiner, who can be one of the signers or a designated third party, aggregates the individual signatures and outputs the accountable subgroup multi-signature of the subgroup  $\mathcal{S}$ . Given the message, the signature and the information  $\mathcal{S}$ , any verifier can verify whether the signature is valid or not.

The main difference among the existing pairing-based ASM schemes is their group setup phase. In [12], authors utilize another multi signature scheme to generate the users' membership keys, and authenticate each other to show that they are authorized to sign on behalf of the group  $\mathcal{G}$ . However, in vASM scheme we use verifiable secret sharing scheme for the same purposes. Users share their secret keys via a joint verifiable secret sharing scheme to authorize other users in group  $\mathcal{G}$  as described in Section 3.1.

Due to the fact that pairing-based cryptography is susceptible to quantum attacks we consider to transform our vASM scheme into a quantum-resistant ASM scheme. We simply ask the following question:

*If we apply the same group setup method to another multi-signature, can we have an accountable subgroup multi-signature with another assumption?*

We focus on the multi-signatures based on Dilithium [23] which is based on module-LWE and module-SIS problems and is one of the winner signature schemes of the

NIST's (National Institute of Standards and Technology) post-quantum cryptography standardization process.

In Section 2.3, we give the notation, assumptions, definitions of hard problems and the protocols used in our construction. In Chapter 5, we propose a novel lattice-based accountable subgroup multi-signature scheme by combining the group setup method of our vASM scheme given in Section 3.1 and Damgård et al.'s lattice-based  $MS_2$  multi-signature scheme [20].

## CHAPTER 2

### PRELIMINARIES

In this chapter, we give preliminaries, including definitions, lemmata, and notations, for the proposed constructions. For the sake of simplicity we split the preliminary information into three subsections as follows.

- Section 2.1 includes the preliminaries for the constructions in Chapter 3,
- Section 2.2 includes the preliminaries for the constructions in the Chapter 4, and
- Section 2.3 includes the preliminaries for the constructions in the Chapter 5.

#### 2.1 Preliminaries for pairing-based constructions

In order to provide sufficient background information for readers, we give the definitions of notions that we mainly use throughout Chapter 3. Namely, we give the definitions of bilinear pairings, computationally hard problems, digital signatures, multi-signatures and accountable subgroup multi-signatures, generalized forking lemma, and the definitions of BLS signature scheme and Boneh-Drijvers-Neven ASM Scheme.

##### 2.1.1 Bilinear pairings and computational hard problems

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

**Definition 2.1.1.** A pairing is a map  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$  which satisfies the bilinearity and non-degeneracy properties:

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

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

**Definition 2.1.2** (Computational co-Diffie-Hellman Problem [13]). For groups  $\mathbb{G}_1 = \langle g_1 \rangle$  and  $\mathbb{G}_2 = \langle g_2 \rangle$  of prime order  $q$ , define  $Adv_{\mathbb{G}_1, \mathbb{G}_2}^{co-CDH}$  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  $(\alpha, \beta)$ .  $\mathcal{A}(\tau, \epsilon)$ -breaks the co-CDH problem if it runs in time at most  $\tau$  and has  $Adv_{\mathbb{G}_1, \mathbb{G}_2}^{co-CDH} \geq \epsilon$ . co-CDH is  $(\tau, \epsilon)$ -hard if no such adversary exists.

**Definition 2.1.3** (Computational  $\psi$ -co-Diffie-Hellman Problem [12]). For groups  $\mathbb{G}_1 = \langle g_1 \rangle$  and  $\mathbb{G}_2 = \langle g_2 \rangle$  of prime order  $q$ , let  $\mathcal{O}^\psi(\cdot)$  be an oracle that returns  $g_1^x \in \mathbb{G}_1$  on input  $g_2^x \in \mathbb{G}_2$ . Define  $Adv_{\mathbb{G}_1, \mathbb{G}_2}^{\psi-co-CDH}$  of an adversary  $\mathcal{A}$  as

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

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

## 2.1.2 Digital Signatures, Multi-signatures and Accountable Subgroup Multi-signatures

**Definition 2.1.4** (Digital signature scheme [11]). A digital signature scheme  $\mathcal{DS} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V})$  is specified by four algorithms which are defined below:

- $\mathcal{G}(1^\lambda)$  takes  $1^\lambda$ , where  $\lambda$  is the security parameter as input, and outputs the public system parameters  $par$  including security parameter, hash functions, cyclic groups, generators, etc.
- $\mathcal{K}(par)$  takes system parameters  $par$  as input, and outputs a secret-public key pair  $(sk, pk)$ .
- $\mathcal{S}(par, M, sk)$  takes system parameters  $par$ , message  $M$ , and secret key  $sk$  as inputs, and returns a signature  $\sigma$ .
- $\mathcal{V}(par, M, pk, \sigma)$  takes system parameters  $par$ , message  $M \in \{0, 1\}^*$ , public key  $pk$ , a signature  $\sigma$  as inputs, and outputs accept or reject.

**Definition 2.1.5.** A multi-signature scheme consists of four algorithms, i.e., *ParGen*, *KeyGen*, *Sign*, and *Verify*. Let  $\mathcal{G} = \{P_1, \dots, P_n\}$  be a set of  $n$  players.

- ***ParGen*** $(1^\lambda)$  takes the security parameter  $\lambda$  as input, and outputs the public system parameters  $par$  including security parameter, hash functions, cyclic groups, generators, etc.
- ***KeyGen*** $(par)$  takes the system parameters  $par$  as input, and outputs secret and public key pair, i.e.,  $sk$  and  $pk$ .
- ***Sign*** $(par, sk, m)$  is an interactive protocol which is run by  $\mathcal{G}$ , in two steps, as follows:
  - ***Individual signature generation*** takes the system parameters  $par$ , secret key  $sk_i$  and message  $m$  as inputs, and outputs the individual signature  $\sigma_i$ .
  - ***Individual signature aggregation*** takes a set of individual signatures  $\{\sigma_i\}_{i \in \mathcal{G}}$  as inputs and outputs the multi-signature  $\sigma$ .
- ***Verify*** $(par, \{pk_j\}_{j \in \mathcal{G}}, \sigma, m)$  takes system parameter  $par$ , multi-signature  $\sigma$ , message  $m$ , and public keys of the players in  $\mathcal{G}$  as inputs, and outputs 1 if it is valid or 0 otherwise.

For the definition of an accountable subgroup multi-signature scheme, we add an interactive group setup algorithm **GSetup**, which is a one-time protocol run by all the players in the group  $\mathcal{G}$ . Then we modify the **Sign** and **Verify** algorithms as follows.

**Definition 2.1.6.** An accountable subgroup multi-signature scheme is a tuple of five algorithms, that is *ParGen*, *KeyGen*, *GSetup*, *Sign*, and *Verify*. Let  $\mathcal{G} = \{P_1, \dots, P_n\}$  be a set of  $n$  players, and  $\mathcal{PK} = \{pk_1, \dots, pk_n\}$  is the set of public keys of all the users in group  $\mathcal{G}$ .

- **ParGen**( $1^\lambda$ ) takes the security parameter  $\lambda$  as input, and outputs the public system parameters *par* including security parameter, hash functions, cyclic groups, generators, etc.
- **KeyGen**(*par*) takes the system parameters *par* as input, and outputs secret and public key pair, i.e., *sk* and *pk*.
- **GSetup**(*par*,  $sk_i$ ,  $\mathcal{PK}$ ) is an interactive protocol which is run by all the players in  $\mathcal{G}$ . It takes system parameters *par*, secret keys  $sk_i$  and set of all public keys  $\mathcal{PK}$ , and outputs a membership key  $mk_i$ , a set of membership public keys *MPK*, and a set of commitments *COM*.
- **Sign**(*par*,  $mk_i$ ,  $m$ ) is an interactive protocol which is run by any subset  $\mathcal{S} \subseteq \mathcal{G}$ , in two steps, as follows:
  - **Individual signature generation** takes the system parameters *par*, membership key  $mk_i$  and message  $m$  as inputs, and outputs the individual signature  $\sigma_i$ .
  - **Individual signature aggregation** takes a set of individual signatures  $\{\sigma_i\}_{i \in \mathcal{S}}$  as inputs and outputs the accountable subgroup multi-signature  $\sigma$ .
- **Verify**(*par*, *MPK*, *COM*,  $\mathcal{S}$ ,  $\sigma$ ,  $m$ ) takes system parameter *par*, multi-signature  $\sigma$ , message  $m$ , definition of the subset  $\mathcal{S}$ , the set of membership public keys *MPK*, and the commitment set *COM* as inputs, and outputs 1 if it is valid or 0 otherwise.

*Correctness* and *unforgeability* are two properties that every accountable subgroup multi-signature scheme should meet. *Correctness* means that for any subgroup of signers  $\mathcal{S} \subseteq \mathcal{G}$  and message  $m$ , if the signers  $P_i \in \mathcal{S}$  run the **Sign**( $\cdot$ ) protocol with their membership keys  $mk_i$ , and follow the protocol honestly, then all of the signers

in  $\mathcal{S}$  outputs exactly the same valid accountable subgroup multi-signature  $\sigma$ , such that  $\text{Verify}(par, \text{MPK}, \text{COM}, \mathcal{S}, \sigma, m) = 1$ . Unforgeability means that it is infeasible for an adversary to forge a valid multi-signature where at least one honest user follows the protocol properly. Unforgeability can be described by the following game.

**Setup:** The challenger randomly picks  $n$  values and computes membership public keys MPK and the commitment set COM, with respect to the indices  $\{1, \dots, n\}$ . Finally, it runs the adversary  $\mathcal{A}(par, \text{MPK}, \text{COM})$ , where  $par$  is the system parameters.

**Signature queries:** The adversary  $\mathcal{A}$  makes queries on any message  $m$ , for any subset  $\mathcal{S} \subseteq \{1, \dots, n\}$  of users, and challenger responds with valid signatures.

**Output:** The adversary  $\mathcal{A}$  eventually outputs a subset of indices  $\mathcal{S}$ , a message  $m$ , and an accountable subgroup multi-signature  $\sigma$ . The adversary  $\mathcal{A}$  wins the game if  $\text{Verify}(par, \text{MPK}, \text{COM}, \mathcal{S}, \sigma, m) = 1$ , where the message  $m$  has been never queried as part of a signing query before.

**Definition 2.1.7.** Let  $\Pi = \{\text{ParGen}, \text{KeyGen}, \text{GSetup}, \text{Sign}, \text{Verify}\}$  be an accountable subgroup multi-signature scheme.

We say that an adversary  $\mathcal{A}$  is a  $(t, q_H, q_S, \epsilon)$ -forger if it runs in time  $t$ , makes at most  $q_H$  random oracle queries and  $q_S$  signing queries, and wins the above game with probability at least  $\epsilon$ . We say that the accountable subgroup multi-signature scheme is  $(t, q_H, q_S, \epsilon)$ -secure if no such adversary exists.

### 2.1.3 Generalized forking lemma

In order to prove the security of Schnorr-based signature schemes, Pointcheval and Stern firstly defined the forking lemma in [53]. Bellare and Neven generalized this lemma in [9]. In the security proofs of schemes in [12], the authors use the lemma in [7], which is a generalization of the forking lemma by Bagherzandi, Cheon, and Jarecki. We also use the latter generalized forking lemma in the security proofs of our two constructions.

Let  $\mathcal{A}$  be an algorithm that interacts with random-oracle and takes  $inp$  as input. Let

$f = (\rho, h_1, \dots, h_{q_H})$  be the randomness used in the process of  $\mathcal{A}$ . Let  $\rho$  be  $\mathcal{A}$ 's random tape,  $h_i$  be the response to  $\mathcal{A}$ 's  $i$ -th hash query,  $q_H$  be the maximal number of hash queries. Let  $\Omega$  be the space of all randomness vectors like  $f$ , and  $f|_i$  be defined as  $f|_i = (\rho, h_1, \dots, h_{i-1})$  for any  $i \leq q_H$ .  $\mathcal{A}(inp, f)$  is considered as successful if it returns a pair  $(J, \{out_j\}_{j \in J})$ , where  $J$  is a non-empty multi-set of indexes with  $|J| = n$  and  $\{out_j\}_{j \in J}$  is the multi-set of side outputs. Let  $\epsilon$  be the probability that  $\mathcal{A}(inp, f)$  is successful for fresh randomness  $f \xleftarrow{\$} \Omega$  and for an input  $inp \xleftarrow{\$} \text{IG}$  generated by an input generator IG. On input  $inp$ , the generalized forking algorithm  $\mathcal{GF}_{\mathcal{A}}$  proceeds as follows:

- $f = (\rho, h_1, \dots, h_{q_H}) \xleftarrow{\$} \Omega$
- $(J, \{out_j\}_{j \in J}) \leftarrow \mathcal{A}(inp, f)$
- If  $J = \emptyset$ , then output “fail”
- Let  $J = \{j_1, \dots, j_n\}$  such that  $j_1 \leq \dots \leq j_n$ .
- For  $i = 1$  to  $n$  do
  - $succ_i \leftarrow 0$ ,
  - $k_i \leftarrow 0$ ,
  - $k_{max} = 8nq_H/\epsilon \cdot \ln(8n/\epsilon)$
  - Repeat until  $succ_i = 1$  or  $k_i > k_{max}$ 
    - \*  $k_i = k_i + 1$
    - \*  $f'' \xleftarrow{\$} \Omega$  such that  $f''|_{j_i} = f|_{j_i}$
    - \* Let  $f'' = (\rho, h_1, \dots, h_{j_i-1}, h''_{j_i}, \dots, h''_{q_H})$
    - \*  $(J'', \{out''_j\}_{j \in J''}) \leftarrow \mathcal{A}(inp, f'')$
    - \* If  $h''_{j_i} \neq h_{j_i}$  and  $J'' \neq \emptyset$  and  $j_i \in J''$  then
      - $out'_{j_i} \leftarrow out''_{j_i}$
      - $succ_i \leftarrow 1$
- If  $succ_i = 1$  for all  $i = 1, \dots, n$ , then output  $(J, \{out_j\}_{j \in J}, \{out'_j\}_{j \in J})$
- Else output “fail”

The  $\mathcal{GF}_A$  algorithm is considered to be successful if it does not return “fail”.

**Lemma 2.1.1** (Generalized Forking Lemma [7]). *Let  $IG$  be a randomized algorithm generating  $inp$ . Let  $\mathcal{A}$  be also a randomized algorithm that makes at most  $q_H$  random-oracle queries in time  $\tau$ , which succeeds with probability  $\epsilon$ . If  $q > 8nq_H/\epsilon$ , then  $\mathcal{GF}_A(inp)$  runs in time at most  $\tau \cdot 8n^2q_H/\epsilon \ln(8n/\epsilon)$  and succeeds with probability at least  $\epsilon/8$ , where the probability is over the choice of  $inp \xleftarrow{\$} IG$  and the coins of  $\mathcal{GF}_A$ .*

The forking lemma tells us that if an adversary obtains a successful forgery, it can obtain another successful forgery on the same message but with different random oracle query values. We will use this lemma to prove the security of two of our schemes in the random oracle model.

#### 2.1.4 BLS Signature Scheme

Let  $e$  be an efficient, non-degenerate bilinear map,  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ , in groups  $(\mathbb{G}_1, \mathbb{G}_2, \text{ and } \mathbb{G}_T)$  and  $(g_1, g_2)$  be generators of the group pair  $(\mathbb{G}_1, \mathbb{G}_2)$ , respectively. Let  $H : \{0, 1\}^* \rightarrow \mathbb{G}_1$  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 [14].

**Definition 2.1.8.** *We say that an adversary  $\mathcal{A}(t, q_S, q_H, \epsilon)$ -breaks a signature scheme if it runs in time at most  $t$ , makes at most  $q_S$  signature queries and at most  $q_H$  hash function queries, and the success probability of  $\mathcal{A}$  is at least  $\epsilon$ . A signature scheme*

is  $(t, q_S, q_H, \epsilon)$ -secure against existential forgery under adaptive chosen-message attacks if no such adversary exists.

**Theorem 2.1.1** (Theorem 3.2. [14]). *If solving the co-CDH problem in  $\mathbb{G}_1 \times \mathbb{G}_2$  is  $(t', \epsilon')$ -hard, then the BLS signature scheme is  $(t, q_S, q_H, \epsilon)$ -secure against existential forgery under adaptive chosen-message attacks, for*

$$t = t' - c_{\mathbb{G}_1}(q_H + 2q_S), \text{ and}$$

$$\epsilon = \epsilon' e(q_S + 1).$$

where  $c_{\mathbb{G}_1}$  is a constant that depends on  $\mathbb{G}_1$ , and  $e$  is the base of natural logarithm.

Although Theorem 2.1.1 was proved in [14] to be secure in the random oracle model, it is not safe to use it as a multi-signature scheme directly because of the “rogue-key” attack [13]. In order to avoid this attack, there are some standard measures, 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 from performing efficient verification [12], and using PoP requires additional verification operations. It is not fully compatible with the applications in cryptocurrencies [45]. In order to eliminate these disadvantages, Boneh, Drijvers and Neven proposed in [12] a multi-signature scheme, called MSP, which is a modified version of the BLS scheme. Moreover, they proposed an accountable subgroup multi-signature (ASM) scheme, a composition of the BLS and MSP schemes.

### 2.1.5 Boneh-Drijvers-Neven ASM Scheme

This section follows the notation given in both [12] and the previous sections. Let  $\mathcal{PK} := \{pk_1, \dots, pk_n\}$  be the set of public keys of the group members of the group  $\mathcal{G}$ , and let  $H_0, H_2 : \{0, 1\}^* \rightarrow \mathbb{G}_1$  and  $H_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_q$  be the hash functions. The ASM scheme given in [12] 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  $i = 1, 2, \dots, n$ .

- 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  $j = 1, 2, \dots, n$  and  $j \neq i$ .
- After receiving  $\mu_{ji}$ , 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  $m$

$$s_i = H_0(apk, m)^{sk_i} \cdot mk_i, \quad (2.1)$$

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

$$e\left(H_0(apk, m), pk\right) \cdot e\left(\prod_{j \in \mathcal{S}} H_2(apk, j), apk\right) \stackrel{?}{=} e(s, g_2). \quad (2.2)$$

**Theorem 2.1.2** (Theorem 3. [12]). *ASM scheme is unforgeable under the  $\psi$ -co-CDH problem (Definition 2.1.3) in the random oracle model. More precisely, ASM scheme is  $(\tau, q_H, q_S, \epsilon)$ -unforgeable in the random oracle model if  $q > 8q_H/\epsilon$  and if  $\psi$ -co-CDH problem is  $(\tau + q_H \cdot \max(\tau_{exp_2^1}, \tau_{exp_2^2}) + q_G(l-1)\tau_{exp_1} + q_S(\tau_{exp_2^1} + \tau_{exp_1}) + 2\tau_{pair} + \tau_{exp_1^3}) \cdot 8q_H^2 / ((1 - (q_S + q_H)/q) \cdot \epsilon) \cdot \ln(8q_H / ((1 - (q_S + q_H)/q)\epsilon))$ ,  $(1 - (q_S + q_H)/q) \cdot \epsilon / (8q_H)$ -hard, where  $l$  is the maximum number of signers involved in any group setup,  $\tau_{exp_1}$  and  $\tau_{exp_2}$  denote the time required to compute exponentiations in  $\mathbb{G}_1$  and  $\mathbb{G}_2$  respectively, and  $\tau_{exp_1^i}$  and  $\tau_{exp_2^i}$  denote the time required to compute  $i$ -multi-exponentiations in  $\mathbb{G}_1$  and  $\mathbb{G}_2$  respectively, and  $\tau_{pair}$  denotes the time required to compute a pairing operation.*

ASM scheme of Boneh, Drijvers and Neven was proved to be secure in Theorem 2.1.2 by [12]. This scheme is 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 (2.1) 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  $i = 1, 2, \dots, n$ .

The *proof of possession* (PoP) of the secret keys is also discussed in [12]. The ASM scheme with PoP includes each user's signature on their 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  $H_3 : \{0, 1\}^* \rightarrow \mathbb{G}_1$  for  $i = 1, 2, \dots, n$ . Then each user has a secret key  $sk_i$  and the public key pair  $(y_i, \pi_i)$ . 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  $H_4 : \{0, 1\}^* \rightarrow \mathbb{Z}_q$  is another hash function. And then, the aggregated public key is  $apk = (Y, h)$ . The signature generation, aggregation and verification phases are the same as 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 verification. In the PoP variant of ASM scheme [12], each user's public key consists of two group elements, and each user computes two extra pairings before computing the aggregated public key  $apk$ .

## 2.2 Preliminaries for compartment-based and hierarchical threshold delegated verifiable accountable subgroup multi-signatures

This section defines notions we mostly use in Chapter 4. In particular, we define proxy signature schemes, Shamir's secret sharing scheme, Feldman's verifiable secret sharing scheme, and hierarchical threshold secret sharing scheme.

In [11], the authors give a detailed definition of a proxy signature and define the first formal security model for this notion. Below we give the formal definition of proxy signature scheme according to [11].

**Definition 2.2.1** (Proxy signature scheme [11]). *A proxy signature scheme is a tuple  $\mathcal{PS} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V}, (\mathcal{D}, \mathcal{P}), \mathcal{PS}, \mathcal{PV}, \mathcal{ID})$ , where  $\mathcal{DS} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V})$  are as defined in Definition 2.1.4, and the other components are defined below:*

- $(\mathcal{D}, \mathcal{P})$  is an interactive proxy-designation protocol composed of a pair of algorithms, i.e.,  $\mathcal{D}$  and  $\mathcal{P}$ . Let  $i, j$  be the designator and the proxy, respectively.
  - $\mathcal{D}(pk_i, sk_i, j, pk_j, \omega)$  takes both sides' public keys, the designator's secret key, the proxy's identity, and the message space  $\omega$  as inputs, and gives no local outputs.
  - $\mathcal{P}(pk_j, sk_j, pk_i)$  takes public keys of both sides and the secret key of the proxy and outputs proxy signing key  $skp$  after the interaction with  $\mathcal{D}$ .
- $\mathcal{PS}(skp, M)$  takes proxy signing key  $skp$  and the message  $M$  as inputs, and outputs a proxy signature  $p\sigma$ .
- $\mathcal{PV}(pk, M, p\sigma)$  takes public key  $pk$ , a message  $M$ , and a proxy signature  $p\sigma$  as inputs, and outputs accept or reject.
- $\mathcal{ID}(p\sigma)$  takes a proxy signature  $p\sigma$  and outputs an identity of a signer  $i$  or  $\perp$ .

We will compare the properties of our constructions with the functionalities given in Definition 2.2.1 in Section 4.2.2.

### 2.2.1 Shamir's Secret Sharing (SSS) Scheme

Shamir's secret sharing (SSS) scheme [58] is a protocol that is used for sharing a secret among some predetermined players. Assume that we have  $n$  players. Let  $\mathbb{F}_q$  be a finite field with prime order  $q$ . The dealer delivers the shares as follows:

- Chooses a polynomial over  $\mathbb{F}_q$  of degree  $t - 1 < q$ ,

$$f(x) = \alpha_{t-1}x^{t-1} + \dots + \alpha_1x + \alpha_0$$

with  $\alpha_k \in \mathbb{F}_q$  for  $k = 0, \dots, t - 1$ , where  $\alpha_0$  is the secret to be shared.

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

If at least  $t$  or more players perform Lagrange interpolation with their shares, they can uniquely determine the secret polynomial  $f(x)$  and  $f(0)$  will yield the secret.

### 2.2.2 Feldman's Verifiable Secret Sharing (VSS) Scheme

Feldman's verifiable secret sharing (VSS) scheme [25] is a protocol that is used for sharing a secret in a verifiable fashion, where Shamir's secret sharing scheme [58] is 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, the dealer computes commitments with the coefficients of the secret polynomial so that users can verify that they receive 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 over  $\mathbb{F}_q$  of degree  $t - 1 < q$ ,

$$f(x) = \alpha_{t-1}x^{t-1} + \dots + \alpha_1x + \alpha_0$$

with  $\alpha_k \in \mathbb{F}_q$  for  $k = 0, \dots, t - 1$ , where  $\alpha_0$  is the secret to be shared.

- Computes a set of commitments  $\text{COM} = \{C_k : C_k = g^{\alpha_k}, k = 0, 1, \dots, t - 1\}$ .
- Sends  $f(i)$  and  $\text{COM}$  to the  $i$ -th player for  $i = 1, 2, \dots, n$ .

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

$$g^{f(i)} \stackrel{?}{=} \prod_{k=0}^{t-1} C_k^{i^k}. \quad (2.3)$$

The received share is consistent only if (2.3) is satisfied. The reconstruction of the secret is identical to Shamir's secret sharing scheme. The users can uniquely determine the secret polynomial by applying the Lagrange interpolation if and only if threshold  $t$  is satisfied.

**Definition 2.2.2** (Lagrange Interpolation). *Given  $t$  points  $(x_i, y_i)$  for distinct  $x_i$ 's and  $i = 1, \dots, t$ , the unique polynomial of degree  $t - 1$  which satisfies all the points is the linear combination of Lagrange basis polynomials, and given by the equation*

$$P(x) = \sum_{i=1}^{t-1} y_i \ell_i(x), \quad (2.4)$$

where the Lagrange basis polynomial  $\ell_i(x)$  is given by the equation

$$\ell_i(x) = \prod_{\substack{0 \leq k \leq t-1 \\ i \neq k}} \frac{x - x_k}{x_i - x_k} \quad (2.5)$$

Now we define another notion called the Lagrange coefficient, which we will use in the signature aggregation and verification phases of our constructions in Section 4.1.2 and 4.1.3.

**Definition 2.2.3** (Lagrange Coefficient). *Given a set of  $t$  points  $(x_i, y_i)$  for distinct  $x_i$ 's and  $i = 1, \dots, t$ , the Lagrange coefficient  $\lambda_i$  is the evaluation of the the Lagrange basis polynomial  $\ell_i(x)$  at 0, i.e.,*

$$\lambda_i = \ell_i(0) = \prod_{\substack{0 \leq k \leq t-1 \\ i \neq k}} \frac{-x_k}{x_i - x_k}. \quad (2.6)$$

### 2.2.3 Hierarchical Threshold Secret Sharing Scheme

Hierarchical threshold secret sharing schemes in [32, 63, 64] were proposed for sharing secrets in a partitioned structure of users. Assume that we have a set  $\mathcal{G}$  of  $n$  players, which is composed of  $m$  disjoint subsets,  $\mathcal{G} = \bigcup_{i=1}^m \mathcal{G}_i$  where  $\mathcal{G}_i \cap \mathcal{G}_j = \emptyset$  for  $i \neq j$ . Let  $0 < k_1 < \dots < k_m$  be a sequence of integers. The access structure  $\Gamma$  of the hierarchical threshold secret sharing scheme in [32] is

$$\Gamma = \left\{ \mathcal{V} \subset \mathcal{G} : \left| \mathcal{V} \cap \left( \bigcup_{j=1}^i \mathcal{G}_j \right) \right| \geq k_i \forall i \in \{1, 2, \dots, m\} \right\}$$

Like in Shamir's SSS, the dealer delivers the shares as follows:

- Chooses a random polynomial of degree  $k_m$

$$f(x) = \sum_{i=0}^{k_m} \alpha_i x^i$$

such that  $\alpha_0 = s$ .

- Sends a share  $(x_u, f^{(k_{i-1}+1)}(x_u))$  to the user  $u \in \mathcal{G}_i$ , where  $f^{(k_{i-1}+1)}$  is the  $(k_{i-1} + 1)$ -th derivative of  $f$ ,  $k_0 = -1$  and  $x_u \in \mathbb{F}_q$  is a part of the share corresponding to the user  $u$ .

The secret reconstruction is performed by Birkhoff interpolation defined below only if the level-specific thresholds are satisfied.

**Definition 2.2.4** (Birkhoff Interpolation). *Let the triplet  $\langle X, E, C \rangle$  be as follows:*

- $X = \{x_1, \dots, x_k\}$  be a given set of points in  $\mathbb{R}$ , where  $x_1 < x_2 < \dots < x_k$ ,
- $E = (e_{i,j})$  for  $i = 0, \dots, k$  and  $j = 0, \dots, \ell$  be a matrix with binary entries,  $I(E) = \{(i, j) : e_{i,j} = 1\}$ ,  $d = |I(E)|$ , and
- $C = \{c_{i,j} : (i, j) \in I(E)\}$  be a set of  $d$  real values (we assume hereafter that the right-most column in  $E$  is nonzero).

Then, the Birkhoff interpolation problem that corresponds to the triplet  $\langle X, E, C \rangle$  is the problem of finding a polynomial  $P(x) \in \mathbb{R}_{d-1}[x]$  that satisfies the  $d$  equalities

$$P^{(j)}(x_i) = c_{i,j}, (i, j) \in I(E). \quad (2.7)$$

The matrix  $E$  is called the interpolation matrix [63].

We summarize the Birkhoff interpolation method as described in [24]. Let  $\phi = \{g_0, g_1, \dots, g_{d-1}\}$  be a system of linearly independent,  $d - 1$  times continuously differentiable real-valued, functions and  $I'(E) = \{\alpha_i : i = 1, \dots, d\}$  be a vector that is obtained by lexicographically ordering of entries of  $I(E)$ . Furthermore, let  $\alpha_i(1)$  and  $\alpha_i(2)$  denote the first and second elements of the pair  $\alpha_i \in I'(E)$ . Finally, let  $C' = \{c'_i : i = 1, \dots, d\}$  be another vector obtained by lexicographically ordering entries of  $C$  (according to the indexes of elements in  $C$ ).

According to the above definition and clarifications, the Birkhoff interpolation problem is solved by the below equation:

$$P(x) = \sum_{j=0}^{d-1} \frac{\det(A(E, X, \phi_j))}{\det(A(E, X, \phi))} g_j(x), \quad (2.8)$$

where

$$A(E, X, \phi_j) = (\theta_{ij})_{d \times d}, \quad (2.9)$$

$\theta_{ij} = g_{j-1}^{(\alpha_i(2))}(x_{\alpha_i(1)})$  for  $i, j = 1, \dots, d$ , and  $A(E, X, \phi)$  can be computed by replacing  $(j + 1)$ -th column of matrix (2.9) with  $C'$ . An explicit example of the application of Birkhoff interpolation using (2.8) can be found in [24].

Although the Birkhoff interpolation problems can be solved by (2.8), we cannot directly use this method. In our constructions, we use Birkhoff interpolation for signature aggregation and verification of the aggregated signature. In order to compute (2.8), any combiner and verifier need to know the sufficient number of shares. However, our constructions require only the shareholders to know their shares, and no one should learn the shares of others. To this end, we use the modified version of (2.8), which is also given in [24]:

$$P(x) = \sum_{i=0}^{d-1} c'_{i+1} \left( \sum_{j=0}^{d-1} (-1)^{(i+j)} \frac{\det(A_i(E, X, \phi_j))}{\det(A(E, X, \phi))} g_j(x) \right) \quad (2.10)$$

Now we can define the Birkhoff coefficient, which we will use in the signature aggregation and verification phases of our last construction in Section 4.1.4.

**Definition 2.2.5.** *Let the triplet  $\langle X, E, C \rangle$  be in Definition 2.2.4 then the Birkhoff coefficient  $\beta_i$  is the evaluation of the polynomial*

$$P_i(x) = \sum_{j=0}^{d-1} (-1)^{(i+j)} \frac{\det(A_i(E, X, \phi_j))}{\det(A(E, X, \phi))} g_j(x),$$

at 0, i.e.,  $\beta_i = P_i(0)$ .

## 2.2.4 Security of the secret sharing schemes

Shamir's secret sharing (SSS) scheme [58] is known to be information-theoretically secure, i.e., secure against even computationally unbounded adversaries. The hierarchical threshold secret sharing scheme (HTSS) [32, 63, 64] is a generalization of Shamir's SSS and is also information-theoretically secure. However, suppose an adversary can behave like a shareholder and interact actively with real shareholders. In that case, he can obtain the secret as described in [65] by Tompa and Woll. The attack works as follows. Consider an adversary behaving actively. He chooses a random *false* share and performs an interaction for reconstruction with  $t - 1$  honest shareholders. As a result, he will not obtain the secret because there are  $t - 1$  true shares where there must be at least  $t$ . However, he obtains a value, let us say  $s'$ . Then he interacts with another group of  $t - 1$  honest shareholders and gets another value  $\hat{s}$ . He

can compute the secret without having a genuine share using the values  $s'$  and  $\hat{s}$  and corresponding Lagrange coefficients. To avoid this attack, one should force the shareholders to behave passively. For example, using Feldman's verifiable secret sharing (VSS) scheme [25] would be a solution. Before the reconstruction phase, one checks whether the shares to be interpolated are consistent with the shared secret. This will avoid the active adversary attack defined in [65]. On the other hand, Feldman's VSS has its own security risks.

As we stated before, Feldman's VSS scheme uses SSS, and so it has the same security arguments about sharing and reconstruction phases. However, in the committing phase, it is not information-theoretically secure anymore. The commitment set contains  $C_0 = g^s$ , where  $g$  is the generator for the cyclic group, and  $s$  is the secret to be shared. This commitment may leak information about the secret  $s$ . The security of the commitments depends on the Discrete Logarithm Problem (DLP), defined over cyclic groups. In some cyclic groups, even with a large order, DLP may not be as hard as it is supposed to be. Therefore the space that we are working in should be chosen carefully. In this paper, all the schemes that we propose are pairing-based constructions. In the literature, there are many secure and efficient pairing-friendly curves that we can choose.

### 2.3 Preliminary for lattice-based constructions

In this section, we give the notation, assumptions, definitions of hard problems and the protocols used in our construction that is presented in Chapter 5.

#### 2.3.1 Assumptions and the notation

Assume that  $R = \mathbb{Z}[x]/(f(x))$  and  $R_q = \mathbb{Z}_q[x]/(f(x))$  where  $N$  is a power of two and  $f(x) = x^N + 1$  is the  $2N$ -th cyclotomic polynomial as in [20, 23]. As common in lattice-based cryptography, through out this report we use *centered reduction*  $\text{mod}^{\pm}q$ , i.e., for any  $a \in \mathbb{Z}_q$ ,  $\bar{a} = a \text{ mod}^{\pm}q$  is defined to be a unique integer in the range  $[-\frac{q-1}{2}, \frac{q-1}{2}]$ . We write the ring elements with lower-case letters and column vector of elements with bold ones, i.e.,  $u \in R$  and  $\mathbf{u} \in R^k$ , respectively. For a set  $X$  we write

$x \stackrel{\S}{\leftarrow} X$  to show that  $x$  is sampled from the uniform distribution defined over the set  $X$ .

For any  $u(x) = u_0 + u_1x + \dots + u_{N-1}x^{N-1} \in R_q$ ,  $\ell^\infty$  norm is defined to be  $\|u\|_\infty = \max_i |u_i|$  for  $i = 0, \dots, N-1$ , and  $\ell^2$  norm of  $u$  is defined to be  $\|u\|_2 = \sqrt{u_0^2 + \dots + u_{N-1}^2}$ . Similarly for  $\mathbf{u} \in R_q^k$ ,  $\|\mathbf{u}\|_\infty = \max_i \|u_i\|_\infty$ , and  $\ell^2$  norm of  $\mathbf{u}$  is defined to be

$$\|\mathbf{u}\|_2 = \sqrt{(\|u_1\|_2)^2 + \dots + (\|u_k\|_2)^2}.$$

Let  $S_\eta \subseteq R$  be the set of small polynomials, i.e., for  $\eta \in \mathbb{Z}$ ,  $S_\eta = \{v \in R : \|v\|_\infty \leq \eta\}$ . Let  $C \subseteq R$  be the challenge space consisting of small polynomials, which will be used as the image of random oracle  $H_0 : \{0, 1\}^* \rightarrow C$ , i.e., for  $\kappa \in \mathbb{Z}$ ,

$$C = \{c \in R : \|c\|_\infty = 1 \wedge \|c\|_1 = \kappa\},$$

where  $\|c\|_1$  is the  $\ell^1$  norm and it is equal to the sum of all the coefficients of the polynomial  $c$ . For constructing such a random oracle, a variant of Fisher-Yates shuffle [26] is used in Dilithium and Dilithium-G ([23, Algorithm 1]). The randomness, which is produced by a collision resistant hash function, is given to the random oracle as input (randomness seed).

We now give the definitions of Module-SIS and Module-LWE problems, which are assumed to be hard in our designs. We begin with the definition of discrete Gaussian distribution.

**Definition 2.3.1** (Discrete Gaussian Distribution over  $R^m$  [20]). *For  $x \in R^m$ , let*

$$\rho_{v,s}(x) = \exp(-\pi\|x - v\|_2^2/s^2)$$

*be a Gaussian function of parameters  $v \in R^m$  and  $s \in R$ . The discrete Gaussian distribution  $D_{v,s}^m$  centered at  $v$  is*

$$D_{v,s}^m(x) = \rho_{v,s}(x)/\rho_{v,s}(R^m),$$

*where  $\rho_{v,s}(R^m) = \sum_{x \in R^m} \rho_{v,s}(x)$ .*

**Definition 2.3.2** (Module-SIS $_{q,k,\ell,\beta}$  Problem [20]). *Given a random matrix  $\mathbf{A} \in R_q^{k \times \ell}$ , find a vector  $\mathbf{x} \in R_q^{\ell+k}$ , such that*

$$[\mathbf{A} \mid \mathbf{I}] \cdot \mathbf{x} = \mathbf{0} \text{ and } \|\mathbf{x}\|_2 \leq \beta.$$

**Definition 2.3.3** (Module-LWE $_{q,k,\ell,\eta}$  Problem [20]). *Given a pair  $(\mathbf{A}, \mathbf{t}) \in R_q^{k \times \ell} \times R_q^k$ , decide whether it is selected uniformly at random from  $R_q^{k \times \ell} \times R_q^k$  or it is generated in a way that  $\mathbf{t} := [\mathbf{A}|\mathbf{I}] \cdot \mathbf{s}$  for some  $\mathbf{s} \xleftarrow{\$} S_\eta^\ell \times S_\eta^k$ .*

### 2.3.2 Dilithium-G

In 2022, the National Institute of Standards and Technology (NIST) announced the winner signature schemes for post-quantum cryptography standardization process. The CRYSTALS-Dilithium signature scheme [23] is one of the winner signature schemes which is based on module-LWE and module-SIS problems. In the original paper, authors proposed two variants, Dilithium and Dilithium-G. The main difference between these variants is the distribution that the variants sample random elements from and the way of rejection sampling procedure. In the Dilithium scheme, the sampling procedure is done from a uniform distribution whereas the Dilithium-G uses a Gaussian one. The Dilithium-G scheme has better parameters in terms of size and number of repetitions. However, it is noted that it has weaknesses against side channel attacks [23].

Below we state a simplified version of Dilithium-G, and then we restate a few lemmata for the signature parameters (see Table 5.1), which were defined in [20].

---

#### Algorithm 1 Key Generation

---

**Require:**  $par := \{R_q, k, \ell, \eta, B, s, M\}$

**Ensure:**  $(sk, pk)$

- 1:  $\mathbf{A} \xleftarrow{\$} R_q^{k \times \ell}$
  - 2:  $\bar{\mathbf{A}} = [\mathbf{A}|\mathbf{I}] \in R_q^{k \times (\ell+k)}$ , where  $\mathbf{I}$  is the  $k \times k$  identity matrix
  - 3:  $(\mathbf{s}_1, \mathbf{s}_2) \xleftarrow{\$} S_\eta^\ell \times S_\eta^k$ ,  $\mathbf{s} := \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}$
  - 4:  $\mathbf{t} := \bar{\mathbf{A}} \cdot \mathbf{s}$
  - 5:  $pk = (\mathbf{A}, \mathbf{t})$  and  $sk = \mathbf{s}$
  - 6: **return**  $(sk, pk)$
-

---

**Algorithm 2 Signature Generation**

---

**Require:**  $sk = \mathbf{s}, pk = (\mathbf{A}, \mathbf{t}), \mu, par := \{R_q, k, \ell, \eta, B, s, M\}$

**Ensure:** A signature pair  $(\mathbf{z}, c)$

- 1:  $(\mathbf{y}_1, \mathbf{y}_2) \xleftarrow{\$} D_s^\ell \times D_s^k, \mathbf{y} := \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$
  - 2:  $\mathbf{w} := \bar{\mathbf{A}} \cdot \mathbf{y}$
  - 3:  $c \leftarrow H_0(\mathbf{w}, \mu, pk)$
  - 4:  $\mathbf{z} := c\mathbf{s} + \mathbf{y}$
  - 5: With probability  $\min(1, D_s^{\ell+k}(\mathbf{z}) / (M \cdot D_{cs,s}^{\ell+k}(\mathbf{z})))$ :
  - 6:     **return**  $(\mathbf{z}, c)$
  - 7: Restart otherwise
- 

---

**Algorithm 3 Verification**

---

**Require:**  $(\mathbf{z}, c), pk = (\mathbf{A}, \mathbf{t}), \mu, par := \{R_q, k, \ell, \eta, B, s, M\}$

**Ensure:** Accept or Reject

- 1: **if**  $\|\mathbf{z}\|_2 \leq B$  and  $c = H_0(\bar{\mathbf{A}}\mathbf{z} - c\mathbf{t}, \mu, pk)$  **then**
  - 2:     **return** Accept
  - 3: **end if**
  - 4: Otherwise, **return** Reject
- 

Aggregating individual signatures inevitably results in a larger size of multi-signature. According to Lemma 1 of [20], the upper bound for the size of the sum of  $n$  Gaussian samples is  $\sqrt{n} \cdot B$ , where  $B$  is the upper bound for only one sample and we restate it below.

**Lemma 2.3.1.** (*Sum of Discrete Gaussian Samples [20, Lemma 1]*) Suppose  $s$  exceeds the smoothing parameter by a factor  $\geq \sqrt{2}$ . Let  $x_i$  for  $i \in [n]$  be independent samples from the distribution  $D_s^m$ . Then the distribution of  $x = \sum_i x_i$  is statistically close to  $D_{s\sqrt{n}}^m$ .

According to [42] the upper bound for the size of a single signature is computed by  $B = \gamma\sigma\sqrt{(\ell+k)N}$ . For ensuring the size of a single signature stays below this bound with high probability, one should set  $\gamma > 1$ .

**Lemma 2.3.2** ([20, Lemma 2] and [42, Lemma 4.4]). *For any  $\gamma > 1$ ,*

$$Pr[\|z\|_2 > B = \gamma\sigma\sqrt{mN} : \mathbf{z} \stackrel{\$}{\leftarrow} D_s^m] < \gamma^{mN} e^{mN(1-\gamma^2)/2}.$$

The following lemma given in [42] determines both the standard deviation and the value  $M$ , i.e., the number of repetition of signing attempts until producing a valid signature. Note that setting  $\alpha = 11$  and  $t = 12$  results in  $M = 3$  which provides  $\epsilon < e^{-100}$ , i.e., at least 100-bit security.

**Lemma 2.3.3** ([20, Lemma 3], and [42, Lemma 4.5]). *For  $\mathbf{V} \subseteq R^m$ , let  $T = \max_{v \in \mathbf{V}} \|v\|_2$ . Fix some  $t$  such that  $t = \omega\sqrt{\log(mN)}$  and  $t = o(\log(mN))$ . If  $\sigma = \alpha T$  for any positive  $\alpha$ , then*

$$Pr[M > D_s^m(\mathbf{z})/D_{v,s}^m(\mathbf{z}) : \mathbf{z} \stackrel{\$}{\leftarrow} D_s^m(\mathbf{z})] \geq 1 - \epsilon$$

where  $M = e^{t/\alpha + 1/(2\alpha^2)}$  and  $\epsilon = 2e^{-t^2/2}$ .

The proofs of the above lemmas can be found in [42].

### 2.3.3 MS<sub>2</sub> Scheme

In [20], authors proposed 3 novel multi-signature schemes and a trapdoor commitment scheme. One of those multi-signature schemes is 3-round  $n$ -out-of- $n$  multi-signature scheme (DS<sub>3</sub>) that utilizes Baum et al.'s additively homomorphic commitment scheme [8]. The second one is a 2-round version of the first one (DS<sub>2</sub>). The only difference is that they use a novel additively homomorphic trapdoor commitment scheme which is indeed a modified version of Baum et al.'s commitment scheme. Moreover, a 2-round multi-signature scheme called MS<sub>2</sub> was proposed in the same paper which is also constructed by the help of the above mentioned trapdoor commitment scheme (TCOM). Below we restate the 2-round MS<sub>2</sub> scheme proposed by Damgård et al. in [20].

1. **Key Generation:** Key generation phase is identical to the **Algorithm 1**. For choosing the matrix  $\mathbf{A} \in R_q^{k \times \ell}$ , a trusted third party can do it or users can jointly compute it interactively.

2. **Signature Generation:** Let  $H_0 : \{0, 1\}^* \rightarrow C$  and  $H_3 : \{0, 1\}^* \rightarrow S_{ck}$  be two random oracles, where  $S_{ck}$  is the commitment key space as defined in [20]. Let the functions **Commit** and **Open** belong to the additively homomorphic trapdoor commitment scheme TCOM which was proposed by Damgård et al. in [20].

- (a) Compute the message-specific commitment key  $ck \leftarrow H_3(\mu, L)$ , where  $\mu$  is the message and  $L = \{pk_1, \dots, pk_n\}$ .
- (b) Pick a random  $\mathbf{y}_i \xleftarrow{\$} D_s^{\ell+k}$  and compute  $\mathbf{w}_i = \bar{\mathbf{A}}\mathbf{y}_i$ , where  $\bar{\mathbf{A}} = [\mathbf{A}|\mathbf{I}] \in R_q^{k \times (\ell+k)}$ .
- (c) Compute  $com_i \leftarrow \text{Commit}_{ck}(\mathbf{w}_i, r_i)$ , where  $r_i$  is a discrete Gaussian vector sampled from the Gaussian distribution  $D(S_r)$  defined in the trapdoor commitment scheme in [20].
- (d) Send  $com_i$  to the other users.
- (e) Upon receiving  $com_j$  for  $i \neq j$ , compute  $com := \sum_{j \in \mathcal{G}} com_j$
- (f) Compute  $c_i \leftarrow H_0(\mathbf{t}_i, com, \mu, L)$
- (g) Compute his individual signature  $\mathbf{z}_i = c_i \mathbf{s}_i + \mathbf{y}_i$
- (h) Run the rejection sampling on input  $(c_i \mathbf{s}_i, \mathbf{z}_i)$ , i.e., with probability

$$\min(1, D_s^{\ell+k}(\mathbf{z}_i) / (M \cdot D_{c_i \mathbf{s}_i, s}^{\ell+k}(\mathbf{z}_i)))$$

send individual signature  $(\mathbf{z}_i, r_i)$  to all other users, otherwise send out RESTART and go to (b).

- (i) After receiving RESTART from some user go to (b). Otherwise, aggregate the individual signatures  $(\mathbf{z}_j, r_j)$  for  $j \neq i$  as follows:
  - i. Compute  $c_j \leftarrow H_0(\mathbf{t}_j, com, \mu, L)$  for all  $j \neq i$ , and compute  $\mathbf{w}_j := \bar{\mathbf{A}}\mathbf{z}_j - c_j \mathbf{t}_j$ .
  - ii. Check  $\|\mathbf{z}_j\|_2 \leq B$  and  $\text{Open}_{ck}(com_j, r_j, \mathbf{w}_j) = 1$ .
  - iii. If the check fails for some  $j$  send out ABORT.
  - iv. Compute  $\mathbf{z} = \sum_{j \in \mathcal{G}} \mathbf{z}_j$  and  $r := \sum_{j \in \mathcal{G}} r_j$
- (j) If the protocol does not ABORT the multi-signature is  $(com, \mathbf{z}, r)$ .

3. **Signature Verification:** Given  $\{(com, \mathbf{z}, r), \mu, L = \{pk_1, \dots, pk_n\}\}$  one can verify the multi-signature as follows:

- (a) Compute  $c_j := \mathbf{H}_0(\mathbf{t}_j, com, \mu, L)$  for  $j \in \mathcal{G}$ , and reconstruct  $\mathbf{w} := \bar{\mathbf{A}}\mathbf{z} - \sum_{j \in \mathcal{G}} c_j \mathbf{t}_j$ .
- (b) Accept if  $\|\mathbf{z}\|_2 \leq B_n$  and  $\text{Open}_{ck}(com, r, \mathbf{w}) = 1$ , where  $B_n = \sqrt{n}B$ .

Throughout the next section, in order to avoid confusion we follow the same notation of [20] and the parameters in Table 5.1.

## CHAPTER 3

### PAIRING-BASED ACCOUNTABLE SUBGROUP MULTI-SIGNATURES WITH VERIFIABLE GROUP SETUP

In this chapter, we focus on pairing-based accountable subgroup multi-signatures (ASM). We propose three novel accountable subgroup multi-signature schemes based on pairings. We prove that our proposed schemes are secure against existential forgery under chosen-message attacks, under co-CDH/ $\psi$ -co-CDH assumption in the random oracle model. The first one is the vASM (verifiable ASM) scheme which is a modified BLS signature. We give a method of generating a membership key via VSS protocol [25], which transforms the BLS signature scheme into an ASM scheme. The proposed vASM scheme, which also solves the open problem in [46], requires fewer multiplications and bilinear pairings than the ASM schemes proposed in [12]. The second one is ASMwSA (ASM with Subgroup Authentication), in which the subgroup of the signers is known before the protocol starts. So the members participating in authentication are decided by the subgroup itself. The third one is ASMwCA (ASM with Combiner Authentication) which also provides a solution to the open problem in [46], in which we construct a scheme such that the subgroup of signers  $\mathcal{S}$  is not predetermined. The ASMwSA and ASMwCA schemes also require fewer multiplications and bilinear pairings than the ASM scheme in [12]. Moreover, we give a method of consecutive and cumulative signing that eliminates the designated combiner in case the subgroup of signers  $\mathcal{S}$  is known before the signature generation. Further, we discuss the aggregated versions of vASM, ASMwSA and ASMwCA schemes for  $N$  distinct accountable subgroup multi-signatures. The aggregated versions of our schemes, i.e., AvASM, AASMwSA and AASMwCA, output aggregated signatures with the size of a single group element and require  $N + 1$  pairings for aggregated signature verifica-

tion, in comparison with the partial aggregated AASM scheme proposed in [12] with the signature size of  $N + 1$  group elements and verification with  $2N + 1$  pairings. One can also find the details in [3].

### 3.1 vASM: An ASM scheme with VSS based group setup

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  $f_i(x) = \alpha_{n-1}^{(i)}x^{n-1} + \dots + \alpha_1^{(i)}x + \alpha_0^{(i)} \in \mathbb{Z}_q[x]$ , where  $\alpha_0^{(i)} = sk_i$  and  $\alpha_k^{(i)}$ 's are all nonzero and distinct, for  $k = 1, \dots, n-1$ .
  - Computes the set of commitments  $\text{COM}_i := \{C_k^{(i)} = g_2^{\alpha_k^{(i)}} \mid k = 0, \dots, n-1\}$ .
  - Sends  $(f_i(j), \text{COM}_i)$  to  $j$ -th user in  $\mathcal{G}$ , for  $j = 1, \dots, n$ .
  - After receiving  $(f_j(i), \text{COM}_j)$ ,
    - computes the membership key  $mk_i = \sum_{j \in \mathcal{G}} f_j(i)$ .
    - computes  $\text{COM} := \{C_k = \prod_{j \in \mathcal{G}} C_k^{(j)} \mid k = 0, \dots, n-1\}$ .
  - 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 and set of membership public keys  $mpk_i$ 's public. Define  $\text{MPK} = \{mpk_i\}_{i \in \mathcal{G}}$ . Note that these public keys can also be computed by the verifier, i.e.  $mpk_i = g_2^{mk_i} = \prod_{k=0}^{n-1} C_k^{i^k}$ .
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  $\{par, MPK, COM, \mathcal{S}, m, \sigma\}$ , can verify the signature  $\sigma$  by checking

$$e(H_0(m), \prod_{i \in \mathcal{S}} mpk_i) \stackrel{?}{=} e(\sigma, g_2). \quad (3.1)$$

Note that a simple proof of concept implementation can be found in [2]. Correctness of the vASM scheme follows from the following equation array.

$$\begin{aligned} e(H_0(m), \prod_{i \in \mathcal{S}} mpk_i) &= e(H_0(m), \prod_{i \in \mathcal{S}} g_2^{mk_i}) \\ &= e(H_0(m), g_2^{\sum_{i \in \mathcal{S}} mk_i}) \\ &= e(H_0(m)^{\sum_{i \in \mathcal{S}} mk_i}, g_2) \\ &= e(\prod_{i \in \mathcal{S}} H_0(m)^{mk_i}, g_2) \\ &= e(\prod_{i \in \mathcal{S}} s_i, g_2) \\ &= e(\sigma, g_2) \end{aligned}$$

**Remark 3.1.1.** *Unlike the threshold multi-signatures [10, 21, 28, 29], ASM schemes [12, 46] 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  $|\mathcal{S}|$  [46].*

**Remark 3.1.2.** *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  $i, j = 1, 2, \dots, n$ , the signature  $\sigma$  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  $\mathcal{G}$ .*

**Remark 3.1.3.** *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 the vASM scheme, no user can set a special rogue key.

**Remark 3.1.4.** The subgroups  $\mathbb{G}_1$  and  $\mathbb{G}_2$  are interchangeable. This means one can choose to have membership public keys belong to  $\mathbb{G}_1$  and the signature belong to  $\mathbb{G}_2$ , or vice versa. For sure, the choice depends on the system that we want to utilize the vASM scheme. In this thesis, we assumed that the signature belongs to  $\mathbb{G}_1$  and the membership public keys belong to  $\mathbb{G}_2$  for having relatively smaller signatures.

**Remark 3.1.5.** We consider the case that the membership public keys, i.e.,  $mpk_i$  for  $i = 1, 2, \dots, n$ , are published. We could also write the verification equation (3.1) as

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

However, in this case, the verifier would incur the cost of  $n$  exponentiations in  $\mathbb{G}_2$  (or  $\mathbb{G}_1$  according to the choice, see the previous remark).

**Remark 3.1.6.** 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 user's secret key since the secret polynomials are of degree  $n - 1$  (see Section 2.2.2). In order to avoid this vulnerability, the polynomials  $f_i$  in the group setup phase of the vASM scheme can be set to degree  $r \geq n - 1$ . In this way, at most  $r - n + 1$  newcomers can be registered to the group  $\mathcal{G}$  without resetting the group setup. On the other hand, this costs extra computational complexity at the group setup phase.

**Remark 3.1.7.** The membership key  $mk_i$  in the 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 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  $mk_i$ .

### 3.1.1 Security

We follow the security reduction of the BLS signature scheme given in [14] with a few modifications to prove the security of the proposed scheme in Section 3.1. First, we give Theorem 3.1.1 which states the security reduction of our proposed vASM scheme.

**Theorem 3.1.1.** *If the  $\psi$ -co-CDH problem (Definition 2.1.3) in  $\mathbb{G}_1 \times \mathbb{G}_2$  is  $(t', \epsilon')$ -hard, then vASM scheme is  $(t, q_H, q_S, \epsilon)$ -secure (see Definition 2.1.7) against existential forgery under adaptive chosen message attacks in the random oracle model for all  $t$  and  $\epsilon$  satisfying*

$$t \leq t' - t_{exp1}(q_H + (n + 1)q_S) \text{ and } \epsilon \geq e(q_S + 1) \cdot \epsilon',$$

where  $t_{exp1}$  is the time required by an exponentiation in  $\mathbb{G}_1$  and  $n$  is the maximum number of potential signers involved in a vASM signature.

*Proof.* Let  $\mathcal{F}$  be a forger that  $(t, q_H, q_S, \epsilon)$ -breaks the vASM signature scheme. We construct an algorithm  $\mathcal{A}$  which  $(t', \epsilon')$ -breaks the  $\psi$ -co-CDH problem (see Definition 2.1.3). Let  $\mathbb{G}_1$  and  $\mathbb{G}_2$  be two groups and  $g_1, g_2$  be their generators, respectively as in Definition 2.1.1. Algorithm  $\mathcal{A}$  is given a triplet  $(g_2, A = g_2^\alpha, B = g_1^\beta)$ , and access to the oracle  $\mathcal{O}^\psi(\cdot)$  which takes  $g_2^x \in \mathbb{G}_2$  as input and outputs  $g_1^x \in \mathbb{G}_1$ . We assume without loss of generality that the set of indices of our target group is  $\{1, 2, \dots, n\}$ .

**Setup.** Algorithm  $\mathcal{A}$  proceeds as follows.

- It chooses  $n$  random  $r_j \xleftarrow{\$} \mathbb{Z}_q$  for  $j = 1, 2, \dots, n$ .
- It computes corresponding  $n$  membership public keys  $mpk_j = A \cdot g_2^{r_j}$  for  $j = 1, \dots, n$ .
- Algorithm  $\mathcal{A}$  computes the commitment set  $\text{COM} = \{C_0, \dots, C_{n-1}\}$  with respect to the membership public keys using the system of equations below.

$$\begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & 2 & \dots & 2^{n-1} \\ & & \vdots & \\ 1 & n & \dots & n^{n-1} \end{bmatrix} \cdot \begin{bmatrix} C_0 \\ C_1 \\ \vdots \\ C_{n-1} \end{bmatrix} = \begin{bmatrix} mpk_1 \\ mpk_2 \\ \vdots \\ mpk_n \end{bmatrix}$$

Because the above leftmost matrix is an  $n \times n$  Vandermonde matrix and its entries are the powers of the indices of the membership public keys, i.e., they are distinct; hence it is invertible. Since the rightmost matrix is known,  $\mathcal{A}$  can compute the matrix in the middle, i.e., the commitment set  $\text{COM} = \{C_0, C_1, \dots, C_{n-1}\}$ . (Note that we operate in additive groups, but we use a multiplicative notation as common in the literature. Therefore, matrix operation above gives the desired result for the vASM commitment set.)

- It gives the set of membership public keys  $\text{MPK} = \{mpk_1, \dots, mpk_n\}$  and the commitment set  $\text{COM}$  to the forger  $\mathcal{F}$ .

**Hash query.** Algorithm  $\mathcal{A}$  maintains a list  $\mathcal{L}$  of tuples  $\langle m_i, w_i, b_i, c_i \rangle$  for the  $i$ -th query, where  $m_i, w_i, b_i, c_i$  are defined below. The list  $\mathcal{L}$  is initially empty, and when  $\mathcal{F}$  queries the oracle  $H$ , for a value  $m \in \{0, 1\}^*$ ,  $\mathcal{A}$  responds as follows.

1. If the query  $m_i$  has already been made before,  $\mathcal{A}$  looks up to the list  $\mathcal{L}$ , finds the tuple  $\langle m_i, w_i, b_i, c_i \rangle$ , and responds with  $H(m_i) = w_i \in \mathbb{G}_1$ .
2. Otherwise,  $\mathcal{A}$  generates a random  $c_i \xleftarrow{\$} \{0, 1\}$  with probability  $\Pr[c_i = 0] = 1/(q_S+1)$ . Then,  $\mathcal{A}$  picks a random  $b_i \xleftarrow{\$} \mathbb{Z}_q$ , and computes  $w_i \leftarrow B^{1-c_i} \psi(g_2)^{b_i}$ .
3. It adds the tuple  $\langle m_i, w_i, b_i, c_i \rangle$  to the list  $\mathcal{L}$ .

**Signature query.**  $\mathcal{A}$  responds to  $\mathcal{F}$ 's  $i$ -th signature query  $(m_i, \mathcal{S}_i)$ , where  $m_i$  is the message to be signed and  $\mathcal{S}_i$  is the set of indices of the subgroup of signers, as follows:

1. It runs the above hash query algorithm to get the tuple  $\langle m_i, w_i, b_i, c_i \rangle$ . If  $c_i = 0$ , then it aborts.
2. Otherwise, it defines  $\sigma_i = \prod_{j \in \mathcal{S}_i} \sigma_j$ , where  $\sigma_j = \psi(A)^{b_i} \psi(g_2)^{r_j b_i} \in \mathbb{G}_1$ , and  $j \in \mathcal{S}_i$ . Note that  $\sigma_i = w_i^{\sum_{j \in \mathcal{S}_i} \alpha + r_j}$  and therefore  $\sigma_i$  is a valid vASM signature on

message  $m_i$  by the subgroup  $\mathcal{S}_i$ . Because

$$\begin{aligned}
e(\sigma_i, g_2) &= e(w_i^{\sum_{j \in \mathcal{S}_i} \alpha + r_j}, g_2) \\
&= e(w_i, g_2^{\sum_{j \in \mathcal{S}_i} \alpha + r_j}) \\
&= e(w_i, \prod_{j \in \mathcal{S}_i} A \cdot g_2^{r_j}) \\
&= e(H(m_i), \prod_{j \in \mathcal{S}_i} mpk_j)
\end{aligned}$$

as in the verification equation of the vASM scheme.

3. Algorithm  $\mathcal{A}$  sends  $\sigma_i$  to forger  $\mathcal{F}$ .

**Output.** Forger  $\mathcal{F}$  outputs a tuple  $(\sigma_f, m_f, \mathcal{S}_f)$ , where  $\sigma_f$  is a valid signature on a message  $m_f$  by a subgroup  $\mathcal{S}_f$ .

1. If the signature query has already been made on message  $m_f$  before, then  $\mathcal{A}$  aborts.
2. If there is no tuple in the list  $\mathcal{L}$  containing  $m_f$ ,  $\mathcal{A}$  runs the hash query algorithm for  $m_f$ .
3.  $\mathcal{A}$  checks if  $\sigma_f$  is a valid signature on  $m_f$  by the signers in  $\mathcal{S}_f$ , i.e.,

$$e(H(m_f), \prod_{j \in \mathcal{S}_f} mpk_j) = e(\sigma_f, g_2).$$

4. If it is not valid,  $\mathcal{A}$  aborts.
5. Otherwise,  $\mathcal{A}$  finds the tuple  $\langle m_f, w, b, c \rangle$  in the list  $\mathcal{L}$ .

- If  $c = 1$ , then  $\mathcal{A}$  aborts.
- Otherwise, we have  $c = 0$  and  $H(m_f) = w = B \cdot \psi(g_2)^b$ . This means that

$$\sigma_f = B^{\sum_{j \in \mathcal{S}_f} \alpha + r_j} \cdot \psi(g_2)^{b \sum_{j \in \mathcal{S}_f} \alpha + r_j}.$$

- Algorithm  $\mathcal{A}$  computes  $B^\alpha$  as follows.

$$B^\alpha = \left( \frac{\sigma_f}{B^{\sum_{j \in \mathcal{S}_f} r_j} \cdot \psi(A)^{b|\mathcal{S}_f|} \cdot \psi(g_2)^{\sum_{j \in \mathcal{S}_f} r_j b}} \right)^{|\mathcal{S}_f|^{-1}}$$

It is easy to check that the value on the right-hand side is indeed equivalent to  $B^\alpha = g_1^{\alpha\beta}$ :

$$\begin{aligned}
B^\alpha &= \left( \frac{\sigma_f}{B^{\sum_{j \in \mathcal{S}_f} r_j} \cdot \psi(A)^{b|\mathcal{S}_f|} \cdot \psi(g_2)^{\sum_{j \in \mathcal{S}_f} r_j b}} \right)^{|\mathcal{S}_f|^{-1}} \\
&= \left( \frac{B^{\sum_{j \in \mathcal{S}_f} \alpha + r_j} \cdot \psi(g_2)^{b \sum_{j \in \mathcal{S}_f} \alpha + r_j}}{g_1^{\sum_{j \in \mathcal{S}_f} \beta r_j} \cdot g_1^{b\alpha|\mathcal{S}_f|} \cdot g_1^{\sum_{j \in \mathcal{S}_f} r_j b}} \right)^{|\mathcal{S}_f|^{-1}} \\
&= \left( \frac{g_1^{\alpha\beta|\mathcal{S}_f|} \cdot g_1^{\beta \sum_{j \in \mathcal{S}_f} r_j} \cdot g_1^{b\alpha|\mathcal{S}_f|} \cdot g_1^{b \sum_{j \in \mathcal{S}_f} r_j}}{g_1^{\beta \sum_{j \in \mathcal{S}_f} r_j} \cdot g_1^{b\alpha|\mathcal{S}_f|} \cdot g_1^{b \sum_{j \in \mathcal{S}_f} r_j}} \right)^{|\mathcal{S}_f|^{-1}} \\
&= (g_1^{\alpha\beta|\mathcal{S}_f|})^{|\mathcal{S}_f|^{-1}} \\
&= g_1^{\alpha\beta}
\end{aligned}$$

Above, we gave the construction of Algorithm  $\mathcal{A}$ . Now we give the success probability and the running time of it. The probability of  $\mathcal{A}$  aborts in the signature query phase is equivalent to the probability that  $c_i = 0$ . From the hash query algorithm we know that  $\Pr[c_i = 0] = 1/(q_S + 1)$ . Therefore the probability that  $\mathcal{A}$  does not abort after the  $i$ -th query will be  $(1 - 1/(q_S + 1))^i$ . Since  $\mathcal{F}$  makes  $q_S$  queries, the probability that  $\mathcal{A}$  does not abort after the  $q_S$ -th query is at least  $(1 - 1/(q_S + 1))^{q_S} \geq 1/e$ . Moreover, by Definition 2.1.7, the probability that  $\mathcal{F}$  outputs a valid forgery is at least  $\epsilon$ . After a valid forgery,  $\mathcal{A}$  aborts if  $c_i = 1$ , which has probability  $1 - 1/(q_S + 1)$ . Therefore, given a valid forgery, the probability that  $\mathcal{A}$  does not abort is  $1/(q_S + 1)$ . Hence, the overall success probability of Algorithm  $\mathcal{A}$  is  $1/e \cdot \epsilon \cdot 1/(q_S + 1)$ , that is  $\epsilon/(e(q_S + 1)) \geq \epsilon'$ , as desired.

Moreover,  $\mathcal{A}$ 's running time is nearly identical to  $\mathcal{F}$ 's running time. In addition to  $\mathcal{F}$ 's running time,  $\mathcal{A}$ 's running time includes the time required to respond to  $(q_H + q_S)$  hash queries and  $q_S$  signature queries. Each requires exponentiation in  $\mathbb{G}_1$ , which takes  $t_{exp_1}$  running time. Note that each hash query requires a single exponentiation while each signature query requires at most  $n$  exponentiations. Therefore the overall running time of the algorithm  $\mathcal{A}$  is  $t + t_{exp_1}((q_H + (n + 1)q_S)) \leq t'$  as desired.

□

### 3.2 Accountable Subgroup Multi-signature Scenarios with Subgroup Authentication

In this section, we propose two more ASM schemes whose group setups are different from the vASM scheme. We slightly modify the group setup method of Boneh et al.'s ASM scheme. In the first one, we use components of a membership key to create a subgroup-specific membership key. In the second one, the users keep a single component secret and send other components to the combiner.

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 2.1.5, any subgroup  $\mathcal{S} \subseteq \mathcal{G}$  of signers are authorized to sign any message on behalf of the whole group. Consider that two subgroups make two 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 Equation (2.1) 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 Equation (2.2). In the following, we describe two scenarios.

#### 3.2.1 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 to the ASM scheme given in Section 2.1.5. The ASMwSA is as follows:

1. Key Generation: Identical to the Key Generation in Section 2.1.5.
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  $j = 1, \dots, n$ , 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  $\sigma$  by checking

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

Note that the correctness property follows from the below equation array.

$$\begin{aligned} e(\sigma, g_2) &= e\left(\prod_{i \in \mathcal{S}} (H_0(apk, m)^{a_i sk_i} \cdot smk_i), g_2\right) \\ &= e\left(H_0(apk, m)^{\sum_{i \in \mathcal{S}} a_i sk_i} \cdot \left(\prod_{j \in \mathcal{S}} H_2(apk, j)\right)^{\sum_{i \in \mathcal{S}} a_i sk_i}, g_2\right) \\ &= e\left(H_0(apk, m) \cdot \prod_{j \in \mathcal{S}} H_2(apk, j), g_2^{\sum_{i \in \mathcal{S}} a_i sk_i}\right) \\ &= e\left(H_0(apk, m) \cdot \prod_{j \in \mathcal{S}} H_2(apk, j), spk\right) \end{aligned}$$

**Remark 3.2.1.** In the group setup phase of *ASMwSA*, the signers only compute  $\mu_{ij}$  and store them, but they do not send those  $\mu_{ij}$  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 2.1.5, which also results in a legitimate aggregated signature. We note that this eliminates 1 round of transmission cost.

**Remark 3.2.2.** In the signature generation phase,  $H_0(apk, m)$  may be replaced by  $H_0(sp, k, m)$  because the subgroup  $\mathcal{S}$  is known in advance.

**Remark 3.2.3.** We note that the user  $i \in \mathcal{G}$  may compute her individual signature as

$$s_i = \left( H_0(\text{apk}, m) \cdot \prod_{j \in \mathcal{S}} H_2(\text{apk}, j) \right)^{a_i \text{sk}_i}$$

instead of computing and storing  $\mu_{ij}$ 's in the group setup phase. Although this reduces the storage costs of the signers, computational cost increases by the extra computations of the hash  $H_2$  at the signature generation of each message.

### 3.2.1.1 Security

We follow the security reduction of the MSP signature scheme given in [12] with a few modifications to prove the security of the proposed scheme in Section 3.2.1. Below, we give Theorem 3.2.1 which states the security reduction of our proposed ASMwSA scheme.

**Theorem 3.2.1.** *ASMwSA is unforgeable (as defined in Definition 2.1.7) under the co-CDH problem (Definition 2.1.2) in the random oracle model. More precisely, ASMwSA is  $(t, q_H, q_S, \epsilon)$ -unforgeable in the random oracle model if  $q > 8q_H/\epsilon$  and if co-CDH problem is*

$$(t + t_{\text{exp}_2^n} + q_H \cdot t_{\text{exp}_1^2} + q_S \cdot t_{\text{exp}_1} + (l \cdot t_{\text{mul}_1} + 2t_{\text{pair}})) \cdot 8q_H^2/\epsilon \cdot \ln(8q_H/\epsilon), \epsilon/(8q_H)\text{-hard},$$

where  $n = |\mathcal{G}|$  is the number of potential signers and  $l = |\mathcal{S}|$  is the signers involved in an ASMwSA signature,  $t_{\text{exp}_1}$  and  $t_{\text{exp}_2}$  denote the time required to compute exponentiations in  $\mathbb{G}_1$  and  $\mathbb{G}_2$  respectively,  $t_{\text{exp}_1^i}$  and  $t_{\text{exp}_2^i}$  denote the time required to compute  $i$ -multi-exponentiations in  $\mathbb{G}_1$  and  $\mathbb{G}_2$  respectively, and  $t_{\text{mul}_1}$  denotes the multiplication in  $\mathbb{G}_1$ .

*Proof.* Suppose we have a  $(t, q_H, q_S, \epsilon)$ -forger  $\mathcal{F}$  against ASMwSA scheme. An algorithm  $\mathcal{A}$ , given  $(A = g_1^\alpha, B_1 = g_1^\beta, B_2 = g_2^\beta)$  where  $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_q$ , and a randomness  $f = \{\rho, h_1, \dots, h_{q_S}\}$  as input, proceeds as follows.

- $\mathcal{A}$  picks a random index  $k \xleftarrow{\$} \{1, \dots, q_H\}$ .
- $\mathcal{A}$  sets  $pk_i = B_2$ .
- $\mathcal{A}$  runs the forger  $\mathcal{F}$  on input  $pk_i$  with random tape  $\rho$ .

- $\mathcal{A}$  receives  $\mathcal{PK}$  from  $\mathcal{F}$  such that  $pk_i \in \mathcal{PK}$ .
- $\mathcal{A}$  maintains three lists, i.e.,  $\mathcal{L}_0, \mathcal{L}_1$  and  $\mathcal{L}_2$ , which are initially empty.
- $\mathcal{A}$  responds  $\mathcal{F}$ 's  $j$ -th  $H_1$ -queries on  $(pk_j, \mathcal{PK})$  for  $j = 1, \dots, q_H$  as follows:
  - If this is the first query on  $(pk_j, \mathcal{PK})$  with  $pk_j \in \mathcal{PK}$  and  $j \neq i$ ,  $\mathcal{A}$  chooses a random value  $w_j \xleftarrow{\$} \mathbb{Z}_q$  and sets  $H_1(pk_j, \mathcal{PK}) = w_j$ . If  $j = i$ , it sets  $H_1(pk_i, \mathcal{PK}) = h_i$ .  $\mathcal{A}$  responds to  $\mathcal{F}$  with  $H_1(pk_j, \mathcal{PK})$ .
  - If this query is done before, then  $\mathcal{A}$  responds to  $\mathcal{F}$  with earlier values.
  - For other types of queries,  $\mathcal{A}$  responds with a random value in  $\mathbb{Z}_q$ .
  - Add the response to the list  $\mathcal{L}_1$ .
- We assume that  $\mathcal{F}$  makes no repeated  $H_2$ -queries.  $\mathcal{A}$  responds  $\mathcal{F}$ 's  $H_2$  queries of the form  $(apk, j)$  for  $j = 1, \dots, n$  as follows:
  - $\mathcal{A}$  responds with  $g_1^{r_j^{(2)}}$ , where  $r_j^{(2)}$  is randomly selected from  $\mathbb{Z}_q$ ,
  - Add  $\langle (apk, j), H_2(apk, j), r_j^{(2)} \rangle$  to the list  $\mathcal{L}_2$ .
- We assume that  $\mathcal{F}$  makes no repeated  $H_0$ -queries.  $\mathcal{A}$  responds  $\mathcal{F}$ 's  $j$ -th  $H_0$ -query of the form  $(apk, m_j)$  for  $j = 1, \dots, q_H$  as follows:
  - If  $j \neq k$  then responds with  $g_1^{r_j^{(0)}}$ , where  $r_j^{(0)}$  is randomly selected from  $\mathbb{Z}_q$ .
  - If  $j = k$ , then responds with  $A_0 = A/(g_1^{r_i^{(2)}})$ , where  $r_i^{(2)} \in \mathbb{Z}_q$  is the randomly selected value for the  $i$ -th user's  $H_2$  query.
  - Add  $\langle (apk, m_j), H_0(apk, m_j), r_j^{(0)} \rangle$  to the list  $\mathcal{L}_0$ .
- For signing queries of  $\mathcal{F}$  on a message  $m_j$  and a subset  $\mathcal{S}_j \subseteq \mathcal{G}$  for  $1 \leq j \leq q_S$ ,  $\mathcal{A}$  proceeds as follows:
  - If  $i \notin \mathcal{S}_j$ , then  $\mathcal{A}$  aborts.
  - $\mathcal{A}$  looks up the list  $\mathcal{L}_0$ . If  $H_0(apk, m_j) = A_0$ , then it aborts.
  - If  $H_2(apk, z) \notin \mathcal{L}_2$  for any  $z \in \mathcal{S}_j$ , then it aborts.
  - Otherwise  $\mathcal{A}$  responds with  $\sigma_i = \left( B_1^{r_j^{(0)}} \cdot B_1^{\sum_{u \in \mathcal{S}_j} r_u^{(2)}} \right)^{h_i}$ .

Actually, the resulting signature  $\sigma_i$  is a valid signature and can be verified by the public keys of the signers in  $\mathcal{S}_j$ . Because

$$\begin{aligned} e(\sigma_i, g_2) &= e\left(\left(B_1^{r_j^{(0)}} \cdot \prod_{u \in \mathcal{S}_j} B_1^{r_u^{(2)}}\right)^{h_i}, g_2\right) \\ &= e\left(H_0(apk, m_j) \cdot \prod_{z \in \mathcal{S}_j} H_2(apk, z)^{h_i \beta}, g_2\right) \\ &= e\left(H_0(apk, m_j) \cdot \prod_{z \in \mathcal{S}_j} H_2(apk, z), pk_i^{h_i}\right) \end{aligned}$$

as in the verification equation of the ASMwSA scheme.

- Eventually  $\mathcal{F}$  outputs a triplet  $(\sigma_f, m_f, \mathcal{S}_f)$  where  $\sigma_f$  is an ASMwSA signature on the message  $m_f$  for the subgroup  $\mathcal{S}_f$ .
- $\mathcal{A}$  looks up into list  $\mathcal{L}_0$  for  $m_f$ ,
  - If  $H_0(apk, m_f) \neq A_0$ , then  $\mathcal{A}$  aborts.
  - Otherwise,  $\sigma_f$  is a valid signature on  $m_f$  for the subgroup  $\mathcal{S}_f$  with probability  $\epsilon$  such that

$$e(H_0(apk, m_f) \cdot \prod_{j \in \mathcal{S}_f} H_2(apk, j), spk_f) = e(\sigma_f, g_2). \quad (3.2)$$

The algorithm's running time  $\mathcal{A}$  is equivalent to the running time of the forger  $\mathcal{F}$  plus some extra computations that  $\mathcal{A}$  makes.

- Computing  $apk$ : The running time of computing the aggregated public key  $apk$  is  $t_{exp_2^n}$ , where  $n$  is the number of potential signers in the group  $\mathcal{G}$ .
- Hash queries:
  - For  $H_0$  and  $H_2$ , the running time is at most  $t_{exp_1^2}$ .
  - Therefore, overall running time of the hash queries including  $H_0$  and  $H_2$  will be at most  $q_H \cdot t_{exp_1^2}$ .
- Signing queries:
  - Computing the signature takes  $t_{exp_1}$ . Therefore overall running time of the signature queries is  $q_S \cdot t_{exp_1}$ .

- For verification of the output of  $\mathcal{F}$  takes  $l \cdot t_{mul_1}$  and  $2t_{pair}$ .
- Therefore, overall running time of the algorithm  $\mathcal{A}$  is  $t + t_{exp_2^q} + q_H \cdot t_{exp_1^2} + q_S \cdot t_{exp_1} + (l \cdot t_{mul_1} + 2t_{pair})$ .

The probability that  $\mathcal{A}$  does not abort is equivalent to the probability that  $\mathcal{A}$  correctly guesses the hash index of the valid forgery of  $\mathcal{F}$ , which is  $1/q_H$ . Since the forger  $\mathcal{F}$ 's success probability is  $\epsilon$ , the success probability of the algorithm  $\mathcal{A}$  is  $\epsilon/q_H$ .

Now we construct an algorithm  $\mathcal{B}$  which solves co-CDH problem in  $(\mathbb{G}_1, \mathbb{G}_2)$  on input a co-CDH instance  $(A, B_1, B_2) \in \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_2$  and a forger  $\mathcal{F}$ . The algorithm  $\mathcal{B}$  actually runs generalized forking algorithm  $\mathcal{GF}_A$  on input  $(A, B_1, B_2)$  and algorithm  $\mathcal{A}$  as described above.  $\mathcal{B}$  proceeds as follows

- If  $\mathcal{GF}_A$  outputs fail, then  $\mathcal{B}$  aborts.
- If  $\mathcal{GF}_A$  outputs  $(\{j_f\}, \{out\}, \{out'\})$ , then
  - $\mathcal{B}$  parses  $out$  as  $(\sigma, \mathcal{PK}, spk, a_1, \dots, a_n)$  and  $out'$  as  $(\sigma', \mathcal{PK}', spk', a'_1, \dots, a'_{n'})$ .
  - Note that  $out$  and  $out'$  were obtained from two executions of  $\mathcal{A}$  with randomness  $f$  and  $f'$  such that  $f|_{j_f} = f'|_{j_f}$  for some integer  $j_f \leq q_S$ . In other words, these executions are identical to the  $j_f$ -th  $H_1$ -query of the type that has not been queried before.
  - This means that the arguments of this query are identical, i.e.,  $\mathcal{PK} = \mathcal{PK}'$ , and  $n = n'$ .
  - From the construction of  $\mathcal{GF}_A$ , we know that  $a_i = h_{j_f}$ ,  $a'_i = h'_{j_f}$ , and by the forking lemma (see Lemma 2.1.1) we have  $a_i \neq a'_i$ .
  - We know that  $spk = \prod_{j \in \mathcal{S}} pk^{a_j}$  and  $spk' = \prod_{j \in \mathcal{S}} pk^{a'_j}$ .
  - Since the algorithm  $\mathcal{A}$  assigned  $H_1(pk_j, \mathcal{PK}) \leftarrow a_j$  for all  $j \neq i$ , we have  $a_j = a'_j$ , and therefore  $spk/spk' = pk_i^{a_i - a'_i}$ .
  - Notice that  $\mathcal{A}$ 's output (see (3.2)) satisfies both

$$e(H_0(apk, m_f) \cdot \prod_{j \in \mathcal{S}_f} H_2(apk, pk_j), spk_f) = e(\sigma_f, g_2)$$

and

$$e(H_0(apk, m_f) \cdot \prod_{j \in \mathcal{S}_f} H_2(apk, pk_j), spk'_f) = e(\sigma'_f, g_2).$$

– Then we have

$$e(A, B_2^{(a_i - a'_i)}) = e(\sigma_f / \sigma'_f, g_2).$$

– Hence,  $(\sigma_f / \sigma'_f)^{1/(a_i - a'_i)}$  is the solution to the co-CDH instance  $(A = g_1^\alpha, B_1 = g_1^\beta, B_2 = g_2^\beta)$  which is given to algorithm  $\mathcal{B}$  as input.

Lemma 2.1.1 says that if  $q > 8q_H/\epsilon$ , then Algorithm  $\mathcal{B}$  runs in time at most  $(t + t_{exp_2^n} + q_H \cdot t_{exp_1^2} + q_S \cdot t_{exp_1} + (l \cdot t_{mul_1} + 2t_{pair})) \cdot 8q_H^2/\epsilon \cdot \ln(8q_H/\epsilon)$ , and succeeds with probability at most  $\epsilon/(8q_H)$ .  $\square$

### 3.2.2 ASMWCA: Accountable Subgroup Multi-signature with Combiner Authentication

In ASMWCA, each user  $i \in \mathcal{G}$  sends the membership key components, i.e.,  $\mu_{ij}$  for  $j \neq i$ , to the combiner so that the signers perform fewer computations, and the main workload passes to the combiner. The ASMWCA is as follows:

1. Key Generation: Identical to Key Generation in Section 2.1.5.

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  $j = 1, \dots, n$  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  $\sigma = \prod_{i \in \mathcal{S}} s_i \cdot \prod_{i \in \mathcal{S}} \prod_{j \in \mathcal{S}} \mu_{ij}$  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 signature  $\sigma$  by checking

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

Since ASMwCA is a modified version of ASMwSA, its correctness property follows from the correctness of the ASMwSA schemes as shown in Section 3.2.1.

**Remark 3.2.4.** *The subgroup  $\mathcal{S} \in \mathcal{G}$  of the signers is determined by the combiner from the set of received individual signatures. Hence, no signer knows her co-signers.*

**Remark 3.2.5.** *In the signature generation phase,  $H_0(apk, m)$  may be replaced by  $H_0(spkn, m)$  so that the signer could set the subgroup  $\mathcal{S}$  in advance. This also eliminates the 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  $\sigma$ . However, in this case, computing  $spk$  brings additional cost, i.e.,  $l$  multi-exponentiations for each signer.*

**Remark 3.2.6.** *Each user  $i \in \mathcal{G}$  sends  $\mu_{ij}$  to the combiner for  $i, j = 1, 2, \dots, n$  and  $j \neq i$ . Unlike to scenarios in Sections 2.1.5 and 3.2.1, each signer  $i \in \mathcal{G}$  does not compute the membership key, but uses only  $\mu_{ii}$  for the signature generation. The other components,  $\mu_{ij}$  for  $i, j = 1, 2, \dots, n$  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 workload of the combiner is increased by the number of signers.*

**Remark 3.2.7.** *We note that the user  $i \in \mathcal{G}$  may compute her individual signature as*

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

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

### 3.2.2.1 Security

Here, without loss of generality, we assume that the combiner is the algorithm  $\mathcal{A}$ . Then the security reduction of the following theorem will be identical to the security reduction of Theorem 3.2.1.

**Theorem 3.2.2.** *ASMwCA is unforgeable (as defined in Definition 2.1.7) under the co-CDH problem (Definition 2.1.2) in the random oracle model. More precisely, ASMwSA is  $(t, q_H, q_S, \epsilon)$ -unforgeable in the random oracle model if  $q > 8q_H/\epsilon$  and if co-CDH problem is*

$$(t + t_{exp_2^n} + q_H \cdot t_{exp_1^2} + q_S \cdot t_{exp_1} + (l \cdot t_{mul_1} + 2t_{pair})) \cdot 8q_H^2/\epsilon \cdot \ln(8q_H/\epsilon), \epsilon/(8q_H)\text{-hard,}$$

where  $n = |\mathcal{G}|$  is the number of potential signers and  $l = |\mathcal{S}|$  is the signers involved in a ASMwCA signature,  $t_{exp_1}$  and  $t_{exp_2}$  denote the time required to compute exponentiations in  $\mathbb{G}_1$  and  $\mathbb{G}_2$  respectively,  $t_{exp_1^i}$  and  $t_{exp_2^i}$  denote the time required to compute  $i$ -multi-exponentiations in  $\mathbb{G}_1$  and  $\mathbb{G}_2$  respectively, and  $t_{mul_1}$  denotes the multiplication in  $\mathbb{G}_1$ .

*Proof.* As we state above, the only elements to be kept secret are the membership key components and the secret keys. Since the algorithm  $\mathcal{A}$  acts as an honest signer, we may assume -without loss of generality- that Algorithm  $\mathcal{A}$  also acts as the designated combiner. Hence, the proof follows from the security proof of Theorem 3.2.1.  $\square$

### 3.2.3 Eliminating the combiner

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

- Computes the  $i$ -th aggregated signature  $s_{k_i} = s_{k_{i-1}} \cdot (H_0(sp_k, m)^{a_{k_i} sk_{k_i}} \cdot sm_{k_{k_i}})$ , 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  $sp_{k_{k_i}} = sp_{k_{k_{i-1}}} \cdot pk_{k_{k_i}}^{a_{k_i}}$ , where  $sp_{k_{k_{i-1}}}$

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 3.1.

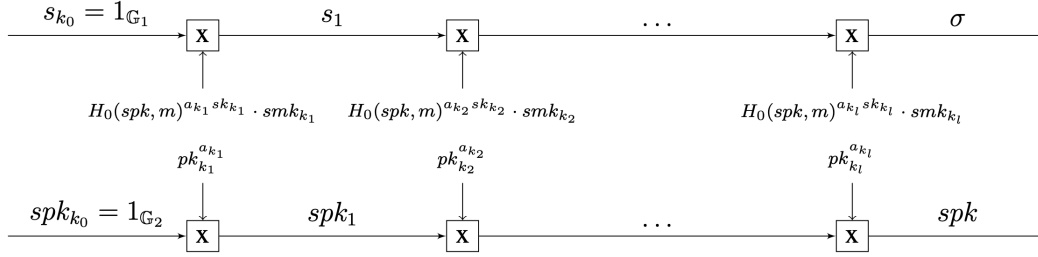


Figure 3.1: ASM scheme without a combiner

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, that is, the transmission size is doubled.

### 3.2.4 Advantages of subgroup-specific membership key

1. Group-specific membership keys in Boneh-Drijvers-Neven ASM scheme given in Section 2.1.5 are generated by one round of interactive protocol, in which all members participate. But in case of using subgroup-specific ones given in Sections 3.2.1 and 3.2.2, 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 the case of using the subgroup-specific membership key  $smk$  instead of group-specific one, i.e.,  $mk_i$ . If we use  $H_2(pk_j)$  instead of  $H_2(apk, j)$  for computing the components of membership keys, registering/unregistering a member to/from the group  $\mathcal{G}$  does not require a new group setup.

3. Users in  $\mathcal{S} \subseteq \mathcal{G}$  do multiplications to get  $smk$ ; but all group members in  $\mathcal{G}$  need to do multiplications 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  $apk$ .

### 3.3 Aggregated versions of ASM schemes

In this section, we extend the ASM scheme to a partial aggregated version of ASM, i.e., AASM scheme. Remember that the ASM signature scheme described in Section 2.1.5 outputs a signature  $\sigma = (s, pk)$ . Consider  $N$  many ASM signatures  $(apk_1, \mathcal{S}_1, m_1, \sigma_1), \dots, (apk_N, \mathcal{S}_N, m_N, \sigma_N)$ . The AASM scheme outputs the partial aggregated signature  $\Sigma = (pk_1, \dots, pk_N, s)$ , where  $s$  is the aggregation of  $s_1, \dots, s_N$ . In Section 3.3.1, we describe the AASM scheme that is given in [12].

#### 3.3.1 Partially Aggregated ASM Scheme (AASM)

The aggregation of several ASM signatures is discussed in [12], 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 [12]. In addition to the phases of the ASM scheme, two more phases are defined in [12]. The signature aggregation and aggregated signature verification phases are as follows. Let  $H_5 : \{0, 1\}^* \rightarrow \mathbb{Z}_q$  be another hash function.

- **Signature Aggregation:** Given  $(par, \{(apk_i, \mathcal{S}_i, m_i, \sigma_i)_{i=1}^N\})$ .
  - Parse  $\sigma_i$  as  $(s_i, pk_i)$ .
  - Compute  $b_i \leftarrow H_5((apk_i, \mathcal{S}_i, m_i, pk_i), \{(apk_j, \mathcal{S}_j, m_j, pk_j)_{j=1}^N\})$  for  $i = 1, \dots, N$ .
  - Compute  $s \leftarrow \prod_{i=1}^N s_i^{b_i}$  and output  $\Sigma \leftarrow (pk_1, \dots, pk_N, s)$ .
- **Aggregate Signature Verification:** Given  $(par, \{(apk_i, \mathcal{S}_i, m_i)_{i=1}^N\}, \Sigma)$ .
  - Parse  $\Sigma$  as  $(pk_1, \dots, pk_N, s)$ .

- For  $i = 1, \dots, N$ , compute  $b_i \leftarrow H_5((apk_i, \mathcal{S}_i, m_i, pk_i), \{(apk_j, \mathcal{S}_j, m_j, pk_j)_{j=1}^N\})$ .
- Accept if and only if

$$\prod_{i=1}^N \left( e(H_0(apk_i, m_i), pk_i^{b_i}) \cdot e\left(\prod_{j \in \mathcal{S}_i} H_2(apk_i, j), apk_i^{b_i}\right) \right) \stackrel{?}{=} e(s, g_2). \quad (3.3)$$

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 (3.3) of the AASM scheme requires  $(2N + 1)$  pairings.

### 3.3.2 Aggregated versions of proposed schemes

Our schemes defined in Sections 3.1, 3.2.1 and 3.2.2 can also be further turned into aggregated schemes. Unlike the partial aggregation in the AASM scheme, our proposed schemes can be fully aggregated, resulting in a single signature size.

#### 3.3.2.1 Aggregated vASM Scheme (AvASM)

The vASM scheme described in Section 3.1 can be extended to aggregated version AvASM as in Section 3.3.1. Let  $H_5$  be the hash function defined in the previous section.

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

$$\prod_{i=1}^N e\left(H_0(m_i), \prod_{j \in \mathcal{S}_i} mpk_j\right) = e(\Sigma, g_2). \quad (3.4)$$

Correctness of the AvASM scheme follows from the following equation array.

$$\begin{aligned}
\prod_{i=1}^N e\left(H_0(m_i), \prod_{j \in \mathcal{S}_i} mpk_j\right) &= \prod_{i=1}^N e\left(H_0(m_i), \prod_{j \in \mathcal{S}_i} g_2^{mk_j}\right) \\
&= \prod_{i=1}^N e\left(H_0(m_i), g_2^{\sum_{j \in \mathcal{S}_i} mk_j}\right) \\
&= \prod_{i=1}^N e\left(H_0(m_i)^{\sum_{j \in \mathcal{S}_i} mk_j}, g_2\right) \\
&= \prod_{i=1}^N e\left(\sigma_i, g_2\right) \\
&= e\left(\prod_{i=1}^N \sigma_i, g_2\right) \\
&= e(\Sigma, g_2)
\end{aligned}$$

Next, we reduce the security of the AvASM scheme to the security of the vASM scheme.

**Theorem 3.3.1.** *If the vASM scheme is unforgeable in the random oracle model, then the AvASM scheme is also unforgeable in the random oracle model. More precisely, AvASM is  $(t, q_H, q_S, \epsilon)$ -secure against existential forgery under adaptive chosen message attacks in the random oracle model, for all  $t$  and  $\epsilon$  satisfying*

$$t \leq t' - t_{exp_1}(q_H + (n + 1)q_S) + q_S \cdot t_{exp_1^N} \text{ and } \epsilon \geq e(q_S + 1) \cdot \epsilon'$$

where  $t_{exp_1}$  is the time required by an exponentiation in  $\mathbb{G}_1$  and  $n$  is the maximum number of potential signers involved in a single vASM signature.

*Proof.* Consider the forger  $\mathcal{F}$  in the security proof of Theorem 3.1.1. We construct an algorithm  $\mathcal{A}$  as in the security proof of Theorem 3.1.1, except that  $\mathcal{F}$  now returns an aggregated vASM signature instead of a single vASM signature, i.e., an AvASM signature  $\Sigma$ , a set  $\{\text{MPK}_j, \mathcal{S}_j, m_j\}_{j=1}^N$ , and the set  $\{\text{MPK}^*, \mathcal{S}^*, m^*\}$ .

Assume that  $\langle m^*, w^*, b^*, c^* \rangle$  is in the list  $\mathcal{L}$  such that  $c^* = 0$  and  $w^* = B \cdot \psi(g_2)^b$ .

This gives us

$$e(B \cdot \psi(g_2)^b, \prod_{j \in \mathcal{S}^*} mpk_j) \cdot \prod_{i=1}^n e\left(H_0(m_i), \prod_{j \in \mathcal{S}_i} mpk_j\right) = e(\Sigma, g_2). \quad (3.5)$$

Then,  $\mathcal{A}$  computes  $\sigma \leftarrow \Sigma \cdot \prod_{i=1}^N \left( \prod_{j \in \mathcal{S}_i} \psi(\text{mpk}_j) \right)^{-\sum_{j \in \mathcal{S}_i} (r_j + b_i)}$ , and this signature  $\sigma$  satisfies

$$e(H(m^*), \prod_{j \in \mathcal{S}^*} \text{mpk}_j) = e(\sigma, g_2), \quad (3.6)$$

which is a single vASM signature on message  $m^*$  by the subgroup  $\mathcal{S}^*$ .

Note that the success probability is the same as the forger in the security proof of the vASM scheme. On the other hand, Algorithm  $\mathcal{A}$  computes extra  $t_{\text{exp}_1^N}$  multi-exponentiations for extracting  $\sigma$  from  $\Sigma$ .  $\square$

### 3.3.2.2 Aggregated versions of ASMwSA / ASMwCA Schemes

The ASMwSA and ASMwCA schemes described in Section 3.2 can be extended to aggregated versions AASMwSA and AASMwCA as in Section 3.3.1. In addition to the phases of the described schemes in Section 3.2, signature aggregation and aggregate signature verification phases are given below:

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

$$\prod_{i=1}^N e \left( H_0(apk_i, m_i) \cdot \prod_{j \in \mathcal{S}_i} H_2(apk_i, j), spk_i \right) = e(\Sigma, g_2). \quad (3.7)$$

Note that the correctness property follows from the below equation array.

$$\begin{aligned}
e(\Sigma, g_2) &= e\left(\prod_{i=1}^N \sigma_i, g_2\right) \\
&= e\left(\prod_{i=1}^N s_i \cdot smk_i, g_2\right) \\
&= \prod_{i=1}^N e(s_i \cdot smk_i, g_2) \\
&= \prod_{i=1}^N e\left(H_0(apk_i, m_i)^{\sum_{j \in \mathcal{S}_i} a_j sk_j} \cdot \left(\prod_{j \in \mathcal{S}_i} H_2(apk_i, j)\right)^{\sum_{j \in \mathcal{S}_i} a_j sk_j}, g_2\right) \\
&= \prod_{i=1}^N e\left(H_0(apk_i, m_i) \cdot \left(\prod_{j \in \mathcal{S}_i} H_2(apk_i, j)\right), g_2^{\sum_{j \in \mathcal{S}_i} a_j sk_j}\right) \\
&= \prod_{i=1}^N e\left(H_0(apk_i, m_i) \cdot \prod_{j \in \mathcal{S}_i} H_2(apk_i, j), spk_i\right)
\end{aligned}$$

As we did in the security proof of AvASM scheme above, we next reduce the security of AASMwSA and AASMwCA schemes to the security of ASMwSA/ASMwCA schemes.

**Theorem 3.3.2.** *AASMwSA and AASMwCA are unforgeable (as defined in Definition 2.1.7) under the  $\psi$ -co-CDH problem (Definition 2.1.3) in the random oracle model. More precisely, AASMwSA and AASMwCA are  $(t, q_H, q_S, \epsilon)$ -unforgeable in the random oracle model if  $q > 8q_H/\epsilon$  and if  $\psi$ -co-CDH problem is*

$$\begin{aligned}
&(t + q_H \cdot \max(t_{exp_2^n}, t_{exp_1^2}) + n \cdot q_S \cdot (t_{exp_2^n} + t_{exp_1}) + 2t_{pair} + \\
&\quad + t_{exp_1^N}) \cdot 8q_H^2/\epsilon \cdot \ln(8q_H/\epsilon), \epsilon/(8q_H))\text{-hard},
\end{aligned}$$

where  $n$  is the number of potential signers involved in a AASMwSA and ASMwCA signatures,  $t_{exp_1}$  and  $t_{exp_2}$  denote the time required to compute exponentiations in  $\mathbb{G}_1$  and  $\mathbb{G}_2$  respectively, and  $t_{exp_1^i}$  and  $t_{exp_2^i}$  denote the time required to compute  $i$ -multi-exponentiations in  $\mathbb{G}_1$  and  $\mathbb{G}_2$  respectively.

*Proof.* Consider the security proof of Theorem 3.2.1. We construct the same algorithm  $\mathcal{A}$ , but this time it's forger  $\mathcal{F}$  gives an aggregate multi-signature, i.e., an aggregate multi-signature  $\Sigma$ , a set of tuples  $\{(apk_i, spk_i, \mathcal{S}_i, m_i, \sigma_i)_{i=1}^N\}$ , a set of public

keys  $\mathcal{PK}$ , a subgroup  $\mathcal{S}^*$  and a message  $m^*$ . We know that  $apk^*$  and  $spk^*$  can be generated from  $\mathcal{PK}$  and  $\mathcal{S}^*$ .

Assume that the algorithm  $\mathcal{A}$  guesses the  $k$ -th query  $H_0(apk^*, m^*)$  correctly. Then the following equality holds

$$e(A_0, spk^*) \cdot \prod_{i=1}^N e\left(H_0(apk_i, m_i) \cdot \prod_{j \in \mathcal{S}_i} H_2(apk_i, j), spk_i\right) = e(\Sigma, g_2). \quad (3.8)$$

$\mathcal{A}$  looks up the tables  $\mathcal{L}_0$  and  $\mathcal{L}_2$ :

- $\mathcal{L}_0$ , for the pairs  $(apk_j, m_j)$ , such that  $H_0(apk_j, m_j) = g_1^{r_j^{(0)}}$ .
- $\mathcal{L}_2$ , for the pairs  $(apk_j, i_j)$ , such that  $H_2(apk_j, i_j) = g_1^{r_{i_j}^{(2)}}$ .

Then  $\mathcal{A}$  computes  $\sigma \leftarrow \Sigma \cdot \prod_{i=1}^N \mathcal{O}(spk_i)^{-\sum_{j \in \mathcal{S}_i} (r_i^{(0)} + r_{i_j}^{(2)})}$ , and this signature satisfies

$$e(H_0(apk^*, m^*) \cdot \prod_{j \in \mathcal{S}^*} H_2(apk^*, j^*), spk^*) = e(\sigma, g_2).$$

Notice that we have  $A_0$  for a single **ASMwSA/ASMwCA** forgery. Therefore, the rest will be the same as in the security proof of Theorem 3.2.1. The running time of Algorithm  $\mathcal{A}$  will increase because of the extraction of the  $\sigma$ , which costs extra  $t_{exp_1^N}$  running time. On the other hand, the success probability stays the same.  $\square$

### 3.4 Comparison

In Table 3.1, we compare **ASM**, **ASM-PoP**, and the proposed schemes in this paper. Our comparison contains the number of operations required in each phase of those schemes. For example, consider a comparison of group setup phases of the schemes **ASM** and **vASM**. It can be seen in Table 3.1 that the **ASM** scheme costs of  $n$   $H_2$  hashes (onto  $\mathbb{G}_1$ ),  $n$  exponentiations and  $(n - 1)$  multiplications in  $\mathbb{G}_1$  in the group setup phase. On the other hand, the **vASM** scheme has  $2n$  exponentiations and  $(n^2 + n - 2)$  multiplications  $\mathbb{G}_2$  in the group setup.

It can also be seen from Table 3.1 that the **vASM** scheme requires fewer operations than **ASM** and **ASM-PoP** in the verification phase. On the other hand, since the set of

membership public keys MPK and the set of commitments COM are published, the broadcasted data size in vASM is linear in the number of users in  $\mathcal{G}$ .

The ASMwSA scheme provides efficient group setup, signature aggregation and verification phases; however, its signature generation requires more multiplications in  $\mathbb{G}_1$  and extra storage for  $(n - 1) \mathbb{G}_1$  elements.

The ASMwCA scheme has efficient group setup and verification phases, but it requires more computations in the signature aggregation phase. On the other hand, it requires extra storage for  $(n^2 - n - 1) \mathbb{G}_1$  elements.

Consider  $N$  distinct accountable subgroup multi-signatures. The ASM scheme proposed by Boneh et al. supports a partial aggregation. Therefore the AASM scheme outputs an aggregated signature consists of  $N$  many  $\mathbb{G}_2$  elements and one  $\mathbb{G}_1$  element. On the other hand, the AvASM outputs an aggregated signature with only one  $\mathbb{G}_1$  element. In terms of verification efficiency, we only compare the number of pairings required in verification equations (since the pairing operations dominate the cost of verification). All the proposed schemes, i.e., AvASM, AASMwSA, and AASMwCA, require  $N + 1$  pairings for verification whereas the AASM requires  $2N + 1$ .

Table 3.1: Comparison of our pairing-based ASM schemes

Phases	ASM [12]	ASM with PoP [12]	vASM	ASMwSA	ASMwCA
Key Generation	1 $\text{Exp}_{\mathbb{G}_2}$	<u>For <math>pk</math>:</u> 1 $\text{Exp}_{\mathbb{G}_2}$ <u>For proof:</u> 1 Hash ( $H_3$ ) 1 $\text{Exp}_{\mathbb{G}_2}$	1 $\text{Exp}_{\mathbb{G}_2}$	1 $\text{Exp}_{\mathbb{G}_2}$	1 $\text{Exp}_{\mathbb{G}_2}$
Key Aggregation	$n$ Hash ( $H_1$ ) $n$ $\text{Exp}_{\mathbb{G}_2}$ $(n - 1)$ $\text{Mul}_{\mathbb{G}_2}$	<u>For proof:</u> $n$ Hash ( $H_3$ ) $2n$ pairings <u>For <math>apk</math>:</u> 1 Hash ( $H_4$ ) $(n - 1)$ $\text{Mul}_{\mathbb{G}_2}$		$n$ Hash ( $H_1$ ) $n$ $\text{Exp}_{\mathbb{G}_2}$ $(n - 1)$ $\text{Mul}_{\mathbb{G}_2}$	$n$ Hash ( $H_1$ ) $n$ $\text{Exp}_{\mathbb{G}_2}$ $(n - 1)$ $\text{Mul}_{\mathbb{G}_2}$
Group Setup	$n$ Hash ( $H_2$ ) $n$ $\text{Exp}_{\mathbb{G}_2}$ $(n - 1)$ $\text{Mul}_{\mathbb{G}_1}$	$n$ Hash ( $H_2$ ) $n$ $\text{Exp}_{\mathbb{G}_1}$ $(n - 1)$ $\text{Mul}_{\mathbb{G}_1}$	$2n$ $\text{Exp}_{\mathbb{G}_2}$ $(n^2 + n - 2)$ $\text{Mul}_{\mathbb{G}_2}$	$n$ Hash ( $H_2$ ) $n$ $\text{Exp}_{\mathbb{G}_1}$	$n$ Hash ( $H_2$ ) $n$ $\text{Exp}_{\mathbb{G}_1}$
Signature Generation	1 Hash ( $H_0$ ) 1 $\text{Exp}_{\mathbb{G}_1}$ 1 $\text{Mul}_{\mathbb{G}_1}$	1 Hash ( $H_0$ ) 1 $\text{Exp}_{\mathbb{G}_1}$ 1 $\text{Mul}_{\mathbb{G}_1}$	1 Hash ( $H_0$ ) 1 $\text{Exp}_{\mathbb{G}_1}$	1 Hash ( $H_0$ ) 1 $\text{Exp}_{\mathbb{G}_1}$ $l$ $\text{Mul}_{\mathbb{G}_1}$	1 Hash ( $H_0$ ) 1 $\text{Exp}_{\mathbb{G}_1}$ 1 $\text{Mul}_{\mathbb{G}_1}$
Signature Aggregation	$(l - 1)$ $\text{Mul}_{\mathbb{G}_1}$ $(l - 1)$ $\text{Mul}_{\mathbb{G}_2}$	$(l - 1)$ $\text{Mul}_{\mathbb{G}_1}$ $(l - 1)$ $\text{Mul}_{\mathbb{G}_2}$	$(l - 1)$ $\text{Mul}_{\mathbb{G}_1}$	$(l - 1)$ $\text{Mul}_{\mathbb{G}_1}$	$(l^2 - l)$ $\text{Mul}_{\mathbb{G}_1}$
Verification	<u>For <math>apk^{(c)}</math>:</u> $n$ Hash ( $H_1$ ) $n$ $\text{Exp}_{\mathbb{G}_2}$ $(n - 1)$ $\text{Mul}_{\mathbb{G}_2}$ <u>For verification:</u> 1 Hash ( $H_0$ ) $l$ Hash ( $H_2$ ) $(l - 1)$ $\text{Mul}_{\mathbb{G}_1}$ 1 $\text{Mul}_{\mathbb{G}_T}$ 3 Pairings	<u>For proof:</u> $n$ Hash ( $H_3$ ) $n$ $\text{Exp}_{\mathbb{G}_2}$ $2n$ pairings <u>For <math>apk^{(c)}</math>:</u> 1 Hash ( $H_4$ ) $(n - 1)$ $\text{Mul}_{\mathbb{G}_2}$ <u>For verification:</u> 1 Hash ( $H_0$ ) $l$ Hash ( $H_2$ ) $(l - 1)$ $\text{Mul}_{\mathbb{G}_1}$ 1 $\text{Mul}_{\mathbb{G}_T}$ 3 Pairings	1 Hash ( $H_0$ ) $(l - 1)$ $\text{Mul}_{\mathbb{G}_2}$ 2 Pairings	<u>For <math>apk/spk^{(a)}</math>:</u> $n$ Hash ( $H_1$ ) $n$ $\text{Exp}_{\mathbb{G}_2}$ $(n - 1)$ $\text{Mul}_{\mathbb{G}_2}$ <u>For verification:</u> 1 Hash ( $H_0$ ) $l$ Hash ( $H_2$ ) $l$ $\text{Mul}_{\mathbb{G}_1}$ 2 Pairings	<u>For <math>apk/spk^{(a)}</math>:</u> $n$ Hash ( $H_1$ ) $n$ $\text{Exp}_{\mathbb{G}_2}$ $(n - 1)$ $\text{Mul}_{\mathbb{G}_2}$ <u>For verification:</u> 1 Hash ( $H_0$ ) $l$ Hash ( $H_2$ ) $l$ $\text{Mul}_{\mathbb{G}_1}$ 2 Pairings
Transmission	<u>For group setup:</u> $(n - 1)$ $\mathbb{G}_1$ Elt. <u>For signature:</u> 1 $\mathbb{G}_1$ Elt. 1 $\mathbb{G}_2$ Elt.	<u>For group setup:</u> $(n - 1)$ $\mathbb{G}_1$ Elt. <u>For signature:</u> 1 $\mathbb{G}_1$ Elt. 1 $\mathbb{G}_2$ Elt.	<u>For signature:</u> 1 $\mathbb{G}_1$ Elt.	<u>For signature:</u> 1 $\mathbb{G}_1$ Elt.	<u>For group setup:</u> $(n - 1)$ $\mathbb{G}_1$ Elt. <u>For signature:</u> 1 $\mathbb{G}_1$ Elt.
Broadcasting	<u>For <math>pk</math>:</u> 1 $\mathbb{G}_2$ Elt.	<u>For <math>pk</math>:</u> 1 $\mathbb{G}_2$ Elt. <u>For Proof:</u> 1 $\mathbb{G}_1$ Elt.	<u>For <math>pk</math>:</u> 1 $\mathbb{G}_2$ Elt. <u>For <math>mpk</math>:</u> 1 $\mathbb{G}_2$ Elt. <u>For <math>COM_i</math>:</u> $n$ $\mathbb{G}_2$ Elt. <u>For <math>COM^{(b)}</math>:</u> $n$ $\mathbb{G}_2$ Elt.	<u>For <math>pk</math>:</u> 1 $\mathbb{G}_2$ Elt.	<u>For <math>pk</math>:</u> 1 $\mathbb{G}_2$ Elt.
Storage	<u>For <math>sk</math>:</u> 1 integer ( $\mathcal{O}(\lambda)$ -bit) <u>For <math>mk</math>:</u> 1 $\mathbb{G}_1$ Elt.	<u>For <math>sk</math>:</u> 1 integer ( $\mathcal{O}(\lambda)$ -bit) <u>For <math>mk</math>:</u> 1 $\mathbb{G}_1$ Elt.	<u>For <math>mk</math>:</u> 1 integer ( $\mathcal{O}(\lambda)$ -bit)	<u>For <math>sk</math>:</u> 1 integer ( $\mathcal{O}(\lambda)$ -bit) <u>For <math>mk</math>:</u> $n$ $\mathbb{G}_1$ Elt.	<u>For <math>sk</math>:</u> 1 integer ( $\mathcal{O}(\lambda)$ -bit) <u>For <math>mk</math>:</u> 1 $\mathbb{G}_1$ Elt. <u>Combiner:</u> $(n^2 - n)$ $\mathbb{G}_1$ Elt.

- <sup>(a)</sup> As  $apk$  contains the components of  $spk$ , no extra cost is needed for computing  $spk$ .
- <sup>(b)</sup> Since all the users compute the same commitment set  $COM$ , it is enough to be broadcast by only one user.
- <sup>(c)</sup>  $apk$  is computed by the verifier. But to avoid the extra cost, it would be given to the combiner.
- $\text{Exp}_{\mathbb{G}_i}$  Exponentiation in group  $\mathbb{G}_i$  for  $i \in \{1, 2\}$ .
- $\text{Mul}_{\mathbb{G}_i}$  Multiplication in group  $\mathbb{G}_i$  for  $i \in \{1, 2, T\}$ .  $\mathbb{G}_i$  Elt. Group elements are written as  $\mathbb{G}_i$  Elt. for  $i \in \{1, 2\}$ .
- $H_1, H_2, H_3, H_4, H_5$  Hash functions are as in the schemes.
- $\lambda$  Security parameter.

## CHAPTER 4

### COMPARTMENT-BASED AND HIERARCHICAL THRESHOLD DELEGATED VERIFIABLE ACCOUNTABLE SUBGROUP MULTI-SIGNATURES

Individuals possessing signatory authority carry out their transactions by appointing a proxy to wield their signing power in situations where their physical presence at the organization is impeded. Nevertheless, the process of designating a proxy may encounter challenges depending on the organizational structure. Consider an organization with a complex topology consisting of many compartments; each has many sub-compartments and multi-level hierarchical structures. In this case, the issue of who will deputize for whom would emerge as a challenging problem.

In this chapter, we propose four constructions using the **vASM** scheme and several threshold secret sharing schemes to present alternative solutions to the problem of delegation of signing authority problem. We propose that one can have the functionalities of a proxy signature scheme via accountable subgroup multi-signature schemes [3, 12, 46], in particular via the **vASM** scheme (see Section 3.1), or a combination of the **vASM** scheme with some proper threshold secret sharing schemes [25, 32, 58, 63, 64]. Assume a scenario that there exists an organization with lots of compartments, and each has an authorized signer and many unauthorized users, just like managers and their subordinates in a hierarchical organizational structure. The authorized signers have signing rights on behalf of their compartments (also on behalf of the entire organization). They want to assign proxies among unauthorized users in their compartments. To solve this assignment problem, firstly, we apply the **vASM** scheme recursively. The authorized signer (original signer) and the unautho-

rized users (proxy signer candidates) jointly participate in a *vASM* group setup. At the end of this setup phase, each unauthorized user has a *compartment membership key* which can be used to sign on behalf of the authorized user and the entire organization. Then we combine the methods from threshold signatures [10, 21, 28, 29] and *vASM* scheme to construct solutions for the delegation that supports one or more original/proxy signers and provides accountability at the same time. As a second method, we consider that the authorized users share their membership key via Shamir's secret sharing (SSS) scheme [58] with unauthorized users in their compartment. Then unauthorized users can sign with the sum of their shares and secret keys. Thirdly, we assume a trusted user exists in each compartment and apply Shamir's SSS in a nested fashion. The authorized users first share their membership key via a 2-out-of-2 Shamir's SSS, give one of the shares to the trusted user in their compartment, and then recursively apply another SSS to the second share. At the end of this nested SSS protocol, each unauthorized user has a compartment membership key to sign a proxy signature on behalf of the authorized signer. Finally, we consider a hierarchical structure and think the trusted user(s) are in the first level of the compartment. In this case, the authorized signers share their membership keys via a hierarchical threshold secret sharing (HTSS) scheme [32, 63, 64]. Each unauthorized user can sign on behalf of the authorized signers in an accountable way. After describing the constructions, we see that they all have the main functionalities of a proxy signature according to [11]. One can also find the details in [4].

#### **4.1 Compartment-based and Hierarchical Threshold Delegation of *vASM* Authority**

In general, the delegation of signing capability can be achieved by giving a power of attorney. For example, consider an organization with a sophisticated and complicated nature whose structure expands in vertical and horizontal directions. In such a case, signing authorities may need to assign multiple proxies simultaneously. In this section, we propose four constructions. In the first one, we apply the *vASM* scheme recursively. In the second construction, we use Shamir's secret sharing scheme (SSS) [58] directly among the users of each compartment. In the third one, we assume the

existence of at least one trusted user in each compartment, and we share the vASM authority by a secret sharing scheme in a nested fashion. In the last one, we use the hierarchical threshold secret sharing scheme [32, 63] to delegate the vASM signing authority of an authorized users to the unauthorized users in the same compartment, which is partitioned hierarchically.

Consider a group of users  $\mathcal{G} = \bigcup_{i=1}^m \mathcal{U}_i$  which is a union of distinct compartments  $\mathcal{U}_i$  for  $i = 1, \dots, m$ . Without loss of generality, we assume that only one authorized user (or original signer) exists in each compartment  $\mathcal{U}_i$ , that is  $AU_i$ . Assume that each authorized user  $AU_i \in \mathcal{U}_i$  participates in a vASM group setup, obtains her membership key  $mk_i$ , and wants to delegate his vASM signing authority to some unauthorized users (or proxy signers) in her compartment, i.e.,  $u_{ij} \in \mathcal{U}_i$  for  $j = 1, \dots, k_i$ , where  $k_i \in \mathbb{Z}$  is the number of proxy candidates (unauthorized users) in the  $i$ -th compartment. In the remaining part, we use the notation above. Let the functions  $e$  and  $H$  be as in Section 3.1.

#### 4.1.1 Recursive vASM as a proxy signature

The output of a vASM group setup is a membership key and a membership public key for each participant, which are used for signing and verification, respectively. Consider an authorized user  $AU_i \in \mathcal{U}_i$  has her membership key  $mk_i$  that she calculated with the other authorized users  $AU_j \in \mathcal{U}_j$ , for  $j = 1, \dots, m$ , in a group setup as described in Section 3.1. Moreover, they also publish a set of membership public keys MPK, and a global commitment set COM at the end of that group setup. Assume that  $AU_i$  wants to delegate the signing power of her membership key to a certain number of unauthorized users in her compartment. Let  $I_i$  be the union of the indices of  $AU_i$  and the set of unauthorized users in  $i$ -th compartments that  $AU_i$  wants to designate as proxies. To that end, the authorized user  $AU_i$  and the unauthorized users  $u_{ij} \in I_i$  jointly participate in another vASM group setup, which we call as *compartment setup*. Each unauthorized participant  $u_{ij} \in I_i$  joins the compartment setup protocol with his secret key  $sk_{ij}$  for  $j = 1, \dots, k_i$ . However, the authorized user  $AU_i$  participates in the compartment setup with her membership key  $mk_i$ . At the end of this compartment setup, each unauthorized user  $u_{ij} \in I_i$

obtains a compartment membership key  $cmk_{ij}$  for  $j = 1, \dots, k_i$ , along with compartment public key set  $CPK := \{cpk_j : j = 1, \dots, k_i\}$ , a proxy commitment set  $PCS := \{PC_j : j = 1, \dots, k_i\}$  that contains the commitments of the VSS protocol they participate in, see Figure 4.1. Here the compartment membership keys can be seen as proxy signing keys, and the compartment public keys in CPK can be seen as the proxy verification keys. Below we give the steps of this construction.

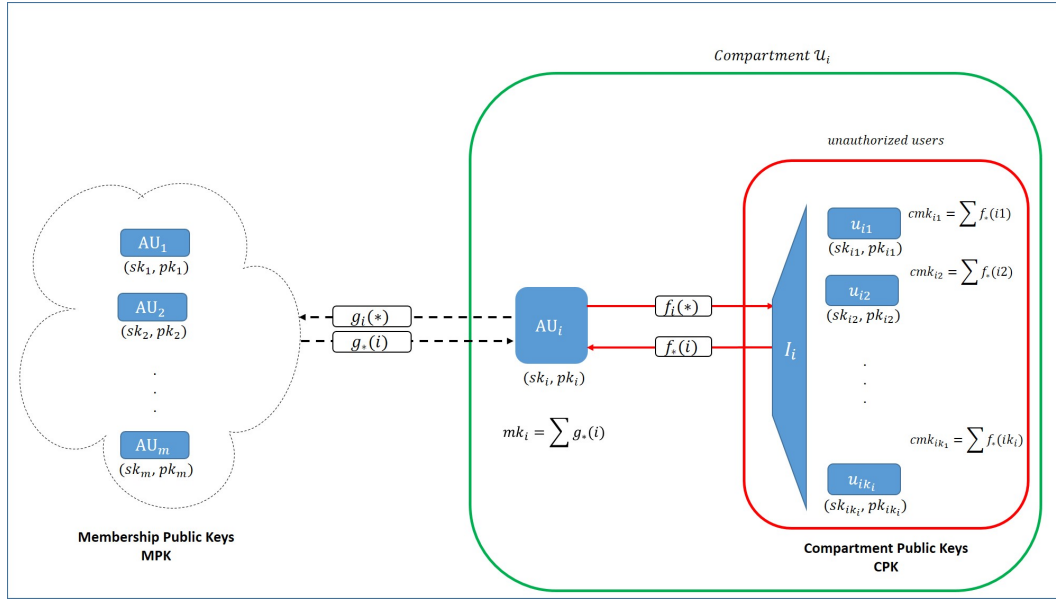


Figure 4.1: Membership key generation of recursive vASM

## 1. Key Generation:

- The authorized user  $AU_i$  has her membership key  $mk_i$ , membership public key  $mpk_i$ , and corresponding commitment set COM by performing a group setup with other authorized users.
- Each unauthorized user  $u_{ij} \in I_i$  has their secret and public key pairs  $sk_{ij}, pk_{ij}$  as described in Section 3.1.

2. **Compartment Setup:** Both  $AU_i$  and  $u_{ij} \in I_i$  proceed as follows, see also Figure 4.1:

- Choose a polynomial  $f_j(x) = \alpha_{k_i-1}^{(j)}x^{k_i-1} + \dots + \alpha_1^{(j)}x + \alpha_0^{(j)} \in \mathbb{Z}_q[x]$ , where  $\alpha_0^{(j)} = sk_{ij}$  (or  $mk_i$  for  $AU_i$ ) and  $\alpha_w^{(j)}$ 's are all nonzero and distinct, for  $w = 1, \dots, k_i - 1$ .

- Compute the set of commitments  $PCS_j := \{PC_w^{(j)} = g_2^{\alpha_w^{(j)}} \mid w = 0, \dots, k_i - 1\}$ .
- Send  $(f_j(z), PCS_j)$  to  $z$ -th user in  $I_i$ , for  $z = 1, \dots, k_i$ .
- After receiving  $(f_z(j), PCS_z)$ ,
  - computes the membership key  $cmk_{ij} = \sum_{z \in I_i} f_z(j)$ , and
  - computes  $PCS_i := \{PC_w = \prod_{z \in I_i} PC_w^{(z)} \mid w = 0, \dots, k_i\}$ .
- Checks:
  - (a)  $PC_0 \stackrel{?}{=} mpk_i \cdot \prod_{j \in I_i} pk_{ij}$  ( $mpk_i \in MPK$  is the membership public key of the authorized user  $AU_i$ )
  - (b)  $g_2^{cmk_{ij}} \stackrel{?}{=} \prod_{w=0}^{k_i-1} (PC_w)^{j^w}$
- If either (a) or (b) fails, then they abort. Else, define  $CPK_i = \{cpk_{ij} = g_2^{cmk_{ij}}\}_{j \in \mathcal{U}_i}$ , and make  $CPK_i$  and  $PCS_i$  public.

- Signature Generation:** A designated proxy signer (among the unauthorized users)  $u_{ij} \in I_i$  computes his individual signature  $s_{ij} = H(M)^{cmk_{ij}}$  on the message  $M$  and sends  $s_{ij}$  to the designated combiner.
- Signature Aggregation:** After receiving the individual signatures of the proxy signers, the designated combiner first forms the subgroup of proxy signers  $\mathcal{S}_i \subseteq I_i$ . Then, she computes the aggregated subgroup multi-signature  $\sigma_i = \prod_{j \in \mathcal{S}_i} s_{ij}$ .
- Verification:** Anyone, who is given  $\{par, CPK_i, \mathcal{S}_i, M, \sigma_i\}$ , can verify the signature  $\sigma_i$  by checking

$$e(H(M), \prod_{j \in \mathcal{S}_i} cpk_{ij}) \stackrel{?}{=} e(\sigma_i, g_2). \quad (4.1)$$

Verification satisfies correctness as given below:

$$\begin{aligned}
 e(\sigma_i, g_2) &= e\left(\prod_{j \in \mathcal{S}_i} s_{ij}, g_2\right) \\
 &= e\left(H(M)^{\sum_{j \in \mathcal{S}_i} cmk_{ij}}, g_2\right) \\
 &= e\left(H(M), g_2^{\sum_{j \in \mathcal{S}_i} cmk_{ij}}\right) \\
 &= e\left(H(M), \prod_{j \in \mathcal{S}_i} cpk_{ij}\right).
 \end{aligned}$$

**Security.** Note that this is indeed a vASM signature scheme. The only difference from the known vASM is the secret that is shared in the compartment setup phase. The authorized user  $AU_i$  shares his membership key  $mk_i$  from the previous group setup with the other authorized users of the other compartments (see Figure 4.1) while the unauthorized users share their secret keys. Hence, the security of the construction follows from the security of the vASM scheme.

For the compartment setup phase, we should clarify the purpose of consistency checks. There are two consistency checks in the compartment setup. The first check is performed to ensure that the participants know their shared secret. Since the first proxy commitment  $PC_0^{(j)} = g_2^{\alpha_0^{(j)}}$ , and  $\alpha_0^{(j)} = sk_j$  (or  $mk_j$ ), the aggregation of the first commitments  $PC_0 = mpk_i \cdot \prod_{j \in I_i} pk_{ij}$  is nothing but the aggregation of the public keys  $pk_{ij}$  of the corresponding shared secret keys  $sk_{ij}$  (and  $mpk_i$  for  $mk_i$ ). The second check is a standard consistency check for the VSS used in group setup, i.e., to guarantee that the received shares are consistent with the shared secrets. On the other hand, threshold solutions require a trusted combiner and honest majority assumption because any sufficient number (threshold) of malicious users can forge a valid signature. However, using the vASM scheme, each signer is responsible only for his signature. Even if all the unauthorized users (proxy) come together, they cannot forge the membership key of the authorized user.

**Remark 4.1.1.** *Any authorized user can assign a proxy via a vASM signature scheme. If she wants to delegate her own or organizational signing power, she participates in the compartment setup with her secret or membership key, respectively.*

**Remark 4.1.2.** *One can also consider sharing the membership key  $mk_i$  of the authorized user  $AU_i$  via a verifiable secret sharing scheme instead of an interactive compartment setup. However, in this case, one should be aware of the need for honest majority assumption.*

In the following, we propose three more constructions in which we consider the authorized user  $AU_i$  to share her membership key  $mk_i$  with different threshold SSS solutions with a trusted user and/or honest majority assumption.

#### 4.1.2 Shamir's Secret Sharing Scheme Based Delegation

In this construction of the delegation, each authorized user  $AU_i \in \mathcal{U}_i$  delegates her signing authority by sharing her membership key  $mk_i$  to a subset of unauthorized users  $u_{ij} \in \mathcal{U}_i$  via  $(t_i, k_i)$ -Shamir's SSS for  $j = 1, \dots, k_i$ , where  $t_i$  is the threshold and  $k_i$  is the number of users in compartment  $\mathcal{U}_i$  as shown in Figure 4.2.

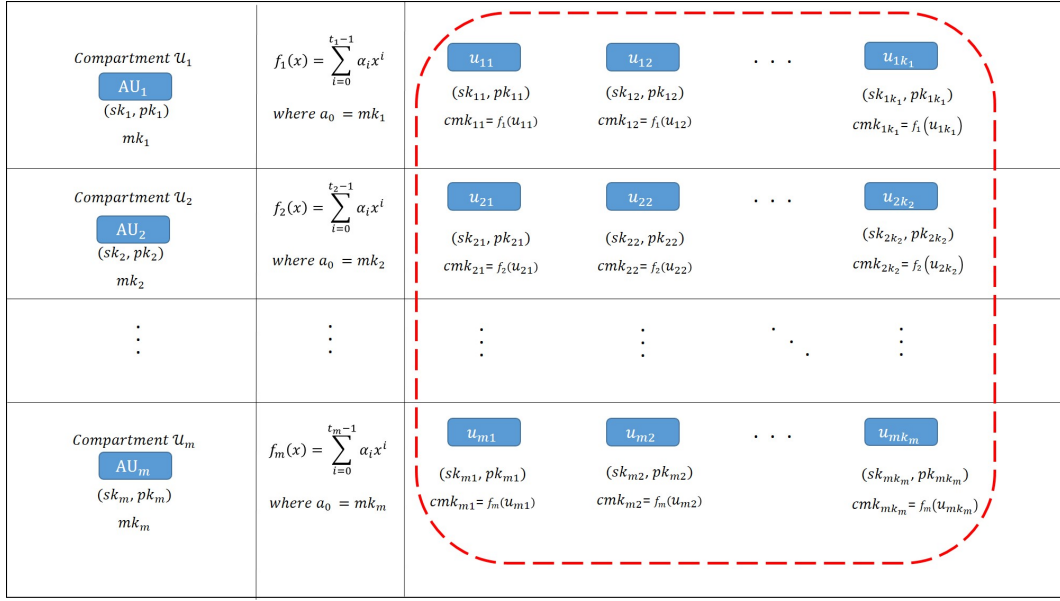


Figure 4.2: Shamir's Secret Sharing Scheme Based Delegation

Then, the unauthorized users  $u_{ij} \in \mathcal{G}$  for  $i = 1, 2, \dots, m$  and  $j = 1, 2, \dots, k_i$  sign as follows.

1. **Key Generation:** Each user  $u_{ij} \in \mathcal{G}$  picks uniformly at random a secret key  $sk_{ij} \xleftarrow{\$} \mathbb{Z}_q$ , and computes the public key  $pk_{ij} \leftarrow g_2^{sk_{ij}}$ , where  $g_2$  is a generator of  $\mathbb{G}_2$ .
2. **Compartment Setup:** Each  $AU_i$  shares her membership key  $mk_i$  via a  $(t_i, k_i)$ -Shamir's SSS [58] to a subset of unauthorized users  $u_{ij} \in \mathcal{U}_i$  as shown in Figure 4.2. At the end of this secret sharing procedure, each user  $u_{ij} \in \mathcal{U}_i$  obtains a compartment membership key  $cmk_{ij} = f_i(u_{ij})$ .
3. **Signature Generation:** Each user  $u_{ij} \in \mathcal{U}_i$  computes his individual signature on the message  $M$  as  $s_{ij} = H(M)^{sk_{ij} + cmk_{ij}}$ , and sends it to the combiner.
4. **Signature Aggregation:** After receiving the individual signatures from the

users, the combiner

- (a) forms the subgroup of signers  $\mathcal{S} := \{\mathcal{S}_1, \dots, \mathcal{S}_m\}$ , where  $\mathcal{S}_i$  is the subgroup of signers from compartment  $\mathcal{U}_i$ , for  $i = 1, \dots, m$ .
- (b) computes  $\sigma = \prod_{i=1}^m \prod_{j \in \mathcal{S}_i} s_{ij}^{\lambda_{ij}}$ , where  $\lambda_{ij}$  is the appropriate Lagrange coefficient as described in Definition 2.2.3.

5. **Verification:** If  $|\mathcal{S}_i| \geq t_i$  for  $i = 1, \dots, m$ , any verifier given  $(par, MPK, \mathcal{PK}, \mathcal{S}, M, \sigma)$  can verify the signature by checking the below equation

$$e\left(\mathbf{H}(M), \prod_{i=1}^m mpk_i \prod_{j \in \mathcal{S}_i} pk_{ij}^{\lambda_{ij}}\right) \stackrel{?}{=} e(\sigma, g_2).$$

The verification equation satisfies correctness as given below:

$$\begin{aligned} e(\sigma, g_2) &= e\left(\prod_{i=1}^m \prod_{j \in \mathcal{S}_i} s_{ij}^{\lambda_{ij}}, g_2\right) \\ &= e\left(\mathbf{H}(M)^{\sum_{i=1}^m \sum_{j \in \mathcal{S}_i} \lambda_{ij} sk_{ij} + \lambda_{ij} cmk_{ij}}, g_2\right) \\ &= e\left(\mathbf{H}(M), g_2^{\sum_{i=1}^m \sum_{j \in \mathcal{S}_i} \lambda_{ij} sk_{ij} + \lambda_{ij} cmk_{ij}}\right) \\ &= e\left(\mathbf{H}(M), g_2^{\sum_{i=1}^m \sum_{j \in \mathcal{S}_i} \lambda_{ij} cmk_{ij}} \cdot g_2^{\sum_{i=1}^m \sum_{j \in \mathcal{S}_i} \lambda_{ij} sk_{ij}}\right) \\ &= e\left(\mathbf{H}(M), \prod_{i=1}^m mpk_i \prod_{j \in \mathcal{S}_i} pk_{ij}^{\lambda_{ij}}\right). \end{aligned}$$

**Security.** In this scenario, each user  $u_{ij}$  signs a common message  $M$  with the sum of his secret and compartment membership keys, i.e.,  $sk_{ij} + cmk_{ij}$ . The resulting individual signature of user  $u_{ij}$  is  $s_{ij} = H(M)^{sk_{ij} + cmk_{ij}}$  which can be seen as a BLS signature [14] on message  $M$  with the key  $sk_{ij} + cmk_{ij}$ . Since  $sk_{ij}$  is sampled randomly and the  $cmk_{ij}$  is computed via an information-theoretically secure SSS scheme, the signing key  $sk_{ij} + cmk_{ij}$  will also be random-looking. Assume that the membership key  $cmk_{ij}$  is somehow obtained by an adversary. Even in this case, if the secret key  $sk_{ij}$  is secure, the individual signature  $s_{ij}$  of the user  $u_{ij}$  cannot be forged. On the other hand, if a sufficient number of unauthorized users are corrupted, then they can reconstruct the membership key  $mk_i$  of the authorized user  $AU_i$ . Therefore, we assume for this scenario that the number of corrupted users is less than threshold  $t$ .

Assigning a trusted user is a solution to this problem. In the following constructions, unauthorized users cannot obtain the membership key of the authorized user  $AU_i$  without compromising with the trusted user(s).

### 4.1.3 Trusted User Based Delegation

In this construction, each authorized user  $AU_i \in \mathcal{U}_i$  delegates her signing authority by sharing her membership key  $mk_i$  to a certain number of unauthorized users  $u_{ij} \in \mathcal{U}_i$ , including at least one trusted user. For simplicity, we assume that exactly one trusted user exists in each compartment. Note that one can also consider a group of trusted users. The authorized user  $AU_i \in \mathcal{U}_i$  uses Shamir's SSS in a nested way as described in Figure 4.3. Without loss of generality, we assume that  $u_{i1} \in \mathcal{U}_i$  is the trusted user in the  $i$ -th compartment.

The authorized user  $AU_i \in \mathcal{U}_i$  chooses two polynomials and distributes the shares as follows:

1. Chooses  $f_{i1}(x) = a_i x + mk_i$  for a random secret  $a_i \in \mathbb{Z}_q$ .
2. Sends  $f_{i1}(1)$  to the trusted user  $u_{i1}$ .
3. Chooses  $f_{i2}(x) = a_{t_i-2}^{(i)} x^{t_i-2} + \dots + a_1^{(i)} x_1 + f_{i1}(2)$ , where  $a_k^{(i)} \in \mathbb{Z}_q$  for  $k = 1, \dots, t_i - 2$ .
4. Sends  $f_{i2}(u_{ij})$  to the user  $u_{ij}$  for  $j = 2, \dots, k_i$ .

With the first polynomial evaluation, the authorized user  $AU_i$  shares her membership key  $mk_i$  via  $(2, 2)$ -Shamir's SSS and sends one of the shares to the trusted user. For the second share, she applies one more  $(t_i - 1, k_i - 1)$ -Shamir's SSS, where  $t_i$  is the threshold, and  $k_i$  is the number of users in compartment  $\mathcal{U}_i$ .

The unauthorized users  $u_{ij} \in \mathcal{G}$  for  $i = 1, 2, \dots, m$  and  $j = 1, 2, \dots, k_i$  sign as follows.

1. **Key Generation:** Each user  $u_{ij} \in \mathcal{G}$  picks uniformly at random a secret key  $sk_{ij} \xleftarrow{\$} \mathbb{Z}_q$ , and computes her public key  $pk_{ij} \leftarrow g_2^{sk_{ij}}$ , where  $g_2$  is a generator of  $\mathbb{G}_2$ .

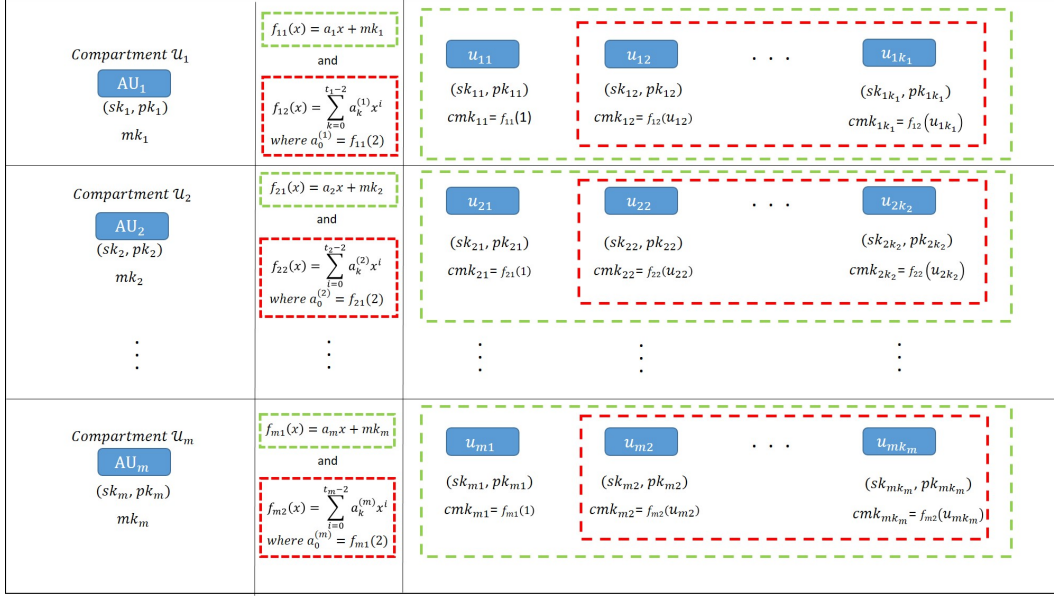


Figure 4.3: Trusted User Based Delegation

2. **Compartment Setup:** Each authorized user  $AU_i$  shares her membership key  $mk_i$  via Shamir's SSS [58] (first  $(2, 2)$ -Shamir's SSS, then  $(t_i - 1, k_i - 1)$ -Shamir's SSS) as shown in Figure 4.3. At the end of this nested secret sharing procedure, each user in  $\mathcal{U}_i$  obtains a compartment membership key  $cmk_{ij}$ .
3. **Signature Generation:** Each user  $u_{ij} \in \mathcal{U}_i$  computes his individual signature on the message  $M$  as  $s_{ij} = H(M)^{sk_{ij} + cmk_{ij}}$ , and sends it to the combiner.
4. **Signature Aggregation:** After receiving the individual signatures from the users, the combiner
  - (a) forms the subgroup of signers  $\mathcal{S} := \{\mathcal{S}_1, \dots, \mathcal{S}_m\}$ , where  $\mathcal{S}_i$  is the subgroup of signers from the compartment  $\mathcal{U}_i$ , for  $i = 1, \dots, m$ .
  - (b) computes  $\sigma = \prod_{i=1}^m s_{i1}^{\gamma_{i1}} \left( \prod_{\substack{j \in \mathcal{S}_i \\ j \neq 1}} s_{ij}^{\lambda_{ij}} \right)^{\gamma_{i2}}$ , where  $\gamma_{i1}$ ,  $\gamma_{i2}$  and  $\lambda_{ij}$  are the appropriate Lagrange coefficients as described in Definition 2.2.3, for the first and the second secret sharing scheme, respectively.
5. **Verification:** If the thresholds are satisfied and the trusted users participate in the signing, then any verifier given  $(par, MPK, \mathcal{PK}, \mathcal{S}, M, \sigma)$  can verify the signature by checking the below equation

$$e \left( H(M), \prod_{i=1}^m mpk_i \cdot pk_{i1}^{\gamma_{i1}} \left( \prod_{\substack{j \in \mathcal{S}_i \\ j \neq 1}} pk_{ij}^{\lambda_{ij}} \right)^{\gamma_{i2}} \right) \stackrel{?}{=} e(\sigma, g_2)$$

The verification equation satisfies correctness as given below:

$$\begin{aligned}
e(\sigma, g_2) &= e\left(\prod_{i=1}^m s_{i1}^{\gamma_{i1}} \left(\prod_{\substack{j \in \mathcal{S}_i \\ j \neq 1}} s_{ij}^{\lambda_{ij}}\right)^{\gamma_{i2}}, g_2\right) \\
&= e\left(\mathbf{H}(M)^{\sum_{i=1}^m \gamma_{i1} sk_{i1} + \gamma_{i1} cmk_{i1} + \sum_{i=1}^m \gamma_{i2} \left(\sum_{\substack{j \in \mathcal{S}_i \\ j \neq 1}} \lambda_{ij} sk_{ij} + \lambda_{ij} cmk_{ij}\right)}, g_2\right) \\
&= e\left(\mathbf{H}(M), g_2^{\sum_{i=1}^m \gamma_{i1} sk_{i1} + \gamma_{i1} cmk_{i1} + \sum_{i=1}^m \gamma_{i2} \left(\sum_{\substack{j \in \mathcal{S}_i \\ j \neq 1}} \lambda_{ij} sk_{ij} + \lambda_{ij} cmk_{ij}\right)}\right) \\
&= e\left(\mathbf{H}(M), g_2^{\sum_{i=1}^m \gamma_{i1} sk_{i1} + \sum_{i=1}^m \gamma_{i2} \left(\sum_{\substack{j \in \mathcal{S}_i \\ j \neq 1}} \lambda_{ij} sk_{ij}\right) + \sum_{i=1}^m \gamma_{i1} cmk_{i1} + \sum_{i=1}^m \gamma_{i2} \left(\sum_{\substack{j \in \mathcal{S}_i \\ j \neq 1}} \lambda_{ij} cmk_{ij}\right)}\right) \\
&= e\left(\mathbf{H}(M), g_2^{\sum_{i=1}^m \gamma_{i1} sk_{i1} + \sum_{i=1}^m \gamma_{i2} \left(\sum_{\substack{j \in \mathcal{S}_i \\ j \neq 1}} \lambda_{ij} sk_{ij}\right) + \sum_{i=1}^m mk_i}\right) \\
&= e\left(\mathbf{H}(M), \prod_{i=1}^m mpk_i \cdot pk_{i1}^{\gamma_{i1}} \left(\prod_{\substack{j \in \mathcal{S}_i \\ j \neq 1}} pk_{ij}^{\lambda_{ij}}\right)^{\gamma_{i2}}\right)
\end{aligned}$$

**Security.** Security of this construction similarly follows from the security discussion of the previous construction. In this scenario, like in the previous one, each user  $u_{ij}$  signs a message  $M$  with the sum of his secret and membership keys, i.e.,  $sk_{ij} + cmk_{ij}$ . The resulting individual signature is  $s_{ij} = H(M)^{sk_{ij} + cmk_{ij}}$  which can also be seen as a BLS signature [14] on the message  $M$  under the key  $sk_{ij} + cmk_{ij}$ . Because  $sk_{ij}$  is sampled uniformly at random, and  $cmk_{ij}$  is computed via the SSS scheme, the signing key  $sk_{ij} + cmk_{ij}$  will be uniformly random. Moreover, computing membership keys are done in a nested way. Namely, the membership key  $mk_i$  of the authorized user  $AU_i$  is partitioned into two parts. The first one is given to the trusted unauthorized user, and the other one is shared again among the other untrusted unauthorized users. Assume for a moment that the untrusted unauthorized users are corrupted, and they gather their membership keys  $cmk_{ij}$ . Even in this case, since the trusted user has the other half, corrupted users cannot forge the membership key  $mk_i$  of the authorized user  $AU_i$ .

We can assign a group of trusted users instead of a single one, and instead of using

SSS recursively, we can use a hierarchical threshold secret sharing scheme for the same purpose.

#### 4.1.4 Hierarchical Threshold Secret Sharing Scheme Based Delegation

Consider an organization with  $m$  compartments  $\mathcal{G} = \bigcup_{i=1}^m \mathcal{U}_i$  and each compartment  $\mathcal{U}_i$  has a hierarchical structure with  $r$  levels as shown in Figure 4.4. Assume that each authorized user  $AU_i \in \mathcal{U}_i$  delegates her signing capability to the unauthorized users via the hierarchical threshold secret sharing (HTSS) scheme proposed in [32] as defined in Section 2.2.3.

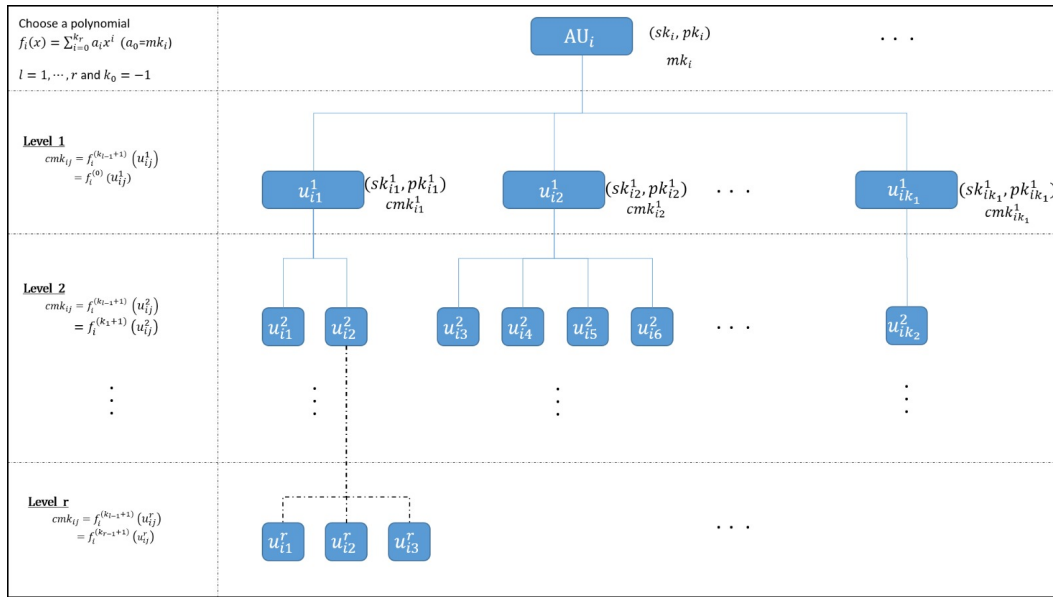


Figure 4.4: Hierarchical Threshold Secret Sharing Scheme Based Delegation

The unauthorized users  $u_{ij} \in \mathcal{G}$  for  $i = 1, 2, \dots, m$  and  $j = 1, 2, \dots, k_i$  sign as follows.

1. **Key Generation:** Each user  $u_{ij} \in \mathcal{G}$  picks uniformly at random a secret key  $sk_{ij} \xleftarrow{\$} \mathbb{Z}_q$ , and computes her public key  $pk_{ij} \leftarrow g_2^{sk_{ij}}$ , where  $g_2$  is a generator of  $\mathbb{G}_2$ .
2. **Compartment Setup:** Each  $AU_i$  shares her membership key  $mk_i$  via a HTSS scheme [32] as described in Figure 4.4. She sends compartment membership keys  $cmk_{ij} = f_i^{(k_{l-1}+1)}(u_{ij})$  to each user  $u_{ij} \in \mathcal{U}_i$ , for  $j = 1, \dots, k_i$ , and where  $l = 1, \dots, r$  is the level of the user  $u_{ij}$  belongs to.

3. **Signature Generation:** Each user  $u_{ij}$  computes his individual signature on the message  $M$  as  $s_{ij} = H(M)^{sk_{ij}+cmk_{ij}}$ , and sends it to the combiner.
4. **Signature aggregation:** After receiving the individual signatures from the users, the combiner
  - (a) forms the subgroup of signers  $\mathcal{S} := \{\mathcal{S}_1, \dots, \mathcal{S}_m\}$ , where  $\mathcal{S}_i$  is the subgroup of signers from the compartment  $\mathcal{U}_i$ , for  $i = 1, \dots, m$ .
  - (b) computes  $\sigma = \prod_{i=1}^m \prod_{j \in \mathcal{S}_i} s_{ij}^{\beta_{ij}}$ , where  $\beta_{ij}$  is the appropriate Birkhoff coefficients as described in Definition 2.2.5.
5. **Verification:** If the level-specific thresholds are satisfied, then any verifier given  $(par, MPK, \mathcal{PK}, \mathcal{S}, M, \sigma)$  can verify the signature by checking the below equation

$$e\left(\mathbf{H}(M), \prod_{i=1}^m mpk_i \prod_{j \in \mathcal{S}_i} pk_{ij}^{\beta_{ij}}\right) \stackrel{?}{=} e(\sigma, g_2)$$

The verification equation satisfies correctness as given below:

$$\begin{aligned} e(\sigma, g_2) &= e\left(\prod_{i=1}^m \prod_{j \in \mathcal{S}_i} s_{ij}^{\beta_{ij}}, g_2\right) \\ &= e\left(\mathbf{H}(M)^{\sum_{i=1}^m \sum_{j \in \mathcal{S}_i} \beta_{ij} sk_{ij} + \beta_{ij} cmk_{ij}}, g_2\right) \\ &= e\left(\mathbf{H}(M), g_2^{\sum_{i=1}^m \sum_{j \in \mathcal{S}_i} \beta_{ij} sk_{ij} + \beta_{ij} cmk_{ij}}\right) \\ &= e\left(\mathbf{H}(M), g_2^{\sum_{i=1}^m \sum_{j \in \mathcal{S}_i} \beta_{ij} cmk_{ij}} \cdot g_2^{\sum_{i=1}^m \sum_{j \in \mathcal{S}_i} \beta_{ij} sk_{ij}}\right) \\ &= e\left(\mathbf{H}(M), \prod_{i=1}^m mpk_i \prod_{j \in \mathcal{S}_i} pk_{ij}^{\beta_{ij}}\right). \end{aligned}$$

**Security.** Like in the previous ones, each user  $u_{ij}$  signs a common message  $M$  with a combination of his secret and compartment membership keys, i.e.,  $sk_{ij} + cmk_{ij}$ . Then the signature  $s_{ij} = H(M)^{sk_{ij}+cmk_{ij}}$  can be also seen as a BLS signature [14] on the message  $M$  with the key  $sk_{ij} + cmk_{ij}$ . Because  $sk_{ij}$  is picked uniformly at random and the  $cmk_{ij}$  is computed in an HTSS scheme, the signing key  $sk_{ij} + cmk_{ij}$  will also be random. Moreover, the authorized user  $AU_i$  chooses a secret polynomial such that the constant term is her membership key  $mk_i$  and shares it among unauthorized

users (subordinates) according to their levels. The unauthorized users in the first level get their shares like in Shamir's SSS. Other users in the following levels get their shares from the polynomials' evaluations, which are some certain order derivatives of the first level polynomial. Since an order  $d > 0$  derivative kills the constant term, the users in the lower levels cannot obtain the  $mk_i$  without cooperating with the first level (i.e., trusted) users. Therefore, while distributing the shares, the authorized user  $AU_i$  assumes that her trusted subordinates are in the first level.

**Remark 4.1.3.** *In the original vASM scheme, the messages are signed by using only the membership keys. However, in our constructions, we assume that messages are signed using the sum of secret keys and compartment membership keys. In this way, we discard the natural anonymity of threshold secret sharing schemes; and ensure accountability.*

**Remark 4.1.4.** *Notice that the methods we give above, except Section 4.1.1, provide conditional security. If enough shareholders are corrupted, they can easily forge the membership key of the authorized user in their compartment. Since it is an accountable subgroup multi-signature, they can sign on behalf of the entire organization. Therefore one should be aware of the need for a trusted user/combiner in each construction except the one in Section 4.1.1.*

## 4.2 Comparison

### 4.2.1 Comparison of the proposed constructions

We compare our proposed constructions with each other and give the number of operations in each phase of the proposed schemes in Table 4.1. The number of main operations, such that group operations and bilinear pairings are the same for Shamir's SSS-based, Trusted user-based, and HTSS-based constructions. The main difference between these three constructions emerges in computing Lagrange and Birkhoff coefficients. For Shamir's SSS-based and trusted user-based delegation, Lagrange coefficients should be computed in the signature aggregation and in the verification phase by the designated combiner and the verifier, respectively. For HTSS-based delegation, Birkhoff coefficients should also be computed for the same phases. One needs

to perform simple integer addition and multiplication to compute the Lagrange coefficients, whereas additional matrix operations should be conducted to compute the Birkhoff coefficients. Finally, since they all use threshold schemes, they all are secure when an adversary can corrupt up to  $t$  signers, where  $t$  is the threshold of the construction.

The number of operations required by the recursive vASM construction is less than the others. Only the compartment setup phase requires more operation than the others constructions. Since it is a one-time setup, it can be omitted. Moreover, the recursive vASM construction has a more robust assumption regarding the number of users an adversary can corrupt. Since the group setup is performed via an interactive  $(n, n)$ -VSS scheme, only one honest participant (possibly the authorized user or original signer) suffices to ensure the system's security.

Table 4.1: Comparison of the delegation methods

Phases	Recursive vASM (Section 4.1.1)	Shamir's SSS based (Section 4.1.2)	Trusted user based (Section 4.1.3)	HTSS based (Section 4.1.4)
Key Generation	1 $\text{Exp}_{\mathbb{G}_2}$	1 $\text{Exp}_{\mathbb{G}_2}$	1 $\text{Exp}_{\mathbb{G}_2}$	1 $\text{Exp}_{\mathbb{G}_2}$
Compartment Setup	$m + 2k_i \text{Exp}_{\mathbb{G}_2}$ $m + k_i(k_i + 1) - 2 \text{Mul}_{\mathbb{G}_2}$	$(t_i, k_i)$ -SSS	(2, 2)-SSS $(t_i - 1, k_i - 1)$ -SSS	HTSS with $m$ levels
Signature Generation	1 Hash 1 $\text{Exp}_{\mathbb{G}_1}$	1 Hash 1 $\text{Exp}_{\mathbb{G}_1}$	1 Hash 1 $\text{Exp}_{\mathbb{G}_1}$	1 Hash 1 $\text{Exp}_{\mathbb{G}_1}$
Signature Aggregation	$k_i - 1 \text{Mul}_{\mathbb{G}_1}$	$l \text{Exp}_{\mathbb{G}_1}$ $l - 1 \text{Mul}_{\mathbb{G}_1}$ $l$ Lagrange Coef.	$l \text{Exp}_{\mathbb{G}_1}$ $l - 1 \text{Mul}_{\mathbb{G}_1}$ $l + 1$ Lagrange Coef.	$l \text{Exp}_{\mathbb{G}_1}$ $l - 1 \text{Mul}_{\mathbb{G}_1}$ $l$ Birkhoff Coef.
Verification	1 Hash $k_i - 1 \text{Mul}_{\mathbb{G}_2}$ 2 pairings	1 Hash $m + l \text{Exp}_{\mathbb{G}_2}$ $m + l - 1 \text{Mul}_{\mathbb{G}_2}$ 2 pairings $l$ Lagrange Coef.	1 Hash $m + l \text{Exp}_{\mathbb{G}_2}$ $m + l - 1 \text{Mul}_{\mathbb{G}_2}$ 2 pairings $l + 1$ Lagrange Coef.	1 Hash $m + l \text{Exp}_{\mathbb{G}_2}$ $m + l - 1 \text{Mul}_{\mathbb{G}_2}$ 2 pairings $l$ Birkhoff Coef.

$\text{Exp}_{\mathbb{G}_i}$  : Exponentiation in group  $\mathbb{G}_i$ ,  $i \in \{1, 2\}$

$\text{Mul}_{\mathbb{G}_i}$  : Multiplication in group  $\mathbb{G}_i$ ,  $i \in \{1, 2\}$

$k_i$  : Number of signers in  $\mathcal{U}_i$ , i.e.,  $i$ -th compartment

$m$  : Number of compartments in the organization  $\mathcal{G}$

$l$  : Number of all signers in the organization  $\mathcal{G}$ , i.e.,  $l = m \cdot k_i$

#### 4.2.2 Comparison of vASM scheme with the existing proxy signature schemes

In literature, many proxy signature schemes and variants of this notion have similar but not the same definitions. There are mainly three parties in all kinds of proxy signatures, i.e., the original signer, the proxy signer, and the verifier. Here is the high level description of existing proxy signatures. An original signer (or a set of original signers) generates and signs a warrant that contains detailed information about the delegation, such as proxy IDs, validity time, etc. Then the original signer sends this warrant signature to the proxy signer (or a set of proxy signers). The proxy signer

signs on behalf of the original signer using this warrant signature and the corresponding public keys of the two sides. However, the verifier’s job is the same as in standard signature schemes, with the difference that the warrant signature needs to be verified along with the signature on a message.

Boldyreva et al. formally defined the proxy signatures in [11] for the first time, and defined what functionalities a proxy signature scheme should provide to the users. We compare the functionalities of the formal definition of proxy signature that is given in Section 2.2.1 with our constructions.

1. **Proxy-designation protocol  $(\mathcal{D}, \mathcal{P})$  functionality:** In the recursive vASM construction, instead of the protocols  $\mathcal{D}$  and  $\mathcal{P}$ , we use a compartment setup in which all the original signers and the proxy signers jointly participate in an interactive Feldman’s VSS. In the end, proxy signers get their compartment membership keys  $cmk_{ij}$  as an analog of proxy signing keys  $skp$ . In the other constructions, we use secret sharing schemes in the compartment setup phases, and the outputs of these phases are the compartment membership keys used for proxy signing.
2. **Signing and verification functionalities:** Analogy between signing and verification functionalities are straightforward.
3. **Identification functionality:** All of our constructions have accountability properties because of the underlying signature scheme, i.e., vASM. The verifiers should know the identities of the proxy signers to verify the proxy signature properly.

Although the vASM scheme is an accountable subgroup multi-signature scheme, it provides the functionalities of a proxy signature with a flexible number of original and proxy signers. It can serve as several types of proxy signatures according to the number of original and proxy signers ( $n$  and  $l$ , respectively) as described in the following subsections.

#### 4.2.2.1 vASM signature scheme can serve as a proxy signature ( $n = l = 1$ )

To that end the original signer and the proxy signer generate their secret and public keys, participate in a group setup and obtain their membership keys. Then the proxy signer-using his membership key- can sign any message on behalf of the original signer in an accountable way. We compare vASM scheme with existing proxy signatures in Table 4.2. We choose the best -up to our knowledge- pairing-based proxy signature schemes for comparison. It can be seen from the table that the proxy key generation (group setup) of vASM scheme requires less operations than the existing ones which require at least one or more pairings and a number of elliptic curve additions, scalar multiplications and hash computations. In signature generation and the verification phases, vASM requires less operations than most of the schemes but not all. In terms of the signature size, vASM is better than all other schemes. The size of a vASM signature is just one group element whereas all others have more than one. Note that the actual efficiency depends on the system parameters (e.g., elliptic curve groups, hash functions, etc.) that are used to instantiate the schemes. Moreover, we also add the Schnorr-based scheme proposed by Boldyreva et al. [11] to the table. Although it is not a pairing-based one we add it in our table to make the readers able to compare the number of operations.

#### 4.2.2.2 vASM signature scheme can serve as a proxy multi-signature ( $n > 1$ and $l = 1$ )

$n$  original signers and the proxy signer jointly participate in the group setup of vASM scheme. This way, each original signer authorizes the proxy signer to sign on behalf of himself. We give an efficiency comparison of vASM and the existing pairing-based proxy multi-signature schemes in Table 4.3. It can be easily seen that vASM scheme is much more efficient than the existing pairing-based proxy multi-signature schemes in terms of all comparison parameters. vASM does not require pairing operations for proxy key generation phases, whereas the others require many pairings. vASM has also better computational efficiency than the others in terms of signature generation and verification phases. The signature size of vASM is only one group element, while others result in more.

Table 4.2: Comparison with the existing proxy signature schemes

Schemes	Proxy key generation		Signature generation	Verification	Signature size
	Original signer	Proxy signer			
<b>Boldyreva et al.</b> <sup>(*)</sup> [11]	1 $Exp_p$ 1 $H_{\mathbb{Z}_q}$	2 $Exp_p$ 3 $H_{\mathbb{Z}_q}$ 1 $Mul_q$	1 $Exp_p$ 3 $H_{\mathbb{Z}_q}$	3 $Exp_p$ 1 $H_{\mathbb{Z}_q}$ 3 $Mul_q$	$3 \mathbb{Z}_p  +  \mathbb{Z}_q  +  M_w $
<b>Lee et al.</b> [36]	1 $\mathcal{P}$ 1 $Exp_{\mathbb{G}}$ 1 $H_{\mathbb{G}}$	1 $\mathcal{P}$ 1 $H_{\mathbb{G}}$	2 $\mathcal{P}$ 2 $Exp_{\mathbb{G}}$ 1 $H_{\mathbb{Z}_q}$ 1 $Exp_{\mathbb{G}_T}$	1 $\mathcal{P}$ 1 $Exp_{\mathbb{G}}$ 1 $H_{\mathbb{Z}_q}$	$ \mathbb{G}  +  \mathbb{G}_T $
<b>Shim</b> [59]	1 $Exp_{\mathbb{G}}$ 1 $H_{\mathbb{G}}$	2 $\mathcal{P}$ 1 $Exp_{\mathbb{G}}$ 1 $Mul_{\mathbb{G}}$ 2 $H_{\mathbb{G}}$	1 $\mathcal{P}$ 1 $Exp_{\mathbb{G}}$ 1 $Mul_{\mathbb{G}}$ 1 $H_{\mathbb{G}}$	2 $\mathcal{P}$ 2 $Exp_{\mathbb{G}}$ 1 $Mul_{\mathbb{G}}$ 3 $H_{\mathbb{G}}$ 1 $Mul_{\mathbb{G}_T}$	$ \mathbb{G}_T  +  M_w $
<b>Zhang et al.</b> [71]	3 $Exp_{\mathbb{G}}$ 2 $Mul_{\mathbb{G}}$ 2 $H_{\mathbb{G}}$	4 $\mathcal{P}$ 1 $Exp_{\mathbb{G}}$ 2 $Mul_{\mathbb{G}}$ 4 $H_{\mathbb{G}}$ 2 $Mul_{\mathbb{G}_T}$	2 $Exp_{\mathbb{G}}$ 1 $Mul_{\mathbb{G}}$ 2 $H_{\mathbb{G}}$	5 $\mathcal{P}$ 2 $Mul_{\mathbb{G}}$ 5 $H_{\mathbb{G}}$ 3 $Mul_{\mathbb{G}_T}$	$3 \mathbb{G}  +  M_w $
<b>Seo et al.</b> [57]	2 $Exp_{\mathbb{G}}$ 1 $Mul_{\mathbb{G}}$ 1 $H_{\mathbb{G}}$ 1 $H_{\mathbb{Z}_q}$	3 $\mathcal{P}$ 1 $Exp_{\mathbb{G}}$ 2 $Mul_{\mathbb{G}}$ 1 $H_{\mathbb{G}}$ 1 $Mul_{\mathbb{G}_T}$	2 $Exp_{\mathbb{G}}$ 1 $Mul_{\mathbb{G}}$ 1 $H_{\mathbb{G}}$ 1 $H_{\mathbb{Z}_q}$	4 $\mathcal{P}$ 2 $Exp_{\mathbb{G}}$ 3 $Mul_{\mathbb{G}}$ , 4 $H_{\mathbb{G}}$ 2 $H_{\mathbb{Z}_q}$ 2 $Mul_{\mathbb{G}_T}$	$3 \mathbb{G}  +  M_w $
<b>Verma &amp; Singh</b> [66]	1 $Exp_{\mathbb{G}}$ 1 $H_{\mathbb{G}}$	2 $\mathcal{P}$ 1 $H_{\mathbb{G}}$	1 $Exp_{\mathbb{G}}$ 1 $H_{\mathbb{Z}_q}$	2 $\mathcal{P}$ 1 $Exp_{\mathbb{G}}$ 1 $Mul_{\mathbb{G}}$ 1 $H_{\mathbb{G}}$ 1 $H_{\mathbb{Z}_q}$	$ \mathbb{G}  +  M_w $
<b>vASM</b> [3]		4 $Exp_{\mathbb{G}_2}$ 4 $Mul_{\mathbb{G}_2}$	1 $H_{\mathbb{G}_1}$ 1 $Exp_{\mathbb{G}_1}$	2 $\mathcal{P}$ 1 $Mul_{\mathbb{G}_2}$ 1 $H_{\mathbb{G}_1}$	$ \mathbb{G}_1 $

Note that since the group operations determine the efficiency numbers we ignore the integer operations.

(\*) Notice that all the schemes except [11] are pairing-based schemes.

$\mathcal{P}$ : Bilinear pairing operation.

$Exp_A$ : Exponentiation in group  $A$ , where  $A \in \{\mathbb{G}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T\}$ .

$Mul_A$ : Multiplication in group  $A$ , where  $A \in \{\mathbb{G}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T\}$ .

$H_A$ : Hash onto the set  $A$ , where  $A \in \{\mathbb{G}, \mathbb{G}_1, \mathbb{Z}_q\}$ .

$|M_w|$ : Size of the warrant in bits.

$|A|$ : Size of the elements of the set  $A$ , where  $A \in \{G, G_1\}$

#### 4.2.2.3 vASM signature scheme can serve as a multi-proxy signature ( $n = 1$ and $l > 1$ )

If an original signer participates in a group setup procedure with  $l$  proxy signers, he authorizes the proxy signers by sharing his own secret key in the group setup (for

Table 4.3: Comparison with the existing proxy multi-signature schemes

Schemes	Proxy key generation		Signature generation	Verification	Signature size
	Original signer	Proxy signer			
Li & Chen [38]	$3 \text{Exp}_{\mathbb{G}}$ $n \text{Mul}_{\mathbb{G}}$ $1 H_{\mathbb{Z}_q}$	$3n \mathcal{P}$ $1 \text{Exp}_{\mathbb{G}}$ $n \text{Mul}_{\mathbb{G}}$ $1 H_{\mathbb{Z}_q}$ $n \text{Exp}_{\mathbb{G}_T}$ $n \text{Mul}_{\mathbb{G}_T}$	$1 \mathcal{P}$ $2 \text{Exp}_{\mathbb{G}}$ $1 \text{Mul}_{\mathbb{G}}$ $1 H_{\mathbb{Z}_q}$ $1 \text{Exp}_{\mathbb{G}_T}$	$3 \mathcal{P}$ $n \text{Mul}_{\mathbb{G}}$ $2 H_{\mathbb{Z}_q}$ $2 \text{Exp}_{\mathbb{G}_T}$ $2 \text{Mul}_{\mathbb{G}_T}$	$2 \mathbb{G}  +  \mathbb{Z}_q  +  M_w $
Du & Wen [22]	$3 \text{Exp}_{\mathbb{G}}$ $(n+1) \text{Mul}_{\mathbb{G}}$ $2 H_{\mathbb{G}}$	$4n \mathcal{P}$ $(2n-1) \text{Mul}_{\mathbb{G}}$ $2n H_{\mathbb{G}}$ $2n \text{Mul}_{\mathbb{G}_T}$	$3 \text{Exp}_{\mathbb{G}}$ $3 \text{Mul}_{\mathbb{G}}$ $2 H_{\mathbb{G}}$	$6 \mathcal{P}$ $(2n-1) \text{Mul}_{\mathbb{G}}$ $4 H_{\mathbb{G}}$ $4 \text{Mul}_{\mathbb{G}_T}$	$3 \mathbb{G}  +  M_w $
vASM [3]	$2n \text{Exp}_{\mathbb{G}_2}$ $(n^2 + n - 2) \text{Mul}_{\mathbb{G}_2}$		$1 H_{\mathbb{G}_1}$ $1 \text{Exp}_{\mathbb{G}_1}$	$2 \mathcal{P}$ $(n-1) \text{Mul}_{\mathbb{G}_2}$ $1 H_{\mathbb{G}_1}$	$ \mathbb{G}_1 $

Note that since the group operations determine the efficiency numbers we ignore the integer operations.

$\mathcal{P}$ : Bilinear pairing operation.

$\text{Exp}_A$ : Exponentiation in group  $A$ , where  $A \in \{\mathbb{G}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T\}$ .

$\text{Mul}_A$ : Multiplication in group  $A$ , where  $A \in \{\mathbb{G}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T\}$ .

$H_A$ : Hash onto the set  $A$ , where  $A \in \{\mathbb{G}, \mathbb{G}_1, \mathbb{Z}_q\}$ .

$n$ : Number of original signers.

$|M_w|$ : Size of the warrant in bits.

$|A|$ : Size of the elements of the set  $A$ , where  $A \in \{\mathbb{G}, \mathbb{G}_1\}$

details see Section 3.1). In Table 4.4, we compare the vASM scheme with existing pairing-based multi-proxy signature schemes. In general, vASM scheme requires less number of operations that are hard to compute (e.g. bilinear pairings). But the other schemes may have better computational efficiency as the number of proxies increase. For example, for a very large  $l$ , the cost of proxy key generation (group setup) of vASM requires more time than a few pairings which are required by the other schemes in the same phase. But for other comparison parameters, i.e. signature generation, aggregation, verification and signature size, vASM has better efficiency numbers than the other existing pairing-based multi-proxy signature schemes.

#### 4.2.2.4 vASM signature scheme can serve as a multi-proxy multi-signature ( $n > 1$ and $l > 1$ )

To this end,  $n$  original signers and  $l$  proxy signers jointly perform a group setup. As a result of this group setup, each proxy signer has a membership key, which can be used as a proxy key. Any number of proxy signers can sign on behalf of the original signers using these membership keys. We compare the vASM scheme with the existing pairing-based multi-proxy multi-signature schemes in Table 4.5. vASM scheme has slightly better efficiency numbers than the other schemes.

**Remark 4.2.1.** Note that vASM scheme provides only a general delegation of signing

Table 4.4: Comparison with the existing multi-proxy signature schemes

Scheme	Proxy key generation		Signature generation	Aggregation	Verification	Signature size
	Original signer	Proxy signer				
Li & Chen [38]	$3 \text{Exp}_{\mathbb{G}}$ $1 \text{Mul}_{\mathbb{G}}$ $1 H_{\mathbb{Z}_q}$	$3 \mathcal{P}$ $2 \text{Exp}_{\mathbb{G}}$ $3 \text{Mul}_{\mathbb{G}}$ $1 H_{\mathbb{Z}_q}$ $1 \text{Exp}_{\mathbb{G}_T}$ $1 \text{Mul}_{\mathbb{G}_T}$	$3 \text{Exp}_{\mathbb{G}}$ $l \text{Mul}_{\mathbb{G}}$ $1 H_{\mathbb{Z}_q}$	$3l \mathcal{P}$ $(l-1) \text{Mul}_{\mathbb{G}}$ $l H_{\mathbb{Z}_q}$ $l \text{Exp}_{\mathbb{G}_T}$ $l \text{Mul}_{\mathbb{G}_T}$	$3 \mathcal{P}$ $l \text{Exp}_{\mathbb{G}}$ $(3l-1) \text{Mul}_{\mathbb{G}}$ $(l+1) H_{\mathbb{G}}$ $2 H_{\mathbb{Z}_q}$ $1 \text{Exp}_{\mathbb{G}_T}$	$3 \mathbb{G}  +  M_w $
Cao & Cao [18]	$2 \text{Exp}_{\mathbb{G}}$ $1 \text{Mul}_{\mathbb{G}}$ $1 H_{\mathbb{G}}$	$3 \mathcal{P}$ $1 \text{Exp}_{\mathbb{G}}$ $1 \text{Mul}_{\mathbb{G}}$ $2 H_{\mathbb{G}}$ $1 H_{\mathbb{Z}_q}$	$2 \text{Exp}_{\mathbb{G}}$ $l \text{Mul}_{\mathbb{G}}$ $1 H_{\mathbb{G}}$	$5 \mathcal{P}$ $(l-1) \text{Mul}_{\mathbb{G}}$ $2 H_{\mathbb{G}}$ $1 H_{\mathbb{Z}_q}$ $1 \text{Exp}_{\mathbb{G}_T}$ $3 \text{Mul}_{\mathbb{G}_T}$	$5 \mathcal{P}$ $3 \text{Exp}_{\mathbb{G}}$ $l \text{Mul}_{\mathbb{G}}$ $2 H_{\mathbb{G}}$ $1 H_{\mathbb{Z}_q}$ $3 \text{Mul}_{\mathbb{G}_T}$	$3 \mathbb{G}  +  M_w $
Liu et al. [41]	$7 \text{Exp}_{\mathbb{G}}$ $2 M_w  \text{Mul}_{\mathbb{G}}$	$2 \mathcal{P}$ $ M_w  \text{Mul}_{\mathbb{G}}$ $2 \text{Mul}_{\mathbb{G}_T}$	$3 \mathcal{P}$ $5 \text{Exp}_{\mathbb{G}}$ $(2 M_w  + 7) \text{Mul}_{\mathbb{G}}$ $3 \text{Mul}_{\mathbb{G}_T}$	$3l \mathcal{P}$ $(4l-2) \text{Mul}_{\mathbb{G}}$ $3 \text{Mul}_{\mathbb{G}_T}$	$3 \mathcal{P}$ $2 M_w  \text{Mul}_{\mathbb{G}}$ $(l+2) \text{Mul}_{\mathbb{G}_T}$	$3 \mathbb{G}  +  M_w $
vASM [3]	$2l \text{Exp}_{\mathbb{G}_2}$ $(l^2 + l - 2) \text{Mul}_{\mathbb{G}_2}$		$1 H_{\mathbb{G}_1}$ $1 \text{Exp}_{\mathbb{G}_1}$	$(l-1) \text{Mul}_{\mathbb{G}_1}$	$2 \mathcal{P}$ $(l-1) \text{Mul}_{\mathbb{G}_2}$ $1 H_{\mathbb{G}_1}$	$ \mathbb{G}_1 $

Note that since the group operations determine the efficiency numbers we ignore the integer operations.

$\mathcal{P}$ : Bilinear pairing operation.

$\text{Exp}_A$ : Exponentiation in group  $A$ , where  $A \in \{\mathbb{G}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T\}$ .

$\text{Mul}_A$ : Multiplication in group  $A$ , where  $A \in \{\mathbb{G}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T\}$ .

$H_A$ : Hash onto the set  $A$ , where  $A \in \{\mathbb{G}, \mathbb{G}_1, \mathbb{Z}_q\}$ .

$l$ : Number of proxy signers

$|M_w|$ : Size of the warrant in bits.

$|A|$ : Size of the elements of the set  $A$ , where  $A \in \{G, G_1\}$

rights. Since there is no warrant-like information the delegation is valid until it is canceled (by invalidating the commitment set  $COM$ ).

**Remark 4.2.2.** One important property of vASM is that it has aggregation property. For  $N$  distinct vASM signatures, one can aggregate all signatures into one which is a single  $\mathbb{G}_1$  element. In this case, verification of the aggregated vASM signature requires  $N + 1$  pairing operations instead of  $3N + 1$ . Further information about aggregation of vASM signatures can be found in [3].

Table 4.5: Comparison with the existing multi-proxy multi-signature schemes

Scheme	Proxy key generation			Signature generation	Aggregation	Verification	Signature size
	Original signer	Proxy signer	Chairman				
<b>Li &amp; Chen</b> [38]	$3 \text{Exp}_{\mathbb{G}}$ $(n+l) \text{Mul}_{\mathbb{G}}$ $1 \text{H}_{\mathbb{Z}_q}$	$3 \text{Exp}_{\mathbb{G}}$ $(n+l) \text{Mul}_{\mathbb{G}}$ $1 \text{H}_{\mathbb{Z}_q}$	$3(n+l) \mathcal{P}$ $(n+l-1) \text{Mul}_{\mathbb{G}}$ $1 \text{H}_{\mathbb{Z}_q}$ $(n+l) \text{Exp}_{\mathbb{G}_T}$ $(n+l) \text{Mul}_{\mathbb{G}_T}$	$3 \text{Exp}_{\mathbb{G}}$ $l \text{Mul}_{\mathbb{G}}$ $1 \text{H}_{\mathbb{Z}_q}$	$3l \mathcal{P}$ $(l-1) \text{Mul}_{\mathbb{G}}$ $l \text{H}_{\mathbb{Z}_q}$ $l \text{Exp}_{\mathbb{G}_T}$ $l \text{Mul}_{\mathbb{G}_T}$	$6 \mathcal{P}$ $(n+2l-2) \text{Mul}_{\mathbb{G}}$ $2 \text{H}_{\mathbb{Z}_q}$ $2 \text{Exp}_{\mathbb{G}_T}$ $2 \text{Mul}_{\mathbb{G}_T}$	$4 \mathbb{G}  +  M_w $
<b>Sahu &amp; Padhye</b> [55]	$2 \text{Exp}_{\mathbb{G}}$ $(n-1) \text{Mul}_{\mathbb{G}}$ $1 \text{H}_{\mathbb{Z}_q}$	$2n \mathcal{P}$ $(n+l) \text{Exp}_{\mathbb{G}}$ $(n+l-1) \text{Mul}_{\mathbb{G}}$ $(n+1) \text{H}_{\mathbb{Z}_q}$	NA	$1 \mathcal{P}$ $2 \text{Exp}_{\mathbb{G}}$ $1 \text{Mul}_{\mathbb{G}}$ $1 \text{H}_{\mathbb{Z}_q}$ $1 \text{Exp}_{\mathbb{G}_T}$ $(l-1) \text{Mul}_{\mathbb{G}_T}$	$2 \mathcal{P}$ $l \text{Exp}_{\mathbb{G}}$ $(nl+l-1) \text{Mul}_{\mathbb{G}}$ $l \text{H}_{\mathbb{Z}_q}$ $l \text{Exp}_{\mathbb{G}_T}$ $l \text{Mul}_{\mathbb{G}_T}$	$2 \mathcal{P}$ $2 \text{Exp}_{\mathbb{G}}$ $(nl-1) \text{Mul}_{\mathbb{G}}$ $2 \text{H}_{\mathbb{Z}_q}$ $1 \text{Exp}_{\mathbb{G}_T}$ $1 \text{Mul}_{\mathbb{G}_T}$	$3 \mathbb{G}  +  M_w $
VASM [3]	$2(n+l) \text{Exp}_{\mathbb{G}_2}$ $((n+l)^2 + (n+l) - 2) \text{Mul}_{\mathbb{G}_2}$			$1 \text{H}_{\mathbb{G}_1}$ $1 \text{Exp}_{\mathbb{G}_1}$	$(l-1) \text{Mul}_{\mathbb{G}_1}$	$2 \mathcal{P}$ $(n+l-1) \text{Mul}_{\mathbb{G}_2}$ $1 \text{H}_{\mathbb{G}_1}$	$ \mathbb{G}_1 $

Note that since the group operations determine the efficiency numbers we ignore the integer operations.

$\mathcal{P}$ : Bilinear pairing operation.

$\text{Exp}_A$ : Exponentiation in group  $A$ , where  $A \in \{\mathbb{G}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T\}$ .

$\text{Mul}_A$ : Multiplication in group  $A$ , where  $A \in \{\mathbb{G}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T\}$ .

$\text{H}_A$ : Hash onto the set  $A$ , where  $A \in \{\mathbb{G}, \mathbb{G}_1, \mathbb{Z}_q\}$ .

$n$ : Number of original signers.

$l$ : Number of proxy signers

$|M_w|$ : Size of the warrant in bits.  $|A|$ : Size of the elements of the set  $A$ , where  $A \in \{G, G_1\}$



## CHAPTER 5

### A LATTICE-BASED ACCOUNTABLE SUBGROUP MULTI-SIGNATURE SCHEME

In this chapter, we propose a novel lattice-based accountable subgroup multi-signature scheme. We use Damgård et al.'s lattice-based  $\text{MS}_2$  multi-signature scheme [20] which is based on the Dilithium scheme. We follow the same group setup method in Section 3.1 for the group setup phase in our construction. Each user in the group  $\mathcal{G}$  share her secret key via a VSS, and upon receiving other shares from other users, each user computes her membership keys by simply summing up the shares she received. However, summing the shares entails a problem in the size of the membership keys since the size (infinity norm, see Section 2.3.1) of the signing keys are crucial in the underlying  $\text{MS}_2$  scheme. To tackle this issue, we allow users in  $\mathcal{G}$  to sample a normalizer vector from a special distribution to ensure that the resulting membership keys are of the desired size. We should also note that adding accountability is not a cost-free operation, it comes with a cost of one-time one-round group setup phase. One can also find the details in [6].

#### 5.1 $\text{vMS}_2$ : A lattice-based ASM scheme with verifiable group setup

In the  $\text{vASM}$  scheme (see section 3.1), it is considered to sign a message with a special signing key, i.e., membership key ( $mk$ ). Each user in the group  $\mathcal{G}$  participate in a 1-round interactive group setup protocol in which each user shares his own secret key ( $sk$ ) via a verifiable secret sharing scheme. Upon receiving the shares from other users, each user sums those shares up and computes his membership key ( $mk$ ).

Therefore, each membership key is composed of the secret keys of all users in the group  $\mathcal{G}$ . So, this construction provides accountability to any signer who signs with a membership key. Below we apply the above mentioned group setup method to the  $\text{MS}_2$  scheme [20].

1. **Key Generation Phase:** Let  $\mathcal{G}$  be a group of  $n$  potential signers and let  $\mathcal{S} \subseteq \mathcal{G}$  be the subgroup of  $\tau$  signers among  $n$ . Let  $H_1 : \{0, 1\}^* \rightarrow \{0, 1\}^{l_1}$  be another random oracle in addition to  $H_0$  and  $H_3$ . (Note that the length  $l_1$  should be long enough for the random oracle to be secure [20].) Although the matrix  $\mathbf{A}$  can be determined by a trusted party, here we assume that it is computed interactively. Given public parameters  $par := \{R_q, k, \ell, \eta, B, s, M\}$   $i$ -th user proceeds as follows:

- (a)  $\mathbf{A}_i \xleftarrow{\$} R_q^{k \times \ell}$
- (b) Computes  $g_i := H_1(\mathbf{A}_i, i)$  and sends  $g_i$  to the  $j$ -th user.
- (c) Upon receiving  $g_j$  for  $j \neq i$  sends  $\mathbf{A}_i$  to  $j$ -th user.
- (d) Upon receiving  $\mathbf{A}_j$  for  $j \neq i$ :
  - i. If  $g_j \neq H_1(\mathbf{A}_j, j)$  for some  $j$  then send out ABORT.
  - ii. Otherwise set  $\mathbf{A} := \sum_{j=1}^n \mathbf{A}_j$  and  $\bar{\mathbf{A}} = [\mathbf{A} | \mathbf{I}] \in R_q^{k \times (\ell+k)}$ , where  $\mathbf{I}$  is  $k \times k$  identity matrix.
- (e) Picks random secrets  $(\mathbf{s}_{i_1}, \mathbf{s}_{i_2}) \xleftarrow{\$} S_\eta^\ell \times S_\eta^k$ ,  $\mathbf{s}_i := \begin{bmatrix} s_{i_1} \\ s_{i_2} \end{bmatrix}$
- (f) Compute  $\mathbf{t}_i := \bar{\mathbf{A}} \cdot \mathbf{s}_i$
- (g)  $\mathbf{s}_i$  is the secret key and  $(\bar{\mathbf{A}}, \mathbf{t}_i)$  is the public key pair of  $i$ -th user.

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

- (a) Choose a polynomial  $f_i(x) = \alpha_{n-1}^{(i)} x^{n-1} + \dots + \alpha_1^{(i)} x + \alpha_0^{(i)}$  where  $\alpha_0^{(i)} = \mathbf{s}_i$  and  $\alpha_1^{(i)}, \dots, \alpha_{n-1}^{(i)}$  are all nonzero and chosen randomly from  $S_\eta^{\ell+k}$ .
- (b) Compute the set of commitments  $\text{CS}_i := \{\mathbf{C}_u^{(i)} = \bar{\mathbf{A}} \cdot \alpha_u^{(i)} : u = 0, \dots, n-1\}$ .
- (c) Sends  $(f_i(j), \text{CS}_i)$  to the  $j$ -th user in  $\mathcal{G}$ , for  $j = 1, \dots, n$ , along with a signature  $\varpi_i$  on  $\mathbf{t}_i$  using the secret key  $\mathbf{s}_i$ . Note that signature  $\varpi_i$  is used

for proof of possession of the secret key  $\mathbf{s}_i$  that is being shared in the group setup phase. (See the Remark 5.1.2)

(d) After receiving  $(f_j(i), \text{CS}_j)$  and the proof of possession signature  $\varpi_j$ :

- Compute the raw membership key  $\mathbf{rmk}_i = \sum_{j \in \mathcal{G}} f_j(i)$ . Notice that  $\|\mathbf{rmk}_i\|_\infty$  would be larger than  $\eta$ . Therefore it has to be normalized. But before that the following consistency check is done.
- Compute  $\text{CS} := \{\mathbf{C}_u = \sum_{j \in \mathcal{G}} \mathbf{C}_u^{(j)} : u = 0, \dots, n-1\}$ , and check:
  - $\bar{\mathbf{A}} \cdot \mathbf{rmk}_i \stackrel{?}{=} \sum_{u=0}^{n-1} \mathbf{C}_u i^u$ ,
  - $\mathbf{C}_0 \stackrel{?}{=} \sum_{i \in \mathcal{G}} \mathbf{t}_i$ ,
  - Verify the proof of possession signature  $\varpi_j$  using  $\mathbf{t}_j$ .

If one of the above checks fail, then ABORT.

- Otherwise, in order to normalize the raw membership key, i.e.,

$$\mathbf{rmk}_i = \begin{pmatrix} b_{N-1}^{(1)}x^{N-1} + \dots + b_1^{(1)}x + b_0^{(1)} \\ b_{N-1}^{(2)}x^{N-1} + \dots + b_1^{(2)}x + b_0^{(2)} \\ \vdots \\ b_{N-1}^{(\ell+k)}x^{N-1} + \dots + b_1^{(\ell+k)}x + b_0^{(\ell+k)} \end{pmatrix},$$

construct a vector  $\Delta_i \in R_q^{\ell+k}$ , that is

$$\Delta_i = \begin{pmatrix} \delta^{(1)} \\ \delta^{(2)} \\ \vdots \\ \delta^{(\ell+k)} \end{pmatrix} = \begin{pmatrix} \delta_{N-1}^{(1)}x^{N-1} + \dots + \delta_1^{(1)}x + \delta_0^{(1)} \\ \delta_{N-1}^{(2)}x^{N-1} + \dots + \delta_1^{(2)}x + \delta_0^{(2)} \\ \vdots \\ \delta_{N-1}^{(\ell+k)}x^{N-1} + \dots + \delta_1^{(\ell+k)}x + \delta_0^{(\ell+k)} \end{pmatrix},$$

where  $\delta_u^{(v)}$  is sampled uniformly random from the interval  $[q - b_u^{(v)} - \eta, q - b_u^{(v)} + \eta]$ , for  $u = 0, \dots, N-1$  and  $v = 1, \dots, \ell+k$ .

- Then compute the membership key  $\mathbf{mk}_i = \mathbf{rmk}_i + \Delta_i$  and make the set of membership public keys, i.e.,  $\text{MPK} = \{\mathbf{T}_i = \bar{\mathbf{A}} \cdot \mathbf{mk}_i\}_{i \in \mathcal{G}}$ , public.

Notice that if the normalization is not performed then the coefficients of the membership keys will be in the interval  $[-\frac{q-1}{2}, \frac{q-1}{2}]$ , but after the normalization process they will be small enough, i.e., in the interval  $[-\eta, \eta]$  as desired for a signing key.

3. **Signature Generation:** Let  $H_0$  and  $H_3$  be the random oracles as in Section 2.3.3. Given a message  $\mu \in \{0, 1\}^*$ ,  $i$ -th user signs as follows:

- (a) Computes the commitment key  $ck \leftarrow H_3(\mu, L)$ , where  $L = \{pk_1, \dots, pk_n\}$ .
- (b) Picks a random  $\mathbf{y}_i \xleftarrow{\$} D_s^{\ell+k}$  and computes  $\mathbf{w}_i = \bar{\mathbf{A}}\mathbf{y}_i$
- (c) Computes  $com_i \leftarrow \text{Commit}_{ck}(\mathbf{w}_i, r_i)$ , where  $r_i$  is a discrete Gaussian vector sampled from the Gaussian distribution  $D(S_r)$  defined in the trapdoor commitment scheme in [20].
- (d) Sends  $com_i$  to the  $j$ -th user in  $\mathcal{S}$ , for  $j \neq i$ .
- (e) Upon receiving  $com_j$  from other users in  $\mathcal{S}$ , computes  $com := \sum_{j \in \mathcal{S}} com_j$
- (f) Computes  $c_i \leftarrow H_0(\mathbf{t}_i, com, \mu, L)$
- (g) Computes his individual signature  $\mathbf{z}_i = c_i \mathbf{m}\mathbf{k}_i + \mathbf{y}_i$
- (h) Runs the rejection sampling on input  $(c_i \mathbf{m}\mathbf{k}_i, \mathbf{z}_i)$ , i.e., with probability

$$\min(1, D_s^{\ell+k}(\mathbf{z}_i) / (M \cdot D_{c_i \mathbf{m}\mathbf{k}_i, s}^{\ell+k}(\mathbf{z}_i)))$$

sends out  $(\mathbf{z}_i, r_i)$ , otherwise sends out RESTART and goes to (b).

- (i) If he receives RESTART from some user, then goes to (b). Otherwise, upon receiving  $(\mathbf{z}_j, r_j)$  for  $j \in \mathcal{S}$  computes the aggregated signature as follows:

- i. Computes  $c_j \leftarrow H_0(\mathbf{t}_j, com, \mu, L)$  for all  $j \in \mathcal{S}$ , and computes

$$\mathbf{w}_j := \bar{\mathbf{A}}\mathbf{z}_j - c_j \mathbf{T}_j$$

- ii. Checks  $\|\mathbf{z}_j\|_2 \leq B$  and  $\text{Open}_{ck}(com_j, r_j, \mathbf{w}_j) = 1$ .

- iii. If the check fails for some  $j \in \mathcal{S}$ , he sends out ABORT.

- iv. Otherwise, he computes  $\mathbf{z} = \sum_{j \in \mathcal{S}} \mathbf{z}_j$  and  $r := \sum_{j \in \mathcal{S}} r_j$

- (j) If the protocol does not ABORT, then the accountable subgroup multi-signature is  $(com, \mathbf{z}, r)$ .

4. **Verification:** Given  $\{(com, \mathbf{z}, r), \mu, \mathcal{S}, L = \{pk_1, \dots, pk_n\}, \text{MPK} = \{\mathbf{T}_1, \dots, \mathbf{T}_n\}\}$  one can verify the  $\text{vMS}_2$  signature as follows:

- (a) Compute  $c_j := H_0(\mathbf{t}_j, com, \mu, L)$  for  $j \in \mathcal{S}$ , and reconstruct  $\mathbf{w} := \bar{\mathbf{A}}\mathbf{z} - \sum_{j \in \mathcal{S}} c_j \mathbf{T}_j$ .

- (b) Accept if  $\|\mathbf{z}\|_2 \leq B_\tau$  and  $\text{Open}_{ck}(com, r, \mathbf{w}) = 1$ , where  $B_\tau = \sqrt{\tau} \cdot B$  (see Lemma 2.3.1).

**Remark 5.1.1.** *Since the underlying multi-signature scheme [20] has interactive signing phase, in our construction we assume that the subgroup  $\mathcal{S}$  is known to all signers before the protocol starts. In other words, each signer in  $\mathcal{S}$  knows his co-signers.*

**Remark 5.1.2.** *The checks in the group setup phase is performed to verify whether the users shared their secret keys honestly. The first one checks whether the shares are consistent with the shared secrets. The second and the third ones ensures that the users shared their secret keys honestly. Notice that if we do not enforce the users to send a proof of possession signature on their public keys, a malicious user may share an arbitrary value  $\gamma_j$  with  $\|\gamma_j\|_\infty > \eta$  such that  $\mathbf{t}_j = \bar{\mathbf{A}} \cdot \gamma_j$ , which satisfies the second check without knowing the  $\mathbf{s}_j$ .*

**Remark 5.1.3.** *Notice that each  $\|\alpha_i^{(j)}\|_\infty \leq \eta$  for  $i = 0, \dots, n-1$  and  $j = 1, \dots, n$ . Therefore neither  $\|f_j(i)\|_\infty$  nor  $\|\mathbf{rmk}_i\|_\infty = \|\sum_{j \in \mathcal{G}} f_j(i)\|_\infty$  is under control (i.e., their infinity norms may be larger than  $\eta$ ). In order to have a membership key with  $\ell^\infty$  norm at most  $\eta$ , we normalize the  $\mathbf{rmk}_i$  with a user-specific normalizer vector, i.e.,  $\Delta_i$  which has entries whose coefficients are chosen from a coefficient-specific interval  $[q - b_u^{(v)} - \eta, q - b_u^{(v)} + \eta]$ , for  $u = 0, \dots, N-1$  and  $v = 1, \dots, \ell + k$ . Then the membership key is  $\mathbf{mk}_i = \mathbf{rmk}_i + \Delta_i$  and  $\|\mathbf{mk}_i\|_\infty \leq \eta$  as desired.*

**Remark 5.1.4.** *Note that one may publish the set of public normalizer vectors, i.e.,  $NS = \{N_i = \bar{\mathbf{A}}\Delta_i\}$  and the commitment set  $CS := \{\mathbf{C}_u = \sum_{j \in \mathcal{G}} \mathbf{C}_u^{(j)} : u = 0, \dots, n-1\}$  to make the membership public keys publicly verifiable. In this case, the verifier can also compute  $\mathbf{w} := \bar{\mathbf{A}}\mathbf{z} - \sum_{j \in \mathcal{S}} c_j \cdot \left( \sum_{u=0}^{n-1} \mathbf{C}_u j^u + N_j \right)$ . However, in this case the we incur additional operations in the verification equation. Moreover this modification requires  $2nk$  ring elements, i.e., for  $CS$  and  $NS$ , to be public.*

**Remark 5.1.5.** *Assume that  $B$  is the upper bound for the size of the individual signatures according to Lemma 2.3.2-2.3.3. From Lemma 2.3.1, the upper bound of our  $\mathbf{vMS}_2$  signature is  $B_\tau \leq \sqrt{\tau} \cdot B$ , where  $\tau = |\mathcal{S}|$ .*

**Remark 5.1.6.** *In our proposed  $\mathbf{vMS}_2$  scheme, we have a 2-round interaction in signing phase. In addition to that we have also a one-time 1-round interaction in*

the group setup phase. In the key generation phase we assume that the matrix  $A$  is computed interactively. Therefore we also assume 2 more rounds of interactions for once. Note that, in case of a trusted third party, our key generation phase becomes non-interactive.

**Remark 5.1.7.** Before ending technical remarks, we would like to point out that our group setup method which we use in  $vMS_2$  could be applied to any future multi-signature scheme based on Fiat Shamir with Aborts paradigm.

## 5.2 Security Analysis

We simply change the signing key of  $MS_2$  scheme [20] by adding a group setup phase which includes a secure joint verifiable secret sharing scheme. Namely a  $vMS_2$  signature is nothing but a  $MS_2$  signature with  $\tau$  signers, which is signed by the membership keys  $(mk_i)$  instead of secret keys  $(s_i)$ . Therefore security of our  $vMS_2$  scheme simply follows from the security of the  $MS_2$  scheme.

**Lemma 5.2.1.** *The group setup phase is secure.*

*Proof.* The security of the group setup phase follows from the security of VSS scheme whose security depends on Module LWE and Module SIS problems.  $\square$

**Lemma 5.2.2.** *If the  $MS_2$  scheme is secure, then the  $vMS_2$  scheme is secure.*

*Proof.* Notice that key generation phase is the same as  $MS_2$  scheme. In the group setup phase, each user shares his secret key via a VSS protocol. The output of the group setup phase is the membership keys which are of the same size and from the same distribution as of secret keys. Signature generation, signature aggregation and verification phases are the same as in  $MS_2$ .  $\square$

Next, we modify the security theorem of the  $MS_2$  scheme according to  $vMS_2$  scheme, and state it below.

**Theorem 5.2.1** (Theorem 3 [20]). *Suppose the trapdoor commitment scheme TCOM is secure, additively homomorphic and has uniform keys. For any probabilistic*

polynomial-time adversary  $\mathcal{A}$  that initiates  $\mathcal{Q}_s$  signature generation protocols by querying  $\mathcal{O}_\tau^{\text{vMS}_2}$ , and makes  $\mathcal{Q}_h$  queries to the random oracle  $H_0, H_3$ , the protocol  $\text{vMS}_2$  is MS-UF-CMA secure under  $\text{Module-SIS}_{q,k,\ell+1,\beta}$  and  $\text{Module-LWE}_{q,k,\ell,\eta}$  assumptions, where  $\beta = 2\sqrt{B_\tau^2} + \kappa$ . Concretely, using other parameters specified in Table 5.1, the advantage of  $\mathcal{A}$  is bounded as follows.

$$\begin{aligned} \text{Adv}_{\text{vMS}_2}^{\text{MS-UF-CMA}}(\mathcal{A}) &\leq e \cdot (\mathcal{Q}_h + \mathcal{Q}_s + 1) \cdot \left( (\mathcal{Q}_h + \mathcal{Q}_s)\epsilon_{td} + \mathcal{Q}_s \cdot \frac{2e^{-t^2/2}}{M} \right. \\ &\quad \left. + \text{Adv}_{\text{Module-LWE}_{q,k,\ell,\eta}} + \frac{(\mathcal{Q}_h + \mathcal{Q}_s + 1)}{|C|} \right) \\ &\quad + \sqrt{(\mathcal{Q}_h + \mathcal{Q}_s + 1) \cdot (\epsilon_{bind} + \text{Adv}_{\text{Module-SIS}_{q,k,\ell+1,\beta}})} \end{aligned}$$

Table 5.1: Parameters for  $\text{MS}_2$  [20] and  $\text{vMS}_2$  schemes

Parameter	Description
$n$	Number of users in group $\mathcal{G}$
$\tau$	Number of signers in subgroup $\mathcal{S} \subseteq \mathcal{G}$
$N$	the degree of $f(X)$ which is a power of 2
$f(X) = X^N + 1$	The $2N$ -th cyclotomic polynomial
$q$	Prime modulus
$R = \mathbb{Z}[x]/(f(X))$	Cyclotomic ring
$R_q = \mathbb{Z}_q[x]/(f(X))$	Ring
$k$	The height of the public matrix $A$
$\ell$	The width of the public matrix $A$
$\gamma$	Parameter defining the tail-bound of Lemma 2.3.2
$B = \gamma\sigma\sqrt{N(\ell+k)}$	The maximum $\ell^2$ norm of the individual signatures $z_j \in R^{\ell+k}$ for $j = 0, \dots, n$
$B_n = \sqrt{n}B$	The maximum $\ell^2$ norm of combined signature $z \in R^{\ell+k}$
$\kappa$	The maximum $\ell^1$ norm of challenge vector $c$
$C = \{c \in R : \ c\ _\infty = 1 \wedge \ c\ _1 = \kappa\}$	Challenge space where $ C  = \binom{N}{\kappa} 2^\kappa$
$S_\eta = \{x \in R : \ x\ _\infty \leq \eta\}$	Set of small secrets
$T = \kappa\eta\sqrt{N(\ell+k)}$	Chosen such that Lemma 4 of [20] holds
$\alpha$	Parameter defining $\sigma$ and $M$
$\sigma = s/\sqrt{2\pi} = \alpha T$	Standard deviation of the Gaussian distribution
$t = \omega(\sqrt{\log(mN)}) \wedge t = o(\log(mN))$	Parameter defining $M$ such that Lemma 2.3.3 holds
$M = e^{t/\alpha + 1/(2\alpha^2)}$	The expected number of restarts until a single party can proceed
$M_n = M^n$	The expected number of restarts until all $n$ parties proceed simultaneously
$l_1$	Output bit length of random oracles $H_1$
TCOM	Additively homomorphic trapdoor commitment scheme proposed in [20].
(Commit, Open)	"Committing" and "Opening" algorithms of TCOM

### 5.3 Computational Analysis

Since our underlying hard problems are Module-SIS and Module-LWE, our computations are done in the base ring  $R_q = \mathbb{Z}_q[x]/(x^N + 1)$ , i.e., polynomial addition and multiplication modulo  $x^N + 1$ . In Table 5.2 we give the number of computations

required in each phase of our proposed  $\text{vMS}_2$  scheme in comparison with the  $\text{MS}_2$ . In the last column of the comparison table we state the cost of accountability under the assumption of  $\tau = n$ , i.e., all of the users in the group  $\mathcal{G}$  participate in the  $\text{vMS}_2$  signature. In other words, if the same number of signers participate in  $\text{vMS}_2$  and  $\text{MS}_2$  schemes at the same time, the  $\text{vMS}_2$  scheme requires more operations than  $\text{MS}_2$  as a cost of accountability.

More precisely, in comparison with  $\text{MS}_2$  scheme,

- Matrix generation and key generation phases of our  $\text{vMS}_2$  scheme are identical to the  $\text{MS}_2$  scheme,
- $\text{vMS}_2$  scheme has an additional one time group setup phase.
- In  $\text{vMS}_2$  scheme, the broadcasted data size also grows with extra  $nk$  ring elements.
- Since each user has an additional membership key (with same size) in  $\text{vMS}_2$  scheme, the need of storage doubles.

Table 5.2: Comparison of the  $MS_2$  and  $vMS_2$  schemes

Phases	$MS_2$ Scheme [20]	$vMS_2$ Scheme	# extra operations in $vMS_2$ (for $\tau = n$ )
Matrix Generation	Uniform Sampling: $k\ell$ $R_q$ -Elt. Computation: $n$ Hashes ( $H_0$ ) $k\ell(n-1)$ Add.	Uniform Sampling: $k\ell$ $R_q$ -Elt. Computation: $n$ Hashes ( $H_0$ ) $k\ell(n-1)$ Add.	none
Key Generation	Uniform Sampling: $k + \ell$ $R_q$ -Elt. Computation: $k(k + \ell)$ Mult. $k(k + \ell - 1)$ Add.	Uniform Sampling: $k + \ell$ $R_q$ -Elt. Computation: $k(k + \ell)$ Mult. $k(k + \ell - 1)$ Add.	none
Group Setup	None	Multiplication: $(n+1)(k^2 + k\ell) + n^2$ Addition: $(n+1)(k^2 + k\ell - k) + n^2(2k + \ell) - 1$ Uniform Sampling ( $f(x)$ ): $(n-1)(k + \ell)$ $R_q$ -Elt. Uniform Sampling ( $\Delta_i$ ): $N(k + \ell)$ integers	Multiplication: $(n+1)(k^2 + k\ell) + n^2$ Addition: $(n+1)(k^2 + k\ell - k) + n^2(2k + \ell) - 1$ Uniform Sampling ( $f(x)$ ): $(n-1)(k + \ell)$ $R_q$ -Elt. Uniform Sampling ( $\Delta_i$ ): $N(k + \ell)$ integers
Signature Generation	1 Hash ( $H_3$ ) 1 Commit $(n-1)$ Open $n$ Hashes ( $H_0$ ) Sampling $(k + \ell)$ $R_q$ -Elt. Sampling $(\ell + 2w)$ $R_q$ -Elt <sup>(*)</sup> . $n(k^2 + k\ell + k) + \ell$ Mult. $n(k^2 + k\ell + k + \ell)$ Add.	1 Hash ( $H_3$ ) 1 Commit $(\tau-1)$ Open $\tau$ Hashes ( $H_0$ ) Sampling $(k + \ell)$ $R_q$ -Elt. Sampling $(\ell + 2w)$ $R_q$ -Elt <sup>(*)</sup> . $\tau(k^2 + k\ell + 2k) + \ell$ Mult. $\tau(k^2 + k\ell + k + \ell)$ Add.	none
Verification	$n$ Hashes $H_0$ $k(k + \ell + n)$ Mult. $k(k + \ell + n - 2)$ Add. 1 Open	$\tau$ Hashes $H_0$ $k(k + \ell + \tau)$ Mult. $k(k + \ell + \tau - 2)$ Add. 1 Open	none
Transmission ( $R_q$ Elements)	$k\ell + k + \ell$ 1 (integer) $\ell + 2w + 2$ for TCOM <sup>(*)</sup>	$k(n + \ell) + 2(k + \ell)$ 1 (integer) $\ell + 2w + 2$ for TCOM <sup>(*)</sup>	$k(n + 1) + \ell$
Broadcasting ( $R_q$ Elements)	$k(\ell + k + 1)$	$k(n + k + \ell + 1)$	$nk$
Storage ( $R_q$ Elements)	$(k + \ell)$	$2(k + \ell)$	$(k + \ell)$

[e] (\*) Notice that  $r \in R_q^{\ell+2w}$  and  $com \in R_q^2$ . (see TCOM definition in [20])

Add. - Addition

Mult. - Multiplication

Elt. - Element



## CHAPTER 6

### CONCLUSION

In this thesis, we have proposed a novel BLS-based ASM scheme (**vASM**) that is more efficient than the ones in [12] in terms of signature generation, signature aggregation, and verification. On the other hand, our **vASM** scheme requires a one-time group setup with more multiplications. We have proposed 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 [12]. Moreover, we have compared the aggregated versions of our proposed schemes with the aggregated version of Boneh et al.'s ASM scheme [12]. We see that aggregated versions of our schemes, i.e., **AvASM**, **AASMwSA**, and **AASMwCA**, are more efficient regarding aggregated signature size and verification time. According to the requirements of the system involving an ASM scheme, our schemes could be good alternatives, especially when faster verification is desired.

Furthermore, we have proposed four constructions of compartment-based and hierarchical threshold delegation of **vASM** signing authority for different organizational scenarios. We have showed that applying the **vASM** scheme recursively and combining the functionalities of threshold secret sharing schemes with the **vASM** scheme can solve an organizational problem of delegating the signing power of authorized users to single/multiple proxies in an accountable fashion. We have presented the comparison of our constructions with the existing proxy signatures in the literature in terms of their properties and functionalities. Our constructions provide us with all the functionalities of the proxy signatures that are defined in [11]. We have also com-

pared our constructions with each other according to the number of computations required and the security assumptions. Shamir’s SSS-based and trusted user-based delegations require nearly the same number of computations, whereas HTSS-based delegation requires more operations than the others because of the determinant operations performed in signature aggregation and verification phases. On the other hand, all three constructions are insecure in case of an adversary that can corrupt  $t + 1$  signers, where  $t$  is the threshold of the secret sharing schemes used in the constructions. However, our proposed recursive vASM construction provides a much more efficient solution for delegating the vASM signing authority. Moreover, it is more secure than the other constructions based on the threshold secret sharing schemes. Existence of at least one honest participant during the one-time compartment setup is enough to ensure the security of recursive vASM construction. Since the authorized signer (original signer/designator/delegator) also participates in the compartment setup, even though all the unauthorized users are corrupted, they cannot reconstruct the membership key of the authorized user. In contrast, our constructions using threshold secret sharing schemes require at least one trusted unauthorized user. We have also compared the vASM scheme with the existing pairing-based proxy signature variants, i.e., proxy signatures, proxy multi-signatures, multi-proxy signatures, multi-proxy multi-signatures, in terms of efficiency. vASM scheme can be used as a flexible and practical proxy signature scheme because of its simple structure. On the other hand it provides a general delegation, i.e., it doesn’t have a warrant that gives detailed information about the delegation, such as validity time, delegation context, etc.

Finally, we have proposed a novel lattice-based ASM scheme, i.e., vMS<sub>2</sub>, which is based on the two-round MS<sub>2</sub> multi-signature scheme proposed in [20]. By adding the group setup method of our pairing-based vASM scheme to MS<sub>2</sub> scheme, we ensure all the users give authorization to others in group  $\mathcal{G}$  to sign on behalf of the group  $\mathcal{G}$ . We achieve the accountability with the cost of one-time one-round interactive group setup and increased broadcast data size.

Regarding future works, we believe that the following directions may be achieved:

- Having a pairing-based ASM scheme with a single verification key -representing

the group  $\mathcal{G}$ - instead of individual membership public keys or subgroup public keys,

- Examining the feasibility of using some polynomial commitment schemes with batch verification property to commit to the secret polynomials in the group setup phases,
- Adding accountability by the methods given in this thesis to known multi-signature schemes depending on other post-quantum problems.



## REFERENCES

- [1] B. Alomair, K. Sampigethaya, and R. Poovendran, Efficient generic forward-secure signatures and proxy signatures, in *Public Key Infrastructure: 5th European PKI Workshop: Theory and Practice, EuroPKI 2008 Trondheim, Norway, June 16-17, 2008 Proceedings 5*, pp. 166–181, Springer, 2008.
- [2] A. R. Ağırtaş, vasm, <https://github.com/ahmetramazan/vASM>, 2022, accessed: 2024-01-26.
- [3] A. R. Ağırtaş and O. Yayla, Pairing-based accountable subgroup multi-signatures with verifiable group setup, Cryptology ePrint Archive, Report 2022/018, 2022, <https://ia.cr/2022/018>.
- [4] A. R. Ağırtaş and O. Yayla, Compartment-based and hierarchical threshold delegated verifiable accountable subgroup multi-signatures, Cryptology ePrint Archive, Paper 2023/548, 2023, <https://eprint.iacr.org/2023/548>.
- [5] A. R. Ağırtaş and O. Yayla, Delegated verifiable accountable subgroup multi-signatures, ALCOCRYPT 2023: Algebraic and Combinatorial Methods for Coding and Cryptography, CIRM, Luminy, France, 2023.
- [6] A. R. Ağırtaş and O. Yayla, A lattice-based accountable subgroup multi-signature scheme with verifiable group setup, Cryptology ePrint Archive, Report 2024/014, 2024, <https://eprint.iacr.org/2024/014>.
- [7] A. Bagherzandi, J.-H. Cheon, and S. Jarecki, Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma, in *Proceedings of the 15th ACM Conference on Computer and Communications Security, CCS '08*, p. 449–458, Association for Computing Machinery, New York, NY, USA, 2008.
- [8] C. Baum, I. Damgård, V. Lyubashevsky, S. Oechsner, and C. Peikert, More efficient commitments from structured lattice assumptions, in *11th Conference on Security and Cryptography for Networks, SCN 2018 ; Conference date: 05-09-2018 Through 07-09-2018*, Lecture Notes in Computer Science, pp. 368–385, Springer International Publishing, 2018.
- [9] M. Bellare and G. Neven, Multi-signatures in the plain public-key model and a general forking lemma, in *Proceedings of the 13th ACM Conference on Com-*

puter and Communications Security, CCS '06, pp. 390–399, Association for Computing Machinery, 2006.

- [10] A. Boldyreva, Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme, in *Public Key Cryptography—PKC 2003: 6th International Workshop on Practice and Theory in Public Key Cryptography Miami, FL, USA, January 6–8, 2003 Proceedings 6*, pp. 31–46, Springer, 2002.
- [11] A. Boldyreva, A. Palacio, and B. Warinschi, Secure proxy signature schemes for delegation of signing rights, *Journal of Cryptology*, 25, pp. 57–115, 2012.
- [12] D. Boneh, M. Drijvers, and G. Neven, Compact multi-signatures for smaller blockchains, in T. Peyrin and S. Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018*, pp. 435–464, Springer International Publishing, 2018.
- [13] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, Aggregate and verifiably encrypted signatures from bilinear maps, in *Advances in Cryptology — EUROCRYPT 2003*, pp. 416–432, Springer Berlin Heidelberg, 2003.
- [14] D. Boneh, B. Lynn, and H. Shacham, Short signatures from the weil pairing, in *Advances in Cryptology — ASIACRYPT 2001*, pp. 514–532, Springer Berlin Heidelberg, 2001.
- [15] E. Bresson, J. Stern, and M. Szydło, Threshold ring signatures and applications to ad-hoc groups, in *Advances in Cryptology — CRYPTO 2002*, pp. 465–480, Springer Berlin Heidelberg, 2002.
- [16] V. Buterin, Ethereum: A next-generation smart contract and decentralized application platform, <https://github.com/ethereum/wiki/wiki/White-Paper>, 2013.
- [17] J. Camenisch and M. Stadler, Efficient group signature schemes for large groups, in B. S. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, pp. 410–424, Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.
- [18] F. Cao and Z. Cao, A secure identity-based multi-proxy signature scheme, in *Computers & Electrical Engineering*, volume 35, pp. 86–95, 2009.
- [19] D. Chaum and E. van Heyst, Group signatures, in *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, volume 547 of *Lecture Notes in Computer Science*, pp. 257–265, Springer, 1991.
- [20] I. Damgård, C. Orlandi, A. Takahashi, and M. Tibouchi, Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices, *Journal of Cryptology*, 35, pp. 99–130, 2022.

- [21] Y. Desmedt and Y. Frankel, Threshold cryptosystems, in *Advances in Cryptology – CRYPTO’89 Proceedings*, pp. 307–315, Springer New York, 1990.
- [22] H. Du and Q. Wen, Certificateless proxy multi-signature, in *Information Sciences*, volume 276, pp. 21–30, 2014.
- [23] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, Crystals-dilithium algorithm specifications and supporting documentation, 2017.
- [24] Z. Eslami, N. Pakniat, and M. Nojournian, Ideal social secret sharing using birkhoff interpolation method, *Security and Communication Networks*, 9, 10 2016.
- [25] P. Feldman, A practical scheme for non-interactive verifiable secret sharing, in *28th Annual Symposium on Foundations of Computer Science (SFCS 1987)*, pp. 427–438, IEEE, 1987.
- [26] R. A. Fisher and F. Yates, *Statistical tables for biological, agricultural and medical research*, Oliver and Boyd Ltd, London, 1943.
- [27] E. Fujisaki and K. Suzuki, Traceable ring signature, in *Public Key Cryptography – PKC 2007*, pp. 181–200, Springer Berlin Heidelberg, 2007.
- [28] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, Robust threshold dss signatures, in *Advances in Cryptology — EUROCRYPT ’96*, pp. 354–371, Springer Berlin Heidelberg, 1996.
- [29] L. Harn, Group-oriented (t, n) threshold digital signature scheme and digital multisignature, *IEE Proceedings - Computers and Digital Techniques*, 141, pp. 307–313(6), 1994, ISSN 1350-2387.
- [30] J. Herranz and G. Sáez, Revisiting fully distributed proxy signature schemes, in *Progress in Cryptology - INDOCRYPT 2004*, pp. 356–370, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.
- [31] K. ITAKURA, K; NAKAMURA, A public-key cryptosystem suitable for digital multisignatures, NEC Research & Development, 1983.
- [32] E. Käsper, V. Nikov, and S. Nikova, Strongly multiplicative hierarchical threshold secret sharing, in *Information Theoretic Security*, pp. 148–168, Springer Berlin Heidelberg, 2009.
- [33] S. Kim, S. Park, and D. Won, Proxy signatures, revisited, in *Information and Communications Security: First International Conference, ICIS’97 Beijing, China, November 11–14, 1997 Proceedings 1*, pp. 223–232, Springer, 1997.

- [34] B. Lee, H. Kim, and K. Kim, Secure mobile agent using strong non-designated proxy signature, in *Information Security and Privacy: 6th Australasian Conference, ACISP 2001 Sydney, Australia, July 11–13, 2001 Proceedings 6*, pp. 474–486, Springer, 2001.
- [35] B. Lee, H. Kim, and K. Kim, Strong proxy signature and its applications, in *Proceedings of SCIS*, volume Jan. 2001, pp. 474–486, 2001.
- [36] J.-S. Lee, J. H. Chang, and D. H. Lee, Forgery attacks on kang et al.’s identity-based strong designated verifier signature scheme and its improvement with security proof, *Comput. Electr. Eng.*, 36(5), p. 948–954, 2010.
- [37] J. Li, T. H. Yuen, X. Chen, and Y. Wang, Proxy ring signature: Formal definitions, efficient construction and new variant, in *2006 International Conference on Computational Intelligence and Security*, volume 2, pp. 1259–1264, 2006.
- [38] X. Li and K. Chen, Id-based multi-proxy signature, proxy multi-signature and multi-proxy multi-signature schemes from bilinear pairings, *Appl. Math. Comput.*, 169(1), p. 437–450, 2005.
- [39] W. Lin and J.-K. Jan, Security personal learning tools using a proxy blind signature scheme, In *Proceedings of Journal of Chinese Language and Computing*, p. 273–277, 2000.
- [40] J. K. Liu and D. S. Wong, Linkable ring signatures: Security models and new schemes, in *Computational Science and Its Applications – ICCSA 2005*, pp. 614–623, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.
- [41] Z. Liu, Y. Hu, X. Zhang, and H. Ma, Provably secure multi-proxy signature scheme with revocation in the standard model, *Computer Communications*, 34(3), pp. 494–501, 2011.
- [42] V. Lyubashevsky, Lattice signatures without trapdoors, in *Advances in Cryptology – EUROCRYPT 2012*, pp. 738–755, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
- [43] T. Malkin, S. Obana, and M. Yung, The hierarchy of key evolving signatures and a characterization of proxy signatures, in *Advances in Cryptology - EUROCRYPT 2004*, pp. 306–322, Springer Berlin Heidelberg, 2004.
- [44] M. Mambo, K. Usuda, and E. Okamoto, Proxy signatures for delegating signing operation, in *Proceedings of the 3rd ACM Conference on Computer and Communications Security, CCS '96*, p. 48–57, Association for Computing Machinery, New York, NY, USA, 1996.
- [45] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille, Simple Schnorr multi-signatures with applications to Bitcoin, *Designs, Codes and Cryptography*, 87, pp. 2139–2164, 2019.

- [46] S. Micali, K. Ohta, and L. Reyzin, Accountable-subgroup multisignatures: Extended abstract, in *Proceedings of the 8th ACM Conference on Computer and Communications Security, CCS '01*, p. 245–254, Association for Computing Machinery, New York, NY, USA, 2001.
- [47] S. Nakamoto, Bitcoin: A Peer-to-peer Electronic Cash System., <https://bitcoin.org/bitcoin.pdf>, 2008.
- [48] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*, Princeton University Press, USA, 2016.
- [49] K. Ohta and T. Okamoto, A digital multisignature scheme based on the fiat-shamir scheme, in *Advances in Cryptology — ASIACRYPT '91*, pp. 139–148, Springer Berlin Heidelberg, Berlin, Heidelberg, 1993.
- [50] 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, pp. 21–31, 1999.
- [51] T. Okamoto, M. Tada, and E. Okamoto, Extended proxy signatures for smart cards, in *Information Security. ISW 1999. Lecture Notes in Computer Science, vol 1729.*, pp. 247–258, Springer Berlin Heidelberg, 1999.
- [52] D. Pointcheval and J. Stern, Provably secure blind signature schemes, in K. Kim and T. Matsumoto, editors, *Advances in Cryptology — ASIACRYPT '96*, pp. 252–265, Springer Berlin Heidelberg, Berlin, Heidelberg, 1996, ISBN 978-3-540-70707-3.
- [53] D. Pointcheval and J. Stern, Security proofs for signature schemes, in *Advances in Cryptology — EUROCRYPT '96*, pp. 387–398, Springer Berlin Heidelberg, Berlin, Heidelberg, 1996.
- [54] R. L. Rivest, A. Shamir, and Y. Tauman, How to leak a secret: Theory and applications of ring signatures, in *Theoretical Computer Science*, pp. 164–186, Springer, 2006.
- [55] R. A. Sahu and S. Padhye, Identity-based multi-proxy multi-signature scheme provably secure in random oracle model, *Transactions on Emerging Telecommunications Technologies*, 26(4), pp. 547–558, 2015.
- [56] C.-P. Schnorr, Efficient signature generation by smart cards, *Journal of Cryptology*, 4, pp. 161–174, 1991.
- [57] S. Seo, K. Choi, J. Hwang, and S. Kim, Efficient certificateless proxy signature scheme with provable security, *Information Sciences*, 188, pp. 322–337, 2012.

- [58] A. Shamir, How to share a secret, *Commun. ACM*, 22(11), pp. 612–613, 1979.
- [59] K.-A. Shim, Short designated verifier proxy signatures, *Computers & Electrical Engineering*, 37(2), pp. 180–186, 2011.
- [60] H. Shin-Jia and C. Chiu-Chin, New multi-proxy multi-signature schemes, *Applied Mathematics and Computation*, 147(1), pp. 57–67, 2004.
- [61] C. S. S.J. Hwang, A simple multi-proxy signature scheme for electronic commerce, *Proceedings of the 10th National Conference on Information Security*, Hualien Taiwan, ROC., p. 134–138, 2000.
- [62] H.-M. Sun, Design of time-stamped proxy signatures with traceable receivers, *IEE Proceedings-Computers and Digital Techniques*, 147(6), pp. 462–466, 2000.
- [63] T. Tassa, Hierarchical threshold secret sharing, in M. Naor, editor, *Theory of Cryptography*, *Lecture Notes in Computer Science*, vol 2951., pp. 473–490, Springer Berlin Heidelberg, 2004.
- [64] T. Tassa and N. Dyn, Multipartite secret sharing by bivariate interpolation, *Journal of Cryptology*, 22(2), pp. 227–258, 2009.
- [65] M. Tompa and H. Woll, How to share a secret with cheaters, *Journal of Cryptology*, 1(3), pp. 133–138, Oct 1989.
- [66] G. K. Verma and B. B. Singh, Short certificate-based proxy signature scheme from pairings, *Transactions on Emerging Telecommunications Technologies*, 28(12), p. e3214, 2017.
- [67] C.-K. Wu and V. Varadharajan, Modified Chinese Remainder Theorem and its application to proxy signatures, in *Proceedings of the 1999 ICPP Workshops on Collaboration and Mobile Computing (CMC'99). Group Communications (IWGC). Internet '99 (IWI'99). Industrial Applications on Network Computing (INDAP). Multime*, pp. 146–151, 1999.
- [68] L. Yi, G. Bai, and G. Xiao, Proxy multi-signature scheme: A new type of proxy signature scheme, *Electronics Letters*, 36, pp. 527 – 528, 04 2000.
- [69] F. Zhang, R. Safavi-Naini, and C.-Y. Lin, New proxy signature, proxy blind signature and proxy ring signature schemes from bilinear pairing, *Cryptology ePrint Archive*, Report 2003/104, 2003, <https://ia.cr/2003/104>.
- [70] K. Zhang, Threshold proxy signature schemes, in *International Workshop on Information Security*, *Lecture Notes in Computer Science*, vol 1396., pp. 282–290, Springer, 1997.
- [71] L. Zhang, F. Zhang, and Q. Wu, Delegation of signing rights using certificateless proxy signatures, *Inf. Sci.*, 184(1), pp. 298–309, 2012.

# CURRICULUM VITAE

## PERSONAL INFORMATION

**Surname, Name:** Ağırtaş, Ahmet Ramazan

**Nationality:** Turkish (TC)

## EDUCATION

Degree	Institution	Year of Graduation
B.S.	System Engineering, Turkish Military Academy	2003
High School	Kuleli Military High School	1999

## PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
2 years	Nethermind	Blockchain and Cryptography Researcher
18 years	Gendarmerie General Command	Officer

## PUBLICATIONS

### International Conference Publications

A. R. Ağırtaş and O. Yayla, *Delegated Verifiable Accountable Subgroup Multi-signatures*. Paper presented at the ALCOCRYPT 2023: Algebraic and Combinatorial Methods for Coding and Cryptography, February 20-24, 2023, CIRM, Luminy, France.