Politecnico di Torino 1859

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Telsy
A TIM ENTERPRISE BRAND

Doctoral Dissertation
Doctoral Program in Pure and Applied Mathematics (36th cycle)

On the Design and Security of Post-Quantum Aggregate Signatures

Edoardo Signorini

* * * * *

Supervisors
Prof. Danilo Bazzanella, Supervisor
Dr. Guglielmo Morgari, Co-supervisor

Politecnico di Torino
14 October 2024

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

. . . .

Edoardo Signorini
Turin, 14 October 2024

Abstract

Digital signatures are pivotal for ensuring the authenticity and integrity of digital communications, serving as the backbone of numerous cryptographic protocols and systems. However, the physical constraints of network communication pose challenges in the transmission of large number of signatures, including digital certificates in Public Key Infrastructures, routing protocols, and decentralized systems. The advent of quantum computing has heightened these challenges, given that post-quantum cryptographic schemes, like those proposed under NIST's standardization project, generally result in larger signature sizes.

Aggregate signatures represent a powerful cryptographic tool to reduce communication costs in protocols that require the verification of multiple signatures. In fact, this method can reduce the total amount of transmitted data by allowing multiple signers to combine their individual signatures on separate messages into a single aggregate signature. Since general forms of aggregations are difficult to achieve in practice, weaker variants can be considered. For instance, in a Sequential Aggregate Signature (SAS) scheme, signatures are combined in a specific sequence, allowing each subsequent signer to append their signature to an already aggregated one. Although sequential aggregation is more restrictive, it often suits contexts where order or hierarchy are inherent, thus continuing to be beneficial in practical applications. This thesis contributes primarily by exploring aggregate signatures under post-quantum assumptions, particularly focusing on signature schemes that are not based on structured lattices.

Our research delves into two main classes of digital signatures in the post-quantum setting. Firstly, we explore the generalization of SAS schemes to Hash-and-Sign signatures based on generic trapdoor functions. The simple structure of signatures within this paradigm appears to make them ideal candidates for sequential aggregation. However, early SAS proposals required the use of trapdoor permutation (e.g., RSA), while post-quantum trapdoor functions known so-far are not injective. Direct attempts at generalizing permutation-based schemes have been proposed, but they either lack formal security or require additional properties on the trapdoor function, which are typically not available for multivariate or code-based functions. We provide a comprehensive analysis of existing models, discuss their limitations, and prove how direct extensions of original SAS schemes are not possible without additional properties. Then, we propose a novel construction of history-free SAS within the probabilistic Hash-and-Sign with retry paradigm, generalizing existing techniques to generic trapdoor functions. We prove the security of our scheme in the random oracle model, and we instantiate our construction with multivariate and code-based schemes.

Secondly, we investigate the potential of group action-based signatures within the Fiat-Shamir paradigm, proposing new techniques for signature aggregation. Group actions are emerging as a promising tool in post-quantum cryptography, and have been used to build several signature schemes, including some submitted to the recent NIST call for additional post-quantum signatures. We propose two novel aggregation methods for group action-based signatures: a sequential aggregate signature and an interactive aggregation scheme. Although provably secure sequential aggregation can be achieved, we show that the resulting compression is too small for practical applications. Therefore, we investigate a trade-off between aggregation capabilities and the need for interaction between signers. We then obtain an interactive aggregation scheme (or multi-signature), whose security can be reduced directly to the assumptions underlying the group action.

To Carla Barghini

Acknowledgements

From the beginning, I perceived my PhD as a solitary journey, a constant juggling act between industry and academia. For that, I've had my fair share of moments wondering if I was in the right place at the right time. But looking back, I realize it's the people who've been there – sometimes pushing, sometimes supporting, always believing – that have made this journey not just possible, but worthwhile.

First and foremost, I wish to express my deepest thanks to Guglielmo Morgari. It all started with an unexpected (and perhaps annoying) call, and you took a chance on me. Your unwavering dedication and endless support have been invaluable throughout this process. Thank you for constantly pushing me to reach higher, for making me a better communicator, and somehow convincing me that I'm now friends with the lab's oscilloscope. Perhaps most importantly, I want to thank you for imparting me a small piece of the immeasurable faith you've always placed in me.

These acknowledgments naturally extend to Telsy, the company to which I owe this unique opportunity. I'm still amazed and grateful to be part of a team that's not afraid to bet on the value of research. A special thank you goes to Fabrizio Vacca and the colleagues who believed in this project and made it possible.

I am grateful to my academic supervisor, Danilo Bazzanella, and the entire CrypTo group at Politecnico. You have provided me with the precious opportunity to work closely in an environment that is both stimulating and welcoming. Although my primary workspace has never been the department, and my "office" there remains more of a theoretical concept than a physical location I can reliably find, each visit has contributed to making me feel part of a broader research community. Among the group, my sincere thanks to Antonio Di Scala and Carlo Sanna for guiding me through my first academic works with patience and wisdom.

I extend my gratitude to Simone Dutto and Nadir Murru for their meticulous review of my work and the numerous insightful suggestions they provided.

I am deeply indebted to my closest co-authors and colleagues. To Alessio Meneghetti, thank you for being the first who helped me find a thread in a tangle of confused ideas. To Giuseppe D'Alconzo and Andrea Flamini, I am grateful for the numerous discussions, thoughts, and passion you shared. Without you, moving forward would have been immensely challenging, and my visits to Politecnico would have been even fewer.

A heartfelt thank you goes to my colleagues at Telsy, Francesco Stocco, Veronica Cristiano, and Marco Rinaudo. We often travelled on parallel tracks, and I appreciate your patience with my frequent attempts to intersect our paths. Thank you for the countless "frenstoc" moments, the TèPrint sessions, and the time you dedicated to me despite your growing commitments. Your friendship inside and outside the company have made this journey far less lonely.

I will be forever grateful to my family – Mamma, Papà, Tata, Bea, Leo, and Ale. Thank you for your unconditional love and support, for never doubting my choices even though I was never good at explaining what I do. Thank you for your patience, especially when putting up with my reluctance to travel or attend any event that required dusting off the suit. Despite the distance and our differences, to each of you, I owe a part of who I am.

To marghe, my final acknowledgment, and the one closest to my heart. From the very beginning, you've shared in every high and low, and supported me through every setback. Thank you for everything we have experienced during this time, for the places we have called home, for the cards we have exchanged. Above all, thank you for discovering my hidden bit and keeping it close, seen and understood.

Contents

Introduction **1**

1 Preliminaries **7**
1.1 Notation 7
1.2 Provable Security 8
1.3 Cryptographic Primitives 12
1.3.1 One-Way Functions 12
1.3.2 Hash Functions 12
1.3.3 Digital Signatures 13
1.3.4 Aggregate Signatures 15
1.3.5 Sequential Aggregate Signatures 16
1.4 Post-Quantum Cryptography 17
1.4.1 NIST Standardization Processes 18

I Trapdoor-based Signature Aggregation **21**

2 Hash-and-Sign Paradigm **23**
2.1 Trapdoor Functions 24
2.1.1 Preimage Sampleable Functions 25
2.2 Hash-and-Sign Schemes 27
2.2.1 Security Analysis 29
2.3 Post-Quantum Hash-and-Sign Schemes 33
2.3.1 Lattice-based Cryptography 35
2.3.2 Code-based Cryptography 39
2.3.3 Multivariate-based Cryptography 42

3 History-Free Sequential Aggregation of Hash-and-Sign Signatures **49**
3.1 Sequential Aggregation from Trapdoor Permutation 50
3.2 LMRS Scheme for Generic Trapdoor Functions 52
3.2.1 The Scheme 53
3.2.2 Provable Security 53

3.3 Security of Existing Multivariate SAS Schemes 58
3.3.1 Description of the Forgery 58
3.3.2 Discussion 60
3.4 Sequential Aggregation of Hash-and-Sign Signatures 62
3.4.1 History-Free Sequential Aggregate Signature 62
3.4.2 The Scheme 64
3.4.3 Security Proof 65
3.5 Optimizing the Scheme for (Average) Preimage Sampleable Functions 76
3.5.1 PSF-based Signatures 76
3.5.2 APSF-based Signatures 78
3.6 Instantiation and Evaluation 80
3.6.1 Original Unbalanced Oil and Vinegar 81
3.6.2 Provable Unbalanced Oil and Vinegar 83
3.6.3 MAYO 84
3.6.4 Wave 86
3.6.5 Signature-Specific Optimizations 87

II Group Action-based Signature Aggregation 89

4 Signatures from Cryptographic Group Actions 91
4.1 Interactive Proofs 92
4.1.1 Zero-Knowledge Proofs 94
4.1.2 Sigma-Protocols 95
4.1.3 Fiat-Shamir Transform 99
4.2 Group Actions 100
4.2.1 Effective Group Actions 101
4.2.2 Computational Assumptions 102
4.2.3 Digital Signatures 103
4.3 Signature Optimizations 106
4.3.1 Compression of Random Elements 106
4.3.2 Seed Trees 107
4.3.3 Fixed-Weight Challenges 108
4.3.4 Multiple Public Keys 108
4.3.5 Further Optimizations 109
4.4 Post-Quantum Group Actions 110
4.4.1 Code Equivalence 110
4.4.2 Linear Code Equivalence 112
4.4.3 Matrix Code Equivalence 114
4.4.4 Alternating Trilinear Form Equivalence 115

5 Aggregate and Multi-Signatures from Group Actions 119
5.1 Sequential Half-Aggregation of Group Action-Based Signatures . . 120
5.1.1 Security Proof 121
5.1.2 Support for Standard Optimizations 129
5.2 Multi-Signature from Cryptographic Group Action 130
5.2.1 Multi-Signatures 130
5.2.2 Sigma Protocol Variant 132
5.2.3 The Multi-Signature Scheme 137
5.2.4 Security Proof 139
5.2.5 Signature Optimizations 145
5.3 Instantiation and Evaluation 150
5.3.1 LESS 153
5.3.2 MEDS 155
5.3.3 ALTEQ 155

Conclusions 159

Abbreviations 161

List of Tables 163

List of Figures 164

List of Algorithms 165

List of Experiments 166

Bibliography 167

Introduction

Digital signatures are standard cryptographic techniques for proving the authenticity and integrity of digital data. Authentication embodies one of the core problems of cryptography, and digital signature schemes are needed in numerous secure communication protocols. Being used in different contexts, the requirements behind digital signatures can vary widely. In addition to security, the size of a digital signature is a critical measure of its efficiency, affecting both storage and transmission requirements. In fact, while computational requirements can be partially mitigated by continued technological advancement, there are physical limitations for network communications that are difficult to overcome. A concrete example is the use of digital certificates in a Public Key Infrastructure (PKI). PKIs propagate trust from a Certificate Authority (CA) to an end-user (or a server) through a chain of certificates, involving intermediate authorities who add their signatures to lower level certificates. Each certificate includes a public key, tying the identity of an entity to the possession of the corresponding private key. When a cryptographic protocol involves a PKI, the entire certificate chain, including all intermediate signatures, must be communicated in order for the user's identity to be verified.

To date, the most widely used paradigms in the field of digital signatures are those based on Hash-and-Sign schemes, e.g., the RSA scheme [174], and Fiat-Shamir schemes, e.g., the Schnorr Signature [176]. However, these established solutions are being threatened by the rapid progress of quantum computers, for which algorithms are known to efficiently solve the underlying computational problems. The development of new quantum-resistant public-key cryptographic algorithms, i.e., cryptosystems that are not vulnerable to the use of quantum algorithms, is driving an important branch of cryptography that has matured in the last decade, namely, Post-Quantum Cryptography (PQC). The identification of new standard of post-quantum key-exchange and digital signature algorithms took on particular importance with the announcement of the standardization project launched by the American National Institute of Standards and Technology (NIST) in early 2016 [162]. After three rounds of analysis with a deep involvement of the cryptographic community, in 2022 NIST announced [5] an initial selection of algorithms that will proceed to the standardization stage. For digital signatures, the chosen algorithms are CRYSTALS-Dilithium [144], Falcon [172] and SPHINCS$^+$ [124]. NIST's competition has been mainly dominated by solutions from lattice theory, and all the schemes selected for standardization, except for SPHINCS$^+$, are based on structured lattices. With the aim of further differentiating the digital signature landscape, NIST deemed necessary to initiate a new standardization process focused on the selection of post-quantum signatures [161].

Aggregate Signatures An Aggregate Signature (AS) scheme allows *n* users to combine their individual signatures on separate messages to produce a single, directly verifiable aggregate signature. This approach aims to achieve shorter signature lengths compared to trivial concatenation of individual signatures. This property makes aggregate signatures particularly useful in scenarios where a large number of signatures need to be transmitted and the communication costs within the network are not negligible. Typical application scenarios include PKI certificate chains [40], secure routing protocols authentication [49], software authentication [118], and blockchain systems [39]. The notion of aggregate signatures was initially introduced in a seminal paper by Boneh et al. [40]. The authors proposed a method that allows a third party to aggregate signatures from distinct users using a public aggregation algorithm. Although this general aggregation approach is efficient and valuable in many applications, it is notoriously difficult to achieve in practice without the use of bilinear pairing [40, 19] or advanced constructions based on indistinguishability obfuscation [121] or non-interactive arguments (SNARKs) [7, 73, 192].

A restricted variant of aggregate signatures, known as Sequential Aggregated Signature (SAS), was introduced by Lysyanskaya et al. [143]. In SAS schemes, each user combines their signature with a so-far aggregated signature, acting in a specific, but not necessarily predetermined, sequence. Although the public aggregation functionality is lost, the sequential structure is still beneficial in many applications where an order among the signers can be established, such as PKI certificate chains. Numerous works have been pursued in this direction, proposing constructions based on trapdoor permutations (e.g., RSA) [143, 156, 49, 107] or the use of bilinear pairings [142, 19, 98, 137, 136].

Additionally, the hardness in constructing generic aggregation schemes and the possible limitation in using sequential aggregation in some scenarios led to the investigation of additional variants. Typically, the strategy adopted in constructing these alternatives involves some form of interaction between the parties during signature aggregation, as in the case of synchronous aggregate signature and multi-signatures. First introduced by Gentry and Ramzan [109] and later formalized in [3], the synchronous model allows for general aggregation by assuming that parties can produce an aggregate signature only within a predetermined time window. In turn, multi-signatures are a particularly relevant technique in decentralized applications and typically involve a form of interactive aggregation for signing a single message shared by multiple users. Although multi-signatures were introduced separately from aggregate signatures [126, 164, 163] and the usage scenarios are typically distinct, it is well known that a multi-signature can be easily transformed into an interactive aggregate signature by requiring participants to agree on a concatenation of messages to be signed [20]. Numerous multi-signatures have been proposed for Schnorr's signature [160, 20, 12, 145, 184, 148, 84, 159], with recent near-optimal schemes MuSig2 [158] and DWMS [8] requiring only one round of interaction and allowing key aggregation.

Aggregate Signatures from Post-Quantum Assumptions Increasing activity in the development of post-quantum signatures has led the community to explore AS schemes in this field. The strong interest and additional properties inherent in lattice-based schemes have caused most proposals to focus on this class of signatures. The first lattice-based proposal was presented by El Bansarkhani and Buchmann [91] and proposed sequential aggregation within the Hash-and-Sign paradigm, allowing aggregation of Falcon signatures. A further sequential aggregation scheme was proposed in [191], but was later found to be insecure [48]. Within the Fiat-Shamir paradigm, which includes Dilithium, [83] first proposed a partial aggregation scheme, achieving partial compression of signature components. The original scheme had a vulnerability later corrected in [47], resulting, however, in a negative compression of the signature. Recently, [48] have proposed a sequential aggregation scheme for Dilithium, achieving however only limited compression. Considering additional variants of aggregation, Fleischhacker, Simkin, and Zhang [100] proposed a lattice-based aggregated signature in the synchronous model and its subsequent optimization [99]. In the interactive model, there is a long line of work proposing lattice-based multi-signatures [93, 103, 41, 69, 57] culminating with MuSig-L [45], which achieves properties similar to those of MuSig2 for lattices. Finally, the idea of aggregating signatures using lattice-based SNARK was recently investigated in [7] and formalized by Aardal et al. [1].

Besides lattice-based solutions and generic approaches based on SNARKs, there are only a limited number of proposals tailored for other post-quantum assumptions. In the Hash-and-Sign paradigm, the SAS schemes proposed in [91, 191] extend previous trapdoor permutation-based approaches [156, 107], and can potentially be applied to other post-quantum signatures in the same paradigm. Unfortunately, their security relies on the collision-resistance property of lattice trapdoor Preimage Sampleable Functions [108]. These additional properties are not available for generic trapdoor functions employed, for instance, in multivariate-quadratic-based or code-based signature schemes. The first sequential aggregation scheme based on multivariate assumptions was proposed by El Bansarkhani, Mohamed, and Petzoldt [92], with a construction based on

SAS for trapdoor permutations [143] combined with the data-encoding techniques introduced in [156]. Later, a similar SAS scheme tailored for a specific multivariate-based scheme was proposed in [56]. Unfortunately, both [92, 56] lack formal security and there are instances of the underlying function for which they are insecure, as outlined below.

As such, there is a gap in the design of aggregation schemes for post-quantum assumptions that cannot be traced back to lattice theory. In this thesis, we aim to address this gap by analysing two classes of signature schemes.

Organization of the Thesis

The contribution of this thesis is divided into two parts. The first part is devoted to the generalization of SAS schemes to Hash-and-Sign signatures based on generic trapdoor functions. The second part investigates the aggregation of group action-based signatures in the Fiat-Shamir paradigm.

Hash-and-Sign Schemes The Hash-and-Sign paradigm is a standard and intuitive approach to building digital signature schemes from trapdoor functions and hash functions. Classically, this approach is known as Full Domain Hash (FDH) [21, 22] when employed with trapdoor permutations such as RSA. In the post-quantum scenario, numerous signature schemes can be traced back to this paradigm, e.g., Falcon for lattices and several multivariate schemes submitted to the recent NIST call [30, 115, 104, 190, 78, 31, 167]. To date, no trapdoor permutations based on post-quantum assumptions are known, leading us to consider weaker trapdoor functions. In Chapter 2 we provide an overview of the Hash-and-Sign paradigm, analysing how signature schemes should be adapted in the presence of generic trapdoor functions. In Chapter 3 we propose a sequential aggregation scheme that extends existing constructions on trapdoor permutations to generic trapdoor functions, making sequential aggregation possible for a wider range of post-quantum signatures. We also show that previous proposed constructions for multivariate functions are insecure and that a direct extension of [143] is not sufficient in the absence of additional properties.

Group Action-based Signatures Recently, cryptographic group actions have emerged as a promising tool for building post-quantum digital signatures in the Fiat-Shamir paradigm. Starting from isogeny-based constructions [51, 70] and subsequently from numerous non-abelian group actions [17, 61, 185, 89], several signature schemes have been proposed, including three submitted to the recent NIST call [13, 60, 37]. In Chapter 4 we introduce the relevant notions for constructing digital signatures from group actions, analysing the standard optimizations employed in these schemes so that they can be adapted into an aggregation scheme. In Chapter 5, we propose two aggregation techniques for signature schemes based on group actions. First, we study the design of a sequential aggregate signature using an approach similar to that proposed in [48] for lattice-based Fiat-Shamir signatures. The scheme we obtain is provably secure, but the resulting aggregation is very limited for practical applications. Next, we propose an interactive aggregation scheme (multi-signature) whose security can be directly reduced to the underlying signature, achieving good compression compared to the trivial concatenation of signatures.

Other Contributions

The following contributions were produced during the course of doctoral studies but are not included in this thesis. In [74, 75] we investigated the equivalence between the Ring Learning With Errors (RLWE) and the Polynomial Learning With Errors (PLWE), showing that the two problems are not equivalent over cyclotomic fields. In [154, 102] we studied the role of classical authentication within Quantum Key Distribution (QKD) protocols.

RLWE vs PLWE

- Antonio J. Di Scala, Carlo Sanna, and Edoardo Signorini. "On the condition number of the Vandermonde matrix of the nth cyclotomic polynomial." In: *Journal of Mathematical Cryptology* 15.1 (Nov. 2021), pp. 174–178. DOI: 10.1515/JMC-2020-0009

- Antonio J. Di Scala, Carlo Sanna, and Edoardo Signorini. "RLWE and PLWE over cyclotomic fields are not equivalent." In: *Applicable Algebra in Engineering, Communication and Computing* 35.3 (2022), pp. 351–358. DOI: 10.1007/S00200-022-00552-9

Authentication in QKD

- Guglielmo Morgari, Edoardo Signorini, and Francesco Stocco. "On the classical authentication in Quantum Key Distribution." In: *CrypTOrino 2021*. Vol. 4. Collectio Ciphrarum. Aracne, May 2021

- Giacomo Fregona et al. "Authentication Methods for Quantum Key Distribution: Challenges and Perspectives." In: *Toward a Quantum-Safe Communication Infrastructure*. Ed. by André Xuereb Rainer Steinwandt. Vol. 64. NATO Science for Peace and Security Series - D: Information and Communication Security. IOS Press, Apr. 2024, pp. 54–66. DOI: 10.3233/NICSP240007

Chapter 1

Preliminaries

In this chapter, we introduce the basic notation and definitions that will be adopted in the rest of this thesis. We provide an introductory overview on the notions related to provable security and the main cryptographic primitives which are most frequently recalled. Further notation and primitives will be introduced in the relevant chapters.

1.1 Notation

Sets For $a, b \in \mathbb{N}$, we denote by $[a, b]$ the set $\{ a, \dots , b \}$ and by $[b]$ the set $\{ 1, \dots , a \}$. For a finite set S, we write $|S|$ for the cardinality of S and $\mathrm{len}(S)$ for the bit size of elements in S. For $n \in \mathbb{N}$ we denote with $\{0,1\}^n$ the set of bitstrings of length n and with $\{0,1\}^*$ the set of bitstrings of arbitrary length.

Lists and Tables An ordered list of elements is denoted with an arrow $\vec{a} = (a_1, \dots , a_n)$. Given a list $\vec{a} = (a_1, \dots , a_n)$ and $n_1 < n_2 \leq n$ we denote with $\vec{a}[n_1{:}n_2] = (a_{n_1}, \dots , a_{n_2})$ the slicing of the list $\vec{a}$ between indices n_1 and n_2. We also write $\vec{a}[k{:}]$ to denote $\vec{a}[k : n]$.

A table T is a key-value store, where each key has a single associated value. For a key k, we denote with $v \leftarrow \mathsf{T}[k]$ the assignment of v to the value associated with k. Similarly, to assign a value v to the key k, we write $\mathsf{T}[k] \leftarrow v$. If no value is associated with k, we write $\mathsf{T}[k] = \bot$.

Algebraic Structures Given a prime power $q = p^k$, we denote with $\mathbb{F}_q$ a finite field with q elements. Given $n, m \in \mathbb{N}$, we denote by $\mathbb{F}_q^n$ the n-dimensional vector space over $\mathbb{F}_q$, and by $\mathbb{F}_q^{m \times n}$ the set of matrices over $\mathbb{F}_q$ with m rows and n columns. For a field $\mathbb{F}$ and a positive integer $n \in \mathbb{N}$, we denote by $\mathrm{GL}_n(\mathbb{F})$ the *general linear group* of degree n over $\mathbb{F}$. When $\mathbb{F} = \mathbb{F}_q$, we simply write $\mathrm{GL}_n(q)$.

Vectors and Matrices Vectors in a vector space are denoted with a bold letter $\boldsymbol{x} = (x_1, \dots, x_n)$. For a vector $\boldsymbol{x}$ we denote with x_i the i-th element of the vector. Matrices are denoted with a bold capital letter $\mathbf{A}$. $\mathbf{I}_{n\times n}$ is the identity matrix of size n. $\mathbf{0}_{m\times n}$ is the $m \times n$ zero matrix and 0_n is the zero vector in $\mathbb{F}_q^n$.

Distributions and Probabilities Let $\mathcal{D}$ be a probability distribution and X be a random variable. We write $X \sim \mathcal{D}$ to denote that X is distributed according to $\mathcal{D}$. For a set S, we denote by $\mathcal{U}(S)$ the uniform distribution over S. Furthermore, by $s \leftarrow_{\$} S$, we denote the sample of the element s from $\mathcal{U}(S)$. We denote by $\Pr[X = x]$ the probability that the random variable X takes value x.

Algorithms Where not otherwise specified, each algorithm is *probabilistic polynomial time* (PPT). A PPT algorithm is *probabilistic*, meaning that it has to access random material (or *random coins*) during computation, and is *efficient*, meaning that it runs in a time which is polynomial in the size of its input. In particular, given an input x, we write that a PPT algorithm runs in time $\mathsf{poly}(|x|)$, where $|x|$ is the bit length of x. A PPT algorithm is denoted with sans serif typeface A. For a deterministic algorithm A, we write $y \leftarrow \mathsf{A}(x)$ to denote the assignment of y to the output of A on input x. If A is probabilistic, we write $y \leftarrow_{\$} \mathsf{A}(x)$. In a pseudocode, each variable assignment is done by either deterministic assignment ($\leftarrow$) or probabilistic assignment ($\leftarrow_{\$}$), while the symbol = is reserved for equality testing. Furthermore, we use the symbol $\perp$ to denote a failure, e.g., $\perp \leftarrow \mathsf{A}(x)$.

Complexity Classes In a *decision problem*, given an input $x \in \{0,1\}^*$, we are required to verify whether x satisfies some property and return a yes/no answer. Analogously, given a set $\mathcal{L} \subseteq \{0,1\}^*$ of bitstrings, we need to determine whether $x \in \mathcal{L}$. The subset $\mathcal{L}$ is called a *language*, and every decision problem can be specified using a language (and the reverse is also true). The complexity class P is the class of decision problems that can be decided in polynomial time by a deterministic algorithm. The complexity class NP is instead the class of decision problems that can be decided in polynomial time by a *non-deterministic* algorithm. Formally, a language $\mathcal{L}$ is in NP if there exists a binary relation $R_{\mathcal{L}}$ of pairs (x, w) such that $x \in \mathcal{L}$ and w allows verifying that $x \in \mathcal{L}$ in polynomial time, i.e., $|w| = \mathsf{poly}(|x|)$.

Miscellanea Given two bitstrings $x, y \in \{0,1\}^*$, we denote by $x \parallel y$ the bitstring obtained by their concatenation. We write ε to denote an empty string.

1.2 Provable Security

Informally, the security of a cryptosystem can be described with two notions:

- *Informational security*, which describes the theoretical possibility of breaking a scheme.
- *Computational security*, which describes the strength of a scheme with respect to an adversary which has bounded computational power.

While informational security has a great importance in theoretical cryptography, it does not give a practical measure of the strength of a cryptosystem, unless it is proven to be secure against an adversary with unlimited computational power, which is rarely the case in real-world applications.

To determine the security of a cryptosystem, we first need to define the model of an adversary. Once we have defined the capabilities of an adversary, we usually proceed to reduce a mathematical problem, which is known to be computationally hard, to the cryptographic scheme. Thus, if an adversary can break the cryptosystem, or can produce a solution for the underlying computational problem, then they could use the same algorithm to solve the hard mathematical problem. The target security of a cryptosystem is expressed via a *security parameter*, which is denoted as λ. Once a security parameter is defined, we assume that every PPT algorithm runs polynomially in λ and that the parameters of a cryptographic scheme are chosen to obtain λ bits of security, i.e., the best algorithm that can break the security of the scheme requires at least 2^λ operations.

Adversaries An *adversary* $\mathcal{A}$ in a cryptographic protocol with a security parameter λ is modelled using a PPT algorithm that takes as input a bitstring of length λ, denoted by 1^λ.

Oracles An *oracle* O is a generic black-box functionality that takes some inputs and returns some output without specifying the internal computations. Given an adversary $\mathcal{A}$ and an arbitrary function F, we write $x \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{OF}}$ the assignment of x to the output of $\mathcal{A}$ with oracle access to F.

Negligible Function A function $f\colon \mathbb{N} \to \mathbb{R}$ is said to be *negligible*, and we write $f(x) = \mathsf{negl}(x)$, if for every $c \in \mathbb{N}$ there exists $\bar{n} \in \mathbb{N}$ such that

$$|f(n)| < \frac{1}{n^c}, \text{for all } n > \bar{n}.$$

Advantage When we define the security notion of a cryptographic scheme, we define an *attack game* (or *experiment*) played between a *challenger* and an adversary, and we measure the *advantage* of the adversary in winning the game. An attack game for a security notion Exp is often made of two sub-games, Exp_0 and Exp_1, where the attacker is required to distinguish whether is interacting with the former or the latter. The standard template for a distinguishing experiment is given in

Experiment 1.1: Exp_b

The challenger generates some public parameters for the adversary, which runs with oracle access to either Exp_0 or Exp_1.

```
$x \leftarrow\$ \mathsf{Gen}(1^\lambda)$
$b^\star \leftarrow\$ \mathcal{A}^{\mathsf{Exp}_b}(x)$
return $b^\star \in \{0,1\}$
```

```
$\mathsf{Exp}_0$:
  ...
  return $a$
```

```
$\mathsf{Exp}_1$:
  ...
  return $a$
```

Hybrid Argument 1.1: Generic Template for Successive Games

```
$\mathsf{Game_i}$:
  First instruction
  Second instruction
  Third instruction
  return State
```

```
$\mathsf{Game_{i+1}}$:
  First instruction
  Second instruction
  if Condition then
      New instruction
  Third instruction
  return State
```

Experiment 1.1. The advantage of the adversary $\mathcal{A}$ in playing the experiment Exp is defined as

$$\mathsf{Adv}^{\mathsf{Exp}}(\mathcal{A}) = |\Pr\left[\mathsf{Exp}_0(\mathcal{A}) = 1\right] - \Pr\left[\mathsf{Exp}_1(\mathcal{A}) = 1\right]|,$$

where $\Pr\left[\mathsf{Exp}_b(\mathcal{A}) = 1\right]$ is the probability that Exp_b returns 1 when played by $\mathcal{A}$. We say that a cryptographic scheme is secure with respect to the notion Exp if, for all PPT adversaries $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{Exp}}(\mathcal{A})$ is negligible.

Hybrid Arguments Several approaches exist to prove the security of a cryptographic scheme with respect to a particular security notion. In this thesis, we adopt the so-called *game-based* technique, where the security proof is organized by defining a sequence of indistinguishable experiments (or *hybrid arguments*). In more detail, the security proof consists of constructing a reduction from the original experiment to a notion that we assume to be secure. In the reduction, a sequence of games $\mathsf{Game_0}, \mathsf{Game_1}, \ldots, \mathsf{Game_n}$ is presented, where $\mathsf{Game_0}$ is the initial notion Exp with respect to an adversary $\mathcal{A}$, and subsequent games progressively modify it until it can be simulated by an adversary $\mathcal{B}$ against the target notion. Let $\Pr[\mathsf{Game_i}(\mathcal{A}) = 1]$ denote the probability that $\mathsf{Game_i}(\mathcal{A})$ returns 1 when played by $\mathcal{A}$. Usually, the reduction strategy is to define $\mathsf{Game_{i+1}}$ in a way that $\Pr[\mathsf{Game_i}(\mathcal{A}) = 1]$ is indistinguishable from $\Pr[\mathsf{Game_{i+1}}(\mathcal{A}) = 1]$, i.e.,

$$|\Pr[\mathsf{Game_i}(\mathcal{A}) = 1] - \Pr[\mathsf{Game_i}(\mathcal{A}) = 1]|$$

is negligible. When describing complex hybrid arguments, we will typically include descriptions of intermediate games in pseudocode form, highlighting differences between successive games with the notation variant, as shown in Argument 1.1. We refer to [181] for an excellent survey on game-based security proofs.

Random Oracle Model The Random Oracle Model (ROM), formally introduced by Bellare and Rogaway [21], is a widely used paradigm to obtain security proofs for many cryptographic protocols by assuming the existence of "random oracles". Intuitively, a random oracle $\mathsf{O}\colon \{0,1\}^* \to \{0,1\}^k$ behaves like an ideal, black-box, deterministic random function that returns a uniformly random bitstring of length k on each distinct input. In the ROM, a random oracle is instantiated with a concrete hash function H with the assumption that $\mathsf{H} \leftarrow_{\$} \{\mathsf{G}\colon \{0,1\}^* \to \{0,1\}^k\}$. Clearly this is unrealistic, since otherwise the description of H would be unreasonably long, and indeed there are no practical functions that behave like a random oracle. The ROM paradigm states that if a cryptographic protocol can be proved to be secure by assuming the existence of a random oracle, then replacing the random oracle with a (hash) function H that meets certain requirements does not affect the security of the protocol. Although fundamentally heuristic, ROM has proven extremely useful in allowing formal security proofs of cryptographic protocols without undermining the practical security of the protocols; in fact, there are no known attacks that exploit the use of a hash function instead of a random oracle [133]. Nevertheless, an important research direction is to remove the ROM requirement, building security proofs that rely solely on assumptions about the complexity of the underlying problems. Schemes proven secure without the use of idealized versions of cryptographic primitives are said to be secure in the *standard model*.

When we consider a security experiment within the ROM, the adversary has oracle access to the random oracle H. A standard technique to obtain a security proof in the ROM is known as *reprogramming with lazy sampling*. The random oracle is controlled by the challenger, which, instead of sampling for a random function at the beginning of the game, proceeds to program the outputs of H with random values as it receives new queries. More in detail, at the start of the game the challenger initializes a table HT of inputs that are queried to the random oracle. When a new query Q is sent to H, the challenger checks that $\mathsf{HT}[Q] = \perp$, sample $x \leftarrow_{\$} \{0,1\}^k$ and programs $\mathsf{HT}[Q] \leftarrow x$, finally returning x. The core of security proof is often based on appropriate random oracle programming. A standard example of this approach is found in the security analysis of Hash-and-Sign schemes provided in Section 2.2.1. Later, we will make extensive use of this model in the analysis of subsequent aggregation protocols.

Experiment 1.2: OW_F

```
F: $\mathcal{X} \to \mathcal{Y}$ with $\mathrm{len}(\mathcal{X}) = \lambda$.
1: $x \leftarrow_\$ \mathcal{X}$
2: $y \leftarrow \mathsf{F}(x)$
3: $x' \leftarrow_\$ \mathcal{A}(\mathsf{F}, y)$
4: return $\mathsf{F}(x') = y$
```

Quantum Random Oracle Model When we consider an adversary who has access to a quantum computer, it makes sense to consider that they can query the random oracle on an input in *quantum superposition*. This model is known as the Quantum Random Oracle Model (QROM) and was introduced by Boneh et al. [38]. The problem of extending security proofs from the ROM to the QROM is highly non-trivial, as many standard techniques cannot be directly applied in the quantum model. However, numerous works recently have attempted to fill the gap, typically at the cost of introducing slightly larger safety losses in the reductions. In the context of digital signatures, the security of schemes in the QROM has been studied and proved for both the Hash-and-Sign paradigm [194, 141, 134, 63] and the Fiat-Shamir paradigm [67, 189, 140, 82].

1.3 Cryptographic Primitives

In this section, we introduce some basic cryptographic primitives, which are recalled throughout the thesis.

1.3.1 One-Way Functions

The notion of a one-way function is fundamental in cryptography, as it describes a function that is efficient to compute but computationally difficult to invert.

Definition 1.1 (One-Way Function)**.** Let $\mathsf{F}\colon \mathcal{X} \to \mathcal{Y}$ with $\mathrm{len}(\mathcal{X}) = \lambda$ and let $\mathcal{A}$ by an adversary. We define the advantage of $\mathcal{A}$ playing the OW game (Experiment 1.2) against F as

$$\mathsf{Adv}^{\mathsf{OW}}_{\mathsf{F}}(\mathcal{A}) = \Pr[\mathsf{OW}_{\mathsf{F}}(\mathcal{A}) = 1].$$

We say that F is *one-way* if it is deterministic polynomial-time for all input $x \in \mathcal{X}$ and the advantage $\mathsf{Adv}^{\mathsf{OW}}_{\mathsf{F}}(\mathcal{A})$ is negligible for any adversary $\mathcal{A}$.

1.3.2 Hash Functions

Generally speaking, a *hash function* is a compression map $\mathsf{H}\colon \{0,1\}^* \to \{0,1\}^k$ with some additional security properties. As we previously mentioned, hash functions are used in practice as a concrete instantiation of a random oracle and are then expected to behave as a truly random function. Since we don't explicitly reduce to the properties of hash functions, below we give a formal definition without introducing additional security notions.

Definition 1.2 (Hash Function)**.** A *cryptographic hash function* with output of length k is a function $\mathsf{H}\colon \{0,1\}^* \to \{0,1\}^k$ such that the following problems are computationally hard with respect to k:

Preimage Resistance Given an output $y \in \{0,1\}^k$, find any input such that $\mathsf{H}(x) = y$.

Second Preimage Resistance Given an input $x \in \{0,1\}^*$ find any $x' \neq x$ such that $\mathsf{H}(x') = \mathsf{H}(x)$.

Collision Resistance Find any distinct $x, x' \in \{0,1\}^*$ such that $\mathsf{H}(x) = \mathsf{H}(x')$.

A useful generalization of a hash function is given by *extendable-output functions* (XOF), where instead of having outputs of fixed length, the output can be extended to any chosen length. XOFs must satisfy the same cryptographic properties as hash functions and can be treated within the ROM as independent random oracles. More in detail, a XOF is a function $\mathsf{H}\colon \mathbb{N} \times \{0,1\}^* \to \{0,1\}^*$ such that $\mathsf{H}^{(k)} := \mathsf{H}(k, \cdot)\colon \{0,1\}^* \to \{0,1\}^k$ is modelled as a random oracle.

1.3.3 Digital Signatures

A digital signature is a public-key cryptographic scheme that can be thought of as the cryptographic analogue of a handwritten signature. A digital signature allows a signer in possession of the private key to prove the *authenticity* of a message, guaranteeing a *non-repudiation* property on the part of the author. Anyone in possession of the corresponding public key can verify the authenticity of the signature and confirm the message's binding to the signer.

Formally, a digital signature scheme Sig is a tuple of three algorithms (KGen, Sign, Vrfy):

- $\mathsf{KGen}(1^\lambda)$: takes as input a security parameter 1^λ and generates a key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Sign}(\mathsf{sk}, m)$: takes as input a signing key sk and a message m and returns a signature σ.
- $\mathsf{Vrfy}(\mathsf{pk}, m, \sigma)$: takes as input a verification key pk, a message m and a signature σ and returns 1 for acceptance or 0 for rejection.

Experiment 1.3: strong EUF-CMA

```
$(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{KGen}(1^{\lambda})$
$\mathcal{Q} \leftarrow \varnothing$
$(m, \sigma) \leftarrow_{\$} \mathcal{A}^{\mathsf{O},\mathsf{OSign}}(\mathsf{pk})$
**if** $(m, \sigma) \in \mathcal{Q}$ **then**
    **return** $0$
**return** $\mathsf{Vrfy}(\mathsf{pk}, m, \sigma)$

$\mathsf{OSign}(m)$:
  $\sigma \leftarrow_{\$} \mathsf{Sign}(\mathsf{sk}, m)$
  $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma)\}$
  **return** $\sigma$
```

We say that Sig is *correct* if an honest generated signature is always verified, i.e.,

$$\Pr\left[\mathsf{Vrfy}(\mathsf{pk}, m, \sigma) = 1 \;\middle|\; \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{KGen}(1^{\lambda}) \\ \sigma \leftarrow_{\$} \mathsf{Sign}(\mathsf{sk}, m) \end{array}\right] = 1.$$

In terms of security, the minimum acceptable notion asks that no adversary can forge a valid signature without the knowledge of the private key. This is often not enough in real-world scenarios, where an adversary may have access to valid signatures, or even have access to a signature oracle that produces valid signatures for partially adversary-controlled messages. In this model, an adversary might extract enough information to generate a forgery on a message of their choice, or even be able to recover the private key. Therefore, we usually consider a stronger notion of Existential Unforgeability against Chosen-Message Attacks (EUF-CMA), where an adversary has access to a signing oracle that provides a polynomial number of valid message-signature pairs. Although not strictly necessary, this model is typically considered in the ROM, so that the signature scheme can be extended to arbitrary messages as the signature is computed on the output of a suitable hash function.

Definition 1.3 (EUF-CMA security)**.** Let O be a random oracle, let Sig = (KGen, Sign, Vrfy) be a signature scheme, let $\mathcal{A}$ be an adversary. We define the advantage of $\mathcal{A}$ playing the EUF-CMA game (Experiment 1.3) against Sig in the random oracle model as:

$$\mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{Sig}}(\mathcal{A}) = \Pr\left[\text{EUF-CMA}_{\mathsf{Sig}}(\mathcal{A}) = 1\right]$$

We say that Sig is *existential unforgeable against chosen-message* attacks if the advantage $\mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{Sig}}(\mathcal{A})$ is negligible for any adversary $\mathcal{A}$.

A slightly stronger notion can also be considered, where the adversary may produce forgery on messages already queried to the signing oracle, provided that the forged signature is distinct from the oracle's response. We denote this variant as strong EUF-CMA (SUF-CMA).

1.3.4 Aggregate Signatures

An aggregate signature scheme allows compressing n signatures from different users on potentially distinct messages. In its most general form, each user with keys $(\mathsf{pk}_i, \mathsf{sk}_i)$ for the signature scheme Sig produces a signature σ_i on a message m_i. Then, there exists a public aggregation algorithm for Sig that takes as input $\sigma_1, \dots, \sigma_n$ and produces a single compressed signature Σ that can be directly verified. The main desired property for an aggregate signature is that the length of the compressed signature Σ is less than the concatenation of the individual signatures σ_i, i.e., $|\Sigma| \ll \sum_{i=1}^{n} |\sigma_i|$. For practical applications, it is also required that the aggregation process does not involve changing the underlying signature protocol keys, so that signing and verification keys can be used interchangeably between single and aggregate signature scheme. In fact, aggregation schemes are particularly useful for practical and established signature schemes. Sometimes it is possible to satisfy additional properties, such as obtaining that the aggregated signature Σ is still a valid signature for Sig for some public key pk obtained as a combination of the signers' public keys pk_i.

In what follows, we first consider a generic notion of aggregation, and then specialize on a weaker but easier-to-obtain form in the context of post-quantum signatures.

An aggregate signature scheme AS with associated signature scheme Sig is a tuple of three algorithms (KGen, AggSign, AggVrfy):

- $\mathsf{KGen}(1^\lambda)$: takes as input a security parameter 1^λ and generates a key pair $(\mathsf{pk}, \mathsf{sk})$.

- $\mathsf{AggSign}(\overrightarrow{\mathsf{pk}}_n, \vec{m}_n, \vec{\sigma}_n)$: takes as input a list of public keys $\overrightarrow{\mathsf{pk}}_n = (\mathsf{pk}_1, \dots, \mathsf{pk}_n)$ of messages $\vec{m}_n = (m_1, \dots, m_n)$ and signatures $\vec{\sigma}_n = (\sigma_1, \dots, \sigma_n)$, where $\sigma_i \leftarrow\$ \mathsf{Sign}(\mathsf{sk}_i, m_i)$ for all $i \in \{1, \dots, n\}$. The algorithm combines the individual signatures $\sigma_1, \dots, \sigma_n$ and returns an aggregate signature Σ.

- $\mathsf{AggVrfy}(\overrightarrow{\mathsf{pk}}_n, \vec{m}_n, \Sigma_n)$: takes as input a list of public keys $\overrightarrow{\mathsf{pk}}_n = (\mathsf{pk}_1, \dots, \mathsf{pk}_n)$ and messages $\vec{m}_n = (m_1, \dots, m_n)$, and an aggregate signature Σ. The algorithm returns 1 for acceptance and 0 for rejection.

The security notion associated with aggregated signatures generalizes the notion of EUF-CMA described for plain signatures. More in detail, an adversary is given a target public key and oracle access to an aggregate signing oracle. The adversary can query the oracle with arbitrary messages and using their own independently generated public keys. Eventually, the adversary should provide a forgery involving the target public key for some set of messages and public keys not previously queried to the oracle. Since we work primarily with a variant of aggregated signatures, we defer the formalization of this security notion to the next section. The differences with the notion for general aggregation schemes are minimal and concern only the interaction of the adversary with the signature oracle.

1.3.5 Sequential Aggregate Signatures

Generic aggregation schemes allow a (potentially untrusted) third party to aggregate signatures produced by the individual users. Although very powerful, these constructs are difficult to achieve in practice, and it is often sufficient to focus on a slightly weaker variant where aggregation is done sequentially by individual signers combining their own signatures with the so-far aggregated signature. This variant is known as *sequential aggregate signature* (SAS).

Formally, a SAS scheme is a tuple of three algorithms ($\mathsf{KGen}, \mathsf{AggSign}, \mathsf{AggVrfy}$):

- $\mathsf{KGen}(1^\lambda)$: takes as input a security parameter 1^λ and generates a key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{AggSign}(\mathsf{sk}_i, m_i, \Sigma_{i-1})$: takes as input the secret key and the message of the i-th user and the previous aggregate signature Σ_{i-1} and returns an aggregate signature Σ_i.
- $\mathsf{AggVrfy}(L_n, \Sigma_n)$: takes as input the full history $L_n = (\mathsf{pk}_1, m_1), \ldots, (\mathsf{pk}_n, m_n)$ of public key, message pairs, and an aggregate signature Σ_n. Returns 1 if Σ_n is a valid aggregate signature and 0 otherwise.

Every signer has a key pair $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow_\$ \mathsf{KGen}(1^\lambda)$. The signature aggregation process is done iteratively: the first signer with keys $(\mathsf{pk}_1, \mathsf{sk}_1)$ generates a signature Σ_1 for message m_1 with $\Sigma_1 \leftarrow_\$ \mathsf{AggSign}(\mathsf{sk}_1, m_1, \varepsilon)$, where ε represents the empty string to indicate that this is the first signature in the sequence. The i-th signer with keys $(\mathsf{pk}_i, \mathsf{sk}_i)$ receives an aggregate signature Σ_{i-1} from the $(i-1)$-th signer and aggregates his signature on message m_i to obtain the aggregate signature $\Sigma_i \leftarrow_\$ \mathsf{AggSign}(\mathsf{sk}_i, m_i, \Sigma_{i-1})$. Note that, in general, the aggregation algorithm $\mathsf{AggSign}$ does not require the public keys and messages from the previous signers. Finally, the verifier can check the validity of the aggregate signature by running $\mathsf{AggVrfy}(L_n, \Sigma_n)$.

Security model for SAS Below we show the definition of Sequential Existential Unforgeability against Chosen-Message Attacks (SAS-EUF-CMA).

As briefly mentioned above, in this model the forger controls all users' private keys except for at least one honest user. The forger can choose the keys of the rogue signers and adaptively query an aggregate signature oracle. Finally, to win the experiment, the forger must produce a valid, non-trivial aggregate signature involving the public key of the honest user.

Formally, let O be a random oracle, let $\mathsf{SAS} = (\mathsf{KGen}, \mathsf{OAggSign}, \mathsf{OAggVrfy})$ be a SAS scheme, let $\mathcal{A}$ be an adversary. We define the advantage $\mathsf{Adv}_{\mathsf{SAS}}^{\mathsf{SAS\text{-}EUF\text{-}CMA}}(\mathcal{A})$ of $\mathcal{A}$ playing the SAS-EUF-CMA game as its success probability in the following experiment.

Setup The challenger produces a key pair $(\mathsf{pk}^\star, \mathsf{sk}^\star) \leftarrow_{\$} \mathsf{KGen}(1^\lambda)$ and provides $\mathsf{pk}^\star$ to $\mathcal{A}$.

Queries $\mathcal{A}$ can be adaptive and has access to a random oracle O and an aggregate signing oracle $\mathsf{OAggSign}(\mathsf{sk}^\star, \cdot, \cdot)$. The aggregate signing oracle receives a message m and an aggregate signature Σ_i on messages $m_1, \dots, m_i$ under public keys $\mathsf{pk}_1, \dots, \mathsf{pk}_i$ controlled by $\mathcal{A}$; it returns an aggregate signature Σ under the challenge secret key $\mathsf{sk}^\star$. The random oracle provides $\mathcal{A}$ with access to a suitable random function.

Output $\mathcal{A}$ outputs an aggregate signature Σ on n messages $m_1, \dots, m_n$ under public keys $\mathsf{pk}_1, \dots, \mathsf{pk}_n$ for a chosen $n \in \mathbb{N}$. One of the public keys must be the challenge public key $\mathsf{pk}_{i^\star} = \mathsf{pk}^\star$ for some $1 \le i^\star \le n$.

The adversary $\mathcal{A}$ wins the experiment if Σ is a valid aggregate signature and a query with messages $m_1, \dots, m_{i^\star}$ under public keys $\mathsf{pk}_1, \dots, \mathsf{pk}_{i^\star}$ has never been made to the oracle.

In Chapter 3, we further specialize this security notion for *full-history* and (partial-signature) *history-free* variants of sequential aggregation schemes. The respective experiments are described in Experiment 3.1 and Experiment 3.2.

1.4 Post-Quantum Cryptography

The threat of quantum computers has remained purely theoretical for many years. In 1996, Grover [117], however, described a search algorithm for an unordered list of n elements that only requires $\sqrt{n}$ quantum operations. More in detail, given a function f over a discrete domain of n elements and a target y, Grover's algorithm finds an input x such that $f(x) = y$ in $\mathcal{O}(\sqrt{n})$ iterations, without further assumptions on the structure of f. This gives a quadratic speed-up over the classical counterpart, which requires $\mathcal{O}(n)$ operations. This result is particularly interesting for symmetric cryptosystem, for which the best attacks are typically brute force attacks on the key space. Taking into account only the asymptotic behaviour of Grover's algorithm, this means that a key of 128 bits[1] has only 64 bits of security against a quantum adversary. This is relevant mainly from a theoretical point of view, since Grover's algorithm is difficult to parallelize and the

[1] Typical lengths for symmetric keys are 128, 192, and 256 bits.

concrete security of an exhaustive 128-bit attack is plausibly still out of scale [101, 178]. Either way, any attack with a quadratic speed-up can be easily mitigated by doubling the length of the keys for each level of security needed.

A context where quantum computers have a far more worrying advantage is that of public key cryptography. The main schemes used today for key-exchange and digital signatures are RSA, Diffie-Hellman and elliptic curve cryptography, which are based on two hard problems: integer factorization and discrete logarithm. In 1994, Shor [180] described a quantum algorithm capable of solving both problems with an exponential speed-up over the classical counterpart. The realization of a quantum computer would, therefore, have disastrous consequences on today's public key cryptography, making these schemes no longer secure.

While not going into the details of the Shor algorithm, it is interesting to note how this algorithm allows solving a wider class of problems and how they can be exploited for the factoring problem and the discrete logarithm problem. Roughly speaking, given a periodic function f, i.e., there exists a period P such that $f(x+P) = f(x)$ for any x, Shor's algorithm can efficiently find P.

Integer factorization. Given $N = pq$ the integer product of two large primes, the problem is to find p, q. The key observation for the application of Shor's algorithm is that it is easy to factor N if, fixed $a \in \mathbb{Z} \setminus \{0\}$, one can find the period of $a^x \mod N$. In fact, if we find the period P, then

$$a^x \equiv a^{x+P} \pmod{N} \implies a^P \equiv 1 \pmod{N} \implies a^P - 1 \equiv 0 \pmod{N},$$

so that N divides $a^P - 1$. Now we can just factor $a^P - 1 = (a^{P/2} - 1)(a^{P/2} + 1)$ and check if this leads to a factor of N by computing the gcd of N with each factor. We can iterate this procedure with random choices of a until we find a non-trivial factor of N. From a computational point of view, if we let $n = \log_2(N)$, this algorithm takes approx $\mathcal{O}(n^3)$ iterations, which is polynomial in n. While the best classical algorithm is exponential in n.

Discrete logarithm. Given a group $G = \langle g \rangle$ generated by g and given $y = g^x$, the problem is to find x. Since $q = \mathrm{ord}(g)$ is a known parameter, we need to find $x \mod q$. It is enough to find the period of $f(a,b) = g^a y^b$, that is P, P' such that $f(a+P, b+P') = f(a,b)$. In particular, we have

$$g^a y^b = g^{a+P} y^{b+P'} \implies g^P y^{P'} = 1 \implies g^{P+xP'} = 1.$$

Therefore, $P + xP' \equiv 0 \pmod{q}$ and $x = -P/P' \pmod{q}$.

1.4.1 NIST Standardization Processes

Post-Quantum Cryptography aims to design public-key cryptosystem that cannot be broken by an adversary that has access to a quantum computer. Therefore,

Table 1.1: Categories and number of digital signature proposals between the first and second NIST calls. The winners of Round III are highlighted.

Category	**1st Call**			**2nd Call**
	Round 1	Round 2	Round 3	Round 1
Code-based	3	–	–	5
Isogenies	–	–	–	1
Lattices	5	3	2	7
MPC-in-the-Head	–	–	–	7
Multivariate	7	4	2	11
Symmetric	2	1	1	4
Other	2	1	1	5

such schemes have to be based on hard problems for which there is no exponential quantum speed-up, instead of what happens to integer factorization or discrete logarithm.

The cryptographic community began to be particularly interested in the development of such schemes when in 2015 the American National Security Agency (NSA) declared that it was necessary to begin a phase of transition to post-quantum schemes in view of a possible realization of a quantum computer. Although this development may not occur quickly, it has been assessed that, in the absence of timely intervention, the period needed to switch to a new public key encryption standard could take an even longer time. Following this, in 2016, NIST launched a standardization process for public-key encryption, key exchange and signature schemes [162]. Among the 69 starting candidates, only 19 proposed digital signature algorithms. In July 2022, with the conclusion of the third round [5], NIST has identified three signature schemes that will proceed to be standardized: CRYSTALS-Dilithium [144], Falcon [172] and SPHINCS$^+$ [124]. Despite the reduced number, the initial proposals were heterogeneous, with proposals coming from the main categories of post-quantum problems, such as lattices, linear codes, multivariate systems, and symmetric primitives. The diversification gradually decreased in subsequent rounds, leading to an over-representation of algorithms based on structured lattices. With the aim of further differentiating security assumptions and use-case scenarios for standardized signatures, in 2022 NIST launched a new "on-ramp" process dedicated to post-quantum digital signatures [161].

In June 2023, at the end of the new submission phase, NIST received 50 proposals and found 40 to be compatible with the requirements of the call. In addition to the main categories already identified in the original process, new proposals include algorithms based on isogenies of supersingular elliptic curves and others framed in the recent “MPC-in-the-head” paradigm. Table 1.1 shows the main problem families and the number of proposals among the various rounds of NIST processes. In the following, we give a brief description of the main categories of underlying problems.

Code-based Code-based schemes are tied to the theory of error-correcting codes. The underlying computational problems are related to (general) syndrome decoding problem and the problem of finding isometries between (linear) codes.

Isogenies Isogeny-based cryptography is based on the properties of isogenies between supersingular elliptic curves. The security relies on the difficulty of finding an isogeny of a given degree between two supersingular elliptic curves over a finite field.

Lattices Lattice-based cryptography is related to lattice problems over high-dimensional lattices. The hardness assumptions typically relate to the problem of finding a short vector in the lattice or a vector close to a given vector outside the lattice.

MPC-in-the-Head MPC-in-the-Head is a recent paradigm that leverages secure Multi-Party Computation (MPC) protocols to construct zero-knowledge proofs. Combined with the Fiat-Shamir transform, this allows to obtain efficient digital signature schemes.

Multivariate Multivariate-based cryptography is based on multivariate polynomials over a finite field. The security relies on the hardness of solving systems of multivariate quadratic equations over a finite field (the MQ Problem).

Symmetric Symmetric-based cryptography employs symmetric cryptographic primitives, particularly hash functions, for digital signatures. The security reduces to the collision and preimage resistance of hash functions.

In both calls, NIST defined five security levels, for which the submitted schemes must propose different parameterizations. Each level defines the complexity of breaking the security of the proposed scheme by comparison with a standard symmetric primitive. Specifically, levels I, III, and V refer to the complexity of performing an exhaustive key-search attack against AES-128, AES-192, and AES-256, respectively. Levels II and IV, on the other hand, refer to the complexity of finding a collision for SHA-256 and SHA-384, respectively.

Part I

Trapdoor-based Signature Aggregation

Chapter 2

Hash-and-Sign Paradigm

The Hash-and-Sign (HaS) paradigm is a standard technique to build a digital signature from trapdoor one-way functions and hash functions in the random oracle model. On a high level, the signature of a message is obtained by finding a pre-image of the message hash. A signature can be easily verified by applying the public function but, without knowledge of the trapdoor, finding a pre-image should be computationally hard. Historically, this paradigm is known as Full Domain Hash [21, 22] when employed with trapdoor permutations, such as the one underlying the well-known RSA scheme [174].

When we consider post-quantum assumptions, no trapdoor permutations are known to date, and it is therefore necessary to consider non-bijective functions. In principle, to build a digital signature in the Hash-and-Sign paradigm we only need the trapdoor function to be surjective, so that a pre-image always exists. On the other hand, the loss of injectivity presents issues in the formal security of the signature, as it is not obvious that one-wayness alone is sufficient to prove unforgeability. In the literature, this problem has been addressed in different ways. Lattice-based trapdoor functions typically fall into the class of Preimage Sampleable Functions (PSFs) [108], presenting additional properties that allow for an easy recovery of formal security in signature schemes. PSFs are difficult to implement outside of lattices, and a weaker model has been considered for code-based assumptions [52]. Finally, multivariate trapdoor functions typically do not present additional properties, and schemes based on these assumptions have solved this problem independently [175].

In Section 2.1 we introduce the basic notions of trapdoor functions, analysing some classes of functions which are relevant for applications to post-quantum signatures. In Section 2.2 we describe a generalization of the Hash-and-Sign paradigm as recently formalized in [134]. We also provide an overview of the main techniques used in the security proofs of Hash-and-Sign schemes, anticipating some methods that will be employed in the security analysis of sequential aggregation schemes in the next chapter. Finally, in Section 2.3 we present the

Experiment 2.1: OW, INV, and CR

OW:

```
$(\mathsf{F}, \mathsf{I}) \leftarrow\!\!\$\ \mathsf{TrapGen}(1^\lambda)$
$x \leftarrow\!\!\$\ \mathsf{SampDom}(\mathsf{F})$
$y \leftarrow \mathsf{F}(x)$
$x' \leftarrow\!\!\$\ \mathcal{A}(\mathsf{F}, y)$
return $\mathsf{F}(x') = y$
```

INV:

```
$(\mathsf{F}, \mathsf{I}) \leftarrow\!\!\$\ \mathsf{TrapGen}(1^\lambda)$
$y \leftarrow\!\!\$\ \mathcal{Y}$
$x \leftarrow\!\!\$\ \mathcal{A}(\mathsf{F}, y)$
return $\mathsf{F}(x) = y$
```

CR:

```
$(\mathsf{F}, \mathsf{I}) \leftarrow\!\!\$\ \mathsf{TrapGen}(1^\lambda)$
$(x_0, x_1) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{F})$
return $\mathsf{F}(x_0) = \mathsf{F}(x_1)$
```

state of the art of this paradigm in the post-quantum landscape, introducing some of the schemes that will later be analysed with respect to aggregation capabilities.

2.1 Trapdoor Functions

Definition 2.1. A Trapdoor Function (TDF) T is a tuple of four algorithms $(\mathsf{TrapGen}, \mathsf{F}, \mathsf{I}, \mathsf{SampDom})$:

- $\mathsf{TrapGen}(1^\lambda)$: takes as input a security parameter 1^λ and generates an efficiently computable function $\mathsf{F}\colon \mathcal{X} \to \mathcal{Y}$ and a trapdoor I that allow to invert F.

- $\mathsf{F}(x)$: takes as input $x \in \mathcal{X}$ and outputs $\mathsf{F}(x) \in \mathcal{Y}$.

- $\mathsf{I}(y)$: takes as input $y \in \mathcal{Y}$ and outputs $x \in \mathcal{X}$ such that $\mathsf{F}(x) = y$, or it fails by returning $\perp$.

- $\mathsf{SampDom}(\mathsf{F})$ takes as input a function $\mathsf{F}\colon \mathcal{X} \to \mathcal{Y}$ and outputs $x \in \mathcal{X}$.

A special type of trapdoor function is obtained when we consider bijections.

Definition 2.2. A TDF $\mathsf{T} = (\mathsf{TrapGen}, \mathsf{F}, \mathsf{I}, \mathsf{SampDom})$ is said to be a Trapdoor Permutation (TDP) if F and I are permutations.

Remark. A relevant example of trapdoor permutation is given by the RSA function.

The standard form of security for a trapdoor function is given by the notion of One-Wayness (OW) [21].

Definition 2.3. Let $\mathsf{T} = (\mathsf{TrapGen}, \mathsf{F}, \mathsf{I}, \mathsf{SampDom})$ be a TDF and let $\mathcal{A}$ be an adversary. We define the advantage of $\mathcal{A}$ playing the OW game (Experiment 2.1) against T as

$$\mathsf{Adv}_{\mathsf{T}}^{\mathrm{OW}}(\mathcal{A}) = \Pr[\mathrm{OW}_{\mathsf{T}}(\mathcal{A}) = 1].$$

We say that T is *one-way* if the advantage $\mathsf{Adv}_{\mathsf{T}}^{\mathrm{OW}}(\mathcal{A})$ is negligible for any adversary $\mathcal{A}$.

The notion of one-wayness requires that the challenge is obtained as $\mathsf{F}(x)$ with x sampled following some distribution on $\mathcal{X}$. If the image via F of a random input is not uniformly distributed in $\mathcal{Y}$, an alternative notion of Non-Invertibility (INV) [122] where the challenge is uniformly sampled is necessary to prove the security of Hash-and-Sign schemes.

Definition 2.4. Let $\mathsf{T} = (\mathsf{TrapGen}, \mathsf{F}, \mathsf{I}, \mathsf{SampDom})$ be a TDF and let $\mathcal{A}$ be an adversary. We define the advantage of $\mathcal{A}$ playing the INV game (Experiment 2.1) against T as

$$\mathsf{Adv}_{\mathsf{T}}^{\mathrm{INV}}(\mathcal{A}) = \Pr[\mathrm{INV}_{\mathsf{T}}(\mathcal{A}) = 1].$$

We say that T is *non-invertible* if the advantage $\mathsf{Adv}_{\mathsf{T}}^{\mathrm{INV}}(\mathcal{A})$ is negligible for any adversary $\mathcal{A}$.

In the literature, there is not always a clear distinction between these two notions, and both are commonly described as one-wayness [108, 175, 52]. When implementing the Hash-and-Sign paradigm with post-quantum trapdoor functions, we will mainly use the latter concept of non-invertibility. This is because these functions usually do not have a uniform distribution of outputs.

Remark. Note that the INV notion of Definition 2.4 includes trapdoor functions for which the probability that a pre-image exists via F for a random element $y \in \mathcal{Y}$ is negligible. Such functions could not be used as a building block for Hash-and-Sign schemes, as even knowledge of the secret trapdoor would not allow the computation of a pre-image. Nevertheless, a non-negligible failure probability could impact the tightness of a security reduction from INV. In security proofs, it is therefore important to explicitly consider this possibility.

Finally, we can consider an advanced notion of Collision Resistance (CR). Trapdoor functions that satisfy this notion can typically achieve tighter security reductions and enable advanced constructions.

Definition 2.5. Let $\mathsf{T} = (\mathsf{TrapGen}, \mathsf{F}, \mathsf{I}, \mathsf{SampDom})$ be a TDF and let $\mathcal{A}$ be an adversary. We define the advantage of $\mathcal{A}$ playing the CR game (Experiment 2.1) against T as

$$\mathsf{Adv}_{\mathsf{T}}^{\mathrm{CR}}(\mathcal{A}) = \Pr[\mathrm{CR}_{\mathsf{T}}(\mathcal{A}) = 1].$$

We say that T is *collision-resistant* if the advantage $\mathsf{Adv}_{\mathsf{T}}^{\mathrm{CR}}(\mathcal{A})$ is negligible for any adversary $\mathcal{A}$.

2.1.1 Preimage Sampleable Functions

When a trapdoor function is used to construct a digital signature, each signature reveals a preimage $x \leftarrow\!\!\$\; \mathsf{I}(y)$ obtained via the secret trapdoor on a uniformly distributed element $y \leftarrow\!\!\$\; \mathcal{Y}$. To ensure the security of the scheme, it is necessary that knowledge of the pair (x, y) does not reveal information about the trapdoor.

When we consider a trapdoor permutation, this problem does not arise, since the preimage of a uniformly distributed element is still uniformly distributed. This property of *preimage sampleability* is quickly lost when we consider a generic trapdoor function, which is generally not injective. However, building a one-way trapdoor permutation in the post-quantum setting presents significant challenges [123]. Nonetheless, relevant properties for security can be obtained by imposing additional requirements on the trapdoor function. A strong notion of Preimage Sampleable Functions was introduced for lattice-based functions in [108].

Definition 2.6. A TDF $\mathsf{T} = (\mathsf{TrapGen}, \mathsf{F}, \mathsf{I}, \mathsf{SampDom})$ is a Preimage Sampleable Function (PSF) if it satisfies the following properties:

1. $y \leftarrow \mathsf{F}(\mathsf{SampDom}(\mathsf{F}))$ is uniformly distributed over $\mathcal{Y}$.
2. $x \leftarrow\$ \mathsf{I}(y)$, with $y \leftarrow\$ \mathcal{Y}$, is distributed as $x \leftarrow\$ \mathsf{SampDom}(\mathsf{F})$ conditioned on $\mathsf{F}(x) = y$.
3. For any $y \in \mathcal{Y}$, $\mathsf{I}(y)$ always returns $x \in \mathcal{X}$ such that $\mathsf{F}(x) = y$.

A collision-resistant PSF satisfies the following additional property:

4. For any $y \in \mathcal{Y}$, the conditional min-entropy of $x \leftarrow\$ \mathsf{SampDom}(\mathsf{F})$ conditioned on $\mathsf{F}(x) = y$ is at least $\omega(\log \lambda)$.

In [52, 59], a relaxed variant of PSFs is considered where the uniformity property on preimages is weakened and required to hold only on average. This was first considered for code-based functions, for which no PSFs are known.

Definition 2.7. A TDF $\mathsf{T} = (\mathsf{TrapGen}, \mathsf{F}, \mathsf{I}, \mathsf{SampDom})$ is an ε-Average Preimage Sampleable Function (ε-APSF) if it satisfies the following properties:

1. For any $y \in \mathcal{Y}$, $\mathsf{I}(y)$ always returns $x \in \mathcal{X}$ such that $\mathsf{F}(x) = y$.
2. Consider the statistical distance $\varepsilon_{\mathsf{F},\mathsf{I}} = \Delta(\mathsf{SampDom}(\mathsf{F}), \mathsf{I}(\mathcal{U}(\mathcal{Y})))$. Then, it holds that $\mathbb{E}_{(\mathsf{F},\mathsf{I})}\left[\varepsilon_{\mathsf{F},\mathsf{I}}\right] \leq \varepsilon$.

Remark. The definition of PSF requires uniform images with the first property of Definition 2.6. For an ε-APSF this is not required, but it holds that [52, Prop. 2]

$$\Delta(\mathsf{F}(\mathsf{SampDom}(\mathsf{F})), \mathcal{U}(\mathcal{Y})) \leq \varepsilon_{\mathsf{F},\mathsf{I}}.$$

It can be easily observed that a trapdoor permutation satisfies all the properties of a PSF. In particular, it holds that

$$\text{TDP} \implies \text{PSF} \implies \text{Average PSF.}$$

Experiment 2.2: PS_b

```
$(\mathsf{F}, \mathsf{I}) \leftarrow_\$ \mathsf{TrapGen}(1^\lambda)$
$b^\star \leftarrow_\$ \mathcal{A}^{\mathsf{Sample}_b}(\mathsf{F})$
return $b^\star \in \{0,1\}$

$\mathsf{Sample}_1$:
  $r_i \leftarrow_\$ \mathsf{R}$
  $x_i \leftarrow_\$ \mathsf{SampDom}(\mathsf{F})$
  return $(r_i, x_i)$

$\mathsf{Sample}_0$:
  repeat
      $r_i \leftarrow_\$ \mathsf{R}$
      $y_i \leftarrow_\$ \mathcal{Y}$
      $x_i \leftarrow_\$ \mathsf{I}(y_i)$
  until $x_i \neq \perp$
  return $(r_i, x_i)$
```

When we consider a generic trapdoor function that does not satisfy the notion of (A)PSF, such as for multivariate-based functions, preimage sampleability must be ensured with an additional security assumption. This notion is formalized in [134] with the Preimage Sampling (PS) game.

Definition 2.8 (Preimage Sampling)**.** Let $\mathsf{T} = (\mathsf{TrapGen}, \mathsf{F}, \mathsf{I}, \mathsf{SampDom})$ be a TDF, let $\mathcal{A}$ be an adversary. We define the advantage of $\mathcal{A}$ playing the PS game (Experiment 2.2) against T as:

$$\mathrm{Adv}_{\mathsf{T}}^{\mathrm{PS}}(\mathcal{A}) = |\Pr[\mathrm{PS}_0(\mathcal{A}) = 1] - \Pr[\mathrm{PS}_1(\mathcal{A}) = 1]|.$$

We say that T is *preimage-sampleable* if the advantage $\mathrm{Adv}_{\mathsf{T}}^{\mathrm{PS}}(\mathcal{A})$ is negligible for any adversary $\mathcal{A}$.

Remark. The PS game introduces the use of a random salt r to recover a weaker property of preimage indistinguishability. When we consider a PSF we have a stronger property where the preimage of a uniformly random output is indistinguishable from a uniformly random input. In the PS game we are trying to distinguish between pairs (r, x) and pairs (r', x'), where r and r' are uniformly random salts over R, $x \in \mathcal{X}$ is the preimage of a uniformly random output $y \leftarrow_\$ \mathcal{Y}$, and $x' \leftarrow_\$ \mathcal{X}$ is a uniformly random input. In the next section, we will see that the pair (r, x) can be described as the output of a generalized Hash-and-Sign scheme. This notion of indistinguishability can then be used in signature schemes to recover EUF-CMA security from the Full Domain Hash approach, where a random salt is not required.

2.2 Hash-and-Sign Schemes

The Hash-and-Sign (HaS) paradigm is a standard approach to building digital signature schemes in the random oracle model from a trapdoor function T and a hash function $\mathsf{H}\colon \{0,1\}^* \to \mathcal{Y}$. When T is a trapdoor permutation, this construction is known as Full Domain Hash [21, 22]. To sign a message m, a signer with a

Algorithm 2.1: Hash-and-Sign with retry

```
$\mathsf{KGen}(1^\lambda)$:
  $(\mathsf{F}, \mathsf{I}) \xleftarrow{\$} \mathsf{TrapGen}(1^\lambda)$
  return $(\mathsf{F}, \mathsf{I})$

$\mathsf{Vrfy}(\mathsf{F}, m, (r, x))$:
  return $\mathsf{F}(x) = \mathsf{H}(r, m)$

$\mathsf{Sign}(\mathsf{I}, m)$:
  repeat
      $r \xleftarrow{\$} \mathsf{R}$
      $x \xleftarrow{\$} \mathsf{I}(\mathsf{H}(r, m))$
  until $x \neq \bot$
  return $(r, x)$
```

secret key $\mathsf{sk} = \mathsf{I}$ applies the hash function, modelled as a random oracle, to the message $y \leftarrow \mathsf{H}(m)$ and computes its inverse $x \xleftarrow{\$} \mathsf{I}(y)$ through the secret trapdoor. In its original form, it can be formulated using the following algorithms.

- $\mathsf{KGen}(1^\lambda)$: takes as input a security parameter 1^λ and generates $(\mathsf{F}, \mathsf{I}) \xleftarrow{\$} \mathsf{TrapGen}(1^\lambda)$. The verification key for the signature scheme is F, the signing key is I.

- $\mathsf{Sign}(\mathsf{I}, m)$: takes as input the trapdoor I and a message m. Computes $\eta \leftarrow \mathsf{H}(m)$ and returns the signature as the preimage $\sigma \xleftarrow{\$} \mathsf{I}(\eta)$.

- $\mathsf{Vrfy}(\mathsf{F}, m, \sigma)$: takes as input a verification key F, a message m and a signature σ. Computes $\eta \leftarrow \mathsf{H}(m)$ and $\eta' \leftarrow \mathsf{F}(\sigma)$. Returns 1 if $\eta = \eta'$, otherwise 0.

Remark. The role of the hash function is twofold. First, it allows the signature scheme to be extended to arbitrary messages. Indeed, if the signature σ of a message m were calculated directly such that $\mathsf{F}(\sigma) = m$, the scheme would be restricted only to messages that reside in the image of the trapdoor function. In the absence of the hash function, it would also be possible to produce forgeries trivially. An attacker could in fact randomly choose σ and obtain a signature for $m \leftarrow \mathsf{F}(\sigma)$.

In some scenarios, the HaS paradigm requires using a random string r, which acts as a salt for the hash function, i.e., $y \leftarrow \mathsf{H}(m \parallel r)$. This is known as the *probabilistic* Hash-and-Sign (PHaS) paradigm. The resulting signature is the couple $\sigma = (x, r)$. A verifier uses the corresponding public key $\mathsf{pk} = \mathsf{F}$ to verify whether $\mathsf{F}(x) = \mathsf{H}(m \parallel r)$.

More generally, a slightly different paradigm, known as probabilistic Hash-and-Sign *with retry* (PHaSwR), is employed with generic trapdoor functions. With this approach, a random string r is sampled until a preimage for $\mathsf{H}(m \parallel r)$ is found. This approach is described in Algorithm 2.1. The security is based on the one-wayness of the trapdoor function and on the additional condition that the output of the signing algorithm (r, x) is indistinguishable from a couple (r', x') with $r' \xleftarrow{\$} \{0,1\}^\lambda$ and $x' \xleftarrow{\$} \mathsf{SampDom}(\mathsf{F})$.

Table 2.1: Summary of existing security proof for Hash-and-Sign schemes in the ROM.

Function	Assumption	Target scheme			Security bound	Proof
		HaS	PHaS	PHaSwR		
TDP	OW/INV	✓	✓	–	$\mathcal{O}(\mathsf{q}_\mathsf{H}\varepsilon_\mathsf{OW})$	[22]
PSF	OW/INV	✓	✓	–	$\mathcal{O}(\mathsf{q}_\mathsf{H}\varepsilon_\mathsf{OW})$	[108]
PSF	CR	✓	✓	–	$\mathcal{O}(\varepsilon_\mathsf{CR})$	[108]
APSF	OW/INV	–	✓	–	$\mathcal{O}(\mathsf{q}_\mathsf{H}\varepsilon_\mathsf{OW})$	[52]
TDF	INV + PS	–	–	✓	$\mathcal{O}(\mathsf{q}_\mathsf{H}\varepsilon_\mathsf{INV} + \varepsilon_\mathsf{PS})$	[134]

2.2.1 Security Analysis

In this section, we give an overview of the security reductions of Hash-and-Sign schemes as the type of trapdoor function changes. Many of the techniques highlighted below underlie the aggregate signature security demonstration in Chapter 3. A summary of the security proof for Hash-and-Sign schemes in the ROM is given in Table 2.1. The techniques presented cover the classical scenario and are generally extendable to QROM [194, 193, 52, 134].

The core idea in the ROM for an (A)PSF, is to simulate the signature oracle for a message m by taking a random element $x \leftarrow_\$ \mathsf{SampDom}(\mathsf{F})$ as the signature and then programming the random oracle as $\mathsf{H}(m) \leftarrow \mathsf{F}(x)$. In this way, x is a valid signature since its image via the trapdoor function is equal to the hash of the message. In the following, we start with the case of the Full Domain Hash paradigm for a trapdoor permutation. Then, we subsequently weaken the assumptions about the trapdoor function, and we analyse the necessary changes in the security proof.

Theorem 2.9 ([22]). *Let* T *be a TDP. Let* $\mathcal{A}$ *be a* EUF-CMA *adversary against the (probabilistic)* HaS *scheme on* T *in the random oracle model, which makes* q_S *signing queries and* q_H *queries to the random oracle. Then, there exists a* OW *adversary* $\mathcal{B}$ *against* T *such that*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{HaS/PHaS}}(\mathcal{A}) \leq (\mathsf{q}_\mathsf{S} + \mathsf{q}_\mathsf{H} + 1)\mathsf{Adv}^{\mathsf{OW}}_{\mathsf{T}}(\mathcal{B}).$$

Proof Idea. We show that the OW adversary $\mathcal{B}$ can simulate the EUF-CMA game. First, after receiving a challenge (F, y), $\mathcal{B}$ forwards F to $\mathcal{A}$. Then, $\mathcal{B}$ initialized a table HT and simulates the oracles queries as follows.

Random oracle queries Suppose a query $Q = (r, m)$ for H is received. If $\mathsf{HT}[Q] \neq \perp$, $\mathcal{B}$ uniformly samples $x \leftarrow_\$ \mathcal{X}$ and stores $\mathsf{HT}[Q] \leftarrow x$. Finally, it returns

$\mathsf{F}(\mathsf{HT}[Q])$. The simulation is correct since $\mathsf{F}\colon \mathcal{X} \to \mathcal{X}$ is a permutation and the output of $\mathsf{F}(x)$ with $x \leftarrow_{\$} \mathcal{X}$ is uniformly distributed.

Signing oracle queries Suppose a query $Q = m$ for OSign is received. First, $\mathcal{B}$ uniformly samples a random salt $r \leftarrow_{\$} \mathsf{R}$. If $\mathsf{HT}[(r, m)] = \perp$, then $\mathcal{B}$ uniformly samples $x \leftarrow_{\$} \mathsf{SampDom}(\mathsf{F})$ and stores $\mathsf{HT}[(r, m)] \leftarrow x$. Otherwise, $\mathcal{B}$ retrieves $x \leftarrow \mathsf{HT}[(r, m)]$. Finally, it returns (r, x). The simulation is correct since simulated signatures are uniformly distributed in $\mathcal{X}$, while real signatures are obtained as $\mathsf{I}(\eta)$ with $\eta \sim \mathcal{U}(\mathcal{X})$. Since we are dealing with a trapdoor permutation, $\mathsf{I}(\eta)$ is uniformly distributed.

To win the OW game, $\mathcal{B}$ randomly chooses a query $Q^\star = (r^\star, m^\star)$ to H among $(\mathsf{q_S} + \mathsf{q_H} + 1)$ possible ones. For this special query, $\mathcal{B}$ uses their challenge y to program the random oracle, i.e. $\mathsf{HT}[Q^\star] \leftarrow y$. Eventually, $\mathcal{A}$ will produce a valid signature $\sigma = (r, x)$ for a message m. Without loss of generality, we assume that before returning the forgery, $\mathcal{A}$ queries H on input (r, m). With probability $1/(\mathsf{q_S} + \mathsf{q_H} + 1)$, we have that $(r, m) = Q^\star$. Therefore, $\mathsf{F}(x) = \mathsf{H}(r, m) = y$ and $\mathcal{B}$ wins his OW game by returning x. □

The previous reduction cannot be applied to generic trapdoor functions that do not satisfy preimage sampleability. However, through the use of PSFs, it is possible to recover the EUF-CMA security.

Theorem 2.10 ([108])**.** *Let* T *be a PSF. Let* $\mathcal{A}$ *be a* EUF-CMA *adversary against the (probabilistic)* HaS *scheme on* T *in the random oracle model, which makes* $\mathsf{q_S}$ *signing queries and* $\mathsf{q_H}$ *queries to the random oracle. Then, there exists a* OW *adversary* $\mathcal{B}$ *against* T *such that*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{HaS/PHaS}}(\mathcal{A}) \le (\mathsf{q_S} + \mathsf{q_H} + 1)\mathsf{Adv}^{\mathsf{OW}}_{\mathsf{T}}(\mathcal{B}).$$

Proof Idea. The proof is the same as that of the previous theorem, except that sampling elements from $\mathcal{X}$ during the simulation is done according to the distribution of $\mathsf{SampDom}(\mathsf{F})$. Since T is a PSF, Property 1 of Definition 2.6 ensures that $\mathsf{F}(\mathsf{SampDom}(\mathsf{F}))$ is uniformly distributed over $\mathcal{Y}$, so that we can use $\mathsf{F}(x)$ to simulate the output of the random oracle. Moreover, Properties 2 and 3 ensures that the real signature distribution, obtained as $x \leftarrow_{\$} \mathsf{I}(\mathsf{H}(r, m))$, is equal to the simulated distribution, obtained as $x \leftarrow_{\$} \mathsf{SampDom}(\mathsf{F})$. □

A stronger reduction to SUF-CMA is possible using collision-resistant PSFs. This also applies in the QROM [38].

Theorem 2.11 ([108])**.** *Let* T *be a collision-resistant PSF. Let* $\mathcal{A}$ *be a* SUF-CMA *adversary against the (probabilistic)* HaS *scheme on* T *in the random oracle model. Then, there exists a* CR *adversary* $\mathcal{B}$ *against* T *such that*

$$\mathsf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{HaS/PHaS}}(\mathcal{A}) \le \frac{1}{1 - 2^{-\omega(\log \lambda)}}\mathsf{Adv}^{\mathsf{CR}}_{\mathsf{T}}(\mathcal{B}).$$

Proof Idea. The simulation of random oracle queries and signing queries coincides with that of the previous theorem. When $\mathcal{A}$ produces a valid signature $\sigma = (r, x)$ for a message m, $\mathcal{B}$ retrieves $x^\star \leftarrow \mathsf{HT}[(r, m)]$ and returns $(x, x^\star)$ as a collision for F. Notice that $\mathsf{F}(x) = \mathsf{H}(r, m) = \mathsf{F}(x^\star)$, so we are left to bound the probability that $x \neq x^\star$. Suppose that $\mathcal{A}$ made a signature query on m and received $(r^\star, x^\star)$. Since (r, x) is a valid signature for the SUF-CMA game, then either $r \neq r^\star$ and (r, m) was later queried to H, or $r = r^\star$ and $x \neq x^\star$. Otherwise, if $\mathcal{A}$ did not make a signature query on m, then (r, m) was queried to H and $\mathcal{B}$ stored $\mathsf{HT}[(r, m)] \leftarrow x^\star$ for $x^\star \leftarrow_\$ \mathsf{SampDom}(\mathsf{F})$. Then, by Property 4 of Definition 2.6, the min-entropy of $x^\star$ given $\mathsf{F}(x^\star)$ is $\omega(\log \lambda)$. Therefore, $x \neq x^\star$ with probability $1 - 2^{-\omega(\log \lambda)}$. □

When we consider the relaxed notion of APSF, we have the following result[1].

Theorem 2.12 ([52]). *Let* T *be an* ε*-APSF. Let* $\mathcal{A}$ *be a* EUF-CMA *adversary against the* PHaS *scheme on* T *in the random oracle model, which makes* $\mathsf{q_S}$ *signing queries and* $\mathsf{q_H}$ *queries to the random oracle. Then, there exists a* OW *adversary* $\mathcal{B}$ *against* T *such that*

$$\mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{PHaS}}(\mathcal{A}) \leq \mathsf{q_H}\mathsf{Adv}^{\mathsf{OW}}_{\mathsf{T}}(\mathcal{B}) + \mathsf{q_S}\left(\varepsilon + \frac{\mathsf{q_S} + \mathsf{q_H}}{|\mathsf{R}|}\right).$$

Proof Idea. When T is an ε-APSF, Property 2 of Definition 2.6 is replaced with a weaker condition such that the expected value of the statistical distance $\varepsilon_{\mathsf{F},\mathsf{I}} = \Delta(\mathsf{SampDom}(\mathsf{F}), \mathsf{I}(\mathcal{U}(\mathcal{Y})))$ over all $(\mathsf{F}, \mathsf{I}) \leftarrow_\$ \mathsf{KGen}(1^\lambda)$ is bounded by ε. $\mathcal{B}$ simulates the oracle queries as follows.

Random oracle queries Suppose a query $Q = (r, m)$ for H is received. If $\mathsf{HT}[Q] \neq \perp$, $\mathcal{B}$ uniformly samples $y \leftarrow_\$ \mathcal{Y}$ and stores $\mathsf{HT}[Q] \leftarrow y$. Finally, it returns $\mathsf{HT}[Q]$. The simulation is correct since it outputs uniformly random elements in $\mathcal{Y}$.

Signing oracle queries Suppose a query $Q = m$ for OSign is received. First, $\mathcal{B}$ uniformly samples a random salt $r \leftarrow_\$ \mathsf{R}$. If $\mathsf{HT}[(r, m)] \neq \perp$, then $\mathcal{B}$ the simulations fails. Otherwise, $\mathcal{B}$ uniformly samples $x \leftarrow_\$ \mathsf{SampDom}(\mathsf{F})$. Finally, it returns (r, x). The statistical distance between the real distribution and the simulated distribution is bounded by $\varepsilon_{\mathsf{F},\mathsf{I}} + \frac{\mathsf{q_S}+\mathsf{q_H}}{|\mathsf{R}|}$. In fact, the simulation differs with the real execution only when $\mathcal{B}$ samples a random salt r such that $\mathsf{HT}[(r, m)] \neq \perp$. Since there are at most $\mathsf{q_S} + \mathsf{q_H}$ entries in HT, this happens with probability at most $\frac{\mathsf{q_S}+\mathsf{q_H}}{|\mathsf{R}|}$. Otherwise, the statistical distance between the output of the simulation and the real execution is given by $\varepsilon_{\mathsf{F},\mathsf{I}}$. The previous bound then follows using the triangle inequality.

[1] The authors of [52] prove the reduction step from sEUF-NMA to EUF-CMA using a slightly more general problem called *claw with random function problem*. This is equivalent to the sEUF-NMA game for HaS. Moreover, it is easy to show that $\mathsf{Adv}^{\text{sEUF-NMA}}_{\mathsf{HaS}}(\mathcal{C}) \leq \mathsf{q_H}\mathsf{Adv}^{\mathsf{OW}}_{\mathsf{T}}(\mathcal{B})$

The rest of the proof follows from that of PSFs. Averaging over $(\mathsf{F}, \mathsf{I}) \leftarrow_{\$} \mathsf{KGen}(1^{\lambda})$, the real execution differs from the simulated one only by the signing oracle simulation. Since there are q_{S} queries to the singing oracle, we obtain the claimed bound. □

Remark. Notice that for a (A)PSF it holds that $\mathsf{Adv}^{\mathrm{OW}}_{\mathsf{T}}(\mathcal{A}) = \mathsf{Adv}^{\mathrm{INV}}_{\mathsf{T}}(\mathcal{B})$. In fact, $\mathcal{A}$ can simulate the INV game (Definition 2.4) by giving the uniformly distributed $y \leftarrow \mathsf{F}(x)$ to $\mathcal{B}$. On the other hand, $\mathcal{B}$ can simulate the OW game (Definition 2.3) by giving the random $y \leftarrow_{\$} \mathcal{Y}$ to $\mathcal{A}$.

The previous security proofs are valid only when we consider (A)PSF. In fact, if we consider a generic trapdoor function, the previous technique to simulate a signing query presents two issues.

- The simulated signature distribution and the real signature distribution may differ. In fact, in the real distribution, the signature is taken as the preimage of a random element in the codomain of the trapdoor function. So the real distribution is given by $\mathcal{D}_{\mathcal{X}} \sim \mathsf{I}(\mathcal{U}(\mathcal{Y}))$. In the simulated distribution the signature is sampled from a fixed distribution $\mathsf{SampDom}(\mathsf{F})$. To obtain a correct simulation, it is necessary to be able to characterize $\mathcal{D}_{\mathcal{X}}$ and choose $\mathsf{SampDom}(\mathsf{F}) \sim \mathcal{D}_{\mathcal{X}}$. In the case of a TDP, this is straightforward since $\mathcal{D}_{\mathcal{X}} \sim \mathcal{U}(\mathcal{X})$. But in general such a characterization is not possible.

- The output of the reprogrammed random oracle may not be uniformly distributed. During the simulation of a signing query, the random oracle is programmed as $\mathsf{F}(x)$ with $x \leftarrow_{\$} \mathsf{SampDom}(\mathsf{F})$. For a generic trapdoor function, the image of F is not necessarily uniform.

Sakumoto, Shirai, and Hiwatari [175], first tried to solve these problems for generic trapdoor functions by introducing the probabilistic Hash-and-Sign with retry paradigm. Their approach solves the first problem by using successive sampling of random salts, which ensures that the simulated distribution of preimages is indistinguishable from the real one. Unfortunately, as analysed in [55], this is not enough to solve the second issue.

In [134], the authors solved this problem, showing that the additional properties of (A)PSFs are not necessary to prove the EUF-CMA security in the (Q)ROM, provided that the trapdoor function satisfies the preimage sampleability notion of Definition 2.8. The same techniques are also applied in [63] to obtain a tighter result in the ROM. Since $\mathsf{F}(x)$ may not be uniformly distributed, it is not possible to show a general reduction from OW. Instead, in [134] a reduction from INV to EUF-CMA is proved[2].

[2] Recall that if the image of F is uniform, the notions of OW and INV are equivalent.

Theorem 2.13 ([134])**.** *Let* T *be a TDF. Let* $\mathcal{A}$ *be a* EUF-CMA *adversary against the* PHaSwR *scheme on* T *in the random oracle model, which makes* q_S *signing queries and* q_H *queries to the random oracle. Then, there exists a* INV *adversary* $\mathcal{B}$ *against* T *and a* PS *adversary* $\mathcal{D}$ *against* T *issuing* q_S *queries, such that*

$$\mathrm{Adv}^{\mathrm{EUF\text{-}CMA}}_{\mathrm{PHaSwR}}(\mathcal{A}) \leq (q_H + 1)\mathrm{Adv}^{\mathrm{INV}}_{\mathrm{T}}(\mathcal{B}) + \mathrm{Adv}^{\mathrm{PS}}_{\mathrm{T}}(\mathcal{D}) + q'_S \frac{q'_S + q_H + 1}{|\mathrm{R}|} + (q_H + 1)\frac{q'_S - q_S}{|\mathrm{R}|},$$

where q'_S *is a bound on the total number of queries to* H *in all the signing queries.*

Proof Idea. The main idea of the general case is to modify the EUF-CMA game such that in OSign the salt r is chosen uniformly at random in R and the preimage is generated by $x \leftarrow_{\$} \mathsf{SampDom}(\mathsf{F})$ instead of repeatedly sampling the random salt until $\mathsf{I}(\mathsf{H}(r, m)) \neq \perp$. The PS adversary $\mathcal{D}$, can simulate the original game and the modified game by either playing PS_0 or PS_1 (Experiment 2.2). The advantage in distinguishing the two games can therefore be estimated with $\mathrm{Adv}^{\mathrm{PS}}_{\mathrm{T}}(\mathcal{D})$. Modifying the EUF-CMA game requires first to adaptively reprogram the random oracle H with $\mathsf{H}(r, m) \leftarrow y$ for randomly chosen (r, y) until it holds that $\mathsf{I}(y) \neq \perp$. Next, it is required to remove the reprogramming for failed retries. Finally, H is modified such that the adversary cannot make a query to the random oracle on a set of pre-selected salts. The rest of the proof follows as in the previous cases. □

Remark. The previous result can be extended to the QROM by employing relevant techniques [116, 9, 81, 35]. The incurred security loss in the QROM grows from $(q_H + 1)$ to $(2q_H + 1)^2$.

2.3 Post-Quantum Hash-and-Sign Schemes

In its classical form, the Full Domain Hash paradigm uses trapdoor permutations [174] to invert hash outputs, ensuring secure signature generation and verification. As outlined in the previous section, adapting this framework in the post-quantum setting, where no construction of one-way permutation are known, presents unique challenges.

Early attempts to extend the Hash-and-Sign paradigm with lattices can be traced back to the GGH proposal [111] and subsequent variants like NTRUSign [119]. In these proposals, the trapdoor consisted of a “good” lattice base. The signature of a message was then obtained by finding a lattice point near the message hash encoding in the space. This was one of the first attempts to construct HaS schemes without the use of trapdoor permutations. Unfortunately, these proposals lacked preimage sampleability and the choice of a preimage was deterministically linked to the private key. These flaws were exploited and led to key-recovery attacks [157, 86]. Gentry, Peikert, and Vaikuntanathan [108] proposed in 2008 a general approach, that will later be known as the GPV framework, to build provably secure Hash-and-Sign schemes based on PSFs (Definition 2.6). Numerous subsequent works have proposed practical instances of the GPV framework with optimized implementations [182, 151, 85, 88]. This line of research culminates with the NIST PQC selected algorithm Falcon [172], and with the recent proposal of Hawk [87] to the NIST additional call for post-quantum signatures [44].

Code-based schemes have also sought to adapt the FDH framework. The CFS scheme, introduced in [64], was one of the initial attempts to create such a scheme using high-rate Goppa codes. However, the security proof for this scheme was later invalidated [95] and the scheme's bit-security scales logarithmically with the key size, requiring impractically large public keys and long signature times for typical security parameters [97, 135]. Other proposals have explored different code families [14, 110, 138] but have also been broken [171, 153]. Additionally, the use of the rank metric has been investigated with the RankSign scheme [105], but it was also later broken in [72]. The use of PSFs, as seen in the GPV framework, have been instrumental in developing secure lattice-based Hash-and-Sign schemes. Unfortunately, it is not known how to build efficient PSFs from code-based assumptions. Nonetheless, [71] introduces the Wave family of functions, based on generalized $(U, U+V)$-codes, that extends this approach to code-based cryptography, recovering preimage sampling on average. This approach has been later formalized in [52] with the notion of APSFs (Definition 2.7).

For multivariate-based assumptions, it is not known how to build (A)PSFs, and ad-hoc version of the PHaSwR paradigm has been employed with multivariate signature schemes. Patarin introduced the Oil and Vinegar (OV) signature scheme [166] using quadratic polynomials with a particular hidden structure. The scheme was later broken in [131] and modified in the Unbalanced Oil and Vinegar scheme [130]. Two variants of UOV, Rainbow [79] and LUOV [33], were later submitted to the NIST competition and subsequently broken [77, 27]. Nevertheless, the assumption behind UOV maintained its security. Because it is considered a practical and conservative design, it has recently been employed as the basis of numerous schemes submitted to NIST additional call for post-quantum signatures [30, 115, 104, 190, 78, 31, 167]. Patarin also introduced the Hidden Field Equation (HFE) construction [165]. Schemes based on HFE, and particularly with the *Minus* and *Vinegar* modifiers (HFEv-) [130], constitute another important line of multivariate signature schemes [168, 80], including GeMSS [50]. Although the HFE assumption is not broken, a key recovered attack presented in [186] showed how the considered modifiers do not increase the security of the scheme, making the construction uncompetitive compared to UOV. In [175], Sakumoto, Shirai, and Hiwatari notices that both UOV and HFE lack preimage sampleability and that the original schemes are not provably secure in the ROM. They first proposed a security proof by considering a variant within the probabilistic Hash-and-Sign with retry paradigm. Although there is a flaw in their proof [55], this construction was later proved to be secure in the (Q)ROM by [134].

In the remainder of this chapter, we will provide a description of some post-quantum Hash-and-Sign schemes. In the next chapter, the proposed aggregate signature will be applied to these schemes in order to evaluate their signature compression and the applicability of the security proof.

2.3.1 Lattice-based Cryptography

A lattice is a subset $\mathcal{L}$ of points in $\mathbb{R}^n$ such that

1. $\mathcal{L}$ is an additive subgroup.
2. $\mathcal{L}$ is discrete. That is for every $\boldsymbol{x} \in \mathcal{L}$ there exists some $\gamma > 0$ such that $B(\boldsymbol{x}, \gamma) \cap \mathcal{L} = \{\boldsymbol{x}\}$.

Equivalently, a lattice can be described as the integer linear combination of independent vectors in $\boldsymbol{b}_1, \dots, \boldsymbol{b}_k \in \mathbb{R}^n$. Let $\mathbf{B} = [\boldsymbol{b}_1 \cdots \boldsymbol{b}_k] \in \mathbb{R}^{n \times k}$. The lattice generated by $\mathbf{B}$ is defined as

$$\mathcal{L}(\mathbf{B}) = \left\{ \mathbf{B}\boldsymbol{x} \;\middle|\; \boldsymbol{x} \in \mathbb{Z}^k \right\} = \left\{ \sum_{i=1}^{k} x_i \boldsymbol{b}_i \;\middle|\; x_i \in \mathbb{Z} \right\}$$

The notion of duality, for lattices, is closely related to the abstract definition of dual of a vector space.

Let $\Lambda \subset \mathbb{R}^n$ be a lattice. The *dual* of Λ is defined as

$$\Lambda^{\vee} := \left\{ \boldsymbol{x} \in \mathbb{R}^n \;\middle|\; \langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathbb{Z},\ \boldsymbol{y} \in \Lambda \right\}.$$

An interesting class of lattices, often used in applications, is that of q-ary lattices. A lattice Λ is said to be q-ary if it satisfies

$$q\mathbb{Z}^n \subseteq \Lambda \subseteq \mathbb{Z}^n$$

for some $q \in \mathbb{Z}$. Clearly, if Λ is q-ary, then for any $\boldsymbol{x} \in \mathbb{Z}^n$ we have that

$$\boldsymbol{x} \in \Lambda \iff \boldsymbol{x} \pmod{q} \in \Lambda.$$

We can define two particular classes of q-ary lattices, which often appear in cryptographic contexts.

Definition 2.14. Let $\mathbf{A} \in \mathbb{Z}^{m \times n}$. Define the q-ary lattices generated by the columns of $\mathbf{A}$ as

$$\Lambda_q^{\perp}(\mathbf{A}) = \left\{ \boldsymbol{y} \in \mathbb{Z}^n \;\middle|\; \mathbf{A}\boldsymbol{y} \equiv 0 \pmod{q} \right\}$$

It can be shown that, for $n > m$, the lattice $\Lambda_q(\mathbf{A})$ is a full-rank lattice in $\mathbb{Z}^n$. If, otherwise, $n \leq m$, the kernel of $\mathbf{A}$ will be the nullspace in most of the cases. So that $\Lambda_q^\perp(\mathbf{A}) = q\mathbb{Z}^n$ and $\det\left(\Lambda_q^\perp(\mathbf{A})\right) = q^n$.

Similarly, we can consider the lattice generated by the rows of $\mathbf{A}$.

Definition 2.15. Let $\mathbf{A} \in \mathbb{Z}^{m \times n}$. Define the q-ary lattices generated by the rows of $\mathbf{A}$ as

$$\Lambda_q(\mathbf{A}) = \left\{ \boldsymbol{y} \in \mathbb{Z}^n \;\middle|\; \boldsymbol{y} \equiv \mathbf{A}^\intercal \boldsymbol{s} \pmod{q}, \boldsymbol{s} \in \mathbb{Z}^m \right\}.$$

These two lattices are linked through the dual in the following relations.

$$\Lambda_q^\perp(\mathbf{A}) = q \cdot (\Lambda_q(\mathbf{A}))^\vee \qquad \text{and} \qquad \Lambda_q(\mathbf{A}) = q \cdot (\Lambda_q^\perp(\mathbf{A}))^\vee.$$

Hard Problems. Many computational problems have been studied on lattices; we will focus on those that are central to the GPV framework. Let us start by introducing the fundamental problem of finding the shortest vector in a lattice.

Definition 2.16 (SVP)**.** Let $\mathbf{B}$ be a lattice basis for $\mathcal{L} = \mathcal{L}(\mathbf{B})$. The *Shortest Vector Problem* asks to find a non-zero vector $\boldsymbol{x} \in \mathcal{L} \setminus \{0\}$, such that

$$\|\boldsymbol{x}\| = \lambda(\mathcal{L}) := \min_{\boldsymbol{v} \in \mathcal{L} \setminus \{0\}} \|\boldsymbol{v}\|.$$

There are no known algorithms to solve the exact version of SVP in polynomial time; for this reason, an approximate and weaker version of this problem has been developed. *Approximation problems* are widely used in lattice cryptography, and algorithms for these versions provide solutions which approximate the exact one within a factor $\gamma \geq 1$. The approximation factor is usually taken as a function of the lattice dimension n. The following is the approximate version of SVP:

Definition 2.17 (SVP$_\gamma$)**.** Let $\mathbf{B}$ be an n-dimensional lattice basis for $\mathcal{L} = \mathcal{L}(\mathbf{B})$. The *$\gamma$-approximate Shortest Vector Problem* asks to find a non-zero vector $\boldsymbol{x} \in \mathcal{L} \setminus \{0\}$ such that

$$\|\boldsymbol{x}\| \leq \gamma(n)\lambda(\mathcal{L}).$$

The problem underlying a GPV-based signature is the *Short Integer Solution* (SIS) problem, that was first introduced by Ajtai in 1996 [4].

Definition 2.18 (SIS Problem)**.** Given a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, find a non-zero integer vector $\boldsymbol{z} \in \mathbb{Z}^m$ of norm $\|\boldsymbol{z}\| \leq \beta$ such that

$$\mathbf{A}\boldsymbol{z} = 0 \in \mathbb{Z}_q^n.$$

The SIS problem can be seen as a SVP$_\gamma$ problem on the m-dimensional q-ary lattice $\Lambda_q^\perp(\mathbf{A})$ generated by $\mathbf{A}$. Starting from the work of Ajtai, a long sequence of results have established the hardness of the SIS problem, linking it to worst-case lattice problems [152, 108, 150].

Algorithm 2.2: Falcon Signature Scheme

```
$\mathsf{TrapGen}_{\mathsf{falc}}(1^\lambda)$:
  $\phi \leftarrow_\$ X^n + 1$
  Find short $f, g \in \mathbb{Z}[X]/\langle\phi\rangle$ such that
    $f$ is invertible $\mod q$
  Find corresponding $F, G \in \mathbb{Z}[X]/\langle\phi\rangle$
    such that $fG - gF = q \pmod{\phi}$
  $h \leftarrow g \cdot f^{-1} \pmod{q}$
  return $\mathsf{F}_{\mathsf{falc}} \leftarrow h, \mathsf{I}_{\mathsf{falc}} \leftarrow (f, g, F, G)$

$\mathsf{I}_{\mathsf{falc}}(t)$:
  $c_0 \leftarrow (\mathbf{S}^\top)^{-1} t$
  repeat
    $v \leftarrow_\$ \mathsf{ffSampling}(\mathbf{S}, c_0)$
    $(s_1, s_2) \leftarrow c_0 - v$
  until $\|(s_1, s_2)\| \le \beta$
  return $(s_1, s_2)$

$\mathsf{KGen}(1^\lambda)$:
  $(\mathsf{F}_{\mathsf{falc}}, \mathsf{I}_{\mathsf{falc}}) \leftarrow_\$ \mathsf{TrapGen}_{\mathsf{falc}}(1^\lambda)$
  return $(\mathsf{F}_{\mathsf{falc}}, \mathsf{I}_{\mathsf{falc}})$

$\mathsf{Sign}(\mathsf{I}_{\mathsf{falc}}, m)$:
  $r \leftarrow_\$ \mathsf{R}$
  $(s_1, s_2) \leftarrow_\$ \mathsf{I}_{\mathsf{falc}}(\mathsf{H}(r, m))$
  return $(r, (s_1, s_2))$

$\mathsf{Vrfy}(\mathsf{F}_{\mathsf{falc}} = h, m, \sigma = (r, (s_1, s_2)))$:
  $t \leftarrow \mathsf{H}(r, m)$
  $t' \leftarrow s_1 + h \cdot s_2$
  return $t = t'$.
```

The GPV Framework

At the core of the work of Gentry, Peikert, and Vaikuntanathan [108], there is a technique on how to use a *short* basis for an arbitrary lattice as a trapdoor for a HaS scheme. This idea had already been used in previous proposals such as GGH and NTRUSign [111, 119], but with a different approach that leaked information about the secret basis.

On a high level, the public key is described by a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, with $m \ge n$, that generates a q-ary lattice $\Lambda = \Lambda_q(\mathbf{A})$. $\mathbf{A}$ is generated together with a "short" trapdoor basis $\mathbf{S} \in \mathbb{Z}^{m \times m}$ such that $\mathbf{S} \cdot \mathbf{A} = 0 \pmod{q}$, so that $\mathbf{S}$ generates $\Lambda_q^\perp = \Lambda_q^\perp(\mathbf{S})$, the lattice orthogonal to Λ_q modulo q. The trapdoor public function is then given by the map $f_{\mathbf{A}} : \mathbb{Z}_q^m \to \mathbb{Z}_q^n$ where $f_{\mathbf{A}}(\boldsymbol{s}) = \mathbf{A}^\top \boldsymbol{s}$.

Given a target $\boldsymbol{t} \in \mathbb{Z}_q^n$ we need to find a "short" value $\boldsymbol{s} \in \mathbb{Z}_q^m$ such that $f_{\mathbf{A}}(\boldsymbol{s}) = \boldsymbol{t}$. To find $\boldsymbol{s}$, first a solution $\boldsymbol{c}_0 \in \mathbb{Z}_q^m$ for $f_{\mathbf{A}}(\boldsymbol{c}_0) = \mathbf{A}^\top \boldsymbol{c}_0 = \boldsymbol{t}$ is computed. Without imposing a constraint on the norm of $\boldsymbol{c}_0$, a solution can be found with gaussian elimination. Then, the secret basis $\mathbf{S}$ is used to find a lattice vector $\boldsymbol{v} \in \Lambda_q^\perp$ close to $\boldsymbol{c}_0$. The difference $\boldsymbol{s} = \boldsymbol{c}_0 - \boldsymbol{v}$ is still a solution since $\mathbf{A}^\top \boldsymbol{s} = \mathbf{A}^\top \boldsymbol{c}_0 - \mathbf{A}^\top \boldsymbol{v} = \boldsymbol{t} - 0$. Moreover, if $\boldsymbol{v}$ is close to $\boldsymbol{c}_0$, then the difference $\boldsymbol{s}$ is short.

One of the main contributions of [108] is the method to sample the near lattice vector $\boldsymbol{v}$ without leaking information about the trapdoor basis $\mathbf{S}$ using a randomized variant of Klein's nearest plane algorithm [132].

Falcon

To instantiate the GPV framework, it is required to choose a class of lattices that will be employed during the trapdoor generation. Falcon relies on the class of structured NTRU lattices, first introduced in [120]. The NTRU lattice is defined over the ring $R_q = \mathbb{Z}_q[X]/\langle\phi\rangle$ where $\phi = X^n + 1$ for some $n = 2^k$ a power-of-two and some prime modulus q. The trapdoor generation algorithm produces four small polynomials $f, g, F, G \in \mathbb{Z}[X]/\langle\phi\rangle$ such that f is invertible modulo q and

$$fG - gF = q \pmod{\phi}.$$

The public key can be reduced to a single polynomial $h \leftarrow g \cdot f^{-1} \pmod{q}$. Given h, the problem of finding small polynomials f', g' such that $h = g' \cdot (f')^{-1}$ constitutes the NTRU assumption.

Given these polynomials, it is possible to instantiate the GPV framework by taking the public basis for Λ_q

$$\mathbf{A} = \left[\begin{array}{c} \mathbf{I}_n \\ \hline h \end{array}\right],$$

where the matrix associated with h is an $n \times n$ matrix where the i-th rows is given by $X^i h \pmod{\phi}$. The corresponding private basis for $\Lambda_q^\perp$ is given by

$$\mathbf{S} = \left[\begin{array}{c|c} g & -f \\ \hline G & -F \end{array}\right]$$

Let $\mathsf{T}_{\text{falc}} = (\mathsf{TrapGen}_{\text{falc}}\mathsf{F}_{\text{falc}}, \mathsf{I}_{\text{falc}})$ be the PSF of Falcon. F_{falc} is given by $f_h \colon R_q \times R_q \to R_q$ such that $f_h(s_1, s_2) = s_1 + h \cdot s_2$. I_{falc} use the knowledge of the secret basis $\mathbf{S}$ to sample a pair (s_1, s_2) from a Gaussian distribution with standard deviation σ, such that $\|(s_1, s_2)\| \leq \beta$ for some short parameter β. This process is known as Fast Fourier Sampling. The key generation and the signing procedure are shown in Algorithm 2.2.

T_{falc} is the underlying trapdoor function of the Falcon scheme [172], which is a selected scheme to the NIST PQC Standardization process. The proposed parameters for Falcon are shown in Table 2.2.

Table 2.2: Proposed parameters for Falcon [172] with corresponding key/signature sizes.

Set	NIST SL	Parameters			|pk|	|sig|
		n	q	σ	(B)	(B)
`Falcon-512`	I	512	12289	165.737	897	666
`Falcon-1024`	V	1024	12289	168.389	1793	1280

2.3.2 Code-based Cryptography

A q-ary linear code $\mathfrak{C}$ of length n and dimension k is a k-dimensional subspace of $\mathbb{F}_q^n$. A linear code $\mathfrak{C}$ can be represented using a *generator matrix* $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ which has the code as image, i.e.,

$$\boldsymbol{v} \in \mathfrak{C} \iff \exists \boldsymbol{x} \in \mathbb{F}_q^k \text{ s.t. } \boldsymbol{v} = \boldsymbol{x}\mathbf{G}.$$

Alternatively, $\mathfrak{C}$ is usually represented using a *parity-check matrix* $\mathbf{H} \in \mathbb{F}_q^{n-k \times n}$ which has the code as kernel, i.e.,

$$\boldsymbol{v} \in \mathfrak{C} \iff \mathbf{H}\boldsymbol{v}^\intercal = 0.$$

Usually, linear codes are endowed with the metric induced by the Hamming weight.

Definition 2.19 (Hamming Weight)**.** Let $\boldsymbol{x} \in \mathbb{F}_q^n$. The *Hamming weight* of $\boldsymbol{x}$ is given by the number of its non-zero components, i.e.,

$$\mathrm{wt}(\boldsymbol{x}) = |\{i \in \{1, \dots, n\} \mid x_i \neq 0\}|.$$

The Hamming weight naturally induces a distance on $\mathbb{F}_q^n$ given by $\mathrm{d}(\boldsymbol{x}, \boldsymbol{y}) = \mathrm{wt}(\boldsymbol{x} - \boldsymbol{y})$.

Hard Problems. Linear codes were introduced to provide error correction capabilities. The goal is to be able to decode a message after is sent through a noisy communication channel, which adds some unknown error. With q-ary linear codes, a message $\boldsymbol{m}$ is an element of $\mathbb{F}_q^k$. After choosing a k-dimensional liner code $\mathfrak{C} \subset \mathbb{F}_q^n$ with generator matrix $\mathbf{G}$, the message is encoded as codeword $\boldsymbol{c} = \boldsymbol{m}\mathbf{G} \in \mathfrak{C}$. After $\boldsymbol{c}$ is sent through the communication channel, a receiver gets a vector $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e} \in \mathbb{F}_q^n$ for some unknown error $\boldsymbol{e} \in \mathbb{F}_q^n$. A decoder for $\mathfrak{C}$ should be able to remove the error and retrieve $\boldsymbol{c}$, and consequently the message $\boldsymbol{m}$. The associated problem comes with additional constraints on the weight of the error.

Algorithm 2.3: Wave Signature Scheme

```
$\mathsf{TrapGen}_{\mathsf{wave}}(1^\lambda)$:
    $\mathbf{H}_{\mathsf{sk}} \in \mathbb{F}_q^{(n-k)\times n} \leftarrow\$$ generalized $(U, U+V)$-code
    $\mathbf{S} \leftarrow\$ \mathrm{GL}_{n-k}(\mathbb{F}_q)$
    $\mathbf{P} \leftarrow\$ n \times n$ permutation matrix
    $\mathbf{H}_{\mathsf{pk}} \leftarrow \mathbf{S}\mathbf{H}_{\mathsf{sk}}\mathbf{P}$
    return $\mathsf{F}_{\mathsf{wave}} \leftarrow \mathbf{H}_{\mathsf{pk}}, \mathsf{I}_{\mathsf{wave}} \leftarrow (\mathbf{H}_{\mathsf{sk}}, \mathbf{S}, \mathbf{P})$

$\mathsf{I}_{\mathsf{wave}}(\boldsymbol{y})$:
    $\boldsymbol{e} \leftarrow\$ D_{\mathbf{H}_{\mathsf{sk}}}(\boldsymbol{y}(\mathbf{S}^\top)^{-1})$
    $\boldsymbol{x} \leftarrow \boldsymbol{e}\mathbf{P}$
    return $\boldsymbol{x}$

$\mathsf{KGen}(1^\lambda)$:
    $(\mathsf{F}_{\mathsf{wave}}, \mathsf{I}_{\mathsf{wave}}) \leftarrow\$ \mathsf{TrapGen}_{\mathsf{wave}}(1^\lambda)$
    return $(\mathsf{F}_{\mathsf{wave}}, \mathsf{I}_{\mathsf{wave}})$

$\mathsf{Sign}(\mathsf{I}_{\mathsf{wave}}, m)$:
    $r \leftarrow\$ \mathsf{R}$
    $\boldsymbol{x} \leftarrow\$ \mathsf{I}_{\mathsf{wave}}(\mathsf{H}(r, m))$
    return $(r, \boldsymbol{x})$

$\mathsf{Vrfy}(\mathsf{F}_{\mathsf{wave}} = \mathbf{H}_{\mathsf{pk}}, m, \sigma = (r, \boldsymbol{x}))$:
    $\boldsymbol{t} \leftarrow \mathsf{H}(r, m)$
    $\boldsymbol{t}' \leftarrow \boldsymbol{x}\mathbf{H}_{\mathsf{pk}}^\top$
    return $\boldsymbol{t} = \boldsymbol{t}'$.
```

Definition 2.20 (Decoding Problem)**.** Let $\mathfrak{C}$ be a linear code with generator matrix $\mathbf{G} \in \mathbb{F}_q^{k\times n}$. Given a target vector $\boldsymbol{y} \in \mathbb{F}_q^n$, and a distance t such that $\boldsymbol{y} = \boldsymbol{m}\mathbf{G} + \boldsymbol{e}$ for some $\boldsymbol{m} \in \mathbb{F}_q^k$ and $\boldsymbol{e} \in \mathbb{F}_q^n$ with $\mathrm{wt}(\boldsymbol{e}) = t$, find the error vector $\boldsymbol{e}$.

There is an equivalent problem expressed in terms of the parity-check matrix, which is typically the one considered in practice. Given a parity-check matrix $\mathbf{H}$ for $\mathfrak{C}$ and a target $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$ it is possible to compute its *syndrome* $\boldsymbol{s}^\top = \mathbf{H}\boldsymbol{y}^\top = \mathbf{H}\boldsymbol{c}^\top + \mathbf{H}\boldsymbol{e}^\top = \mathbf{H}\boldsymbol{e}^\top \in \mathbb{F}_q^{n-k}$.

Definition 2.21 (Syndrome Decoding Problem)**.** Let $\mathfrak{C}$ be a linear code with parity-check matrix $\mathbf{H} \in \mathbb{F}_q^{n-k\times n}$. Given a syndrome vector $\boldsymbol{s} \in \mathbb{F}_q^{n-k}$, and a distance t such that $\boldsymbol{s}^\top = \mathbf{H}\boldsymbol{e}^\top$ for some $\boldsymbol{e} \in \mathbb{F}_q^n$ with $\mathrm{wt}(\boldsymbol{e}) = t$, find the error vector $\boldsymbol{e}$.

The (Syndrome) Decoding Problem is NP-complete [26] and believed to be hard on average [187, 46]. For a recent study on the hardness of the Decoding Problem we point to [94].

A standard approach to building code-based signature schemes, is to find a class of codes $\mathcal{D}$ for which an efficient decoding algorithm is known. The private key will be a code $\mathfrak{C} \in \mathcal{D}$, while the public key will be a representative of $\mathfrak{C}$ given by the generator matrix $\mathbf{G}$ (or the parity-check matrix). Since the Decoding Problem is hard for a random code, for the scheme to be secure $\mathbf{G}$ needs to be *indistinguishable* from a random matrix. There are no general results on the hardness of this Distinguishing Problem, and many proposals based on different code families have been broken.

Table 2.3: Proposed parameters for Wave [15] with corresponding key/signature sizes.

Set	NIST SL	Parameters			|pk|	|sig|
		n	k	w	(B)	(B)
Wave822	I	8576	4288	7668	3,677,390	822
Wave1249	III	12544	6272	11226	7,867,598	1249
Wave1644	V	16512	8256	14784	13,632,308	1644

Wave

Wave [71] is a HaS signature scheme based on the family of the generalized $(U, U+V)$-codes. Let $\mathsf{T}_{\text{wave}} = (\mathsf{TrapGen}_{\text{wave}}, \mathsf{F}_{\text{wave}}, \mathsf{I}_{\text{wave}})$ be the TDF of Wave. The OW security of F_{wave} is based on the indistinguishability of $(U, U+V)$-codes from random codes and the Syndrome Decoding (SD) problem. The indistinguishability problem is NP-complete for large finite fields $\mathbb{F}_q$, while the SD problem is NP-hard for arbitrary finite fields. The trapdoor secret information is a random generalized $(U, U+V)$-code over $\mathbb{F}_q$ of length n and dimension $k = k_U + k_V$, described by its parity check matrix $\mathbf{H}_{\text{sk}} \in \mathbb{F}_q^{(n-k)\times n}$, an invertible matrix $\mathbf{S} \in \mathbb{F}_q^{(n-k)\times(n-k)}$ and a permutation matrix $\mathbf{P} \in \mathbb{F}_q^{n\times n}$. Using the underlying structure of the $(U, U+V)$-code, an efficient decoding algorithm $D_{\mathbf{H}_{\text{sk}}}$ is produced. The public function F_{wave} is obtained from the parity check matrix $\mathbf{H}_{\text{pk}} = \mathbf{S}\mathbf{H}_{\text{sk}}\mathbf{P}$. Let $S_{w,n}$ be the subset of vectors in $\mathbb{F}_q^n$ with Hamming weight w. The weight w is chosen such that the public function $\mathsf{F}_{\text{wave}} \colon \boldsymbol{e} \in S_{w,n} \mapsto \boldsymbol{e}\mathbf{H}_{\text{pk}}^{\intercal} \in \mathbb{F}_q^{n-k}$ is a surjection. To find a preimage for $\boldsymbol{y} \in \mathbb{F}_q^{n-k}$, the signer uses the decoding algorithm $D_{\mathbf{H}_{\text{sk}}}$ on $\boldsymbol{y}(\mathbf{S}^{\intercal})^{-1}$ to find $\boldsymbol{e} \in S_{w,n}$, and finally returns $\boldsymbol{e}\mathbf{P}$. The key generation and the preimage computation via I_{wave} are shown in Algorithm 2.3.

Wave can be described in the HaS *without* retry paradigm. Moreover, T_{wave} is an APSF [52], a weaker notion of PSF where the uniformity property on preimages is required to hold only on average (Definition 2.7). In particular, for any $(\mathsf{F}, \mathsf{I}) \xleftarrow{\$} \mathsf{TrapGen}_{\text{wave}}(1^{\lambda})$, consider the statistical distance $\varepsilon_{\mathsf{F},\mathsf{I}} = \Delta(\mathsf{SampDom}(\mathsf{F}), \mathsf{I}(\mathcal{U}(\mathcal{Y})))$. Then, it holds that $\mathbb{E}_{(\mathsf{F},\mathsf{I})}[\varepsilon_{\mathsf{F},\mathsf{I}}] \leq \varepsilon$, where ε is negligible in the security parameter λ. This condition can be used to bound the distinguishing advantage on PS with ε, obtaining $\mathsf{Adv}^{\text{PS}}_{\mathsf{T}_{\text{wave}}}(\mathcal{D}) \leq \mathsf{q}_{\mathsf{S}}\varepsilon$.

T_{wave} is the underlying trapdoor function of the Wave scheme [15] submitted to the NIST PQC Standardization of Additional Digital Signature. The proposed parameters for Wave are shown in Table 2.3.

2.3.3 Multivariate-based Cryptography

A *multivariate quadratic map* $\mathcal{P}\colon \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^m$ with m components and n variables is defined by m multivariate quadratic polynomials $p_1(\boldsymbol{x}), \dots, p_m(\boldsymbol{x})$ in n variables $\boldsymbol{x} = (x_1, \dots, x_n)$ with coefficients in a finite field $\mathbb{F}_q$. The evaluation of $\mathcal{P}$ at $\boldsymbol{v} \in \mathbb{F}_q^n$ is $\boldsymbol{t} = (t_1, \dots, t_m) \in \mathbb{F}_q^m$, where $t_i = p_i(\boldsymbol{v})$ for $i = 1, \dots, m$.

The main mathematical problem underlying the security of multivariate-based schemes lies in the hardness of solving a system of multivariate equations over a finite field. Typically, this problem is restricted to quadratic polynomials and is known as the Multivariate Quadratic (MQ) problem.

Definition 2.22 (MQ Problem)**.** Let $\mathcal{P}\colon \mathbb{F}_q^n \rightarrow \mathbb{F}_q^m$ be a multivariate quadratic map and let $\boldsymbol{t} \in \mathbb{F}_q^m$ be a *target vector*. Find a preimage $\boldsymbol{v} \in \mathbb{F}_q^n$ such that $\mathcal{P}(\boldsymbol{v}) = \boldsymbol{t}$.

The MQ problem is NP-hard over a finite field, and is believed hard also on average when $n \sim m$. The decision version of the MQ problem is also known to be NP-complete. To date, the best algorithms capable of solving the MQ problem for relevant instances use algebraic techniques based on Gröbner's bases. For an overview of the hardness of the MQ problem, we point to [24].

One of the foundational schemes of multivariate cryptography was proposed by Matsumoto and Imai (MI) in 1988 [147]. The MI scheme proposed for the first time the use of multivariate systems for public key encryption. In particular, the scheme introduced an innovative technique that is still the basis of many multivariate schemes today. The idea is to start with a quadratic map, called a *central map*, that is easy to invert and then to mask it to obtain an apparently random quadratic map. This approach can be traced to a broader class of constructions, called *bipolar*. The private key of a bipolar cryptosystem is given by the central map, a quadratic system $\mathcal{F}\colon \mathbb{F}_q^n \rightarrow \mathbb{F}_q^m$ of m polynomials and n variables that can be easily inverted, and two affine maps $\mathcal{S}\colon \mathbb{F}_q^m \rightarrow \mathbb{F}_q^m$ and $\mathcal{T}\colon \mathbb{F}_q^n \rightarrow \mathbb{F}_q^n$. The public key is the quadratic map $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$ obtained by the composition of the secret maps. With the knowledge of the secret maps is easy to invert $\mathcal{P}$ and, in principle, this construction can be employed for both encryption and signature schemes.

Since the public map of a bipolar system is not a random quadratic map, the security of the scheme cannot be directly reduced to the MQ problem. Instead, due to the presence of a hidden central map masked with affine maps, we also need to consider a variant of the *Isomorphism of Polynomials* (IP) problem. In the following, we consider the extended version of this problem where the central map is not known, as it is the one relevant for UOV-based schemes.

Definition 2.23 (EIP Problem)**.** Let $\mathfrak{C}$ be a class of nonlinear multivariate systems and let $\mathcal{F}\colon \mathbb{F}_q^n \rightarrow \mathbb{F}_q^m \in \mathfrak{C}$. Given $\mathcal{P}\colon \mathcal{F}_q^n \rightarrow \mathbb{F}_q^m$ such that $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$ for some affine maps $\mathcal{S}, \mathcal{T}$, find a decomposition $\mathcal{P} = \mathcal{S}' \circ \mathcal{F}' \circ \mathcal{T}'$ with $\mathcal{S}', \mathcal{T}'$ affine maps and $\mathcal{F}' \in \mathfrak{C}$.

Similarly to the Distinguishing Problem for codes, there are no known results on the difficulty of solving the (E)IP problem. For this reason, bivariate schemes do not enjoy direct reduction to difficult problems and require ad-hoc assumptions.

UOV Trapdoor Function

The Unbalanced Oil and Vinegar (UOV) signature scheme is based on the bivariate construction, so that there is a hidden central map $\mathcal{F}\colon \mathbb{F}_q^n \to \mathbb{F}_q^m$ which is easy to invert. In the original proposal [166], called only Oil and Vinegar, the map was chosen with $n = 2m$. Subsequently, this construction was broken in [131] and the current constructions use $n > 2m$. The UOV signature scheme is built following the probabilistic Hash-and-Sign with retry paradigm. The use of the random salt r with resampling was introduced by [175] to recover EUF-CMA security. As discussed in the previous section, the security proof of [175] has a flaw that was later fixed in [134].

Instead of defining the central map, for the description of the UOV trapdoor function we mainly use the formalism introduced by Beullens in [28]. The trapdoor secret information is a linear subspace $O \subset \mathbb{F}_q^n$ of dimension $\dim(O) = m$. The trapdoor public function is a multivariate quadratic map $\mathcal{P}\colon \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^m$ that vanishes on O, that is:

$$\mathcal{P}(\boldsymbol{o}) = 0, \qquad \text{for all } \boldsymbol{o} \in O.$$

In principle, key pair generation consists of choosing uniformly at random an m-dimensional subspace $O \subset \mathbb{F}_q^n$ and then choosing uniformly at random a multivariate quadratic map $\mathcal{P}\colon \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^m$ that vanishes on O. In practice, a large part of $\mathcal{P}$ can be chosen randomly before choosing O, for example by deterministically expanding a seed to reduce the size of the public key, following the approach of [170].

For a multivariate quadratic polynomial p we can define its *polar form*:

$$p'(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{x} + \boldsymbol{y}) - p(\boldsymbol{x}) - p(\boldsymbol{y}) + p(0).$$

Similarly, for a multivariate quadratic map $\mathcal{P}(\boldsymbol{x}) = \big(p_1(\boldsymbol{x}), \dots, p_m(\boldsymbol{x})\big)$, its polar form is defined as $\mathcal{P}'(\boldsymbol{x}, \boldsymbol{y}) = \big(p_1'(\boldsymbol{x}, \boldsymbol{y}), \dots, p_m'(\boldsymbol{x}, \boldsymbol{y})\big)$. It can be shown that the polar form of a multivariate quadratic map is a symmetric and bilinear map. Given a target $\boldsymbol{t} \in \mathbb{F}_q^m$, we can use the secret information O to find a preimage $\boldsymbol{s} \in \mathbb{F}_q^n$ reducing the MQ problem to a linear system. In detail, one randomly chooses a vector $\boldsymbol{v} \in \mathbb{F}_q^n$ and solves $\mathcal{P}(\boldsymbol{v} + \boldsymbol{o}) = \boldsymbol{t}$ for $\boldsymbol{o} \in O$. Since

$$\boldsymbol{t} = \mathcal{P}(\boldsymbol{v} + \boldsymbol{o}) = \underbrace{\mathcal{P}(\boldsymbol{v})}_{\text{fixed}} + \underbrace{\mathcal{P}(\boldsymbol{o})}_{=0} + \underbrace{\mathcal{P}'(\boldsymbol{v}, \boldsymbol{o})}_{\text{linear in } \boldsymbol{o}},$$

the system reduces to the linear system $\mathcal{P}'(\boldsymbol{v}, \boldsymbol{o}) = \boldsymbol{t} - \mathcal{P}(\boldsymbol{v})$ of m equations and

Algorithm 2.4: UOV Signature Scheme

```
$\mathsf{TrapGen}_{\mathrm{uov}}(1^\lambda)$:
  $O \leftarrow_\$$ $m$-dimensional subspace of $\mathbb{F}_q^n$
  $\mathcal{P} \leftarrow_\$$ quadratic map $\mathbb{F}_q^n \to \mathbb{F}_q^m$ that vanishes on $O$
  return $\mathsf{F}_{\mathrm{uov}} \leftarrow \mathcal{P}, \mathsf{I}_{\mathrm{uov}} \leftarrow (\mathcal{P}, O)$

$\mathsf{I}_{\mathrm{uov}}(\boldsymbol{t})$:
  repeat
      $\boldsymbol{v} \leftarrow_\$ \mathbb{F}_q^n$
  until $\{\boldsymbol{o} \in O \mid \mathcal{P}'(\boldsymbol{v}, \boldsymbol{o}) = \boldsymbol{t} - \mathcal{P}(\boldsymbol{v})\} \neq \emptyset$
  $\boldsymbol{o} \leftarrow_\$ \{\boldsymbol{o} \in O \mid \mathcal{P}'(\boldsymbol{v}, \boldsymbol{o}) = \boldsymbol{t} - \mathcal{P}(\boldsymbol{v})\}$
  $\boldsymbol{x} \leftarrow \boldsymbol{v} + \boldsymbol{o}$
  return $\boldsymbol{x}$
```

```
$\mathsf{KGen}(1^\lambda)$:
  $(\mathsf{F}_{\mathrm{uov}}, \mathsf{I}_{\mathrm{uov}}) \leftarrow_\$ \mathsf{TrapGen}_{\mathrm{uov}}(1^\lambda)$
  return $(\mathsf{F}_{\mathrm{uov}}, \mathsf{I}_{\mathrm{uov}})$

$\mathsf{Sign}(\mathsf{I}_{\mathrm{uov}}, m)$:
  $r \leftarrow_\$ \mathsf{R}$
  $\boldsymbol{x} \leftarrow_\$ \mathsf{I}_{\mathrm{uov}}(\mathsf{H}(r, m))$
  return $(r, \boldsymbol{x})$

$\mathsf{Vrfy}(\mathsf{F}_{\mathrm{uov}} = \mathcal{P}, m, \sigma = (r, \boldsymbol{x}))$:
  $\boldsymbol{t} \leftarrow \mathsf{H}(r, m)$
  $\boldsymbol{t}' \leftarrow \mathcal{P}(\boldsymbol{x})$
  return $\boldsymbol{t} = \boldsymbol{t}'$.
```

m variables $\boldsymbol{o}$. Notice that whenever the linear map $\mathcal{P}'(\boldsymbol{v}, \cdot)$ is non-singular[3], the system has a unique solution $\boldsymbol{o} \in O$ and the preimage is $\boldsymbol{s} = \boldsymbol{v} + \boldsymbol{o}$. If the linear map is singular, one can simply repeat by choosing a new random value for $\boldsymbol{v}$.

Original Unbalanced Oil and Vinegar

Let $\mathsf{T}_{\mathrm{uov}} = (\mathsf{TrapGen}_{\mathrm{uov}} \mathsf{F}_{\mathrm{uov}}, \mathsf{I}_{\mathrm{uov}})$ be the TDF based on the description of the previous section. Unbalanced Oil and Vinegar (UOV) [130] is a PHaS signature scheme based on $\mathsf{T}_{\mathrm{uov}}$. The key generation and the signing procedure with the trapdoor functions are shown in Algorithm 2.4.

In the original version of the UOV signature, the signer samples a random salt $r \leftarrow_\$ \{0,1\}^\lambda$ and repeatedly samples $\boldsymbol{v} \leftarrow_\$ \mathbb{F}_q^n$ until there is a solution to the linear system $\mathcal{P}'(\boldsymbol{v}, \boldsymbol{o}) = \mathsf{H}(m, r) - \mathcal{P}(\boldsymbol{v})$. Notice that since r is fixed, the UOV signature lies in the PHaS *without* retry paradigm. Moreover, a preimage sampled with $\boldsymbol{x} \leftarrow_\$ \mathsf{I}_{\mathrm{uov}}(\boldsymbol{y})$ for $\boldsymbol{y} \leftarrow_\$ \mathbb{F}_q^m$ is not uniformly distributed in $\mathbb{F}_q^n$. Therefore, to apply Theorem 2.13 we need to assume the preimage sampleability of $\mathsf{T}_{\mathrm{uov}}$, i.e. $\mathsf{Adv}^{\mathsf{PS}}_{\mathsf{T}_{\mathrm{uov}}}(\mathcal{D})$ must be negligible.

$\mathsf{T}_{\mathrm{uov}}$ is the underlying trapdoor function of the UOV scheme [31] submitted to the NIST PQC Standardization of Additional Digital Signature. The proposed parameters for UOV are shown in Table 2.4.

Algorithm 2.5: Provable UOV Signature Scheme

```
$\mathsf{TrapGen}_{\mathrm{puov}}(1^\lambda)$:
    $O \leftarrow\$$ $o$-dimensional subspace of $\mathbb{F}_q^n$
    $\mathcal{P} \leftarrow\$$ quadratic map $\mathbb{F}_q^n \to \mathbb{F}_q^m$ that
    vanishes on $O$
    return $\mathsf{F}_{\mathrm{puov}} \leftarrow \mathcal{P}, \mathsf{I}_{\mathrm{puov}} \leftarrow (\mathcal{P}, O)$

$\mathsf{I}^1_{\mathrm{puov}}()$:
    $\boldsymbol{v} \leftarrow\$ \mathbb{F}_q^n$
    return $\boldsymbol{v}$

$\mathsf{I}^2_{\mathrm{puov}}(\boldsymbol{v}, \boldsymbol{t})$:
    if $\{\boldsymbol{o} \in O \mid \mathcal{P}'(\boldsymbol{v}, \boldsymbol{o}) = \boldsymbol{t} - \mathcal{P}(\boldsymbol{v})\} \neq \emptyset$
    then return $\perp$
    $\boldsymbol{o} \leftarrow\$ \{\boldsymbol{o} in O \mid \mathcal{P}'(\boldsymbol{v}, \boldsymbol{o}) = \boldsymbol{t} - \mathcal{P}(\boldsymbol{v})\}$
    $\boldsymbol{x} \leftarrow \boldsymbol{v} + \boldsymbol{o}$
    return $\boldsymbol{x}$

$\mathsf{KGen}(1^\lambda)$:
    $(\mathsf{F}_{\mathrm{puov}}, \mathsf{I}_{\mathrm{puov}}) \leftarrow\$ \mathsf{TrapGen}_{\mathrm{puov}}(1^\lambda)$
    return $(\mathsf{F}_{\mathrm{puov}}, \mathsf{I}_{\mathrm{puov}})$

$\mathsf{Sign}(\mathsf{I}_{\mathrm{puov}}, m)$:
    $\boldsymbol{v} \leftarrow\$ \mathsf{I}^1_{\mathrm{puov}}()$
    repeat
        $r \leftarrow\$ \mathsf{R}$
        $\boldsymbol{x} \leftarrow\$ \mathsf{I}^2_{\mathrm{puov}}(\boldsymbol{v}, \mathsf{H}(r, m))$
    until $\boldsymbol{x} \neq \perp$
    return $(r, \boldsymbol{x})$

$\mathsf{Vrfy}(\mathsf{F}_{\mathrm{puov}} = \mathcal{P}, m, \sigma = (r, \boldsymbol{x}))$:
    $\boldsymbol{t} \leftarrow \mathsf{H}(r, m)$
    $\boldsymbol{t}' \leftarrow \mathcal{P}(\boldsymbol{x})$
    return $\boldsymbol{t} = \boldsymbol{t}'$.
```

Provable Unbalanced Oil and Vinegar

By adopting the probabilistic HaS with retry paradigm, the UOV signature scheme can be proven EUF-CMA secure in the random oracle model [134]. Let $\mathsf{T}_{\mathrm{puov}} = (\mathsf{TrapGen}_{\mathrm{puov}}\mathsf{F}_{\mathrm{puov}}, \mathsf{I}_{\mathrm{puov}})$ be the TDF of the provable variant of UOV. To obtain uniform preimages over $\mathbb{F}_q^n$, the *provable* UOV (PUOV) signing procedure is slightly different from the generic one described in Algorithm 2.1. The signer starts by fixing a random $\boldsymbol{v} \leftarrow\$ \mathbb{F}_q^n$, then it repeatedly samples $r \leftarrow\$ \mathsf{R}$ until there is a solution to the linear system $\mathcal{P}'(\boldsymbol{v}, \boldsymbol{o}) = \mathsf{H}(m, r) - \mathcal{P}(\boldsymbol{v})$. Equivalently, the trapdoor $\mathsf{I}_{\mathrm{puov}}$ can be split in two distinct functions $\mathsf{I}^1_{\mathrm{puov}}$ and $\mathsf{I}^2_{\mathrm{puov}}$. The former is invoked only once and randomly chooses $\boldsymbol{v} \leftarrow\$ \mathbb{F}_q^n$. The latter is part of the repeat loop and tries to find a preimage $\boldsymbol{s}$ of the corresponding linear system. The key generation and the signing procedure with the modified trapdoor functions are shown in Algorithm 2.5.

With this procedure, the authors of [175] proved that the preimages produced from $\mathsf{Sign}(\mathsf{I}_{\mathrm{puov}}, \cdot)$ are indistinguishable from the output of $\mathsf{SampDom}(\mathsf{F}_{\mathrm{puov}})$, so that $\mathsf{Adv}^{\mathrm{PS}}_{\mathsf{T}_{\mathrm{puov}}}(\mathcal{D}) = 0$. The signing procedure start by sampling $\boldsymbol{v} \leftarrow\$ \mathsf{I}^1_{\mathrm{puov}}$ uniformly at random in $\mathbb{F}_q^n$. Suppose $\dim O = o \geq m$, once $\boldsymbol{v}$ is fixed, $\mathcal{P}'(\boldsymbol{v}, \cdot)$ restricted to O is a linear map from $\mathbb{F}_q^m$ to itself of rank at most m. Let i be the rank of $\mathcal{P}'(\boldsymbol{v}, \cdot)$, if H is a random oracle over $\mathbb{F}_q^m$, then after about q^{m-i} tries a random salt r is chosen such that $\mathcal{P}'(\boldsymbol{v}, \cdot) = \mathsf{H}(r, m)$ admits a solution. Then $\boldsymbol{o}$ is sampled from the q^{m-i}

[3] This happens with probability approximately $1 - 1/q$

Algorithm 2.6: MAYO Signature Scheme

```
$\mathsf{TrapGen}_{\mathsf{mayo}}(1^\lambda)$:
    $\mathbf{O} \leftarrow\!\!\$\; \mathbb{F}_q^{o\times(n-o)}$
    $O \leftarrow\!\!\$\; \mathrm{RowSpace}(\mathbf{OI}_o)$
    $\mathcal{P} \leftarrow\!\!\$$ quadratic map $\mathbb{F}_q^n \rightarrow \mathbb{F}_q^m$ that vanishes on $O$
    return $\mathsf{F}_{\mathsf{mayo}} \leftarrow \mathcal{P}, \mathsf{I}_{\mathsf{mayo}} \leftarrow (\mathcal{P}, \mathbf{O})$

$\mathsf{I}_{\mathsf{mayo}}(\boldsymbol{t})$:
    $\mathcal{P}^*(\boldsymbol{x}_1,\dots,\boldsymbol{x}_k) \leftarrow \sum_{i=1}^k \mathbf{E}_{i,i}\mathcal{P}(\boldsymbol{x}_i) + \sum_{1\le i<j\le k}\mathbf{E}_{i,j}\mathcal{P}'(\boldsymbol{x}_i,\boldsymbol{x}_j)$
    $\boldsymbol{v}_1,\dots,\boldsymbol{v}_k \leftarrow\!\!\$\; (\mathbb{F}_q^n \times 0_m)^k$
    if $\mathcal{P}^*(\boldsymbol{v}_1+\boldsymbol{o}_1,\dots,\boldsymbol{v}_k+\boldsymbol{o}_k)$ does not have full rank then
        return $\perp$
    $\boldsymbol{o}_1,\dots,\boldsymbol{o}_k \leftarrow\!\!\$\; \{\boldsymbol{o}_1,\dots,\boldsymbol{o}_k \in O \mid \mathcal{P}^*(\boldsymbol{v}_1+\boldsymbol{o}_1,\dots,\boldsymbol{v}_k+\boldsymbol{o}_k) = \boldsymbol{t}\}$
    $\boldsymbol{s} \leftarrow (\boldsymbol{v}_1+\boldsymbol{o}_1,\dots,\boldsymbol{v}_k+\boldsymbol{o}_k)$
    return $\boldsymbol{s}$
```

```
$\mathsf{KGen}(1^\lambda)$:
    $(\mathsf{F}_{\mathsf{mayo}}, \mathsf{I}_{\mathsf{mayo}}) \leftarrow\!\!\$\; \mathsf{TrapGen}_{\mathsf{mayo}}(1^\lambda)$
    return $(\mathsf{F}_{\mathsf{mayo}}, \mathsf{I}_{\mathsf{mayo}})$

$\mathsf{Sign}(\mathsf{I}_{\mathsf{mayo}}, m)$:
    repeat
        $r \leftarrow\!\!\$\; \mathsf{R}$
        $\boldsymbol{s} \leftarrow\!\!\$\; \mathsf{I}_{\mathsf{mayo}}(\mathsf{H}(r, m))$
    until $\boldsymbol{s} \neq \perp$
    return $(r, \boldsymbol{s})$

$\mathsf{Vrfy}(\mathsf{F}_{\mathsf{mayo}} = \mathcal{P}, m, \sigma = (r, \boldsymbol{s}))$:
    $\mathcal{P}^*(\boldsymbol{x}_1,\dots,\boldsymbol{x}_k) \leftarrow \sum_{i=1}^k \mathbf{E}_{i,i}\mathcal{P}(\boldsymbol{x}_i) + \sum_{1\le i<j\le k}\mathbf{E}_{i,j}\mathcal{P}'(\boldsymbol{x}_i,\boldsymbol{x}_j)$
    $\boldsymbol{t} \leftarrow \mathsf{H}(r, m)$
    $\boldsymbol{t}' \leftarrow \mathcal{P}^*(\boldsymbol{s})$
    return $\boldsymbol{t} = \boldsymbol{t}'$.
```

solutions in O and $\boldsymbol{x} = \boldsymbol{v} + \boldsymbol{o}$ is uniformly distributed over $\mathbb{F}_q^n$.

$\mathsf{T}_{\mathsf{puov}}$ is the underlying trapdoor function of the PROV scheme [115] submitted to the NIST PQC Standardization of Additional Digital Signature. The parameters of PROV are selected so that the dimension of the trapdoor subspace is $o = m + \delta$. This choice significantly reduces the probability that the rank of $\mathcal{P}'(\boldsymbol{v}, \cdot)$ is smaller than m, reducing the number of retries. The proposed parameters for PROV are shown in Table 2.5.

MAYO

MAYO [29] is a PHaS signature scheme based on the UOV trapdoor function that employs a new technique to use a smaller secret subspace O of dimension $\dim(O) = o < m$. Let $\mathsf{T}_{\mathsf{mayo}} = (\mathsf{TrapGen}_{\mathsf{mayo}}, \mathsf{F}_{\mathsf{mayo}}, \mathsf{I}_{\mathsf{mayo}})$ be the TDF of MAYO. The key generation process is the same as for UOV and produces a multivariate quadratic map $\mathcal{P}\colon \mathbb{F}_q^n \rightarrow \mathbb{F}_q^m$ that vanishes on a subspace $O \subset \mathbb{F}_q^n$ with $\dim O = o$. In the signing procedure, $\mathcal{P}$ is deterministically transformed into a larger (*whipped*) map $\mathcal{P}^*\colon \mathbb{F}_q^{kn} \rightarrow \mathbb{F}_q^{m}$, for some $k > 1$, which vanishes on $O^k \subset \mathbb{F}_q^{kn}$ of dimension $ko \geq m$. In [29], the whipping transformation is obtained by choosing $k(k+1)/2$

random invertible matrices $\{\, \mathbf{E}_{i,j} \in \mathrm{GL}_m(\mathbb{F}_q) \,\}_{1 \le i \le j \le k}$ and defining

$$\mathcal{P}^*(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k) = \sum_{i=1}^{k} \mathbf{E}_{i,i}\, \mathcal{P}(\boldsymbol{x}_i) + \sum_{1 \le i < j \le k} \mathbf{E}_{i,j}\, \mathcal{P}'(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

Similarly to UOV, to find a preimage for $\boldsymbol{t} \in \mathbb{F}_q^m$, we randomly choose $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k \in \mathbb{F}_q^{n-m} \times 0_m$. Then, $\mathcal{P}^*(\boldsymbol{v}_1 + \boldsymbol{o}_1, \ldots, \boldsymbol{v}_k + \boldsymbol{o}_k) = \boldsymbol{t}$ is a system of m linear equation in $ko \ge m$ variables, so it will be solvable with high probability. The key generation and the preimage computation via $\mathsf{I}_{\mathrm{mayo}}$ are shown in Algorithm 2.6.

Instead of evaluating the advantage $\mathsf{Adv}^{\mathrm{PS}}_{\mathsf{T}_{\mathrm{mayo}}}(\mathcal{D})$ of the PS adversary, we can use the result of [29, Lemma 2] that bounds the probability B that $\mathcal{P}^*(\boldsymbol{v}_1 + \boldsymbol{o}_1, \ldots, \boldsymbol{v}_k + \boldsymbol{o}_k)$ does not have full rank. It can be shown that if $\mathsf{I}_{\mathrm{mayo}}$ has never output $\perp$, then the preimages produced by $\mathsf{Sign}(\mathsf{I}_{\mathrm{mayo}}, \cdot)$ are indistinguishable from $\mathsf{SampDom}(\mathsf{F}_{\mathrm{mayo}})$.

$\mathsf{T}_{\mathrm{mayo}}$ is the underlying trapdoor function of the MAYO scheme [30] submitted to the NIST PQC Standardization of Additional Digital Signature. The proposed parameters for MAYO are shown in Table 2.6.

Table 2.4: Proposed parameters for UOV [31] with corresponding key/signature sizes.

Set	**NIST SL**	**Parameters**			**|pk|**	**|sig|**
		n	m	q	(B)	(B)
`ov-Ip`	I	112	44	256	43,576	128
`ov-Is`	I	160	64	16	66,576	96
`ov-III`	III	184	72	256	189,232	200
`ov-V`	V	244	96	256	446,992	200

Table 2.5: Proposed parameters for PROV [115] with corresponding key/signature sizes.

Set	**NIST SL**	**Parameters**				**|pk|**	**|sig|**
		n	m	δ	q	(B)	(B)
`PROV-I`	I	142	49	8	256	81,045	166
`PROV-III`	III	206	74	8	256	251,894	238
`PROV-V`	V	270	100	8	256	588,696	310

Table 2.6: Proposed parameters for MAYO [30] with corresponding key/signature sizes.

Set	**NIST SL**	**Parameters**					**|pk|**	**|sig|**
		n	m	o	k	q	(B)	(B)
MAYO_1	I	66	64	8	9	16	1168	321
MAYO_2	I	78	64	18	4	16	5488	180
MAYO_3	III	99	96	10	11	16	2656	577
MAYO_5	V	133	128	12	12	16	5008	838

Chapter 3

History-Free Sequential Aggregation of Hash-and-Sign Signatures

In this chapter, we address the extension of permutation-based SAS schemes to generic trapdoor functions, making them applicable to a broader range of post-quantum signatures. We start by presenting the original SAS scheme for trapdoor permutation of [143, 156], following up in Section 3.2 with an analysis on the possibility of extending these techniques to generic trapdoor functions. In particular, we argue that these simpler approaches are not viable without additional properties on trapdoor functions. As evidence, we show in Section 3.3 how two existing multivariate-quadratic-based aggregate signature schemes [92, 56] are universally forgeable when instantiated with UOV and discuss their lack of provable security.

In Section 3.4, we present a partial-signature history-free SAS scheme based on generic trapdoor functions. In a history-free SAS, introduced in [49], signers receive only the so-far aggregated signature without requiring previous users' public keys and messages. The partial-signature variant, initially presented in [58], reduces the amount of information the signer needs to receive from the previous one, but requires a final (public) aggregation step.

Our approach can be seen as a generalization of the work of Brogle, Goldberg, and Reyzin [49] for trapdoor permutations, adapting the encoding technique of [156, 91] to include trapdoor functions beyond permutations. The main novelty of our work is the extension of the probabilistic Hash-and-Sign with retry paradigm, introduced in Section 2.2, to sequential aggregate signature schemes. As a result, we are able to reduce the security of our scheme to the non-invertibility (Definition 2.4) of the trapdoor function and to the additional notion of Preimage Sampling (PS) indistinguishability (Definition 2.8). While this further notion may appear restrictive in the choice of trapdoor functions, it turns out to be quite natural in security proofs of post-quantum Hash-and-Sign schemes, as recently shown in [134].

In Section 3.5 we show that the scheme and security reduction can be further refined if the considered trapdoor functions feature the additional properties of (A)PSFs. Finally, in Section 3.6, we apply our scheme to MQ-based signature schemes, specifically UOV [130] and MAYO [29], and the code-based scheme Wave [71]. For each scheme, we evaluate its compression capabilities and review its PS security so that it can be covered in our security proof.

The results of this chapter are contained in [149] and have been presented at CT-RSA 2024.

3.1 Sequential Aggregation from Trapdoor Permutation

SAS schemes were originally introduced by Lysyanskaya et al. in [143] for trapdoor permutations with the FDH approach. The main intuition behind the aggregation process is to "embed" the previous aggregate signature into the new message to be signed in order for it to be recovered during verification. Using TDP, a public key is a permutation $\mathsf{pk}_i = \pi_i$ and the corresponding private key is $\mathsf{sk}_i = \pi_i^{-1}$. To sign a message m_1 with the FDH approach, the first signer computes $h_1 \leftarrow \mathsf{H}(\mathsf{pk}_1, m_1)$, for an opportune hash function over the domain of π_i, and the signature is $\Sigma_1 \leftarrow \pi_1^{-1}(h_1)$. Now, if the second signer wants to add a signature of a message m_2 on Σ_1, they compute $h_2 \leftarrow \Sigma_1 \oplus \mathsf{H}(\mathsf{pk}_1, \mathsf{pk}_2, m_1, m_2)$ and gets the aggregate signature $\Sigma_2 \leftarrow \pi_2^{-1}(h_2)$. The overall signed data is (m_1, m_2, Σ_2) and the verifier can recover $\Sigma_1 \leftarrow \pi_2(\Sigma_2) \oplus \mathsf{H}(\mathsf{pk}_1, \mathsf{pk}_2, m_1, m_2)$ and accepts if and only if $\pi_1(\Sigma_1) = \mathsf{H}(\mathsf{pk}_1, m_1)$. More in detail, the LMRS scheme is a tuple of three algorithms $\mathsf{LMRS} = (\mathsf{KGen}, \mathsf{AggSign}, \mathsf{AggVrfy})$:

- $\mathsf{KGen}(1^\lambda)$: takes as input a security parameter 1^λ and generates a key pair $(\mathsf{pk}, \mathsf{sk}) = (\mathsf{F}, \mathsf{I}) \leftarrow\!\!\$\ \mathsf{TrapGen}(1^\lambda)$.

- $\mathsf{AggSign}(\mathsf{sk}_i, m_i, L_{i-1}, \Sigma_{i-1})$: takes as input the secret key sk_i and the message m_i of the i-th user and the previous aggregate signature Σ_{i-1} for the full history $L_{i-1} = (\mathsf{pk}_1, m_1), \ldots, (\mathsf{pk}_{i-1}, m_{i-1})$ of public key, message pairs. If $i = 1$, it simply returns $\Sigma_1 \leftarrow\!\!\$\ \mathsf{I}_1(\mathsf{H}(\mathsf{pk}_1, m_1))$. Otherwise, the algorithm checks that $\mathsf{AggVrfy}(L_{i-1}, \Sigma_{i-1}) = 1$ and computes
$$\Sigma_i \leftarrow\!\!\$\ \mathsf{I}_i(\Sigma_{i-1} \oplus \mathsf{H}(L_{i-1} \cup \{(\mathsf{pk}_i, m_i)\})).$$

- $\mathsf{AggVrfy}(L_n, \Sigma_n)$: takes as input the full history $L_n = (\mathsf{pk}_1, m_1), \ldots, (\mathsf{pk}_n, m_n)$ of public key, message pairs, and an aggregate signature Σ_n. The algorithm recovers the previous signature Σ_{n-1} as follows:
$$\Sigma_{n-1} \leftarrow \mathsf{F}(\Sigma_n) \oplus \mathsf{H}(L_n).$$

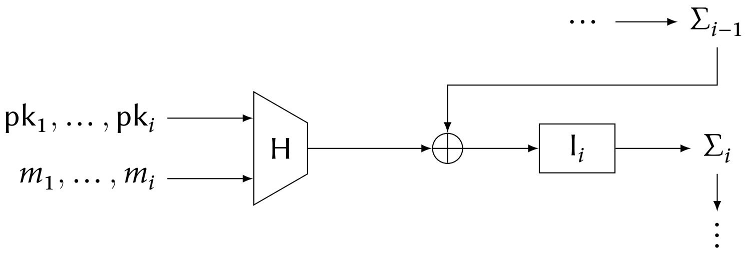


Figure 3.1: Simplified description of SAS scheme from [143]

> Next, the algorithm iterates the process until obtaining Σ_1. It returns 1 if and only if $\mathsf{F}_1(\Sigma_1) = \mathsf{H}(\mathsf{pk}_1, m_1)$ and 0 otherwise.

A simplified diagram of the aggregation process is shown in Figure 3.1.

This is a streamlined idea to provide an insight, to achieve a safe construction some additional steps are needed. The generic construction provided in [143] requires certified trapdoor permutation [23] for the security proof. A trapdoor permutation family is *certified* if for any $\mathsf{F}\colon \mathcal{X} \to \mathcal{X}$ is easy to determine whether F has been obtained as the output of TrapGen, ensuring that F is a permutation.

Remark. This is important when F can be generated by an adversary and the use of a malicious function may compromise the security of a cryptographic scheme. For instance, proving that a RSA public key (N, e) generated by a third party determines a permutation over $\mathbb{Z}_N^*$ is non-trivial [128].

This was later addressed in [156] where a new construction was presented, removing the requirement for certified permutations. Both schemes further require that each signer in the sequence must verify the current aggregate signature before adding their own. In particular, unlike the generic SAS scheme described in Section 1.3.5, here the aggregation algorithm AggSign also requires the messages and public keys of previous signers as input. This problem was investigated in [90, 49]. The latter proposed the first sequential aggregate signature scheme with *lazy verification*, in which verification is deferred until the final signature. As a consequence, the full-history of the aggregate signature, corresponding to the list of public-keys and messages of previous signers, are not required during aggregation. This approach is also known as *history-free* aggregation [98].

The main challenge in extending previous schemes to trapdoor functions that are not permutations lies in their lack of injectivity. This issue was addressed in [91] within the context of lattice-based signatures by employing an encoding technique derived from [156]. Subsequently, this idea was applied to MQ-based schemes instantiated with HFEv- [92] and UOV [56]. The proposed solution is to use a suitable encoding function, which splits the signature into two components. The first component can be injected into the codomain of the trapdoor function and subsequently made part of the computation of the aggregate signature, similar to the approach used in [143]. The second component is transmitted to the next signer and becomes part of the final aggregate signature. During the verification phase, this component is used to recover the partial aggregate signature through a corresponding decoding function. Notice that this method has a drawback in terms of the efficiency of the SAS scheme. In fact, storing part of the signature of each user without further aggregation causes a linear growth of the aggregate signature in the number of users. Therefore, it is currently unknown how to achieve sequential aggregate signatures of constant size in the post-quantum setting, where there are no one-way TDPs.

The proposed schemes employ concrete lattice-based and multivariate-based trapdoor functions, but for the latter no explicit use is made of unique features of these functions. In this regard, [91] can be considered an extension of [156] for PSFs, while [92, 56] an extension of [143] for generic TDFs, which only differ in the use of signers' public identities in the signature calculation. In Section 3.2 we will describe the construction of [92], which is slightly more general, and discuss the security of its approach in the case of generic TDFs, finally showing an explicit universal forgery of the scheme when instantiated with UOV in Section 3.3.

3.2 LMRS Scheme for Generic Trapdoor Functions

In this section, we adapt the Multivariate Quadratic SAS schemes from [92, 56] for the general case of TDFs and analyse its formal security. Both schemes are based on the variant with encoding of [143] and require an alternative definition of SAS with the notion of *full-history*: at each aggregation step, the signer needs the so-far aggregated signature and the complete list of messages and public keys of previous signers. Moreover, knowledge of the full description of the aggregate signature is required, as the signer needs to check its validity before adding its own.

Definition 3.1. A Full-History Sequential Aggregate Signature (FH-SAS) is a tuple of three algorithms $(\mathsf{KGen}, \mathsf{AggSign}, \mathsf{AggVrfy})$:

- $\mathsf{KGen}(1^\lambda)$: takes as input a security parameter 1^λ and generates a key pair $(\mathsf{pk}, \mathsf{sk})$.

- $\mathsf{AggSign}(\mathsf{sk}_i, m_i, L_{i-1}, \Sigma_{i-1})$: takes as input the secret key sk_i and the message m_i of the i-th user, a list $L_{i-1} = (\mathsf{pk}_1, m_1), \ldots, (\mathsf{pk}_{i-1}, m_{i-1})$ of public key, message pairs, and the previous aggregate signature Σ_{i-1}. If $\mathsf{AggVrfy}(L_{i-1}, \Sigma_{i-1}) = 1$, it returns an updated aggregate signature Σ_i.

- $\mathsf{AggVrfy}(L_n, \Sigma_n)$: takes as input the full history $L_n = (\mathsf{pk}_1, m_1), \ldots, (\mathsf{pk}_n, m_n)$ of public key, message pairs, and an aggregate signature Σ_n. Returns 1 if Σ_n is a valid aggregate signature and 0 otherwise.

Experiment 3.1: FH-UF-CMA$_{\mathsf{SAS}}$

```
$(\mathsf{pk}^\star, \mathsf{sk}^\star) \leftarrow_\$ \mathsf{KGen}(1^\lambda)$
$\mathcal{Q} \leftarrow \varnothing$
$(L_n, \Sigma_n) \leftarrow_\$ \mathcal{A}^{\mathsf{O},\mathsf{OAggSign}}(\mathsf{pk}^\star)$
$(\mathsf{pk}_1, m_1), \dots, (\mathsf{pk}_n, m_n) \leftarrow L_n$
if $\nexists i^\star : (\mathsf{pk}_{i^\star} = \mathsf{pk}^\star \wedge (m_{i^\star}, L_{i^\star}) \notin \mathcal{Q})$ then
    return $\perp$
return $\mathsf{AggVrfy}(L_n, \Sigma_n)$
```

```
$\mathsf{OAggSign}(m_i, L_{i-1}, \Sigma_{i-1})$:
if $\mathsf{AggVrfy}(L_{i-1}, \Sigma_{i-1}) = 0$ then
    return $\perp$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m_i, L_{i-1})\}$
$\Sigma_i \leftarrow_\$ \mathsf{AggSign}(\mathsf{sk}^\star, m_i, L_{i-1}, \Sigma_{i-1})$
return $\Sigma_i$
```

Below we show the definition of Full-History existential Unforgeability against Chosen-Message Attacks (FH-UF-CMA). Compared to the generic security notion for SAS described in Section 1.3.5, the signing oracle OAggSign requires sending the list L_{i-1} of public keys and messages along with the aggregate signature Σ_{i-1} and returns the updated signature if and only if Σ_{i-1} is valid.

Definition 3.2 (FH-UF-CMA Security)**.** Let O be a random oracle, let SAS = (KGen, AggSign, AggVrfy) be a FH-SAS scheme, let $\mathcal{A}$ be an adversary. We define the advantage of $\mathcal{A}$ playing the FH-UF-CMA game (Experiment 3.1) against SAS as follows:

$$\mathsf{Adv}^{\text{FH-UF-CMA}}_{\mathsf{SAS}}(\mathcal{A}) = \Pr[\text{FH-UF-CMA}_{\mathsf{SAS}}(\mathcal{A}) = 1].$$

We say that SAS is *full-history unforgeable against chosen message* attacks if the advantage $\mathsf{Adv}^{\text{FH-UF-CMA}}_{\mathsf{SAS}}(\mathcal{A})$ is negligible for any adversary $\mathcal{A}$.

3.2.1 The Scheme

The FH-SAS schemes of [92, 56] are instantiated with HFEv- and UOV, respectively, but no explicit use is made of unique features of these trapdoor functions. The description of Algorithm 3.1 refers to a generic trapdoor function T (as in Section 2.1) and is based on the construction of [92], which is slightly more general.

In Algorithm 3.1, the random oracle is $\mathsf{H}\colon \{0,1\}^* \to \mathcal{Y}$. The encoding function is $\mathsf{enc}\colon \mathcal{X} \to \mathcal{Y} \times \mathcal{X}'$ that splits an element x_i as $\mathsf{enc}(x_i) = (\alpha_i, \beta_i)$ and the corresponding decoding function is $\mathsf{dec}\colon \mathcal{Y} \times \mathcal{X}' \to \mathcal{X}$ such that $\mathsf{dec}(\mathsf{enc}(x)) = x$. To simplify the description, we will also use the notation $\alpha(x_i) = \alpha_i$ and $\beta(x_i) = \beta_i$, where $\alpha(\cdot), \beta(\cdot)$ are implicitly defined by enc.

A simplified diagram of the aggregation process is shown in Figure 3.2.

3.2.2 Provable Security

Both [92] and [56] provide a similar claim on the formal security of their sequential aggregate signature scheme. In the following, we are considering a generic trapdoor function, since its choice does not influence the security claim.

Algorithm 3.1: FH-SAS$_\mathsf{T}$ Scheme for Generic TDF

```
Let $\Sigma_0 = (\emptyset, \varepsilon)$.

KGen($1^\lambda$):
  $(\mathsf{F}, \mathsf{I}) \leftarrow\$ \mathsf{TrapGen}(1^\lambda)$
  return $\mathsf{pk} \leftarrow \mathsf{F}, \mathsf{sk} \leftarrow (\mathsf{F}, \mathsf{I})$

AggVrfy($L_n$, $\Sigma_n$):
  $(\mathsf{F}_1, m_1), \dots, (\mathsf{F}_n, m_n) \leftarrow L_n$
  $(\vec{\beta}_{n-1}, x_n) \leftarrow \Sigma_n$
  for $i \leftarrow n, \dots, 2$ do
      $L_i \leftarrow (\mathsf{F}_1, m_1), \dots, (\mathsf{F}_i, m_i)$
      $h_i \leftarrow \mathsf{H}(L_i)$
      $\alpha_{i-1} \leftarrow \mathsf{F}_i(x_i) \oplus h_i$
      $x_{i-1} \leftarrow \mathsf{dec}(\alpha_{i-1}, \beta_{i-1})$
  return $\mathsf{F}_1(x_1) = \mathsf{H}(\mathsf{F}_1, m_1)$

AggSign($(\mathsf{F}_i, \mathsf{I}_i)$, $m_i$, $L_{i-1}$, $\Sigma_{i-1}$):
  $(\vec{\beta}_{i-2}, x_{i-1}) \leftarrow \Sigma_{i-1}$
  if AggVrfy($L_{i-1}, \Sigma_{i-1}$) $= 0$ then
      return $\perp$
  $L_i \leftarrow L_{i-1} \cup \{(\mathsf{F}_i, m_i)\}$
  $(\alpha_{i-1}, \beta_{i-1}) \leftarrow \mathsf{enc}(x_{i-1})$
  $h_i \leftarrow \mathsf{H}(L_i)$
  $x_i \leftarrow\$ \mathsf{I}_i(\alpha_{i-1} \oplus h_i)$
  $\vec{\beta}_{i-1} \leftarrow \vec{\beta}_{i-2} \cup \{\beta_{i-1}\}$
  return $\Sigma_i \leftarrow (\vec{\beta}_{i-1}, x_i)$
```

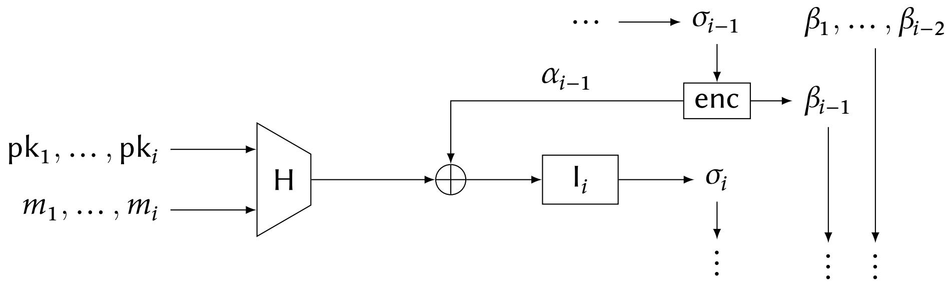


Figure 3.2: Simplified description of FH-SAS

Theorem 3.3 ([92]). *Let* T *be a trapdoor function. Let $\mathcal{A}$ be a* FH-UF-CMA *adversary against the* FH-SAS *scheme on* T *in the random oracle model, which makes* $\mathsf{q_S}$ *signing queries and* $\mathsf{q_H}$ *queries to the random oracle. Then, there exists an* OW *adversary $\mathcal{B}$ against* T *such that*

$$\mathsf{Adv}^{\mathsf{FH\text{-}UF\text{-}CMA}}_{\mathsf{FH\text{-}SAS_T}}(\mathcal{A}) \leq (\mathsf{q_S} + \mathsf{q_H} + 1) \cdot \mathsf{Adv}^{\mathsf{OW}}_{\mathsf{T}}(\mathcal{B})$$

and the running time of $\mathcal{B}$ is about that of $\mathcal{A}$.

In [56], the authors omit the proof for their security claim, while in [92] the authors provide a sketch of the proof in which they state that almost all the steps of the security proof follow [143] with only some slight modifications taking into account the use of the encoding function.

In the remaining of this section, we provide some insights into why the security proof of [143] cannot be applied to multivariate schemes, and more generally to signature schemes based on trapdoor functions that are not permutations. Then, in Section 3.3 we show an explicit universal forgery on FH-SAS when instantiated with the Unbalanced Oil and Vinegar signature scheme.

The proof for trapdoor permutations

To facilitate the discussion in the case of generic trapdoor functions, we give below a sketch of the proof of [143] for trapdoor permutations. We refer to the original paper for an in-depth discussion of the proof.

We show that if there exists a forger $\mathcal{A}$ for the aggregate signature scheme, then we can build a forger $\mathcal{B}$ that uses $\mathcal{A}$ to break the one-wayness assumption of the underlying trapdoor permutation. The goal for $\mathcal{B}$ is to find a preimage of a target public key $\mathsf{pk}^\star = \mathsf{F}^\star$ on a given input $y \in \mathcal{Y}$. $\mathcal{B}$ will use the resulting forged aggregate signature Σ of $\mathcal{A}$ on n messages $m_1, \dots, m_n$ under public keys $\mathsf{pk}_1, \dots, \mathsf{pk}_n$.

For simplicity, we will make the following assumptions on the forger $\mathcal{A}$:

- In the resulting forged aggregate signature, the target public key appears last, that is $\mathsf{pk}_n = \mathsf{pk}^\star$. Every adversary can be modified to achieve this property: in fact, suppose that $\mathsf{pk}^\star$ appears at position $i^\star < n$, then we apply the verification procedure of AggVrfy on $\Sigma = (\beta_1, \dots, \beta_{n-1}, \sigma_n)$ for $n - i^\star$ iterations to recover $\sigma_{i^\star}$ and output $\Sigma_{i^\star} = (\beta_1, \dots, \beta_{i^\star-1}, \sigma_{i^\star})$ as a new forgery on messages $m_1, \dots, m_{i^\star}$ under public keys $\mathsf{pk}_1, \dots, \mathsf{pk}_{i^\star}$ with $\mathsf{pk}_{i^\star} = \mathsf{pk}^\star$.
- The resulting forged aggregate signature Σ is valid, and the forger computes the verification procedure on Σ before returning it by making all the necessary queries to the random oracle. Moreover, we assume that the forger only makes queries for valid input data.

For the reduction, $\mathcal{B}$ supplies $\mathcal{A}$ with the target public key $\mathsf{pk}^\star$ and answers their oracle queries as follows.

Hash Queries $\mathcal{A}$ asks a query $Q = (\overrightarrow{\mathsf{pk}}, \vec{m})$ and expects an answer $\mathsf{H}(Q) = h \in \mathcal{Y}$. To answer on query Q, $\mathcal{B}$ maintains an originally empty list HT of tuples (w, r, c). If $\mathsf{HT}[Q] \neq \perp$, then it returns $\mathsf{H}(Q) \leftarrow w$. Otherwise, it sets $i = |\overrightarrow{\mathsf{pk}}| = |\vec{m}|$ and proceeds as follows:

- If $i > 1$, $\mathcal{B}$ gets the values $(w', r', c') \leftarrow \mathsf{HT}[\overrightarrow{\mathsf{pk}}_{i-1}, \vec{m}_{i-1}]$. If $i = 1$, it fixes $r' = \varepsilon$.
- If the challenge public key $\mathsf{pk}^\star$ does not belong to $\overrightarrow{\mathsf{pk}}$, $\mathcal{B}$ samples $r \leftarrow_\$ \mathcal{Y}$, computes $w \leftarrow \mathsf{F}_i(r) \oplus r'$ and sets $c \leftarrow \perp$.
- Otherwise, if $\mathsf{pk}^\star \in \overrightarrow{\mathsf{pk}}$ (assume for simplicity that $\mathsf{pk}_i = \mathsf{pk}^\star$), then for every query, with the exception of one randomly chosen among all $q_H + q_S + 1$

Hash Queries, $\mathcal{B}$ sample $r \leftarrow_{\$} \mathcal{Y}$, computes $w \leftarrow \mathsf{F}^\star(r) \oplus r'$ and sets $c \leftarrow 1$. In the only special case, $\mathcal{B}$ sets $r \leftarrow \perp$, computes $w \leftarrow y \oplus r'$ and sets $c \leftarrow 0$.

- Finally, $\mathcal{B}$ records $\mathsf{HT}[Q] \leftarrow (w, r, c)$ and returns $\mathsf{H}(Q) \leftarrow w$.

Aggregate Signature Queries $\mathcal{A}$ asks query $Q = (\overrightarrow{\mathsf{pk}}, \vec{m}, \sigma_{n-1})$ with $n = |\overrightarrow{\mathsf{pk}}| = |\vec{m}|$, $\mathsf{pk}_n = \mathsf{pk}^\star$ and σ_{n-1} a valid aggregate signature for messages $\vec{m}_{n-1}$ under public keys $\overrightarrow{\mathsf{pk}}_{n-1}$, and expects an answer σ_n which aggregates the message m_n with $\mathsf{pk}^\star$ on σ_{n-1}.

$\mathcal{B}$ use the previous hash algorithm on $Q' \leftarrow (\overrightarrow{\mathsf{pk}}, \vec{m})$ to obtain $(w, r, c) \leftarrow \mathsf{HT}[Q']$. If Q' is the special query with $c = 0$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ returns $\sigma_n \leftarrow r$.

Output Eventually, $\mathcal{A}$ will output a valid non-trivial aggregate signature σ_n on n messages $\vec{m}$ under public keys $\overrightarrow{\mathsf{pk}}$ with $\mathsf{pk}_n = \mathsf{pk}^\star$. The forgery will satisfy $\mathsf{F}^\star(\sigma_n) = t$ for some t produced in the hash algorithm. $\mathcal{B}$ can then examine the value $(w, r, c) \leftarrow \mathsf{HT}[(\overrightarrow{\mathsf{pk}}, \vec{m})]$. If $c = 1$ it aborts; otherwise, it returns σ_n as a preimage of $\mathsf{F}^\star$ on y. Observe that this preimage, produced without aborts with probability reduced by a factor of $q_H + q_S + 1$ with respect to the forging probability of $\mathcal{A}$, is correct since $\mathsf{F}^\star(\sigma_n) = \mathsf{H}(\overrightarrow{\mathsf{pk}}_n, \vec{m}_n) \oplus \sigma_{n-1}$ and, if $c = 0$, by construction we have $\mathsf{H}(\overrightarrow{\mathsf{pk}}, \vec{m}) = y \oplus \sigma_{n-1}$.

Attempt for generic trapdoor functions

In the following, we will attempt to adapt the previous proof to generic trapdoor functions, arguing how provable security cannot be achieved in this case.

Hash Queries Observe that for a correct simulation of $\mathcal{A}$'s view, $\mathcal{B}$ must answer with a uniformly random value in $\mathcal{Y}$. It is easy to see how this is verified in the case of trapdoor permutation F: $\mathcal{B}$ sample a uniformly random $x \leftarrow_{\$} \mathcal{X}$ and answers with $\mathsf{F}(x) \oplus z$, which is uniformly distributed in $\mathcal{Y} = \mathcal{X}$.

For a generic trapdoor function, we can not assume that the image of F is uniform. Therefore, to provide a correct simulation, we must answer with a uniformly random value in $\mathcal{Y}$. In particular, we modify the original procedure for TDPs by sampling $w \leftarrow_{\$} \mathcal{Y}$, recording $\mathsf{HT}[Q] \leftarrow w$ and finally returning $\mathsf{H}(Q) \leftarrow w$.

Observe that, if the trapdoor function is a PSF, the uniformity property on the image of F follows from Property 1 of Definition 2.6. We will argue in the analysis of the Aggregate Signature Queries that this property is still not enough to maintain the same approach applied to TDPs.

Aggregate Signature Queries In the case of generic TDFs, $\mathcal{A}$ asks query $Q = (\overrightarrow{\mathrm{pk}}, \vec{m}, \Sigma_{n-1})$ with $n = |\overrightarrow{\mathrm{pk}}| = |\vec{m}|, \mathrm{pk}_n = \mathrm{pk}^\star$ and $\Sigma_{n-1} = (\beta_1, \ldots, \beta_{n-2}, \sigma_{n-1})$ a valid aggregate signature for messages $\vec{m}_{n-1}$ under public keys $\overrightarrow{\mathrm{pk}}_{n-1}$.

In the original procedure for TDPs, $\mathcal{B}$ used the hash algorithm to prepare a valid response to the signature query. As we discussed, this approach requires the uniformity property of PSFs and is therefore not applicable to generic TDFs. On the other hand, we argue that this property alone is not sufficient. In fact, the correctness of the response to the aggregate signature query is based on the following fact of TDP-based constructions: fixed an input $(\overrightarrow{\mathrm{pk}}, \vec{m})$, there is a unique aggregate signature on messages $\vec{m}$ under public keys $\overrightarrow{\mathrm{pk}}$. Otherwise, if the aggregate signature is not unique, $\mathcal{B}$ is unable to provide a valid response to the aggregate signature query. In fact, on input Q, $\mathcal{B}$ would take $(w, r, c) \leftarrow \mathsf{HT}[(\overrightarrow{\mathrm{pk}}, \vec{m})]$, where $\mathsf{F}^\star(r) = r' \oplus \mathsf{H}(\overrightarrow{\mathrm{pk}}, \vec{m})$, but without the knowledge that r' is equal[1] to the aggregate signature Σ_{n-1} computed by $\mathcal{A}$ on messages $\vec{m}_{n-1}$ under public keys $\overrightarrow{\mathrm{pk}}_{n-1}$. Therefore, the aggregate signature produced by $\mathcal{B}$ may not be properly verified, invalidating the oracle simulation.

Since the previous aggregate signature is not part of the input of the hash function, it is not possible to construct an a priori uniquely determined signature in the hash algorithm, even in the case of PSFs. We are therefore restricted to the generic case in which $\mathcal{B}$ always responds to a Hash Query with a uniformly random value in $\mathcal{Y}$. In this case, it is still possible to answer the aggregate signature query correctly, as follows:

- From our simplified assumptions, the query is correct. Thus, $\mathcal{B}$ samples $r \leftarrow_\$ \mathcal{X}$ and computes $w \leftarrow \mathsf{F}^\star(r) \oplus \alpha(\sigma_{n-1})$.
- $\mathcal{B}$ checks that there was no previous Hash Query on input $Q' \leftarrow (\overrightarrow{\mathrm{pk}}, \vec{m})$, otherwise it aborts.
- Finally, $\mathcal{B}$ records $\mathsf{HT}[Q'] \leftarrow w$ and returns $\Sigma_n = (\beta_1, \ldots, \beta_{n-2}, \beta(\sigma_{n-1}), r)$.

Observe that in the described procedure $\mathcal{B}$ aborts whenever it receives an input in which $Q' = (\overrightarrow{\mathrm{pk}}, \vec{m})$ has been previously prompted by $\mathcal{A}$ to the hash oracle. This event cannot be ruled out or bounded by a probabilistic argument. Therefore, for this modified approach to be valid, it would be necessary to introduce a random salt during aggregation, similar to what was proposed in [49]. This way, the probability of $\mathcal{B}$ to abort can be arbitrarily bounded by varying the length of the salt.

[1] For simplicity, we are assuming that $\mathcal{B}$ constructed the response to the Hash Query in a way that takes into account the encoding function.

Output Eventually, $\mathcal{A}$ will output a valid non-trivial aggregate signature $\Sigma_n = (\beta_1, \dots, \beta_{n-1}, \sigma_n)$ on n messages $\vec{m}$ under public keys $\overrightarrow{\mathsf{pk}}$ with $\mathsf{pk}_n = \mathsf{pk}^\star$. Since the aggregate signature is correct, it follows that $\mathsf{F}^\star(\sigma_n) = \mathsf{H}(\overrightarrow{\mathsf{pk}}, \vec{m}) \oplus \alpha(\sigma_{n-1}) = t$. In the case of TDPs, we have shown that, with probability $(q_H + q_S + 1)^{-1}$ this t is equal to the target y and $\mathcal{B}$ can output Σ_n as a valid preimage of $\mathsf{F}^\star$ on y.

In the case of generic TDFs, we have seen that the previous approach is not valid, and it is necessary to modify the simulation of oracles as above. On the other hand, we claim that in this setting, it is not possible to correctly simulate the responses to the oracles to obtain a preimage of y. In fact, to obtain a valid preimage, $\mathcal{B}$ would need that $\mathsf{F}^\star(\sigma_n) = \mathsf{H}(\overrightarrow{\mathsf{pk}}, \vec{m}) \oplus \sigma_{n-1} = y$ and therefore $\mathsf{H}(\overrightarrow{\mathsf{pk}}, \vec{m}) = y \oplus \sigma_{n-1}$. $\mathcal{B}$ should then have been able to simulate the hash oracle in order to return $y \oplus \sigma_{n-1}$ on input $(\overrightarrow{\mathsf{pk}}, \vec{m})$. But it is not possible to provide this answer since σ_{n-1} is not part of the query input and is not uniquely determined.

3.3 Security of Existing Multivariate SAS Schemes

In this section, we show a universal forgery for the sequential aggregate signature schemes of Section 3.2.1 when instantiated with the UOV signature scheme.

3.3.1 Description of the Forgery

We recall that in UOV, the trapdoor function is a multivariate quadratic map $\mathcal{P}\colon \mathbb{F}_q^n \to \mathbb{F}_q^m$ that vanishes on a secret linear subspace $O \subset \mathbb{F}_q^n$ of dimension m. A more in-depth description can be found in Section 2.3.3.

In the following, we are assuming that the encoding function $\mathsf{enc}(\boldsymbol{x})$ can be expressed via an appropriate affine map and, accordingly, $\alpha(\boldsymbol{x}) = R(\boldsymbol{x}) = \mathbf{A}\boldsymbol{x} + \boldsymbol{b}$, where $\mathbf{A} \in \mathbb{F}_q^{m \times n}, \boldsymbol{b} \in \mathbb{F}_q^m$. In [92, 56], $\mathsf{enc}(\boldsymbol{x})$ is always the projection in the first m and the last $n - m$ components[2] of $\boldsymbol{x}$. This is a slight generalization that captures the intuition that there must be a corresponding efficient decoding function.

Lemma 3.4. *The* $\mathsf{FH\text{-}SAS_T}$ *scheme of Section 3.2.1, instantiated with UOV, is not* FH-UF-CMA.

Proof. Let $\mathsf{pk}_i = \mathcal{P}_i$ be the target public key and assume that the forger $\mathcal{F}$ knows a valid aggregate signature $\Sigma_i = (\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_{i-1}, \boldsymbol{x}_i)$ for an honest history $L_i = (\mathsf{pk}_1, m_1), \dots, (\mathsf{pk}_i, m_i)$. This is a typical attack environment, much weaker than the notion of FH-UF-CMA that we introduced in Definition 3.2. Then, $\mathcal{F}$ select a message $m_i^\star$ on which it will produce a forged signature for the target user.

[2] In this case we would have that $\alpha(\boldsymbol{x}) = \mathbf{A}\boldsymbol{x}$ with $\mathbf{A} = [\mathbf{I}_m \mid \mathbf{0}_{m,n-m}]$.

The forger $\mathcal{F}$ computes a forged signature by replacing the $(i-1)$-th honest signer, as follows:

1. First, $\mathcal{F}$ appropriately generates a UOV key pair $(\mathsf{pk}_{\mathcal{F}}, \mathsf{sk}_{\mathcal{F}}) = (\mathcal{P}_{\mathcal{F}}, O_{\mathcal{F}})$ by randomly choosing an m-dimensional linear subspace $O_{\mathcal{F}} \subset \ker \mathbf{A}$ and uses the same procedure of $\mathsf{TrapGen}_{\mathsf{uov}}$ (Algorithm 2.4) to sample $\mathcal{P}_{\mathcal{F}}$ that vanishes on $O_{\mathcal{F}}$.

2. Then, $\mathcal{F}$ arbitrarily chooses a message $m_{\mathcal{F}}$, computes a corresponding forged history $L^\star = L_{i-2} \cup \{(\mathsf{pk}_{\mathcal{F}}, m_{\mathcal{F}}), (\mathsf{pk}_i, m_i)\}$ and computes $\boldsymbol{\alpha}^\star \leftarrow \mathcal{P}_i(\boldsymbol{x}_i) \oplus \mathsf{H}(L^\star)$.

3. Finally, $\mathcal{F}$ finds a preimage $\boldsymbol{x}_{\mathcal{F}}$ under $\mathcal{P}_{\mathcal{F}}$ for $L_{\mathcal{F}} = L_{i-2} \cup \{(\mathsf{pk}_{\mathcal{F}}, m_{\mathcal{F}})\}$ such that
$$\mathcal{P}_{\mathcal{F}}(\boldsymbol{x}_{\mathcal{F}}) = \boldsymbol{\alpha}_{i-2} \oplus \mathsf{H}(L_{\mathcal{F}}) \qquad \text{and} \qquad \alpha(\boldsymbol{x}_{\mathcal{F}}) = \boldsymbol{\alpha}^\star. \tag{3.1}$$
Then $\Sigma^\star = (\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_{i-2}, \beta(\boldsymbol{x}_{\mathcal{F}}), \boldsymbol{x}_i)$ is a valid aggregate signature for the forged history $L^\star$.

Finding a preimage $\boldsymbol{x}_{\mathcal{F}}$ that satisfies Equation (3.1) is equivalent to finding a partially fixed preimage for $\mathcal{P}_{\mathcal{F}}$ under the map R. In particular, the forger can use the appropriately generated secret key $O_{\mathcal{F}}$ to restrict the preimage search to an appropriate affine subspace and guarantee the condition $R(\boldsymbol{x}_{\mathcal{F}}) = \boldsymbol{\alpha}^\star$. The forger searches for a preimage of $\boldsymbol{t} = \boldsymbol{\alpha}_{i-2} \oplus \mathsf{H}(L_{\mathcal{F}})$ by using a procedure similar to the Sign procedure described in Algorithm 2.5: on Line 1, instead of randomly sampling the vector $\boldsymbol{v}$ from $\mathbb{F}_q^n$, samples $\boldsymbol{v}$ from $\ker R'$, with $R'(\boldsymbol{x}) = \mathbf{A}\boldsymbol{x} + (\boldsymbol{b} - \boldsymbol{\alpha}^\star)$. Then, when a preimage $\boldsymbol{x}_{\mathcal{F}} \in \mathbb{F}_q^n$ of $\boldsymbol{t}$ is found, the forger would have $\boldsymbol{x}_{\mathcal{F}} \in \ker R'$, since $\boldsymbol{x}_{\mathcal{F}} = \boldsymbol{v} + \boldsymbol{o}$ with $\boldsymbol{v} \in \ker R'$ and $\boldsymbol{o} \in \ker \mathbf{A}$. Therefore, $\alpha(\boldsymbol{x}_{\mathcal{F}}) = R(\boldsymbol{x}_{\mathcal{F}}) = \boldsymbol{\alpha}^\star$.

We then show that $\Sigma^\star$ passes the verification correctly for the forged history $L^\star$:

1. The verifier applies the first iteration of AggVrfy (Algorithm 3.1) to recover the previous signature $\boldsymbol{x}_{\mathcal{F}}$ from $\boldsymbol{x}_i$ as follows:
$$\alpha(\boldsymbol{x}_{\mathcal{F}}) \leftarrow \mathcal{P}_i(\boldsymbol{x}_i) \oplus \mathsf{H}(L^\star) = \alpha^\star, \qquad \boldsymbol{x}_{\mathcal{F}} \leftarrow \mathsf{dec}(\alpha(\boldsymbol{x}_{\mathcal{F}}), \beta(\boldsymbol{x}_{\mathcal{F}}))$$

2. Since $\boldsymbol{x}_{\mathcal{F}}$ is a preimage of $\boldsymbol{\alpha}_{i-2} \oplus \mathsf{H}(L_{\mathcal{F}})$, the verifier correctly obtains $\boldsymbol{x}_{i-2}$ proceeding in the iterations of AggVrfy:
$$\boldsymbol{\alpha}_{i-2} \leftarrow \mathcal{P}_{\mathcal{F}}(\boldsymbol{x}_{\mathcal{F}}) \oplus \mathsf{H}(L_{\mathcal{F}}), \qquad \boldsymbol{x}_{i-2} \leftarrow \mathsf{dec}(\boldsymbol{\alpha}_{i-2}, \boldsymbol{\beta}_{i-2}).$$

3. The $(i-2)$-th signer was not tampered and, hence, the intermediate signature $\Sigma_{i-2} = (\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_{i-3}, \boldsymbol{x}_{i-2})$ can be correctly verified with AggVrfy on honest history L_{i-2}.

Therefore, the verifier determines the forged signature $\Sigma^\star$ as valid. □

3.3.2 Discussion

The previous forging procedure can be directly applied to constructions derived from [143] and instantiated via UOV, such as [92, 56]. In particular, we have shown how the existential unforgeability claims of [92, 56] are incorrect when the schemes are instantiated with UOV. The attack essentially involves finding a partially fixed preimage, following an adversarial key generation based on the public parameters of the aggregate signature scheme, specifically the encoding function. Although this attack may have applicability beyond UOV, it is not a universal forgery for generic trapdoor functions. However, this result aligns with the analysis of critical issues encountered when attempting to extend the security proof of [143] to generic trapdoor functions, as summarized in the following.

Programming the Random Oracle

In [143], the random oracle can be simulated to determine preimages for any permutation $\pi : \mathcal{X} \to \mathcal{X}$. This is typically achieved by sampling a uniformly random $x \leftarrow_{\$} \mathcal{X}$ and returning $\pi(x)$, which is uniformly distributed in $\mathcal{Y} = \mathcal{X}$. However, in the case of a generic trapdoor function, we cannot assume that the image of F is uniform. Consequently, to provide an accurate simulation, we must sample and return a uniformly random value in $\mathcal{Y}$.

Uniqueness of the Aggregate Signature

If we relax the previous condition and assume a uniformity property of the trapdoor functions[3] we may attempt to replicate the process described in [143] to answer the signing oracle. Indeed, on input $Q = (m^\star, L_{n-1}, \Sigma_{n-1})$ the simulator can use the knowledge of appropriate preimages for F_i to craft a valid aggregate signature on $L_{n-1} \cup \{(\mathsf{F}^\star, m^\star)\}$. However, we argue that this property alone is not sufficient for a correct simulation, which is instead based on the following fact of TDP-based constructions: for a fixed input $L_n = (\mathsf{F}_1, m_1), \dots, (\mathsf{F}_n, m_n)$ there exists a unique aggregate signature on L_n. Otherwise, if the aggregate signature is not unique, the simulator would be unable to provide a valid response to the aggregate signature query. In fact, on input Q, the simulator would take the preimage $x^\star$ for $\mathsf{F}^\star$ on $\alpha(x_{n-1}) \oplus \mathsf{H}(L_n)$ associated to the random oracle query on input $L_n = L_{n-1} \cup \{(\mathsf{F}^\star, m^\star)\}$. But, without the knowledge that x_{n-1} is equal to the preimage computed by the adversary on input L_{n-1}, the aggregate signature produced by the simulator may not be properly verified, resulting in an invalid response.

[3] For instance, this is the case for Trapdoor Preimage Sampleable Functions [108].

Reduction to OW

Eventually, the adversary will produce a valid non-trivial aggregate signature Σ_n on input $L_n = (\mathsf{F}_1, m_1), \dots, (\mathsf{F}_n, m_n)$, where we assume, for simplicity, that $\mathsf{F}_n = \mathsf{F}^\star$ is the target public key. Since the aggregate signature is correct, it follows that $\mathsf{F}^\star(x_n) = \alpha(x_{n-1}) \oplus \mathsf{H}(L_n) = y$. In the context of TDPs, [143] shows that y is equal to the target $y^\star$ of the OW game, with probability $(\mathsf{q}_\mathsf{H} + \mathsf{q}_\mathsf{S} + 1)^{-1}$.

When we consider generic TDFs, the previously mentioned approach is not valid, and it is necessary to modify the simulation of H by returning a fresh random value for each query. Moreover, we claim that in this setting, it is not possible to correctly simulate the response to the oracles in order to obtain a preimage of $y^\star$. In fact, to obtain a valid preimage, the simulator would require $\mathsf{F}^\star(x_n) = \alpha(x_{n-1}) \oplus \mathsf{H}(L_n) = y^\star$ and therefore $\mathsf{H}(L_n) = y^\star \oplus \alpha(x_{n-1})$. It should then have been able to simulate the random oracle to return $y^\star \oplus \alpha(x_{n-1})$ when given the input L_n. However, it is not possible to provide this answer, as x_{n-1} is not part of the query input and is not uniquely determined.

Fixing the Forging Vulnerability

The main vulnerability of FH-SAS concerns the overall malleability of the aggregate signature. In the original scheme for TDPs, once the input $L_n = (\mathsf{F}_1, m_1), \dots, (\mathsf{F}_n, m_n)$ is fixed, it was observed that there is a unique aggregate signature on messages $m_1, \dots, m_n$ under public keys $\mathsf{F}_1, \dots, \mathsf{F}_n$. Instead, in the extended version, uniqueness is lost due to the probabilistic nature of the inversion process. Consequently, it is always possible to construct two aggregate signatures on the same input, $\Sigma = (\beta_1, \dots, \beta_{i-1}, x_n)$ and $\Sigma' = (\beta_1, \dots, \beta_{i-1}, x_i')$, which differ only in the aggregation of the last signature. Furthermore, as shown in the forgery presented in Section 3.3.1, it is possible to have two aggregate signatures on the same input $\Sigma = (\beta_1, \dots, \beta_{i-1}, x_i)$ and $\Sigma' = (\beta_1, \dots, \beta_{i-1}', x_i)$ which differ only in the intermediate partial encodings. While the loss of uniqueness is unavoidable, it is possible to modify the scheme to prevent this additional form of malleability by making partial β encodings part of the random oracle input. We modify the aggregation step of $\mathsf{AggSign}((\mathsf{F}_i, \mathsf{I}_i), m_i, L_{i-1}, \Sigma_{i-1})$ (Algorithm 3.1): let $\Sigma_{i-1} = (\beta_1, \dots, \beta_{i-2}, x_{i-1})$ and compute

$$(\alpha_{i-1}, \beta_{i-1}) \leftarrow \mathsf{enc}(x_{i-1}), \qquad x_i \leftarrow \mathsf{I}_i(\alpha_{i-1} \oplus \mathsf{H}(L_i, \overrightarrow{\beta}_{i-1})),$$

where $L_i = L_{i-1} \cup \{(\mathsf{F}_i, m_i)\}$ and $\overrightarrow{\beta}_{i-1} = (\beta_1, \dots, \beta_{i-1})$.

Observe that now, once a new signature has been aggregated, it is no longer possible to modify the previous partial encodings while maintaining the validity of the aggregated signature. That is, if $\Sigma = (\beta_1, \dots, \beta_{i-1}, x_i)$ and $\Sigma' = (\beta_1, \dots, \beta_{i-1}', x_i')$ are valid aggregate signatures on the same input with $\beta_{i-1} \neq \beta_{i-1}'$, then $x_i \neq x_i'$. As a result, the forging procedure described in Section 3.3.1 is no longer applicable, as the adversary now needs to guess the partial encoding $\beta(x_{\mathcal{F}})$ of their own signature. However, in doing so, $\beta(x_{\mathcal{F}})$ becomes fixed and $\alpha^{\star}$ is not under the adversary's direct control. Once $\alpha^{\star}$ is computed, the entire signature $x_{\mathcal{F}}$ is fixed, and with high probability, it is not a valid signature.

This minor modification addresses the vulnerability exploited by our attack. However, from a provable security perspective, this construction presents similar problems to the original attempt to generalize [143]. As a result, we are unable to provide a formal proof of security.

3.4 Sequential Aggregation of Hash-and-Sign Signatures

In this section, we propose a partial-signature history-free sequential aggregate signature based on generic trapdoor functions. To obtain a secure instantiation, we only require the trapdoor function to be non-invertible (Definition 2.4) and preimage-sampleable (Definition 2.8). We argue that these are the weakest properties required to extend the sequential aggregate framework of [143, 156, 49] beyond trapdoor permutations and PSFs [91].

In the following, we introduce the notion of (partial-signature) history-free SAS and the relevant security model. Next, we introduce the scheme by giving a basic intuition and an algorithmic description of the aggregation process. The remainder of the section is devoted to the security proof of the scheme, where we provide a reduction from the non-invertibility and the preimage sampleability of the trapdoor function to the existential unforgeability of the SAS scheme.

3.4.1 History-Free Sequential Aggregate Signature

History-Free Sequential Aggregate Signatures were first introduced in [49, 98] as a variant of the original Full-History construction of [143] that does not require knowledge of previous messages and public keys in the aggregation step.

Definition 3.5. A History-Free Sequential Aggregate Signature (HF-SAS) is a tuple of three algorithms $(\mathsf{KGen}, \mathsf{AggSign}, \mathsf{AggVrfy})$:

- $\mathsf{KGen}(1^{\lambda})$: takes as input a security parameter 1^{λ} and generates a key pair $(\mathsf{pk}, \mathsf{sk})$.

- $\mathsf{AggSign}(\mathsf{sk}_i, m_i, \Sigma_{i-1})$: takes as input the secret key sk_i and the message m_i of the i-th user and the previous aggregate signature Σ_{i-1}. Returns an aggregate signature Σ_i.

Experiment 3.2: strong PS-HF-UF-CMA$_{\mathsf{SAS}}$

```
$(\mathsf{pk}^\star, \mathsf{sk}^\star) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$
$\mathcal{Q} \leftarrow \emptyset$
$(L_n, \bar{\Sigma}_n) \xleftarrow{\$} \mathcal{A}^{\mathsf{O},\mathsf{OAggSign}}(\mathsf{pk}^\star)$
$(\mathsf{pk}_1, m_1), \ldots, (\mathsf{pk}_n, m_n) \leftarrow L_n$
if $\nexists i^\star : (\mathsf{pk}_{i^\star} = \mathsf{pk}^\star \wedge (m_{i^\star}, \varsigma_{i^\star}) \notin \mathcal{Q})$ then
    return $\bot$
return $\mathsf{AggVrfy}(L_n, \bar{\Sigma}_n)$
```

```
$\mathsf{OAggSign}(m, \varrho)$:
    $(\varrho', \varsigma') \xleftarrow{\$} \mathsf{AggSign}(\mathsf{sk}^\star, m, \varrho)$
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \varsigma')\}$
    return $\varrho', \varsigma'$
```

- $\mathsf{AggVrfy}(L_n, \Sigma_n)$: takes as input the full history $L_n = (\mathsf{pk}_1, m_1), \ldots, (\mathsf{pk}_n, m_n)$ of public key, message pairs and an aggregate signature Σ_n. Returns 1 if Σ_n is a valid aggregate signature and 0 otherwise.

Note that the aggregation algorithm AggSign does not require the public keys and messages from the previous signers. Finally, the verifier can check the validity of the aggregate signature by running $\mathsf{AggVrfy}(L_n, \Sigma_n)$.

In the following, we define a slight modification of HF-SAS, as formalized in [48]. In this variant, the aggregation step requires only partial knowledge about the so-far aggregated signature. This description better captures the intuition behind the use of the encoding function and is better suited to our proposed scheme. During each aggregation step, the signer produces a partial signature information, which will be sent to the next signer, along with a complementary component. At the end of the aggregation sequence, an additional Combine step is performed, potentially by a third party. This step combines all the complementary information and the last signature of the sequence, resulting in the complete aggregated signature.

Definition 3.6. A Partial-Signature History-Free Sequential Aggregate Signature (PS-HF-SAS) is a tuple of four algorithms (KGen, AggSign, AggVrfy, Combine):

- KGen and AggVrfy as described in Definition 3.5.

- $\mathsf{AggSign}(\mathsf{sk}_i, m_i, \varrho_{i-1})$: takes as input the secret key sk_i and the message m_i of the i-th user and a partial description ϱ_{i-1} of the previous aggregate signature Σ_{i-1}. Computes an updated aggregate signature Σ_i and returns a partial description ϱ_i and some complementary information ς_i.

- $\mathsf{Combine}(\varsigma_1, \ldots, \varsigma_{n-1}, \Sigma_n)$: takes as input the complementary information ς_i of the first $n-1$ signatures and the full description of the last signature Σ_n. Returns the complete description of the aggregate signature $\bar{\Sigma}_n$.

Below, we show the definition of Partial-Signature History-Free existential Unforgeability against Chosen-Message Attacks (PS-HF-UF-CMA). In this model, the forger controls all signers' private keys except for at least one honest signer. The forger can choose the keys of the rogue signers and adaptively query an aggregate signature oracle. Finally, to win the experiment, the forger must produce a valid, non-trivial aggregate signature involving the public key of the honest signer. A stronger notion is also considered in [48], where the adversary may produce forgery on messages already queried to the signing oracle, provided that the complementary part of the corresponding signature is distinct from the oracle's response. We denote this variant as strong PS-HF-UF-CMA.

Definition 3.7 (PS-HF-UF-CMA Security)**.** Let O be a random oracle, let SAS = (KGen, AggSign, AggVrfy, Combine) be a PS-HF-SAS scheme, let $\mathcal{A}$ be an adversary. We define the advantage of $\mathcal{A}$ playing the strong PS-HF-UF-CMA game (Experiment 3.2) against SAS as follows:

$$\mathrm{Adv}_{\mathrm{SAS}}^{\text{PS-HF-UF-CMA}}(\mathcal{A}) = \Pr[\text{PS-HF-UF-CMA}_{\mathrm{SAS}}(\mathcal{A}) = 1].$$

We say that SAS is *strongly partial-signature history-free unforgeable against chosen message* attacks if the advantage $\mathrm{Adv}_{\mathrm{SAS}}^{\text{PS-HF-UF-CMA}}(\mathcal{A})$ is negligible for any adversary $\mathcal{A}$.

3.4.2 The Scheme

We present a PS-HF-SAS scheme following the probabilistic Hash-and-Sign with retry paradigm. The intuition behind our scheme closely follows the one of [156, 49]. We use a two-step hash procedure: first, the signature x_{i-1} of the previous signer and the message m_i are contracted to a short value h_i, and then expanded to the codomain of the trapdoor function. The value h_i can be aggregated and made available to the verifier, who can expand it before knowing x_{i-1}. In this way, Neven [156] showed that the verifier does not need to check the validity of the signers' public keys. When calculating the first hash, we also require the signer to concatenate a random salt r_i, which will later be part of the aggregate signature. In [49], the salt is introduced to prevent a chosen message attack in the history-free setting. In our scheme, the use of the salt descends from the probabilistic Hash-and-Sign paradigm and provides a solution to overcome the technical challenges of using trapdoor functions that are not permutations. As discussed later in Section 3.3, attempting to remove the random salt would make the construction insecure even in the full-history setting.

A high-level description of the scheme HaS-HF-SAS is shown in Figure 3.3 while a detailed description is given in Algorithm 3.2. The properties of the underlying trapdoor function are as described in Section 2.1.

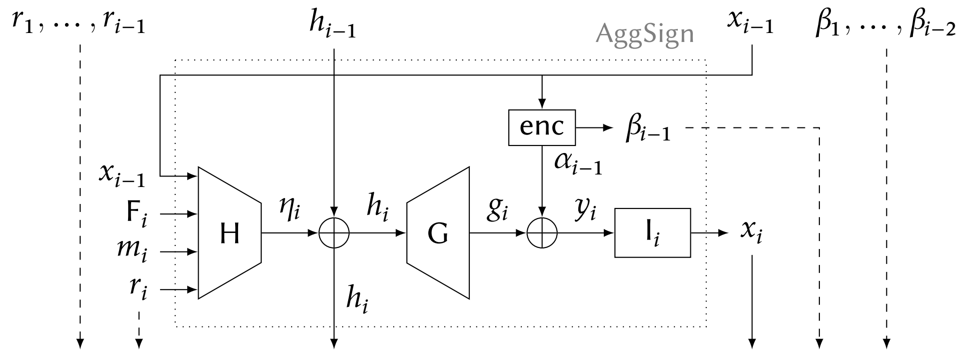


Figure 3.3: High-level description of HaS-HF-SAS scheme. The dashed arrows in the output represent the complementary part of the signature.

Algorithm 3.2: Hash-and-Sign History-Free SAS (HaS-HF-SAS)

Let $h_0 = \varepsilon, x_0 = \varepsilon$. The random oracles are $\mathsf{H}\colon \{0,1\}^* \to \{0,1\}^{2\lambda}$ and $\mathsf{G}\colon \{0,1\}^{2\lambda} \to \mathcal{Y}$. The encoding function is $\mathsf{enc}\colon \mathcal{X} \to \mathcal{Y} \times \mathcal{X}'$ and the corresponding decoding function is $\mathsf{dec}\colon \mathcal{Y} \times \mathcal{X}' \to \mathcal{X}$ such that $\mathsf{dec}(\mathsf{enc}(x)) = x$. ϱ_i and ς_i are the partial description and the complementary information of the aggregate signature Σ_i, respectively.

```
KGen($1^\lambda$):
  $(\mathsf{F}, \mathsf{I}) \leftarrow\$ \mathsf{TrapGen}(1^\lambda)$
  return $\mathsf{pk} \leftarrow \mathsf{F}, \mathsf{sk} \leftarrow (\mathsf{F}, \mathsf{I})$

AggSign($(\mathsf{F}_i, \mathsf{I}_i), m_i, \varrho_{i-1}$):
  $(h_{i-1}, x_{i-1}) \leftarrow \varrho_{i-1}$
  $(\alpha_{i-1}, \beta_{i-1}) \leftarrow \mathsf{enc}(x_{i-1})$
  repeat
      $r_i \leftarrow\$ R$
      $\eta_i \leftarrow \mathsf{H}(\mathsf{F}_i, m_i, r_i, x_{i-1})$
      $h_i \leftarrow h_{i-1} \oplus \eta_i$
      $g_i \leftarrow \mathsf{G}(h_i)$
      $y_i \leftarrow g_i \oplus \alpha_{i-1}$
      $x_i \leftarrow\$ \mathsf{I}_i(y_i)$
  until $x_i \neq \perp$
  $\varrho_i \leftarrow (h_i, x_i)$
  $\varsigma_i \leftarrow (r_i, \beta_{i-1})$
  return $\varrho_i, \varsigma_i$
```

```
AggVrfy($L_n, \bar{\Sigma}_n$):
  $(\mathsf{F}_1, m_1), \ldots, (\mathsf{F}_n, m_n) \leftarrow L_n$
  $(\overrightarrow{r}_n, \overrightarrow{\beta}_{n-1}, h_n, x_n) \leftarrow \bar{\Sigma}_n$
  for $i \leftarrow n, \ldots, 2$ do
      $y_i \leftarrow \mathsf{F}_i(x_i)$
      $g_i \leftarrow \mathsf{G}(h_i)$
      $\alpha_{i-1} \leftarrow g_i \oplus y_i$
      $x_{i-1} \leftarrow \mathsf{dec}(\alpha_{i-1}, \beta_{i-1})$
      $\eta_i \leftarrow \mathsf{H}(\mathsf{F}_i, m_i, r_i, x_{i-1})$
      $h_{i-1} \leftarrow h_i \oplus \eta_i$
  return $h_1 = \mathsf{H}(\mathsf{F}_1, r_1, m_1, \varepsilon) \wedge \mathsf{F}_1(x_1) = \mathsf{G}(h_1)$

Combine($\varsigma_1, \ldots, \varsigma_{n-1}, \Sigma_n$):
  $(r_i, \beta_{i-1}) \leftarrow \varsigma_i$
  $(r_n, \beta_{n-1}, h_n, x_n) \leftarrow \Sigma_n$
  $\overrightarrow{r}_n \leftarrow (r_1, \ldots, r_n)$
  $\overrightarrow{\beta}_{n-1} \leftarrow (\beta_1, \ldots, \beta_{n-1})$
  return $\bar{\Sigma}_n \leftarrow (\overrightarrow{r}_n, \overrightarrow{\beta}_{n-1}, h_n, x_n)$
```

3.4.3 Security Proof

In the following, we prove the strong PS-HF-UF-CMA security of Algorithm 3.2.

Theorem 3.8. *Let* T *be a TDF. Let* $\mathcal{A}$ *be a strong* PS-HF-UF-CMA *adversary against the* $\mathsf{HaS\text{-}HF\text{-}SAS}$ *scheme on* T *in the random oracle model, which runs in time* t *and makes* $\mathsf{q_S}$ *signing queries,* $\mathsf{q_H}$ *queries to the random oracle* H *and* $\mathsf{q_G}$ *queries to the random oracle* G*. Then, there exists an* INV *adversary* $\mathcal{B}$ *against* T *that runs in time* $t + \mathcal{O}((\mathsf{q_H} + \mathsf{q_S} + 1) \cdot \mathsf{poly}(\mathsf{len}(\mathcal{X}), \mathsf{len}(\mathcal{Y})))$*, and a* PS *adversary* $\mathcal{D}$ *against* T *issuing* $\mathsf{q_S}$ *sampling queries that runs in time* $t + \mathcal{O}(\mathsf{q_S} \cdot \mathsf{poly}(\mathsf{len}(\mathcal{X}), \mathsf{len}(\mathcal{Y})))$*, such that*

$$\begin{aligned}\mathsf{Adv}^{\text{PS-HF-UF-CMA}}_{\mathsf{HaS\text{-}HF\text{-}SAS}}(\mathcal{A}) \leq &(\psi \mathsf{q_H}) \cdot \mathsf{Adv}^{\text{INV}}_{\mathsf{T}}(\mathcal{B}) + \mathsf{Adv}^{\text{PS}}_{\mathsf{T}}(\mathcal{D}) + \frac{(\mathsf{q_S} + \mathsf{q_H})(\mathsf{q}'_\mathsf{S} + \mathsf{q_H} + \mathsf{q_G})}{2^{2\lambda}} \\ &+ \frac{\mathsf{q_S}(\mathsf{q}'_\mathsf{S} + \mathsf{q_H})}{|\mathsf{R}|} + \frac{\psi \mathsf{q}_\mathsf{H}^2}{2|\mathcal{Y}|} + \frac{(\psi \mathsf{q_H})^{\psi+1} |\mathcal{X}|}{(\psi+1)! \cdot |\mathcal{Y}|^{\psi+1}},\end{aligned}$$

where $\psi \geq \lceil \mathsf{len}(\mathcal{X}) / \mathsf{len}(\mathcal{Y}) \rceil$*, and* q'_S *is a bound on the total number of queries to* H *in all the signing queries.*

In the following, we sketch the high-level idea of the proof; full details can be found after Lemmas 3.9 to 3.11 at the end of this section. We prove the reduction by showing that the strong PS-HF-UF-CMA game can be simulated by the INV adversary $\mathcal{B}$. First, we modify the PS-HF-UF-CMA game such that in OAggSign the salt r is chosen uniformly at random in R and the preimage is generated by $x \leftarrow_\$ \mathsf{SampDom}(\mathsf{F}^\star)$ instead of iterating until $\mathsf{I}^\star(y) \neq \bot$. The PS adversary $\mathcal{D}$ can simulate the two games by either playing PS_0 or PS_1 and the advantage in distinguishing the two games can therefore be estimated with $\mathsf{Adv}^{\text{PS}}_{\mathsf{T}}(\mathcal{D})$. Once the preimages are produced by $x \leftarrow_\$ \mathsf{SampDom}(\mathsf{F}^\star)$ without retry, we can adapt the techniques for trapdoor permutations of [49] to complete the reduction. In particular, we will use a labelled tree HTree whose nodes will be populated by some of the queries to the random oracle H. The HTree is initialized with a root node with a single value $h_0 = \varepsilon$. Each subsequent node N_i is added following a query to the random oracle H with input $Q_i = (\mathsf{F}_i, m_i, r_i, x_{i-1})$ and will store the following values:

- a reference to its parent node N_{i-1};
- the query Q_i to the random oracle H;
- the hash response to the query $\eta_i \leftarrow \mathsf{H}(Q_i)$;
- the hash state $h_i \leftarrow h_{i-1} \oplus \eta_i$, where h_{i-1} is the hash state stored in the parent node N_{i-1};
- an additional value $y_i \leftarrow \mathsf{G}(h_i) \oplus \alpha_{i-1}$ (where α_{i-1} is computed from $\mathsf{enc}(x_{i-1})$) that will be used to establish if future nodes can be added as children of N_i.

A node N_i can be added as a child of a node N_{i-1} if it satisfies the relation $\mathsf{F}_{i-1}(x_{i-1}) = y_{i-1}$, where F_{i-1} and y_{i-1} are stored in N_{i-1}, while x_{i-1} is stored in N_i. This relationship establishes that the query Q_i can be properly used by the signer with key F_i to aggregate their signature on message m_i with previous signature x_{i-1}, produced by key F_{i-1} and hash state h_{i-1}, which in turn are stored in N_{i-1}. Whenever a query $Q_i = (\mathsf{F}_i, m_i, r_i, x_{i-1})$, with $x_{i-1} \neq \varepsilon$, satisfies this relation with a node N_{i-1} we say that Q_i can be *tethered* to N_{i-1}. If $x_{i-1} = \varepsilon$, then Q_i can always be tethered to the root of the HTree.

Eventually, when the adversary $\mathcal{A}$ outputs a valid aggregate signature $\bar{\Sigma}_n$ for the history $L_n = (\mathsf{pk}_1, m_1), \dots, (\mathsf{pk}_n, m_n)$, the simulator takes $i^\star \in [n]$ such that $\mathsf{pk}_{i^\star} = \mathsf{pk}^\star$ and $(m_{i^\star}, \varsigma_{i^\star}) \notin \mathcal{Q}$ (the index $i^\star$ is guaranteed to exist when $\mathcal{A}$ is winning). It then recovers $x_{i^\star}$ by iterating the procedure of Lines 3 to 9 in AggVrfy for $n - i^\star$ steps. Then, the simulator checks if $x_{i^\star}$ is a preimage of a $y_{i^\star}$ in the HTree as a child of the node $N_{i^\star-1}$ storing $Q_{i^\star-1} = (\mathsf{pk}_{i^\star-1}, m_{i^\star-1}, r_{i^\star-1}, x_{i^\star-2})$, which is itself a child of the node $N_{i^\star-2}$, and so on until the node N_1. If this is not the case, the simulator aborts by raising $\mathsf{bad}_{\mathsf{teth}}$. Otherwise, the value $x_{i^\star}$ produced by the forgery will satisfy $\mathsf{F}^\star(x_{i^\star}) = y_{i^\star}$ for some $y_{i^\star}$ produced either on Line 19 or on Line 21 of H and stored in $N_{i^\star}$. With probability $1/(\psi \mathsf{q_H})$, we have that $y_{i^\star}$ was produced on Line 21 of H and it is equal to $y^\star$. Therefore, $\mathsf{F}^\star(x_{i^\star}) = y^\star$ and $\mathcal{B}$ wins his INV game by returning $x_{i^\star}$.

For the complete proof of Theorem 3.8 we need the following technical lemmas.

Lemma 3.9. *When a new node is added to the* HTree *as a result of a call to* H, *the additional value y' is chosen uniformly at random from $\mathcal{Y}$.*

Proof. When a new node is added to the HTree on Line 23 of H, there are two possibilities for the additional value y'. In both cases, y' is chosen uniformly at random from $\mathcal{Y}$ and is independent of the view of $\mathcal{A}$. In fact, whenever the query to H is not the special random guess $c^\star$ chosen by the simulator, we have $y' \leftarrow \mathsf{G}(h') \oplus \alpha$. Here, $\mathsf{G}(h')$ is guaranteed to be a fresh uniformly random value since, otherwise, H would abort on Line 16 and the node would not be added to the HTree. If, on the other hand, the query $c^\star$ was made to H, then we set $y' \leftarrow y^\star$ for one of the new nodes to be added. Since $c^\star$ was chosen randomly among all queries to H, the assignment of $y^\star$ is made independently of the view of $\mathcal{A}$ and previous interactions with H. □

Lemma 3.10. *For any $k > \psi$ functions $\mathsf{F}_1, \dots, \mathsf{F}_k \colon \mathcal{X} \to \mathcal{Y}$ and uniformly random $y_1, \dots, y_k \in \mathcal{Y}$, there exists $x \in \mathcal{X}$ such that $\mathsf{F}_i(x) = y_i$, for every $i = 1, \dots, k$, with probability at most $|\mathcal{X}|/|\mathcal{Y}|^k$.*

Proof. Let $S_y^{\mathsf{F}} = \{x \in \mathcal{X} : \mathsf{F}(x) = y\}$ be the set of preimages of y under F. For a random choice of y_1 it holds that $|S_{y_1}^{\mathsf{F}_1}| = \alpha$ for some $0 \leq \alpha \leq |\mathcal{X}|$. Then,

Algorithm 3.3: Full Reduction OW $\implies$ PS-HF-UF-CMA

```
$\mathcal{B}(\mathsf{F}^\star, y^\star)$:
$\mathcal{Q} \leftarrow \emptyset; c^\star \leftarrow_\$ [\mathsf{q_H}]; c \leftarrow 0$
$(L_n, \bar{\Sigma}_n) \leftarrow_\$ \mathcal{A}^{\mathsf{H},\mathsf{G},\mathsf{OAggSign}}(\mathsf{F}^\star)$
$(\mathsf{F}_1, m_1), \ldots, (\mathsf{F}_n, m_n) \leftarrow L_n$
**if** $\mathsf{AggVrfy}(L_n, \Sigma_n) \wedge \exists i^\star : (\mathsf{F}_{i^\star} = \mathsf{F}^\star \wedge m_{i^\star} \notin \mathcal{Q})$ **then**
    Recover $x_{i^\star}$ as in AggVrfy
    $\mathsf{NList} \leftarrow \mathsf{Lookup}(x_{i^\star})$
    **if** $\mathsf{NList} = \bot$ **then**
        **raise** $\mathsf{bad_{teth}}$
    **for** $N_{i^\star} \in \mathsf{NList}$ **do**
        Retrieve $y_{i^\star}$ from $N_{i^\star}$
        **if** $y_{i^\star} = y^\star$ **then**
            **return** $x_{i^\star}$
    **raise** $\mathsf{bad_{inv}}$

$\mathsf{OAggSign}(m, \rho = (h, x))$:
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{m\}$
$(\alpha, \beta) \leftarrow \mathsf{enc}(x)$
$r \leftarrow_\$ \mathsf{R}$
**if** $\mathsf{HT}[\mathsf{F}^\star, m, r, x] \neq \bot$ **then**
    **raise** $\mathsf{bad_{hcol}}$
$\eta \leftarrow_\$ \{0,1\}^{2\lambda}$
$\mathsf{HT}[\mathsf{F}^\star, m, r, x] \leftarrow \eta$
$h' \leftarrow h \oplus \eta$
**if** $\mathsf{GT}[h'] \neq \bot$ **then**
    **raise** $\mathsf{bad_{gcol1}}$
$x' \leftarrow_\$ \mathcal{X}$
$y' \leftarrow \mathsf{F}^\star(x')$
$\mathsf{GT}[h'] \leftarrow y' \oplus \alpha$
**return** $(r, \beta), (h', x')$

$\mathsf{G}(h)$:
**if** $\mathsf{GT}[h] = \bot$ **then**
    $g \leftarrow_\$ \mathcal{Y}$
    $\mathsf{GT}[h] \leftarrow g$
**return** $\mathsf{GT}[h]$

$\mathsf{H}(\mathsf{F}, m, r, x)$:
$Q \leftarrow (\mathsf{F}, m, r, x)$
$c \leftarrow c + 1$
**if** $\mathsf{HT}[Q] = \bot$ **then**
    $\eta \leftarrow_\$ \{0,1\}^{2\lambda}$
    $\mathsf{HT}[Q] \leftarrow \eta$
    $\mathsf{NList} \leftarrow \mathsf{Lookup}(x)$
    **if** $\mathsf{F} = \mathsf{F}^\star \wedge c = c^\star$ **then**
        $i^\star \leftarrow_\$ [|\mathsf{NList}|]$
    **for** $i \in [|\mathsf{NList}|]$ **do**
        $N_i \leftarrow \mathsf{NList}[i]$
        $N_i' \leftarrow$ new node with parent $N_i$
        Retrieve $h_i$ from $N_i$
        $h_i' \leftarrow h_i \oplus \eta$
        $(\alpha, \beta) \leftarrow \mathsf{enc}(x)$
        **if** $\mathsf{GT}[h_i'] \neq \bot$ **then**
            **raise** $\mathsf{bad_{gcol2}}$
        **if** $\mathsf{F} \neq \mathsf{F}^\star \vee c \neq c^\star \vee i \neq i^\star$ **then**
            $g_i' \leftarrow \mathsf{G}(h_i')$
            $y_i' \leftarrow g_i' \oplus \alpha$
        **else**
            $y_i' \leftarrow y_i^\star$
            $\mathsf{GT}[h_i'] \leftarrow y_i' \oplus \alpha$
        Populate node $N_i'$ with $Q, \eta, h_i', y_i'$
**return** $\mathsf{HT}[Q]$

$\mathsf{Lookup}(x)$:
**if** $x = \varepsilon$ **then**
    **return** Root of HTree
$\mathsf{NList} \leftarrow \{N \in \mathsf{HTree} : (\mathsf{F}, y) \in N \wedge \mathsf{F}(x) = y\}$
**if** $|\mathsf{NList}| > \psi$ **then**
    **raise** $\mathsf{bad_{tcol}}$
**else if** $|\mathsf{NList}| = 0$ **then**
    **return** $\bot$
**else**
    **return** NList
```

there are at most α possible values for the tuple $(y_2, \dots, y_k)$, corresponding to $\{ (\mathsf{F}_2(x), \dots, \mathsf{F}_k(x)) : x \in S_{y_1}^{\mathsf{F}_1} \}$, such that $\bigcap S_{y_i}^{\mathsf{F}_i} \neq \emptyset$. Since $y_2, \dots, y_k$ are uniformly chosen in $\mathcal{Y}$, the probability of a non-empty intersection is at most $\alpha |\mathcal{Y}|^{1-k}$. Therefore, the desired probability is bounded by varying over the possible values of α:

$$\sum_{\alpha=0}^{|\mathcal{X}|} \frac{\alpha}{|\mathcal{Y}|^{k-1}} \Pr_{y_1 \leftarrow_\$ \mathcal{Y}}\left[|S_{y_1}^{\mathsf{F}_1}| = \alpha \right] = \frac{1}{|\mathcal{Y}|^{k-1}} \sum_{\alpha=0}^{|\mathcal{X}|} \alpha \cdot \frac{|\{ y_1 \in \mathcal{Y} : |S_{y_1}^{\mathsf{F}_1}| = \alpha \}|}{|\mathcal{Y}|} = \frac{|\mathcal{X}|}{|\mathcal{Y}|^k}.$$

□

Lemma 3.11. *If an input Q has not been entered in the* HTree *after being queried to* H*, the probability that it will ever become tethered to a node in* HTree *is at most $\psi \mathsf{q}'/|\mathcal{Y}|$, where q' is the number of queries made to* H *after Q.*

Proof. Suppose that $Q = (\mathsf{F}, m, r, x)$ was queried to H and was not added to the HTree, i.e. $\mathsf{Lookup}(x) = \perp$. Now suppose that a query $Q' = (\mathsf{F}', m', r', x')$ was subsequently sent to H and was added to HTree as part of a node N' with additional value y'. For Q to be tethered to N', it must hold that $\mathsf{F}'(x) = y'$. Following Lemma 3.9, when a new node is added to the HTree as a result of a call to H, the additional value y' is chosen uniformly at random from $\mathcal{Y}$. In particular, y' is random and independent of F' and x. Therefore, the probability of having $\mathsf{F}'(x) = y'$ is $|\mathcal{Y}|^{-1}$. Since there are at most q' queries to H after Q and each query can add at most ψ nodes to the HTree, the desired probability follows by the union bound. □

Proof for strong PS-HF-UF-CMA **security (Theorem 3.8)**

Proof. We prove the reduction by presenting a sequence of hybrid games, modifying the strong PS-HF-UF-CMA game (Experiment 3.2) until it can be simulated by the INV adversary $\mathcal{B}$. The complete reduction is described in Algorithm 3.3. In the following, we use the notation $\Pr[\mathsf{Game}_\mathsf{n}(\mathcal{A}) = 1]$ to denote the probability that Game_n returns 1 when played by $\mathcal{A}$. The game sequence Game_0-Game_3 for OAggSign is detailed in Argument 3.1. The game sequence Game_3-Game_5 for H is detailed in Argument 3.2.

Game_0 This is the original strong PS-HF-UF-CMA game against the HaS-HF-SAS scheme, except that it uses programmable random oracles. At the start of the game, the challenger initializes two tables, HT for H and GT for G. When a query Q for H is received, if $\mathsf{HT}[Q] = \perp$ it uniformly samples $\eta \leftarrow_\$ \{0,1\}^{2\lambda}$ and stores $\mathsf{HT}[Q] \leftarrow \eta$, finally it returns $\mathsf{HT}[Q]$ (similarly for G). It follows that $\Pr[\mathsf{Game}_0(\mathcal{A}) = 1] = \mathsf{Adv}_{\mathsf{HaS\text{-}HF\text{-}SAS}}^{\mathsf{PS\text{-}HF\text{-}UF\text{-}CMA}}(\mathcal{A})$.

Hybrid Argument 3.1: Games for $\mathsf{OAggSign}(m, \varrho = (h, x))$

```
Game_0:
$(\alpha, \beta) \leftarrow \mathsf{enc}(x)$
repeat
    $r \leftarrow_{\$} \mathsf{R}$
    $\eta \leftarrow \mathsf{H}(\mathsf{F}^\star, m, r, x)$
    $h' \leftarrow h \oplus \eta$
    $g' \leftarrow \mathsf{G}(h')$
    $y' \leftarrow g' \oplus \alpha$
    $x' \leftarrow_{\$} \mathsf{I}^\star(y')$
until $x' \neq \perp$
return $(r, \beta), (h', x')$
```

```
Game_1-Game_2:
$(\alpha, \beta) \leftarrow \mathsf{enc}(x)$
repeat
    $r \leftarrow_{\$} \mathsf{R}$
    if $\mathsf{HT}[\mathsf{F}^\star, m, r, x] \neq \perp$ then
        raise $\mathsf{bad_{hcol}}$
    $\eta \leftarrow_{\$} \{0,1\}^{2\lambda}$
    $\mathsf{HT}[\mathsf{F}^\star, m, r, x] \leftarrow \eta$
    $h' \leftarrow h \oplus \eta$
    if $\mathsf{GT}[h'] \neq \perp$ then
        raise $\mathsf{bad_{gcol1}}$
    $y' \leftarrow_{\$} \mathcal{Y}$
    $\mathsf{GT}[h'] \leftarrow y' \oplus \alpha$
    $x' \leftarrow \mathsf{I}^\star(y')$
until $x' \neq \perp$
return $(r, \beta), (h', x')$
```

```
Game_3-Game_5:
$(\alpha, \beta) \leftarrow \mathsf{enc}(x)$
$r \leftarrow_{\$} \mathsf{R}$
if $\mathsf{HT}[\mathsf{F}^\star, m, r, x] \neq \perp$ then
    raise $\mathsf{bad_{hcol}}$
$\eta \leftarrow_{\$} \{0,1\}^{2\lambda}$
$\mathsf{HT}[\mathsf{F}^\star, m, r, x] \leftarrow \eta$
$h' \leftarrow h \oplus \eta$
if $\mathsf{GT}[h'] \neq \perp$ then
    raise $\mathsf{bad_{gcol1}}$
$x' \leftarrow_{\$} \mathsf{SampDom}(\mathsf{F}^\star)$
$y' \leftarrow \mathsf{F}^\star(x')$
$\mathsf{GT}[h'] \leftarrow y' \oplus \alpha$
return $(r, \beta), (h', x')$
```

Game_1 This game is identical to Game_0 except that $\mathsf{OAggSign}$ aborts by raising $\mathsf{bad_{hcol}}$ if on query $(m, \varrho = (h, x))$ it samples a salt r such that the random oracle H was already queried at input $Q = (\mathsf{F}^\star, m, r, x)$, i.e. $\mathsf{HT}[Q] \neq \perp$. Otherwise, it samples $\eta \leftarrow_{\$} \{0,1\}^{2\lambda}$ and programs $\mathsf{HT}[Q] \leftarrow \eta$. It follows that $|\Pr[\mathsf{Game}_0(\mathcal{A}) = 1] - \Pr[\mathsf{Game}_1(\mathcal{A}) = 1]| \leq \Pr[\mathsf{bad_{hcol}}]$.

Game_2 This game is identical to Game_1 except that $\mathsf{OAggSign}$ aborts by raising $\mathsf{bad_{gcol1}}$ if on query $(m, \varrho = (h, x))$, after sampling $\eta \leftarrow_{\$} \{0,1\}^{2\lambda}$ it computes $h' \leftarrow h \oplus \eta$ such that the random oracle G was already queried at input h', i.e. $\mathsf{GT}[h'] \neq \perp$. Otherwise, it samples $y' \leftarrow_{\$} \mathcal{Y}$ and programs $\mathsf{GT}[h'] \leftarrow y' \oplus \alpha$. It follows that $|\Pr[\mathsf{Game}_1(\mathcal{A}) = 1] - \Pr[\mathsf{Game}_2(\mathcal{A}) = 1]| \leq \Pr\left[\mathsf{bad_{gcol1}}\right]$.

Game_3 This game is identical to Game_2 except that $\mathsf{OAggSign}$ directly samples $r \leftarrow_{\$} \mathsf{R}$, $x' \leftarrow_{\$} \mathsf{SampDom}(\mathsf{F}^\star)$ and computes $y' \leftarrow \mathsf{F}^\star(x')$ instead of computing $x' \leftarrow \mathsf{I}^\star(y')$ after sampling $y' \leftarrow_{\$} \mathcal{Y}$. The PS adversary $\mathcal{D}$ can simulate both Game_2 and Game_3, noticing that $y' = \mathsf{F}^\star(x')$ and programming G accordingly. More precisely, on receiving a query $Q = (m, \varrho = (h, x))$ for $\mathsf{OAggSign}$, $\mathcal{D}$ computes $(r, x') \leftarrow_{\$} \mathsf{Sample}_b$ and programs $\mathsf{GT}[h'] \leftarrow \mathsf{F}^\star(x') \oplus \alpha$. Both Game_2 and Game_3 are equivalently modified by moving the programming step of H and G to the end of the $\mathsf{OAggSign}$. It now follows that when $\mathcal{D}$ is playing PS_0 their simulation coincides with Game_2, while when it is playing PS_1 it coincides with Game_3. Either way, $\mathcal{D}$ simulates the games with at most the same running time of $\mathcal{A}$ plus the time required for answering the queries to the sampling oracle. The latter takes $\mathcal{O}(\mathrm{poly}(\mathrm{len}(\mathcal{X}), \mathrm{len}(\mathcal{Y})))$ and is repeated at most $\mathsf{q_S}$ times. Finally, we have that $|\Pr[\mathsf{Game}_2(\mathcal{A}) = 1] - \Pr[\mathsf{Game}_3(\mathcal{A}) = 1]| \leq \mathsf{Adv}^{\mathrm{PS}}_{\mathsf{T}}(\mathcal{D})$.

Game_4 This game is identical to Game_3 except that the random oracle H is simulated as follows. At the start of the game, the challenger initializes a labelled tree HTree, as described at the beginning of the proof. When H receives a query $Q = (\mathsf{F}, m, r, x)$, if $\mathsf{HT}[Q] \neq \perp$ it returns it. Otherwise, it samples a uniformly random $\eta \leftarrow\$ \{0,1\}^{2\lambda}$ and programs $\mathsf{HT}[Q] \leftarrow \eta$. Then, it checks if Q can be added as a child node of existing nodes in HTree. To determine whether this is the case, it uses the Lookup function (see Algorithm 3.3) on input x that checks if it can be tethered to existing nodes, i.e. there exists a node $N_i \in \mathsf{HTree}$ such that $\mathsf{F}_i(x) = y_i$. If Q can be tethered to more than ψ nodes, the game aborts by raising $\mathsf{bad_{tcol}}$. Otherwise, H add a new node N_i' with parent N_i for each node $N_i \in \mathsf{HTree}$ returned by $\mathsf{Lookup}(x)$. N_i' contains the original query Q, the hash response η, the hash state $h_i' \leftarrow h_i \oplus \eta$ (where h_i is stored in N_i) and an additional value $y_i' \leftarrow \mathsf{G}(h_i') \oplus \alpha$ (where α is computed from $\mathsf{enc}(x)$) that will be used to check if a future node can be tethered via Lookup queries. It holds that $|\Pr[\mathsf{Game}_3(\mathcal{A}) = 1] - \Pr[\mathsf{Game}_4(\mathcal{A}) = 1]| \leq \Pr[\mathsf{bad_{tcol}}]$.

Game_5 This game is identical to Game_4 except that the random oracle H is simulated as follows. At the beginning of the game, the challenger uniformly chooses an index $c^\star \leftarrow\$ [\mathsf{q_H}]$ among the queries to the random oracle H, initializes a counter $c \leftarrow 0$ and uniformly samples $y^\star \leftarrow\$ \mathcal{Y}$. When H receives a query $Q = (\mathsf{F}, m, r, x)$ it increments $c \leftarrow c + 1$. Then, if $\mathsf{F} = \mathsf{F}^\star$ and $c = c^\star$, it samples a random index $i^\star$ from the number of nodes in NList. If, for any of the new nodes to be added, it computes $h_i' \leftarrow h_i \oplus \eta$ such that the random oracle G was already queried at input h_i', i.e. $\mathsf{GT}[h_i'] \neq \perp$, it aborts by raising $\mathsf{bad_{gcol2}}$. Otherwise, if $\mathsf{F} = \mathsf{F}^\star$, $c = c^\star$ and $i = i^\star$, it sets $y_i' \leftarrow y^\star$ and programs $\mathsf{GT}[h_i'] \leftarrow y_i' \oplus \alpha$. It holds that $|\Pr[\mathsf{Game}_4(\mathcal{A}) = 1] - \Pr[\mathsf{Game}_5(\mathcal{A}) = 1]| \leq \Pr\left[\mathsf{bad_{gcol2}}\right]$.

If none of the bad events happen, $\mathcal{B}$ perfectly simulate Game_5 and we have that

$$\begin{aligned}
\mathsf{Adv}^{\mathrm{INV}}_{\mathsf{T}}(\mathcal{B}) &= \frac{1}{\psi \mathsf{q_H}} \Pr[\mathsf{Game}_5(\mathcal{A}) = 1] \\
&\geq \frac{1}{\psi \mathsf{q_H}} (\mathsf{Adv}^{\mathrm{PS\text{-}HF\text{-}UF\text{-}CMA}}_{\mathsf{HaS\text{-}HF\text{-}SAS}}(\mathcal{A}) - \Pr[\mathsf{bad_{hcol}}] - \Pr\left[\mathsf{bad_{gcol1}}\right] \\
&\qquad - \mathsf{Adv}^{\mathrm{PS}}_{\mathsf{T}}(\mathcal{D}) - \Pr[\mathsf{bad_{tcol}}] - \Pr\left[\mathsf{bad_{gcol2}}\right] - \Pr[\mathsf{bad_{teth}}]).
\end{aligned}$$

$\mathcal{B}$ can simulate Game_5 with at most the same running time of $\mathcal{A}$ plus the time required for running $\mathsf{AggVrfy}$ and answering the queries to the random oracles H, G,

Hybrid Argument 3.2: Games for $\mathsf{H}(\mathsf{F}, m, r, x)$

```
Game_0-Game_3:
  $Q \leftarrow (\mathsf{F}, m, r, x)$
  if $\mathsf{HT}[Q] = \bot$ then
      $\eta \leftarrow_{\$} \{0,1\}^{2\lambda}$
      $\mathsf{HT}[Q] \leftarrow \eta$
  return $\mathsf{HT}[Q]$

Game_4:
  $Q \leftarrow (\mathsf{F}, m, r, x)$
  if $\mathsf{HT}[Q] = \bot$ then
      $\eta \leftarrow_{\$} \{0,1\}^{2\lambda}$
      $\mathsf{HT}[Q] \leftarrow \eta$
      $\mathsf{NList} \leftarrow \mathsf{Lookup}(x)$
      for $i \in [|\mathsf{NList}|]$ do
          $N_i \leftarrow \mathsf{NList}[i]$
          $N_i' \leftarrow$ new node with parent $N_i$
          Retrieve $h_i$ from $N_i$
          $h_i' \leftarrow h_i \oplus \eta$
          $(\alpha, \beta) \leftarrow \mathsf{enc}(x)$
          $g_i' \leftarrow \mathsf{G}(h_i')$
          $y_i' \leftarrow g_i' \oplus \alpha$
          $N_i' \leftarrow (Q, \eta, h_i', y_i')$
  return $\mathsf{HT}[Q]$

Game_5:
  $Q \leftarrow (\mathsf{F}, m, r, x)$
  $c \leftarrow c + 1$
  if $\mathsf{HT}[Q] = \bot$ then
      $\eta \leftarrow_{\$} \{0,1\}^{2\lambda}$
      $\mathsf{HT}[Q] \leftarrow \eta$
      $\mathsf{NList} \leftarrow \mathsf{Lookup}(x)$
      if $\mathsf{F} = \mathsf{F}^\star \wedge c = c^\star$ then
          $i^\star \leftarrow_{\$} [|\mathsf{NList}|]$
      for $i \in [|\mathsf{NList}|]$ do
          $N_i \leftarrow \mathsf{NList}[i]$
          $N_i' \leftarrow$ new node with parent $N_i$
          Retrieve $h_i$ from $N_i$
          $h_i' \leftarrow h_i \oplus \eta$
          $(\alpha, \beta) \leftarrow \mathsf{enc}(x)$
          if $\mathsf{GT}[h_i'] \neq \bot$ then
              raise $\mathsf{bad}_{\mathsf{gcol2}}$
          if $\mathsf{F} \neq \mathsf{F}^\star \vee c \neq c^\star \vee i \neq i^\star$ then
              $g_i' \leftarrow \mathsf{G}(h_i')$
              $y_i' \leftarrow g_i' \oplus \alpha$
          else
              $y_i' \leftarrow y^\star$
              $\mathsf{GT}[h_i'] \leftarrow y_i' \oplus \alpha$
          $N_i' \leftarrow (Q, \eta, h_i', y_i')$
  return $\mathsf{HT}[Q]$
```

and to the signing oracle OAggSign. These operations take $\mathcal{O}(\mathsf{poly}(\mathsf{len}(\mathcal{X}), \mathsf{len}(\mathcal{Y})))$ and are repeated at most $\mathsf{q_H} + \mathsf{q_S} + 1$ times.

In the following, we bound the probability of each bad event happening.

Probability of $\mathsf{bad}_{\mathsf{hcol}}$ The event $\mathsf{bad}_{\mathsf{hcol}}$ occurs on Line 5 of OAggSign on input $(m, \rho = (h, x))$ when it samples $r \leftarrow_{\$} \mathsf{R}$ such that a value for $Q = (\mathsf{F}^\star, m, r, x)$ was already assigned in the HT. The table HT is populated by either OAggSign or H, so its entries are at most $\mathsf{q_S'} + \mathsf{q_H}$. The probability that a uniformly random r produces a collision with one of the entries is then at most $(\mathsf{q_S'} + \mathsf{q_H})/|\mathsf{R}|$. Since at most $\mathsf{q_S}$ are made to OAggSign, then $\Pr[\mathsf{bad}_{\mathsf{hcol}}] \leq \mathsf{q_S}(\mathsf{q_S'} + \mathsf{q_H})/|\mathsf{R}|$.

Probability of $\mathsf{bad}_{\mathsf{gcol1}}$ The event $\mathsf{bad}_{\mathsf{gcol1}}$ occurs on Line 10 of OAggSign on input $(m, \rho = (h, x))$ when, after sampling $\eta \leftarrow_{\$} \{0,1\}^{2\lambda}$, it computes $h' \leftarrow h \oplus \eta$ such that a value for h' was already assigned in the GT. The table GT is populated by either OAggSign, H or G so its entries are at most $q'_S + q_H + q_G$. The probability that a uniformly random η produces a collision with one of the entries is then at most $(q'_S + q_H + q_G)2^{-2\lambda}$. Since at most q_S are made to OAggSign, then $\Pr[\mathsf{bad}_{\mathsf{gcol1}}] \leq q_S(q'_S + q_H + q_G)2^{-2\lambda}$.

Probability of $\mathsf{bad}_{\mathsf{tcol}}$ The event $\mathsf{bad}_{\mathsf{tcol}}$ occurs on Line 5 of Lookup on input x when the HTree contains $k > \psi$ nodes $N_1, \dots, N_k$ such that $\mathsf{F}_i(x) = y_i$ for $i = 1, \dots, k$, where F_i, y_i are stored in their respective nodes N_i. The HTree is populated by the simulation of the random oracle H. There are at most q_H queries to H and each query contributes a maximum of ψ nodes to the tree. Consequently, the total number of nodes in HTree does not exceed ψq_H. Therefore, we need to bound the probability that any $(\psi + 1)$-tuple of nodes produce a collision on x.

To conclude, we prove that for any $(\psi + 1)$-tuple (possibly adversarially chosen) of functions $\mathsf{F}_i \colon \mathcal{X} \to \mathcal{Y}$ and uniformly random $y_i \in \mathcal{Y}$, there exists $x \in \mathcal{X}$ such that $\mathsf{F}_i(x) = y_i$, for any $i = 1, \dots, \psi + 1$, with probability at most $|\mathcal{X}|/|\mathcal{Y}|^{\psi+1}$ (Lemma 3.10). Indeed, the adversary can issue $\psi + 1$ queries to H with inputs any functions F_i to be stored in $\psi + 1$ nodes N_i in the HTree. However, from Lemma 3.9, we know that when a new node is added to the HTree on Line 23 of H, the value y'_i is chosen uniformly at random from $\mathcal{Y}$ and is independent of the view of $\mathcal{A}$. Therefore, the adversary would receive $\psi + 1$ random, independent values y_i.

Since the number of $(\psi+1)$-tuple of nodes in the HTree are at most $(\psi q_H)^{\psi+1}/(\psi + 1)!$, by the union bound, we obtain $\Pr[\mathsf{bad}_{\mathsf{tcol}}] \leq (\psi q_H)^{\psi+1}|\mathcal{X}|/((\psi + 1)! \cdot |\mathcal{Y}|^{\psi+1})$.

Probability of $\mathsf{bad}_{\mathsf{gcol2}}$ The event $\mathsf{bad}_{\mathsf{gcol2}}$ occurs on Line 16 of H on input (F, m, r, x) when, after sampling $\eta \leftarrow_{\$} \{0,1\}^{2\lambda}$ and retrieving h_{i-1} from the parent node N_{i-1}, it computes $h_i \leftarrow h_{i-1} \oplus \eta$ such that a value for h_i was already assigned in the GT. The same argument from the bound of $\Pr[\mathsf{bad}_{\mathsf{gcol}}]$ can be used to prove that $\Pr[\mathsf{bad}_{\mathsf{gcol2}}] \leq q_H(q'_S + q_H + q_G)2^{-2\lambda}$.

Probability of $\mathsf{bad}_{\mathsf{teth}}$ The event $\mathsf{bad}_{\mathsf{teth}}$ occurs on Line 8 of the simulation of $\mathcal{B}$ when, after the adversary $\mathcal{A}$ outputs a valid aggregate signature $\bar{\Sigma}_n$ for the history $L_n = (\mathsf{pk}_1, m_1), \dots, (\mathsf{pk}_n, m_n)$, the simulator recovers $x_{i^\star}$, with $i^\star \in [n]$ such that $\mathsf{pk}_{i^\star} = \mathsf{pk}^\star$ and $(m_{i^\star}, \varsigma_{i^\star}) \notin \mathcal{Q}$, but $x_{i^\star}$ cannot be tethered to any node in the HTree.

When $\mathsf{bad}_{\mathsf{teth}}$ happens, the aggregate signature $\bar{\Sigma}_n$ must be valid on L_n. In particular, the inputs $Q_1 = (\mathsf{F}_1, m_1, r_1, \varepsilon), Q_2 = (\mathsf{F}_2, m_2, r_2, x_1), \dots, Q_{i^\star} = (\mathsf{F}_{i^\star}, m_{i^\star}, r_{i^\star}, x_{i^\star-1})$ have been queried to H in OAggVrfy. Let $\eta_1, \dots, \eta_{i^\star}$ be the outputs of these queries, so that $\mathsf{HT}[Q_j] = \eta_j$. Each of these entries has been populated by H. In fact, the only exception could occur if $(m_{i^\star}, x_{i^\star-1})$

was queried to OAggSign. Suppose (r, β) is the complementary part of the signature produced by the oracle as a response. Since the forgery is valid, the complementary part $\varsigma_{i^\star} = (r_{i^\star}, \beta_{i^\star-1})$ produced by $\mathcal{A}$ must be different from (r, β). However, both $\beta_{i^\star-i}$ and β must be the same partial encoding of $x_{i^\star-1}$, so that $r_{i^\star} \neq r$. Therefore, $x_{i^\star}$ must have been produced following a query to H with a fresh salt $r_{i^\star}$.

Each step of OAggVrfy also recovers a value $h_j \leftarrow h_{j+1} \oplus \eta_j$ which is the input of the G query. Since the aggregate signature is correct, we obtain that $h_1 = \eta_1$. Observe that since Q_1 was queried to H, it must be tethered to the root of HTree and was therefore inserted as a node of HTree with additional values $\eta_1, h_1 = \eta_1, y_1 = \mathsf{G}(h_1)$. Then, since $\mathsf{F}_1(x_1) = y_1$, the query Q_2 is tethered to N_1. Now we prove that either all $Q_1, \dots, Q_{i^\star}$ are part of a path of nodes in HTree, or there exists an input Q_j that was queried to H, is tethered to a node in HTree and is not itself in a node of HTree. We proceed by induction on $j \leq i^\star$: we have already shown that Q_1 is in HTree; suppose that Q_j is in the HTree, then, since $\mathsf{F}_j(x_j) = y_j$, the query Q_{j+1} is tethered to Q_j and it may or may not be part of HTree. To conclude, we prove that if an input Q has not been entered in the HTree after being queried to H, the probability that it will ever become tethered to a node in HTree is at most $\psi \mathsf{q}'/|\mathcal{Y}|$, where q' is the number of queries made to H after Q (Lemma 3.11). Since there are at most $\mathsf{q_H}$ queries that add new nodes to HTree, we obtain, by the union bound, that $\Pr[\mathsf{bad_{teth}}] \leq \psi \mathsf{q}_\mathsf{H}^2/(2|\mathcal{Y}|)$.

Combining the previous bound on bad events, we obtain the claimed estimate of $\mathsf{Adv}^{\text{PS-HF-UF-CMA}}_{\text{HaS-HF-SAS}}(\mathcal{A})$. □

Incurred Security Loss

The security loss in Theorem 3.8 is comparable to that of EUF-CMA reduction for the single signature (Theorem 2.12), based on the same underlying trapdoor function. In both cases, the main loss is given by a multiplicative factor in $\mathsf{q_H}$. In our reduction, there are also quadratic additive terms in $\mathsf{q_S}$ and $\mathsf{q_H}$, but these are negligible in the security parameter for an appropriate choice of hash functions and salt length. More in detail, the reduction of Theorem 3.8 loses advantage by one multiplicative factor $\psi \cdot \mathsf{q_H}$ and five additive terms corresponding to the occurrence of the bad events $\mathsf{bad_{hcol}}$, $\mathsf{bad_{gcol1}}$, $\mathsf{bad_{tcol}}$, $\mathsf{bad_{gcol2}}$, $\mathsf{bad_{teth}}$. In the following $\mathsf{q_H}, \mathsf{q_G}$ are the number of queries to the random oracles H and G, $\mathsf{q_S}$ is the number of queries to the signing oracle and λ is the security parameter. For a concrete discussion, let us consider typical values $128 \leq \lambda \leq 256, 1 \leq \mathsf{q_H}, \mathsf{q_G} \leq 2^{64}$, and $1 \leq \mathsf{q_S} \leq 2^{32}$.

The additive loss is given by the following terms:

- The terms $(q_S + q_H)(q'_S + q_H + q_G)2^{-2\lambda}$ corresponding to $\mathsf{bad}_{\mathsf{gcol1}}$ and $\mathsf{bad}_{\mathsf{gcol2}}$ are related to the output collision of the hash function H. If we choose H to be a hash function with at least 2λ output bits, the terms are negligible in the security parameter λ.

- The term $(\psi q_H)^{\psi+1}|\mathcal{X}|/((\psi + 1)! \cdot |\mathcal{Y}|^{\psi+1})$ corresponding to $\mathsf{bad}_{\mathsf{tcol}}$ is related to the probability of finding more than ψ tethered nodes during a query to the random oracle H. This term is an artefact of the security proof. For an appropriate choice of ψ, this term is negligible in the security parameter. More details on the role and suitable choice of ψ are provided at the end of this section.

- The term $\psi q_H^2/(2|\mathcal{Y}|)$ corresponding to $\mathsf{bad}_{\mathsf{tcol}}$ is related to the probability that a query become tethered after it was queried to the random oracle H, so that is not in the HTree and cannot be used to win the INV game after the adversary supplies a forged aggregate signature. This term is negligible for appropriate choices of the trapdoor function. For instance, it is negligible for all parameterization of the schemes considered in Section 3.6.

- The term $q_S(q'_S + q_H)/|R|$ corresponding to $\mathsf{bad}_{\mathsf{hcol}}$ is related to the collision of the random salt. To make the term negligible, it is enough to take the salt of appropriate length (e.g. $|R| = 2\lambda$ for $\lambda = 128$). Notice that a similar term also appears in the security proof of the related signature schemes considered in Section 3.6 and in the generic reductions of Section 2.2.1 for Hash-and-Sign schemes with retry. Therefore, the salt length in our constructions will typically be equal to that of the single signature scheme.

The multiplicative loss of $\psi \cdot q_H$ constitutes the main loss in advantage in the security proof. To win the INV game, the INV adversary $\mathcal{B}$ provides their challenge $y^\star$ to the PS-HF-UF-CMA adversary $\mathcal{A}$ in one of q_H queries to the random oracle H (this is the same strategy as the EUF-CMA reduction of Theorem 2.12). For this special query, only one of ψ branches of the HTree lead to finding a preimage for $y^\star$. Hence, if $\mathcal{A}$ outputs a valid forgery, $\mathcal{B}$ wins the INV game with probability $(\psi \cdot q_H)^{-1}$.

Remark. The multiplicative loss is inevitable without stronger assumptions on the trapdoor function. In the next section, we discuss the extension of the scheme to PSFs and APSFs, and we show that a tighter reduction can be achieved by considering collision-resistant PSFs.

Branching Parameter ψ

In Theorem 3.8, the term ψ identifies the maximum number of nodes added to the HTree in each query to the random oracle H. This term is an artefact of the proof and does not address a practical attack. If during a query to H, the simulator identifies a number of nodes tethered to the input greater than ψ, the $\mathsf{bad}_{\mathsf{tcol}}$ event would occur. For a fixed ψ, the probability of this happening is mainly studied in Lemma 3.11. Accordingly, ψ should be taken as the minimum positive integer such that the additive security loss of the $\mathsf{bad}_{\mathsf{tcol}}$ event is negligible in the security parameter. The condition $\psi \geq \lceil \mathrm{len}(\mathcal{X}) / \mathrm{len}(\mathcal{Y}) \rceil$ is necessary to have a meaningful reduction in Theorem 3.8, otherwise the former additive security loss would be greater than 1. By choosing $\psi = \lceil \mathrm{len}(\mathcal{X}) / \mathrm{len}(\mathcal{Y}) \rceil$, Lemma 3.11 tells us that the probability of getting $\psi + 1$ nodes tethered to the input at each query to H is at most $1/|\mathcal{Y}|$. This is often enough to make the additive term negligible, but depending on the specific choices of $\mathcal{X}$ and $\mathcal{Y}$ it will sometimes be necessary to choose $\psi = \lceil \mathrm{len}(\mathcal{X}) / \mathrm{len}(\mathcal{Y}) \rceil + 1$. We note that it is desirable to choose a small ψ since its choice also results in a multiplicative security loss of a factor $\psi \cdot \mathsf{q}_{\mathsf{H}}$, as analysed above.

For the selected schemes in Section 3.6, the minimum value for ψ that meets the previous requirements is provided.

3.5 Optimizing the Scheme for (Average) Preimage Sampleable Functions

In this section we discuss how the construction of Section 3.4 can be modified in the presence of PSFs and APSFs. In particular, we will show how the additional properties of these classes of trapdoor functions allow further optimizations of the security proof.

3.5.1 PSF-based Signatures

In Section 2.3.1 we gave a brief overview of lattice-based Hash-and-Sign schemes within the GPV framework. The trapdoor functions underlying these signature schemes are Preimage Sampleable Functions (PSF). In the following, we consider the notion of PSFs given in Definition 2.6.

First, since for a PSF T it holds that $\mathrm{Adv}_{\mathsf{T}}^{\mathrm{OW}}(\mathcal{A}) = \mathrm{Adv}_{\mathsf{T}}^{\mathrm{INV}}(\mathcal{B})$, when adapting the security proof of Theorem 3.8 we can obtain a reduction from OW. Then, notice that due to Property 3 of Definition 2.6, the signature associated with a PSF can be described in the PHaS without retry paradigm. In particular, in the aggregation procedure of Algorithm 3.2 we can remove the repeat loop on Line 3. It follows that $\mathsf{q}_{\mathsf{S}}' = \mathsf{q}_{\mathsf{S}}$ in Theorem 3.8. Finally, we can remove the advantage of the PS adversary and obtain a tighter reduction when the PSF is collision resistant.

One-way PSFs

We start by considering a PSF without collision resistance. We obtain the following reduction from OW to PS-HF-UF-CMA.

Theorem 3.12. *Let* T *be a PSF. Let* $\mathcal{A}$ *be a strong* PS-HF-UF-CMA *adversary against the* HaS-HF-SAS *scheme on* T *in the random oracle model, which runs in time t and makes* $\mathsf{q_S}$ *signing queries,* $\mathsf{q_H}$ *queries to the random oracle* H *and* $\mathsf{q_G}$ *queries to the random oracle* G*. Then, there exists an* OW *adversary* $\mathcal{B}$ *against* T *that runs in time* $t + \mathcal{O}((\mathsf{q_H} + \mathsf{q_S} + 1) \cdot \mathrm{poly}(\mathrm{len}(\mathcal{X}), \mathrm{len}(\mathcal{Y})))$*, such that*

$$\begin{aligned}\mathsf{Adv}^{\text{PS-HF-UF-CMA}}_{\text{HaS-HF-SAS}}(\mathcal{A}) \leq & (\psi \mathsf{q_H}) \cdot \mathsf{Adv}^{\text{OW}}_{\mathsf{T}}(\mathcal{B}) + \frac{(\mathsf{q_S}+\mathsf{q_H})(\mathsf{q_S}+\mathsf{q_H}+\mathsf{q_G})}{2^{2\lambda}} \\ & + \frac{\mathsf{q_S}(\mathsf{q_S}+\mathsf{q_H})}{|\mathsf{R}|} + \frac{\psi \mathsf{q_H}^2}{2|\mathcal{Y}|} + \frac{(\psi \mathsf{q_H})^{\psi+1}|\mathcal{X}|}{(\psi+1)! \cdot |\mathcal{Y}|^{\psi+1}},\end{aligned}$$

where $\psi \geq \lceil \mathrm{len}(\mathcal{X}) / \mathrm{len}(\mathcal{Y}) \rceil$.

Proof. The proof is identical to that of Theorem 3.8, except that the distinguishing advantage of the PS adversary is 0. In fact, the PS notion of Definition 2.8 is a weaker condition for indistinguishability on preimages than Property 2 of Definition 2.6. As a result, we can modify the reduction of Algorithm 3.3 by merging Game$_2$ and Game$_3$ in Argument 3.1, without the need to introduce the PS adversary. □

Remark. Notice that the proof share the same reduction from OW, incurring in a loss of a factor proportional to $\mathsf{q_H}$.

Collision-Resistant PSFs

Conversely, if we consider a collision-resistant PSF, we can further modify Theorem 3.8 to obtain a tighter reduction from CR to PS-HF-UF-CMA. In fact, when the reduction is performed from the OW/INV game as in the original proof of Theorem 3.8, the OW adversary provides their challenge to the PS-HF-UF-CMA adversary in one of the $\mathsf{q_H}$ queries to the random oracle H. This results in a multiplicative loss of advantage by a factor $\mathsf{q_H}$. However, when the reduction is performed from collision-resistance, the CR adversary can prepare responses that will lead to a collision in each query to H involving the target public key. As a result, we get a tight reduction with only negligible losses from additive terms.

Theorem 3.13. *Let* T *be a collision-resistant PSF. Let* $\mathcal{A}$ *be a strong* PS-HF-UF-CMA *adversary against the* HaS-HF-SAS *scheme on* T *in the random oracle model, which runs in time t and makes* $\mathsf{q_S}$ *signing queries,* $\mathsf{q_H}$ *queries to the random oracle* H *and*

$\mathsf{q_G}$ *queries to the random oracle* G. *Then, there exists a* CR *adversary* $\mathcal{B}$ *against* T *that runs in time* $t + \mathcal{O}((\mathsf{q_H} + \mathsf{q_S} + 1) \cdot \mathsf{poly}(\mathrm{len}(\mathcal{X}), \mathrm{len}(\mathcal{Y})))$, *such that*

$$\mathsf{Adv}^{\mathsf{PS\text{-}HF\text{-}UF\text{-}CMA}}_{\mathsf{HaS\text{-}HF\text{-}SAS}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{CR}}_{\mathsf{T}}(\mathcal{B}) + \frac{(\mathsf{q_S} + \mathsf{q_H})(\mathsf{q_S} + \mathsf{q_H} + \mathsf{q_G})}{2^{2\lambda}} + \frac{\mathsf{q_S}(\mathsf{q_S} + \mathsf{q_H})}{|\mathsf{R}|} + \frac{\psi \mathsf{q_H^2}}{2|\mathcal{Y}|} + \frac{(\psi \mathsf{q_H})^{\psi+1}|\mathcal{X}|}{(\psi+1)! \cdot |\mathcal{Y}|^{\psi+1}},$$

where $\psi \geq \lceil \mathrm{len}(\mathcal{X}) / \mathrm{len}(\mathcal{Y}) \rceil$.

Proof. We modify the proof of Theorem 3.12 by introducing a new intermediate game Game_{5b}. This game is identical to Game_5, except for how, on the simulation of the random oracle H on query $Q = (\mathsf{F}, m, r, x)$, the random oracle G is programmed when $\mathsf{F} = \mathsf{F}^\star$. In particular, in Algorithm 3.3, Line 21 is modified by taking a random input $x' \leftarrow_\$ X$, assigning $y' \leftarrow F^\star(x')$ and programming the G on input h' with $y' \oplus \alpha$. The modified game is described in Argument 3.3. Here, the simulation of G is correct since, from Property 1 of Definition 2.6, y' obtained as $F^\star(\mathcal{U}(\mathcal{X}))$ is uniformly distributed in $\mathcal{Y}$. After the adversary returns a forged aggregate signature, if none of the bad events happen, the value $x_{i^\star}$, from the forgery, and the value x', produced by H and stored in the HTree, constitute a collision for $F^\star$.

Notice that the above changes, does not affect the analysis of the bad events. In particular, all events of Theorem 3.8 occur with the same probability. □

3.5.2 APSF-based Signatures

In the following, we consider the notion of Average PSFs given in Definition 2.7. Similarly to the previous case of PSFs, for an APSF T it holds that $\mathsf{Adv}^{\mathsf{OW}}_{\mathsf{T}}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{INV}}_{\mathsf{T}}(\mathcal{B})$, and following Property 1 of Definition 2.7 the signature associated with an APSF can be described in the PHaS without retry paradigm. Therefore, when adapting the security proof of Theorem 3.8 we can remove the repeat loop in the aggregation procedure, and obtain a reduction from OW with $\mathsf{q_S'} = \mathsf{q_S}$. Moreover, we can use the weakened uniformity property on preimage to bound the advantage of the PS adversary. We obtain the following reduction from OW to PS-HF-UF-CMA.

Theorem 3.14. *Let* T *be an* ε*-APSF. Let* $\mathcal{A}$ *be a strong* PS-HF-UF-CMA *adversary against the* HaS-HF-SAS *scheme on* T *in the random oracle model, which runs in time* t *and makes* $\mathsf{q_S}$ *signing queries,* $\mathsf{q_H}$ *queries to the random oracle* H *and* $\mathsf{q_G}$ *queries to the random oracle* G. *Then, there exists an* OW *adversary* $\mathcal{B}$ *against* T *that runs in time*

Hybrid Argument 3.3: Modified Game$_5$ for Collision-Resistant PSFs on $\mathsf{H}(\mathsf{F}, m, r, x)$

Game$_5$:

```
$Q \leftarrow (\mathsf{F}, m, r, x)$
$c \leftarrow c + 1$
if $\mathsf{HT}[Q] = \bot$ then
    $\eta \leftarrow_\$ \{0,1\}^{2\lambda}$
    $\mathsf{HT}[Q] \leftarrow \eta$
    $\mathsf{NList} \leftarrow \mathsf{Lookup}(x)$
    if $\mathsf{F} = \mathsf{F}^\star \wedge c = c^\star$ then
        $i^\star \leftarrow_\$ [|\mathsf{NList}|]$
    for $i \in [|\mathsf{NList}|]$ do
        $N_i \leftarrow \mathsf{NList}[i]$
        $N_i' \leftarrow$ new node with parent $N_i$
        Retrieve $h_i$ from $N_i$
        $h_i' \leftarrow h_i \oplus \eta$
        $(\alpha, \beta) \leftarrow \mathsf{enc}(x)$
        if $\mathsf{GT}[h_i'] \neq \bot$ then
            raise $\mathsf{bad}_{\mathsf{gcol2}}$
        if $\mathsf{F} \neq \mathsf{F}^\star \vee c \neq c^\star \vee i \neq i^\star$ then
            $g_i' \leftarrow \mathsf{G}(h_i')$
            $y_i' \leftarrow g_i' \oplus \alpha$
        else
            $y_i' \leftarrow y^\star$
        $\mathsf{GT}[h_i'] \leftarrow y_i' \oplus \alpha$
        $N_i' \leftarrow (Q, \eta, h_i', y_i')$
return $\mathsf{HT}[Q]$
```

Game$_{5b}$:

```
$Q \leftarrow (\mathsf{F}, m, r, x)$
if $\mathsf{HT}[Q] = \bot$ then
    $\eta \leftarrow_\$ \{0,1\}^{2\lambda}$
    $\mathsf{HT}[Q] \leftarrow \eta$
    $\mathsf{NList} \leftarrow \mathsf{Lookup}(x)$
    for $i \in [|\mathsf{NList}|]$ do
        $N_i \leftarrow \mathsf{NList}[i]$
        $N_i' \leftarrow$ new node with parent $N_i$
        Retrieve $h_i$ from $N_i$
        $h_i' \leftarrow h_i \oplus \eta$
        $(\alpha, \beta) \leftarrow \mathsf{enc}(x)$
        if $\mathsf{GT}[h_i'] \neq \bot$ then
            raise $\mathsf{bad}_{\mathsf{gcol2}}$
        if $\mathsf{F} \neq \mathsf{F}^\star$ then
            $g_i' \leftarrow \mathsf{G}(h_i')$
            $y_i' \leftarrow g_i' \oplus \alpha$
        else
            $x_i' \leftarrow_\$ \mathcal{X}$
            $y_i' \leftarrow \mathsf{F}^\star(x_i')$
        $\mathsf{GT}[h_i'] \leftarrow y_i' \oplus \alpha$
        $N_i' \leftarrow (Q, \eta, h_i', y_i')$
return $\mathsf{HT}[Q]$
```

$t + \mathcal{O}((\mathsf{q_H} + \mathsf{q_S} + 1) \cdot \mathsf{poly}(\mathsf{len}(\mathcal{X}), \mathsf{len}(\mathcal{Y})))$, *such that*

$$\mathsf{Adv}^{\mathsf{PS\text{-}HF\text{-}UF\text{-}CMA}}_{\mathsf{HaS\text{-}HF\text{-}SAS}}(\mathcal{A}) \leq (\psi \mathsf{q_H}) \cdot \mathsf{Adv}^{\mathsf{OW}}_{\mathsf{T}}(\mathcal{B}) + \mathsf{q_S}\varepsilon + \frac{(\mathsf{q_S} + \mathsf{q_H})(\mathsf{q_S} + \mathsf{q_H} + \mathsf{q_G})}{2^{2\lambda}} + \frac{\mathsf{q_S}(\mathsf{q_S} + \mathsf{q_H})}{|\mathsf{R}|} + \frac{\psi \mathsf{q_H}^2}{2|\mathcal{Y}|} + \frac{(\psi \mathsf{q_H})^{\psi+1} |\mathcal{X}|}{(\psi + 1)! \cdot |\mathcal{Y}|^{\psi+1}},$$

where $\psi \geq \lceil \mathsf{len}(\mathcal{X}) / \mathsf{len}(\mathcal{Y}) \rceil$.

Proof. The proof is identical to that of Theorem 3.8, except that the distinguishing advantage of the PS adversary can be bounded with $\mathsf{q_S}\varepsilon$. In fact, from Property 2 of Definition 2.7, consider the statistical distance $\varepsilon_{\mathsf{F,I}} = \Delta(\mathsf{SampDom}(\mathsf{F}), \mathsf{I}(\mathcal{U}(\mathcal{Y})))$. Then, it holds that $\mathbb{E}_{(\mathsf{F,I})}[\varepsilon_{\mathsf{F,I}}] \leq \varepsilon$, where ε is negligible in the security parameter λ. We can use this condition and [52, Prop. 1] to bound the distinguishing advantage on PS with ε, obtaining $\mathsf{Adv}^{\mathsf{PS}}_{\mathsf{T_{wave}}}(\mathcal{D}) \leq \mathsf{q_S}\varepsilon$. □

3.6 Instantiation and Evaluation

In this section, we will provide some concrete applications of the HaS-HF-SAS of Section 3.4 to MQ-based and code-based HaS signature schemes. In particular, we analyse the compression capabilities of the scheme when instantiated with UOV, MAYO, and Wave. More details on the trapdoor functions and the application of Theorem 3.8 to these schemes are given in Section 2.3. Our scheme could also be extended to lattice-based schemes, such as the NIST PQC selected algorithm Falcon [172], and generally with PSF-based signatures [108]. In particular, applying our construction would allow achieving history-free aggregation over existing schemes[4]. However, we already noted how different design choices become feasible due to the additional properties of trapdoor PSF. Accordingly, a direct application would lead to unnecessary loss of efficiency. An analysis of how the HaS-HF-SAS scheme can be modified with PSF can be found in Section 3.5.1.

The main measure of the efficiency of an aggregate signature scheme is the *compression rate*, i.e., the reduction in the length of the aggregate signature $\bar{\Sigma}_N$ of N users compared to the trivial concatenation of N individual signatures σ. The compression rate of N signatures is defined as $\tau(N) = 1 - \frac{|\bar{\Sigma}_N|}{N \cdot |\sigma|}$. An HaS-HF-SAS signature of N users is the output of the Combine algorithm on $\varsigma_1, \dots, \varsigma_{N-1}, \Sigma_N$ and is given by $\bar{\Sigma}_N = (\vec{r}_N, \vec{\beta}_{N-1}, h_N, x)$. An individual signature of a generic HaS scheme as described in Section 2.2 is given by $\sigma = (r, x)$. In the following, we assume that the aggregation scheme is applied to the signature scheme without further possible optimization, so that we have the same size for salts $|r| = \mathrm{len}(\mathrm{R})$ and preimages $|x| = \mathrm{len}(\mathcal{X})$, where $\mathrm{len}(X)$ denotes the bit size of an element in X.

The compression rate is thus given by

$$\begin{aligned} \tau(N) &= 1 - \frac{N \cdot \mathrm{len}(\mathrm{R}) + (N-1) \cdot \mathrm{len}(\mathcal{X}') + 2\lambda + \mathrm{len}(\mathcal{X})}{N \cdot (\mathrm{len}(\mathrm{R}) + \mathrm{len}(\mathcal{X}))} \\ &= 1 - \frac{N \cdot (\mathrm{len}(\mathrm{R}) + \mathrm{len}(\mathcal{X}) - \mathrm{len}(\mathcal{Y})) + 2\lambda + \mathrm{len}(\mathcal{Y})}{N \cdot (\mathrm{len}(\mathrm{R}) + \mathrm{len}(\mathcal{X}))} \\ &= \frac{\mathrm{len}(\mathcal{Y})}{\mathrm{len}(\mathrm{R}) + \mathrm{len}(\mathcal{X})} - \frac{2\lambda + \mathrm{len}(\mathcal{Y})}{N \cdot (\lambda + \mathrm{len}(\mathcal{X}))}. \end{aligned} \tag{3.2}$$

Notice that the aggregate signature is smaller than the trivial concatenation whenever $N > \frac{2\lambda}{\mathrm{len}(\mathcal{Y})} + 1$, which for typical parameters is as soon as $N > 2$.

In [92, 56], the size of an aggregate signature of N users is $N \cdot (\mathrm{len}(\mathcal{X}) - \mathrm{len}(\mathcal{Y})) + \mathrm{len}(\mathcal{Y})$. Compared to our scheme, we have a small overhead of $N \,\mathrm{len}(\mathrm{R}) + 2\lambda$ bits due to the aggregated hash state and the random salts. However, we already

[4] An attack on the previously unique history-free SAS of [191] was recently proposed in [48].

argued that this increase in signature size is necessary to guarantee the security of the scheme.

3.6.1 Original Unbalanced Oil and Vinegar

We consider the trapdoor function $\mathsf{T}_{\mathsf{uov}}$ described in Section 2.3.3 and the parameters proposed in [31] with respect to NIST security levels I, III, and V. For UOV, the domain $\mathcal{X}$ is given by $\mathbb{F}_q^n$ with elements of length $\mathrm{len}(\mathcal{X}) = n\lceil \log_2 q\rceil$. The codomain $\mathcal{Y}$ is $\mathbb{F}_q^m$ with elements of length $\mathrm{len}(\mathcal{Y}) = m\lceil \log_2 q\rceil$.

$\mathsf{T}_{\mathsf{uov}}$ is a non-invertible trapdoor function; therefore we can build a strongly PS-HF-UF-CMA sequential aggregate signature instantiated with UOV by applying Theorem 3.8.

Corollary 3.15. *Let $\mathcal{A}$ be a strong* PS-HF-UF-CMA *adversary against the* HaS-HF-SAS *scheme on $\mathsf{T}_{\mathsf{uov}}$ in the random oracle model, which makes $\mathsf{q_S}$ signing queries, $\mathsf{q_H}$ queries to the random oracle* H *and $\mathsf{q_G}$ queries to the random oracle* G. *Then, there exists an* INV *adversary $\mathcal{B}$ against $\mathsf{T}_{\mathsf{uov}}$, and a* PS *adversary $\mathcal{D}$ against $\mathsf{T}_{\mathsf{uov}}$ issuing $\mathsf{q_S}$ sampling queries, such that*

$$\mathsf{Adv}^{\mathsf{PS\text{-}HF\text{-}UF\text{-}CMA}}_{\mathsf{HaS\text{-}HF\text{-}SAS}}(\mathcal{A}) \leq (\psi \mathsf{q_H}) \cdot \mathsf{Adv}^{\mathsf{INV}}_{\mathsf{T_{uov}}}(\mathcal{B}) + \mathsf{Adv}^{\mathsf{PS}}_{\mathsf{T_{uov}}}(\mathcal{D}) + \frac{(\mathsf{q_S}+\mathsf{q_H})(\mathsf{q_S}+\mathsf{q_H}+\mathsf{q_G})}{2^{2\lambda}}$$
$$+ \frac{\mathsf{q_S}(\mathsf{q_S}+\mathsf{q_H})}{|\mathsf{R}|} + \frac{\psi \mathsf{q_H^2}}{2q^m} + \frac{(\psi \mathsf{q_H})^{\psi+1} q^n}{(\psi+1)! \cdot q^{m(\psi+1)}},$$

where $\psi \geq \lceil n/m \rceil$, and the running time of $\mathcal{B}$ and $\mathcal{D}$ are about that of $\mathcal{A}$.

Proof. We directly apply the proof of Theorem 3.8 to $\mathsf{T}_{\mathsf{uov}}$. Recall that the UOV signature lies in the PHaS *without* retry paradigm, therefore $\mathsf{q}'_{\mathsf{S}} = \mathsf{q_S}$ holds in Theorem 3.8. On the other hand, the PS advantage term $\mathsf{Adv}^{\mathsf{PS}}_{\mathsf{T_{uov}}}(\mathcal{D})$ cannot be omitted since signature simulation requires knowledge of the trapdoor function. □

For typical parameters, n is chosen equal to $2.5m$. If we choose $\psi = 3$, the additive error terms in Corollary 3.15 are negligible for each parameterization in Table 2.4.

Regardless of the security level, the parameterizations of [31] use 128-bit salts. In our comparison, we consider salts of length $\lambda = 128{,}192$ and 256 bits, respectively. The size of a single UOV signature is then given by

$$|\sigma| = n\lceil \log_2 q\rceil + \lambda.$$

The size of a sequential aggregate signature instantiated with UOV is given by

$$|\bar{\Sigma}_N| = N \cdot (\lambda + (n-m)\lceil \log_2 q\rceil) + 2\lambda + m\lceil \log_2 q\rceil.$$

Table 3.1: Aggregate signature sizes and compression rates for UOV [31].

Parameter NIST SL	ov-Ip I	ov-Is I	ov-III III	ov-V V
(n, m, q)	(112,44,256)	(160,64,16)	(184,72,256)	(244,96,256)
len(R) (bytes)	16	16	24	32
$N \cdot \lvert\sigma\rvert$ (bytes)	$128 \cdot N$	$96 \cdot N$	$208 \cdot N$	$276 \cdot N$
$\lvert\bar{\Sigma}_N\rvert$ (bytes)	$84 \cdot N + 76$	$64 \cdot N + 64$	$136 \cdot N + 120$	$180 \cdot N + 160$
$\tau(5)$	0.23	0.20	0.23	0.23
$\tau(20)$	0.31	0.30	0.32	0.32
$\tau(100)$	0.34	0.33	0.34	0.34
Asym. $\tau(N)$	0.34	0.33	0.35	0.35

Table 3.2: Aggregate signature sizes and compression rates for PROV [115].

Parameter NIST SL	PROV-I I	PROV-III III	PROV-V V
(n, m, δ, q)	(142,49,8,256)	(206,74,8,256)	(270,100,8,256)
len(R) (bytes)	24	32	40
$N \cdot \lvert\sigma\rvert$ (bytes)	$166 \cdot N$	$238 \cdot N$	$310 \cdot N$
$\lvert\bar{\Sigma}_N\rvert$ (bytes)	$117 \cdot N + 97$	$164 \cdot N + 138$	$210 \cdot N + 180$
$\tau(5)$	0.18	0.19	0.21
$\tau(20)$	0.27	0.28	0.29
$\tau(100)$	0.29	0.31	0.32
Asym. $\tau(N)$	0.30	0.31	0.32

Table 3.3: Aggregate signature sizes and compression rates for MAYO [30].

Parameter NIST SL	MAYO_1 I	MAYO_2 I	MAYO_3 III	MAYO_5 V
(n, m, o, k, q)	(66,64,8,9,16)	(78,64,18,4,16)	(99,96,10,11,16)	(133,128,12,12,16)
len(R) (bytes)	24	24	32	40
$N \cdot \lvert\sigma\rvert$ (bytes)	$321 \cdot N$	$180 \cdot N$	$577 \cdot N$	$838 \cdot N$
$\lvert\bar{\Sigma}_N\rvert$ (bytes)	$289 \cdot N + 64$	$148 \cdot N + 64$	$529 \cdot N + 96$	$774 \cdot N + 128$
$\tau(5)$	0.06	0.11	0.05	0.05
$\tau(20)$	0.09	0.16	0.07	0.07
$\tau(100)$	0.10	0.17	0.08	0.07
Asym. $\tau(N)$	0.10	0.18	0.08	0.08

Applying Equation (3.2), we can see that the compression rate asymptotically goes to $m\lceil\log_2 q\rceil/(\lambda + n\lceil\log_2 q\rceil)$. Concrete numbers for different security parameters and the number of signers are given in Table 3.1.

3.6.2 Provable Unbalanced Oil and Vinegar

We consider the trapdoor function $\mathsf{T}_{\mathsf{puov}}$ described in Section 2.3.3 and the parameters proposed in [115] with respect to NIST security levels I, III, and V. For PUOV, the domain $\mathcal{X}$ is given by $\mathbb{F}_q^n$ with elements of length $\mathrm{len}(\mathcal{X}) = n\lceil\log_2 q\rceil$. The codomain $\mathcal{Y}$ is $\mathbb{F}_q^m$ with elements of length $\mathrm{len}(\mathcal{Y}) = m\lceil\log_2 q\rceil$.

$\mathsf{T}_{\mathsf{puov}}$ is a non-invertible trapdoor function; therefore we can build a strongly PS-HF-UF-CMA sequential aggregate signature instantiated with PUOV by applying Theorem 3.8.

Corollary 3.16. *Let $\mathcal{A}$ be a strong* PS-HF-UF-CMA *adversary against the* HaS-HF-SAS *scheme on $\mathsf{T}_{\mathsf{puov}}$ in the random oracle model, which makes q_S signing queries, q_H queries to the random oracle* H *and q_G queries to the random oracle* G*. Then, there exists an* INV *adversary $\mathcal{B}$ against $\mathsf{T}_{\mathsf{puov}}$, such that*

$$\mathsf{Adv}^{\text{PS-HF-UF-CMA}}_{\text{HaS-HF-SAS}}(\mathcal{A}) \leq (\psi \mathsf{q}_\mathsf{H}) \cdot \mathsf{Adv}^{\mathsf{INV}}_{\mathsf{T}_{\mathsf{puov}}}(\mathcal{B}) + \frac{(\mathsf{q}_\mathsf{S} + \mathsf{q}_\mathsf{H})(\mathsf{q}'_\mathsf{S} + \mathsf{q}_\mathsf{H} + \mathsf{q}_\mathsf{G})}{2^{2\lambda}} + \frac{\mathsf{q}_\mathsf{S}(\mathsf{q}'_\mathsf{S} + \mathsf{q}_\mathsf{H})}{|\mathsf{R}|} + \frac{\psi \mathsf{q}_\mathsf{H}^2}{2q^m} + \frac{(\psi \mathsf{q}_\mathsf{H})^{\psi+1} q^n}{(\psi+1)! \cdot q^{m(\psi+1)}},$$

where $\psi \geq \lceil n/m \rceil$, q'_S is a bound on the total number of queries to H *in all the signing queries, and the running time of $\mathcal{B}$ is about that of $\mathcal{A}$.*

Proof. Recall that the trapdoor $\mathsf{I}_{\mathsf{puov}}$ of PUOV can be split in two distinct functions $\mathsf{I}^1_{\mathsf{puov}}$ and $\mathsf{I}^2_{\mathsf{puov}}$ as described in Algorithm 2.5. To apply Theorem 3.8 with a split trapdoor, we can modify the signing oracle OAggSign for Game_0-Game_2 in Argument 3.1 and Sample_0 in the PS game (Experiment 2.2). In particular, we add $\boldsymbol{v} \leftarrow_\$ \mathsf{I}^1_{\mathsf{puov}}$ at the start of the algorithm, and we replace $\boldsymbol{x}' \leftarrow_\$ \mathsf{I}(\boldsymbol{y}')$ with $\boldsymbol{x}' \leftarrow_\$ \mathsf{I}^2_{\mathsf{puov}}(\boldsymbol{v}, \boldsymbol{y}')$. Then, the PS adversary $\mathcal{D}$ can simulate both Game_2 and Game_3 by playing PS_0 and PS_1, respectively. Moreover, in Section 2.3.3, we show that the preimages produced from $\mathsf{Sign}(\mathsf{I}_{\mathsf{puov}}, \cdot)$ are indistinguishable from the output of $\mathsf{SampDom}(\mathsf{F}_{\mathsf{puov}})$, so that $\mathsf{Adv}^{\mathsf{PS}}_{\mathsf{T}_{\mathsf{puov}}}(\mathcal{D}) = 0$. □

Remark. Unlike Corollary 3.15, we cannot explicitly take $\mathsf{q}'_\mathsf{S} = \mathsf{q}_\mathsf{S}$, since in $\mathsf{I}^2_{\mathsf{puov}}$ the probability of $\boldsymbol{x} \neq \perp$ depends on the fixed value of $\boldsymbol{v}$ sampled in $\mathsf{I}^1_{\mathsf{puov}}$. Depending on the concrete parameters of $\mathsf{T}_{\mathsf{puov}}$, we can give a meaningful bound on q'_S so that the probability of having a number of queries to H greater than q'_S is negligible. $\mathsf{I}^2_{\mathsf{puov}}$ returns $\perp$ on input $(\boldsymbol{v}, \boldsymbol{t})$ if $\mathcal{P}'(\boldsymbol{v}, \cdot)$ does not have full rank and $\boldsymbol{t} - \mathcal{P}(\boldsymbol{v})$ does not belong to the image of $\mathcal{P}'(\boldsymbol{v}, \cdot)$. Let q_{ret} be a bound for the number of queries to H in each signing query and let X_i be a random variable on the actual number of queries to H in the i-th query. Then

$$\Pr[X_i > \mathsf{q}_{ret}] = \sum_{j=1}^{m} \Pr[\operatorname{rank}(\mathcal{P}'(\boldsymbol{v}, \cdot)) = j](1 - q^{j-m})^{\mathsf{q}_{ret}}.$$

As done in [175], we can assume that for a random $\boldsymbol{v} \leftarrow_{\$} \mathbb{F}_q^n$, $\mathcal{P}'(\boldsymbol{v}, \cdot)$ is distributed as a random $o \times m$ matrix. For $o \geq m$, the probability that a random $o \times m$ matrix over $\mathbb{F}_q$ has rank $1 \leq j \leq m$ is given in [139]:

$$q^{(j-m)(o-j)} \frac{\prod_{k=o-j+1}^{o}(1 - q^{-k}) \prod_{k=m-j+1}^{m}(1 - q^{-k})}{\prod_{k=1}^{j}(1 - q^{-k})}. \tag{3.3}$$

Then, if we choose q_{ret} such that $\mathsf{q}_\mathsf{S} \Pr[X_i > \mathsf{q}_{ret}]$ is negligible, we can use $\mathsf{q}'_\mathsf{S} = \mathsf{q}_{ret}\mathsf{q}_\mathsf{S}$ in the bound of the corollary.

The parameters of PROV in [115] are selected so that the dimension of the trapdoor subspace is $o = m + \delta$. This choice significantly reduces the probability of Equation (3.3) whenever $j < m$. For instance, with the parameters of PROV-I we have $\Pr[X_i > 1] \leq 2^{-72}$ and $\Pr\left[X_i > 2^{14}\right] \leq 2^{-160}$.

Similarly to Original UOV, if we choose $\psi = 3$, the additive error terms in Corollary 3.16 are negligible for each parameterization in [115].

In [115], the salt length $\operatorname{len}(\mathsf{R})$ is slightly longer than the security parameter λ for consistency with the security proof, and is given by $\operatorname{len}(\mathsf{R}) = \lambda + 64$. The size of a single PROV signature is then given by

$$|\sigma| = n\lceil \log_2 q \rceil + \operatorname{len}(\mathsf{R}).$$

The size of a sequential aggregate signature instantiated with PROV is given by

$$|\bar{\Sigma}_N| = N \cdot (\operatorname{len}(\mathsf{R}) + (n - m)\lceil \log_2 q \rceil) + 2\lambda + m\lceil \log_2 q \rceil.$$

Applying Equation (3.2), we can see that the compression rate asymptotically goes to $m\lceil \log_2 q \rceil / (\operatorname{len}(\mathsf{R}) + n\lceil \log_2 q \rceil)$. Concrete numbers for different security parameters and the number of signers are given in Table 3.2.

3.6.3 MAYO

We consider the trapdoor function $\mathsf{T}_{\mathsf{mayo}}$ described in Section 2.3.3 and the parameters proposed in [30] with respect to NIST security levels I, III, and V. For MAYO, the domain $\mathcal{X}$ is given by $\mathbb{F}_q^{kn}$ with elements of length $\operatorname{len}(\mathcal{X}) = kn\lceil \log_2 q \rceil$. The codomain $\mathcal{Y}$ is $\mathbb{F}_q^m$ with elements of length $\operatorname{len}(\mathcal{Y}) = m\lceil \log_2 q \rceil$.

$\mathsf{T}_{\mathsf{mayo}}$ is a non-invertible trapdoor function; therefore we can build a strongly PS-HF-UF-CMA sequential aggregate signature instantiated with MAYO by applying Theorem 3.8.

Corollary 3.17. *Let $\mathcal{A}$ be a strong* PS-HF-UF-CMA *adversary against the* HaS-HF-SAS *scheme on* $\mathsf{T}_{\mathsf{mayo}}$ *in the random oracle model, which makes* $\mathsf{q_S}$ *signing queries,* $\mathsf{q_H}$ *queries to the random oracle* H *and* $\mathsf{q_G}$ *queries to the random oracle* G. *Then, there exists an* INV *adversary $\mathcal{B}$ against* $\mathsf{T}_{\mathsf{mayo}}$*, such that*

$$\mathsf{Adv}^{\text{PS-HF-UF-CMA}}_{\text{HaS-HF-SAS}}(\mathcal{A}) \leq \frac{\psi \mathsf{q_H}}{1-\mathsf{q_S}\mathsf{B}} \cdot \mathsf{Adv}^{\text{INV}}_{\mathsf{T}_{\mathsf{mayo}}}(\mathcal{B}) + \frac{(\mathsf{q_S}+\mathsf{q_H})(\mathsf{q_S}+\mathsf{q_H}+\mathsf{q_G})}{2^{2\lambda}} + \frac{\mathsf{q_S}(\mathsf{q_S}+\mathsf{q_H})}{|\mathsf{R}|} + \frac{\psi \mathsf{q}_\mathsf{H}^2}{2q^m} + \frac{(\psi \mathsf{q_H})^{\psi+1} q^{kn}}{(\psi+1)! \cdot q^{m(\psi+1)}},$$

where $\psi \geq \lceil kn/m \rceil$, and the running time of $\mathcal{B}$ is about that of $\mathcal{A}$.

Proof. In Section 2.3.3, we observed that that if $\mathsf{I}_{\mathsf{mayo}}$ has never output $\perp$, then the preimages produced by $\mathsf{Sign}(\mathsf{I}_{\mathsf{mayo}}, \cdot)$ are indistinguishable from $\mathsf{SampDom}(\mathsf{F}_{\mathsf{mayo}})$. In [29], the author provides an explicit bound on the failure probability, which we denote as B.

To apply the proof of Theorem 3.8 to $\mathsf{T}_{\mathsf{mayo}}$, we introduce a new intermediate game Game_{2b}. This game is identical to Game_2 except that OAggSign aborts if $\mathsf{I}_{\mathsf{mayo}}$ outputs $\perp$. Since at most $\mathsf{q_S}$ queries are made to OAggSign, the probability that Game_{2b} does not abort is at least $1-\mathsf{q_S}\mathsf{B}$. It follows that $\Pr[\mathsf{Game}_2(\mathcal{A})=1] \leq \frac{1}{1-\mathsf{q_S}\mathsf{B}} \Pr[\mathsf{Game}_{2b}(\mathcal{A})=1]$. Now, when Game_{2b} does not abort, the game is indistinguishable from Game_3, so that $\Pr[\mathsf{Game}_3(\mathcal{A})=1] = \Pr[\mathsf{Game}_{2b}(\mathcal{A})=1]$. The remainder of the proof proceeds as the original without the need to introduce the PS adversary $\mathcal{D}$. Finally, since MAYO now does not repeat any signature attempts, we can set $\mathsf{q}'_\mathsf{S} = \mathsf{q_S}$. □

In order to choose appropriate values for ψ, it is necessary to consider the whipped map $\mathcal{P}^*\colon \mathbb{F}_q^{kn} \to \mathbb{F}_q^m$, from which $\psi \geq \lceil kn/m \rceil$. If we consider the parameter sets of Table 2.6, we can choose ψ equal to 10,5,12 and 13 for MAYO_1, MAYO_2, MAYO_3 and MAYO_5 respectively, to obtain negligible additive terms in Corollary 3.17.

Similarly to PROV, in [30] the salt length len(R) is slightly longer than the security parameter λ and is again given by $\mathsf{len}(\mathsf{R}) = \lambda + 64$. The size of a single MAYO signature is then given by

$$|\sigma| = kn\lceil \log_2 q \rceil + \mathsf{len}(\mathsf{R}).$$

The size of a sequential aggregate signature instantiated with MAYO is given by

$$|\bar{\Sigma}_N| = N \cdot (\mathsf{len}(\mathsf{R}) + (kn-m)\lceil \log_2 q \rceil) + 2\lambda + m\lceil \log_2 q \rceil.$$

Applying Equation (3.2), we can see that the compression rate asymptotically goes to $m\lceil \log_2 q \rceil / (\mathsf{len}(\mathsf{R}) + nk\lceil \log_2 q \rceil)$. Concrete numbers for different security parameters and the number of signers are given in Table 3.3.

Table 3.4: Aggregate signature sizes and compression rates for Wave [71].

Parameter	128g
NIST SL	I
(n, k, w, q)	(8492,5605,7980,3)
len(R) (bytes)	16
$N \cdot \lvert\sigma\rvert$ (bytes)	$1699 \cdot N$
$\lvert\bar{\Sigma}_N\rvert$ (bytes)	$1127 \cdot N + 604$
$\tau(5)$	0.27
$\tau(20)$	0.32
$\tau(100)$	0.33
Asym. $\tau(N)$	0.34

3.6.4 Wave

We consider the trapdoor function $\mathsf{T}_{\mathsf{wave}}$ described in Section 2.3.2 and the parameters proposed in [71] with respect to NIST security level I. Notice that this is not the same scheme later submitted to NIST's call for additional digital signatures. The submitted scheme incorporates an optimization derived from the Wavelet variant [16], which cannot be used during aggregation and for which only an asymptotically trivial compression rate can be obtained. More details on what types of optimizations can be employed during aggregation are provided at the end of this section.

For Wave, the domain $\mathcal{X}$ is given by $S_{w,n}$, the subset of $\mathbb{F}_q^n$ with vectors of Hamming weight w, with elements of length $\mathrm{len}(\mathcal{X}) = \lceil n \log_2 q \rceil$. The codomain $\mathcal{Y}$ is $\mathbb{F}_q^{n-k}$ with elements of length $\mathrm{len}(\mathcal{Y}) = \lceil (n-k) \log_2 q \rceil$.

$\mathsf{T}_{\mathsf{wave}}$ is a one-way ε-APSF; therefore we can build a strongly PS-HF-UF-CMA sequential aggregate signature instantiated with Wave by applying Theorem 3.14.

Corollary 3.18. *Let $\mathcal{A}$ be a strong* PS-HF-UF-CMA *adversary against the* HaS-HF-SAS *scheme on the ε-APSF* $\mathsf{T}_{\mathsf{wave}}$ *in the random oracle model, which makes $\mathsf{q_S}$ signing queries, $\mathsf{q_H}$ queries to the random oracle* H *and $\mathsf{q_G}$ queries to the random oracle* G*. Then, there exists an* OW *adversary $\mathcal{B}$ against* $\mathsf{T}_{\mathsf{wave}}$*, such that*

$$\begin{aligned}\mathsf{Adv}^{\text{PS-HF-UF-CMA}}_{\text{HaS-HF-SAS}}(\mathcal{A}) \leq & (\psi \mathsf{q_H}) \cdot \mathsf{Adv}^{\mathsf{OW}}_{\mathsf{T_{wave}}}(\mathcal{B}) + \mathsf{q_S}\varepsilon + \frac{(\mathsf{q_S}+\mathsf{q_H})(\mathsf{q_S}+\mathsf{q_H}+\mathsf{q_G})}{2^{2\lambda}} \\ & + \frac{\mathsf{q_S}(\mathsf{q_S}+\mathsf{q_H})}{2^{\lambda}} + \frac{\psi \mathsf{q_H}^2}{2q^{n-k}} + \frac{(\psi \mathsf{q_H})^{\psi+1} |S_{w,n}|}{(\psi+1)! \cdot q^{(n-k)(\psi+1)}},\end{aligned}$$

where $\psi \geq \lceil n/(n-k) \rceil$, and the running time of $\mathcal{B}$ is about that of $\mathcal{A}$.

Proof. Since I_{wave} is a ε-APSF, we can directly apply Theorem 3.14 and obtain the claim. □

In Corollary 3.18, we can choose $\psi = 3$ to have negligible additive error terms with respect to the parameterization of Table 2.3.

In our comparison, we consider salts of length equal to the security parameter λ. The size of a single Wave signature is given by

$$|\sigma| = \lceil n \log_2 q \rceil + \lambda.$$

The size of a sequential aggregate signature instantiated with Wave is given by

$$|\bar{\Sigma}_N| = N \cdot (\lambda + \lceil k \log_2 q \rceil) + 2\lambda + \lceil (n-k) \log_2 q \rceil.$$

Applying Equation (3.2), we can see that the compression rate asymptotically goes to $\lceil (n-k) \log_2 q \rceil / (\lambda + \lceil n \log_2 q \rceil)$. Concrete numbers for different security parameters and the number of signers are given in Table 3.4.

3.6.5 Signature-Specific Optimizations

For proper evaluation of the efficiency of SAS, it is necessary to consider any optimizations of the single signature that cannot be used in aggregation. Ignoring possible optimizations can lead to an unfair comparison and an incorrect calculation of the compression rate.

In [48], Boudgoust and Takahashi observe that in the context of lattice-based signatures built on PSFs (e.g., Falcon) it is possible to reduce the size of signatures considerably by slightly modifying the verification process. However, the same variant is not applicable in the context of aggregate signatures. Similarly, as noted in the previous section, the optimization introduced by the Wavelet variant cannot be applied in the aggregation phase, causing a significant loss of efficiency.

More generally, any compression method applied to a single Hash-and-Sign signature can be employed in our construction, provided sufficient information exists in the signature to recover the message hash. To elaborate further, consider the generic Hash-and-Sign scheme outlined in Algorithm 2.1. Suppose that the Sign algorithm returns the pair $(r, C(x))$, where C is a compression algorithm on the preimage x. If, during the verification process, it is possible to recover $\mathsf{H}(r, m)$ from the public key F and $C(x)$, then the same optimization can be effectively employed within the HaS-HF-SAS scheme. In fact, the aggregation process of Algorithm 3.2 can be tweaked to aggregate part of the compressed preimage $C(x)$ without the need to modify the verification step further. However, the optimized versions of Falcon and Wave(let) do not conform to this description, as their verification process does not enable the recovery of the message hash from the signature and the public key. Instead, the verification is based on a custom assertion involving the knowledge of F, $C(x)$, and $\mathsf{H}(r, m)$.

Part II

Group Action-based Signature Aggregation

Chapter 4

Signatures from Cryptographic Group Actions

The Fiat-Shamir transform [96] is a popular paradigm to obtain digital signature from Σ-protocols. A Σ-protocol is a 3-round zero knowledge proof of knowledge that allows a prover to prove their knowledge of a witness for some public statement. A large number of Σ-protocols are known, including the well-known Schnorr identification scheme [177].

Recently, cryptographic group actions have emerged as a promising tool for building post-quantum digital signatures. Schnorr's signature, while enjoying advanced properties not generally available to post-quantum group actions, also falls within this paradigm. The first constructions in the post-quantum domain were introduced by schemes based on isogenies between elliptic curves [51, 70]. More recently, numerous (non-abelian) group actions have been proposed from the equivalence between linear [17] and matrix codes [61], alternate trilinear forms [185] or various isomorphism problems such as that between lattices [89]. In NIST's recent competition for digital signatures [161], three group action-based schemes were submitted: LESS [13], MEDS [60], and ALTEQ [37].

In Section 4.1, we introduce the basic notions of Σ-protocol, and we describe how a digital signature scheme can be derived with the Fiat-Shamir transform. Then, in Section 4.2 we give an overview of cryptographic group actions, discussing the cryptographic assumptions and the relevant constructions for digital signatures. In Section 4.3, we analyse the standard optimization employed in group-action-based signatures, so that they can later be integrated into the aggregation process. Finally, in Section 4.4 we provide some examples focusing on the schemes submitted to NIST on-ramp process, which will be the subject of the applications of aggregation schemes in the following chapter.

4.1 Interactive Proofs

Consider a language $\mathcal{L} \in \mathsf{NP}$, a NP *relation* for $\mathcal{L}$ is a binary relation $R_{\mathcal{L}}$ of pairs (x, w) such that $x \in \mathcal{L}$ and w allows verifying that $x \in \mathcal{L}$ in polynomial time. w is usually called a *witness* for the statement x.

From a language in NP we can then envision a protocol in which a *prover* produces a proof of membership that can be verified in polynomial time. An interactive proof is a generalization of this idea to prove the truth of a statement in an arbitrary language. Interactive proofs were first introduced in two independent works by Goldwasser, Micali, and Rackoff [113] and Babai [11].

Definition 4.1 (Relations and Languages)**.** A *binary relation* is a finite set $R \subseteq X \times W$. Given $(x, w) \in R$, we say that w is a *witness* for the *statement* x. Moreover, the set $\mathcal{L}_R = \{x \in X \mid \exists w \in W \text{ s.t. } (x, w) \in R\}$ of true statements for R is called the *language* of the relation R. We also say that $w \in R(x)$ if $(x, w) \in R$.

Definition 4.2 (Interactive Proof)**.** An *interactive proof* $(\mathcal{P}, \mathcal{V})$ for a binary relation $R \subseteq X \times W$ is an interactive protocol between two probabilistic machines, a prover $\mathcal{P}$ and a polynomial time verifier $\mathcal{V}$. Both $\mathcal{P}$ and $\mathcal{V}$ take as public input a statement x and, additionally, $\mathcal{P}$ takes as private input a witness w for x. We denote the protocol instance as $(\mathcal{P}(w), \mathcal{V})(x)$. As the output of the protocol, $\mathcal{V}$ either accepts (returns 1) or rejects (return 0). Accordingly, we say the corresponding transcript (i.e., the set of all messages exchanged in the protocol execution) is accepting or rejecting.

An interactive proof is usually required to satisfy two additional properties.

Definition 4.3 (Completeness)**.** An interactive proof $(\mathcal{P}, \mathcal{V})$ for a binary relation $R \subseteq X \times W$ is *complete* with *completeness error* $\rho(|x|)$ if, for any $(x, w) \in R$, it holds

$$\Pr[(\mathcal{P}(w), \mathcal{V})(x) = 0] \leq \rho(|x|).$$

If $\rho(x) = 0$ then the protocol is said to be *perfectly complete*.

Definition 4.4 (Soundness)**.** An interactive proof $(\mathcal{P}, \mathcal{V})$ for a binary relation $R \subseteq X \times W$ is *sound* with *soundness error* $\sigma(|x|)$ if, for any $x \notin \mathcal{L}_R$ and any prover $\mathcal{P}^*$, it holds

$$\Pr[(\mathcal{P}^*, \mathcal{V})(x) = 1] \leq \sigma(|x|).$$

In practical applications, we usually require the error probabilities to be negligible in $|x|$. Historically, a protocol is said to be complete (resp. sound) if $\rho(|x|)$ (resp. $\sigma(|x|)$) has the constant value $1/3$. This value is arbitrary and can be replaced by any other value in (0,1), since both the completeness error and the soundness error can be decreased by sequential repetitions.

As a generalization of the class NP we can consider the class of all languages that admits an interactive proof.

Definition 4.5 (Class IP)**.** The complexity class IP is the class of all languages $\mathcal{L}$ for which there exists an interactive proof that $x \in \mathcal{L}$ with a polynomial number of rounds (in $|x|$).

A key result showing that the class IP is strictly larger than the class NP was given in [179], where Shamir proved that IP = PSPACE.

Definition 4.6 (Class PSPACE)**.** The complexity class PSPACE is the class of all languages that can be decided by an algorithm in *polynomial space* and, possibly, with *unbounded running time*.

In an interactive proof for a language in IP, a verifier may keep their internal state hidden to the prover. Such an interactive proof is also called *private-coin*. Conversely, an interactive proof where the verifier is required to reveal their random tape is called *public-coin*.

Definition 4.7 (Public-Coin)**.** An interactive proof $(\mathcal{P}, \mathcal{V})$ is public-coin if all of $\mathcal{V}$'s random choices are made public.

The corresponding complexity class were studied in [11].

Definition 4.8 (Class AM)**.** The complexity class AM (Arthur-Merlin) is the class of all languages $\mathcal{L}$ for which there exists a public-coin interactive proof that $x \in \mathcal{L}$ with a polynomial number of rounds (in $|x|$).

A strong result linking classes IP and AM was proved in [114], showing that any language that admits a private-coin interactive proof with k rounds also admits a public-coin interactive proof with at most $k + 2$ rounds. In particular, this proves that the verifier does not gain particular power by hiding their input. For this reason, in the following we will only consider public-coin interactive proofs.

Interactive proofs provide a proof of membership for a statement $x \in \mathcal{L}_R$. In other words, they provide a proof of existence of a witness w such that $(x, w) \in R$, but there is no guarantee that the prover actually knows w. In fact, a negligible soundness error only states that a prover can not convince a verifier of a false statement $x \notin \mathcal{L}_R$. An interactive proof that also convinces a verifier of the prover's knowledge of a witness is called a *proof of knowledge*.

Informally, an interactive proof is a proof of knowledge if it is possible to *extract* a witness from a prover that correctly answers to the challenges of the verifier with sufficiently high probability. More in detail, it is necessary to describe a *knowledge extractor* algorithm. The extractor Ext is given oracle access to any prover $\mathcal{P}^*$, such that the probability of $(\mathcal{P}^*, \mathcal{V})(x) = 1$ is greater than a fixed threshold $\kappa(\mathsf{Ext}, x)$. Then, Ext should be able to efficiently compute a witness w such that $(x, w) \in R$. In particular, the running time of Ext should be inversely proportional to the success probability of $\mathcal{P}^*$. Given an extractor Ext, and the statement x, the fixed threshold $\kappa(\mathsf{Ext}, x) \in [0,1]$ called *knowledge error*. It follows that no prover who does not know a witness has a higher probability of success (also called *cheating probability*) in interacting with a verifier than the knowledge error.

An interactive proof for which it is possible to build a knowledge extractor is called *knowledge-sound*.

Definition 4.9 (Knowledge Soundness)**.** An interactive proof $(\mathcal{P}, \mathcal{V})$ for a binary relation $R \subseteq X \times W$ is *knowledge-sound* with *knowledge error* $\kappa\colon \mathbb{N} \to [0,1]$ if there exists a positive polynomial q and an algorithm Ext, called a *knowledge extractor*, with the following properties. The extractor Ext, given input x and rewindable oracle access to a (potentially dishonest) prover $\mathcal{P}^*$, runs in an expected number of steps that is polynomial in $|x|$ and outputs a witness $w \in R(x)$ with probability

$$\Pr\left[(x, \mathsf{Ext}^{\mathcal{P}^*}(x)) \in R\right] \geq \frac{\varepsilon(x, \mathcal{P}^*) - \kappa(x)}{q(|x|)},$$

where $\varepsilon(x, \mathcal{P}^*) := \Pr[(\mathcal{P}^*, \mathcal{V})(x) = 1]$.

In many scenarios, we want to use interactive proof systems which are knowledge-sound with a negligible knowledge error. In such case, we can state that *the protocol is knowledge-sound* without mentioning the knowledge error and the associated knowledge extractor. In knowledge-sound interactive proof systems, a prover that can produce a valid response to a verifier with non-negligible probability must know the witness, and therefore a prover who does not know a witness can produce an accepting conversation with a verifier only with negligible probability.

4.1.1 Zero-Knowledge Proofs

In an interactive proof (of knowledge), a verifier may be able to obtain some non-trivial information from his interaction with the prover. Instead, for cryptographic applications we are interested in interactive protocols where a verifier only learns whether a statement is true (for an interactive proof) or if the prover knows a witness (for a proof of knowledge). An interactive proof where the prover is able to convince the verifier without revealing any additional information is called a *zero-knowledge proof*. First introduced by Goldwasser, Micali, and Rackoff in [112], zero-knowledge proofs are pivotal in modern cryptography and the basis of numerous applications.

In [112], the notion of zero-knowledge was formalized through the presence of an efficient simulator for the protocol. In more detail, any information that a malicious verifier could obtain from interacting with an honest prover on a statement x would be computable directly from the output of the simulator.

Definition 4.10 (Zero-Knowledge)**.** An interactive proof $(\mathcal{P}, \mathcal{V})$ for a binary relation R is (Statistical) *Zero-Knowledge* if there exists a probabilistic polynomial time algorithm Sim, called a *simulator*, such that for any $x \in \mathcal{L}_R$ and any verifier $\mathcal{V}^*$, Sim produces a transcript in expected polynomial time which is statistically indistinguishable from the execution $(\mathcal{P}, \mathcal{V}^*)(x)$. If the two distributions are identically distributed, the interactive proof is *Perfect Zero-Knowledge*.

The above definition can be relaxed by requiring that the output of the simulator and the transcript produced by the interaction with $\mathcal{V}^*$ must be *computationally indistinguishable*. This is particularly useful in applications, since in [112] was proved that any language in NP admits a zero-knowledge proof[1].

We can consider a further relaxation of the previous notion, requiring that the algorithm Sim is able to simulate the interaction between a prover and an honest verifier. This weaker notion is called *Honest-Verifier Zero-Knowledge* (HVZK), and can be seen as a specialization of the general notion of zero-knowledge on a single type of verifier.

Definition 4.11 (Honest-Verifier Zero-Knowledge)**.** An interactive proof $(\mathcal{P}, \mathcal{V})$ for a binary relation R is (statistical) *Honest-Verifier Zero-Knowledge* if there exists a probabilistic polynomial time algorithm Sim, called a *simulator*, such that for any $x \in \mathcal{L}_R$, Sim produces a transcript in expected polynomial time which is statistically indistinguishable from the execution $(\mathcal{P}, \mathcal{V})(x)$. If the two distributions are identically distributed, the interactive proof is *perfectly* HVZK. Alternatively, if the two distributions are computationally indistinguishable, the interactive proof is *computationally* HVZK.

Although the notion of HVZK is strictly weaker, it captures roughly the same class of languages for which zero-knowledge interactive proof exists. Particularly, it can be shown that public-coin HVZK interactive proofs can be efficiently transformed to zero-knowledge interactive proofs [68, 106]. Moreover, particularly relevant to the purposes of a digital signature, a public-coin HVZK interactive proof can be made a non-interactive proof by applying the Fiat-Shamir transform [96].

4.1.2 Sigma-Protocols

For the remaining part of this thesis, we will focus solely on a particular type of interactive proof: 3-round public-coin interactive proofs for NP relations.

Definition 4.12 (3-Round Protocol)**.** A *3-round protocol* Π for a NP relation $R \subseteq X \times W$ is a tuple of four probabilistic polynomial-time algorithms $\Pi = (\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$, where $\mathsf{P}_1, \mathsf{P}_2$ are assumed to share states, such that:

[1] No similar results are known for perfect or statistical zero-knowledge proofs.

Algorithm 4.1: 3-Round Protocol

Let $R \subseteq X \times W$ be a NP relation and let $(x, w) \in R$.

```
$\textbf{Prover}(w, x)$                                  $\textbf{Verifier}(x)$
$\mathsf{com} \xleftarrow{\$} \mathsf{P}_1(w, x)$
                          $\xrightarrow{\mathsf{com}}$
                                                         $\mathsf{ch} \xleftarrow{\$} \mathsf{V}_1(\mathsf{com})$
                          $\xleftarrow{\mathsf{ch}}$
$\mathsf{rsp} \leftarrow \mathsf{P}_2(w, x, \mathsf{com}, \mathsf{ch})$
                          $\xrightarrow{\mathsf{rsp}}$
                                                         $\textbf{return } \mathsf{V}_2(x, \mathsf{com}, \mathsf{ch}, \mathsf{rsp})$
```

- $\mathsf{com} \xleftarrow{\$} \mathsf{P}_1(w, x)$: given a statement $x \in X$ and a corresponding witness $w \in R(x)$, the prover generates a *commitment* com.
- $\mathsf{ch} \xleftarrow{\$} \mathsf{V}_1(\mathsf{com})$: given a commitment com, the verifier returns a uniformly random challenge ch in a challenge space Ch.
- $\mathsf{rsp} \leftarrow \mathsf{P}_2(w, x, \mathsf{com}, \mathsf{ch})$: given a statement $x \in X$, the corresponding witness $w \in R(x)$, a commitment com and a challenge ch, the prover computes a response rsp.
- $\{0,1\} \leftarrow \mathsf{V}_2(x, \mathsf{com}, \mathsf{ch}, \mathsf{rsp})$: given a statement $x \in X$, a commitment com, a challenge ch and a response rsp, the verifier either accept (returns 1) or reject (returns 0).

A transcript for an execution of Π is given by the tuple $(\mathsf{com}, \mathsf{ch}, \mathsf{rsp})$.

A schematic representation of the protocol is provided in Algorithm 4.1.

A common strategy to prove the knowledge soundness of a 3-round protocol is showing that it enjoys (general) *special soundness*. Informally, given enough accepting transcripts with a fixed commitment, it is possible to extract a witness.

Definition 4.13 (k-out-of-N Special-Soundness)**.** Let $k, N \in \mathbb{N}$. A 3-round public-coin protocol $(\mathcal{P}, \mathcal{V})$ for relation R, with challenge set of cardinality $N \geq k$, is k-out-of-N *special-sound* if there exists a polynomial time algorithm that, on input a statement x and k accepting transcripts $(\mathsf{com}, \mathsf{ch}_1, \mathsf{rsp}_1), \dots, (\mathsf{com}, \mathsf{ch}_k, \mathsf{rsp}_k)$ with common first message com and pairwise distinct challenges $\mathsf{ch}_1, \dots, \mathsf{ch}_k$, outputs a witness $w \in R(x)$. When N is clear from the context, we also say $(\mathcal{P}, \mathcal{V})$ is k-special-sound and, if $k = 2$, it is simply said to be special-sound.

It is straightforward to prove that k-out-of-N special soundness implies knowledge soundness.

Proposition 4.14. *Let $k, N \in \mathbb{N}$ and let Π be a k-out-of-N special-sound 3-round public-coin protocol for a relation $R \subseteq X \times W$. Then Π is knowledge-sound with knowledge error $(k-1)/N$.*

Proof sketch. Let $\mathcal{P}^*$ be a prover attacking Π on input $x \in X$. We can assume, without loss of generality, that $\mathcal{P}^*$ is deterministic. In fact, it is possible to show that the extractor is well-defined even when restricted to deterministic provers only [10]. Indeed, suppose that $\mathcal{P}^*$ is a probabilistic prover, and denote by $\mathcal{P}^*[r]$ the deterministic prover obtained by setting the randomness of $\mathcal{P}^*$ to r. Let $\varepsilon(x, \mathcal{P}^*)$ be the success probability of $\mathcal{P}^*$ on input x. Then, it is easy to show that $\varepsilon(x, \mathcal{P}^*) = \mathbb{E}[\varepsilon(x, \mathcal{P}^*[r])]$ and $\Pr[(x, \mathsf{Ext}^{\mathcal{P}^*}(x)) \in R] = \mathbb{E}\big[\Pr[(x, \mathsf{Ext}^{\mathcal{P}^*[r]}(x)) \in R]\big]$, where the expected value is taken over the random choice of r.

Since $\mathcal{P}^*$ is deterministic, we can model it as a map $\mathsf{Ch} \to \{0,1\}^*$ that takes as input a challenge $\mathsf{ch} \in \mathsf{Ch}$ and outputs a fixed commitment com and its response rsp, such that (com, ch, rsp) is a transcript for Π. We also define a verification function $\mathsf{V}\colon \mathsf{Ch} \times \{0,1\}^* \to \{0,1\}$ that returns 1 on an accepting transcript and 0 otherwise. Finally, let $\varepsilon = \varepsilon(x, \mathcal{P}^*) = \Pr_{\mathsf{ch} \leftarrow\$ \mathsf{Ch}}[\mathsf{V}(\mathsf{ch}, \mathcal{P}^*(\mathsf{ch})) = 1]$. We build an extractor Ext having oracle access to $\mathcal{P}^*$ as follows.

1. Sample $\mathsf{ch}_1 \leftarrow\$ \mathsf{Ch}$ until $\mathsf{V}(\mathsf{ch}_1, \mathcal{P}^*(\mathsf{ch}_1)) = 1$. For each ch_1, the success probability of $\mathcal{P}^*$ is ε. Therefore, the expected running time of this step is $1/\varepsilon$.

2. Sample $\mathsf{ch}_2 \leftarrow\$ \mathsf{Ch} \setminus \{\mathsf{ch}_1\}$ until $\mathsf{V}(\mathsf{ch}_2, \mathcal{P}^*(\mathsf{ch}_2)) = 1$. For each ch_2, the success probability of $\mathcal{P}^*$ is at least $\varepsilon - 1/N$. Therefore, the expected running time of this step is at most $1/(\varepsilon - 1/N)$.

⋮

k. After finding $k-1$ accepting distinct challenges $\mathsf{ch}_1, \dots, \mathsf{ch}_{k-1} \in \mathsf{Ch}$, sample $\mathsf{ch}_k \leftarrow\$ \mathsf{Ch}$ until $\mathsf{V}(\mathsf{ch}_k, \mathcal{P}^*(\mathsf{ch}_k)) = 1$. For each ch_k, the success probability of $\mathcal{P}^*$ is at least $\varepsilon - (k-1)/N$. Therefore, the expected running time of this step is at most $1/(\varepsilon - (k-1)/N)$.

After the last step, Ext obtained k accepting transcript with a fixed commitment for k pairwise distinct challenges, and can apply the special-soundness extractor to obtain a witness. Combining all the steps of the extractor, the expected running time is at most

$$\frac{k}{\varepsilon(x, \mathcal{P}^*) - (k-1)/N}.$$

It follows that Π is knowledge-sound with knowledge soundness $(k-1)/N$. □

A standard approach to reduce the knowledge error of a 3-round protocol Π is to consider the t-fold parallel repetition of Π. Formally, this transforms Π into a new 3-round protocol Π^t, where t independent repetitions of Π are performed. In particular, the prover first executes P_1 t times, obtaining t commitments which are sent simultaneously to the verifier. The verifier now sends a challenge array $(\mathsf{ch}_1, \dots, \mathsf{ch}_t) \leftarrow\!\!\$\; \mathsf{Ch}^t$ to the prover, which computes t responses for each commitment-challenge pair.

It is easy to notice that the t-fold parallel repetition of a special-sound protocol is still special-sound. In fact, two distinct challenge arrays $(\mathsf{ch}_1, \dots, \mathsf{ch}_t)$ and $(\mathsf{ch}'_1, \dots, \mathsf{ch}'_t)$ must differ in at least one component $j \in \{1, \dots, t\}$, and the special-sound extractor for Π can be applied to the transcripts associated with $\mathsf{ch}_j, \mathsf{ch}'_j$. In particular, Π^t is knowledge-sound and the knowledge error is reduced from $1/|\mathsf{Ch}|$ to $1/|\mathsf{Ch}|^t$. This is not trivial to extend for generic k-special-sound protocols, and was only recently proved in [10], where Attema and Fehr showed that the t-fold parallel repetition of a k-special-sound protocol reduces the knowledge error from κ to κ^t, which is optimal[2].

Theorem 4.15 ([10]). *Let $k, N, t \in \mathbb{N}$ and let Π be a k-out-of-N special-sound 3-round public-coin protocol for a relation $R \subseteq X \times W$ and let Π^t be the t-fold parallel repetition of Π. Then Π^t is knowledge-sound with knowledge error $(k-1)^t/N^t$.*

Finally, we introduce Σ-protocols, which are the main objects discussed in the upcoming sections.

Definition 4.16. A *Σ-protocol* Π for a NP relation $R \subseteq X \times W$ is a 3-round protocol which is perfectly complete, statistically honest-verifier zero-knowledge and knowledge-sound.

A necessary requirement on the Σ-protocol to build digital signatures is to have a sufficiently large commitment space. This is formalized with the notion of *high min-entropy* [2].

Definition 4.17 (α min-entropy)**.** A Σ-protocol Π for binary relation R has α *min-entropy* if, for any $(x, w) \in R$ and for any probabilistic algorithm $\mathcal{A}$ outputting a commitment, it holds

$$\Pr\left[\mathsf{com} = \mathsf{com}' \;\middle|\; \mathsf{com} \leftarrow\!\!\$\; \mathsf{P}_1(w, x), \mathsf{com}' \leftarrow\!\!\$\; \mathcal{A}(w, x)\right] \leq 2^{-\alpha}.$$

Π has *high* min-entropy with respect to λ if $2^{-\alpha}$ is negligible in λ.

In the following sections, we will describe how to build identification protocols and digital signatures starting from Σ-protocols on cryptographic group actions.

[2] The authors also showed that this result extends to a generalization of the notion of special soundness for multi-round protocols.

Algorithm 4.2: Fiat-Shamir Transformation of a Σ-protocol

Π is a Σ-protocol on a binary relation $R \subseteq X \times W$. $\mathsf{H}\colon \{0,1\}^* \to \mathsf{Ch}$ is a random oracle

```
KGen(pp):
    Sample $x \in X, w \in W$ such that $(x, w) \in R$
    return $(\mathsf{pk} = x, \mathsf{sk} = w)$

Vrfy($\mathsf{pk} = x, m, \sigma$):
    $(\mathsf{com}, \mathsf{rsp}) \leftarrow \sigma$
    $\mathsf{ch} \leftarrow \mathsf{H}(\mathsf{com}, m)$
    return $\mathsf{V}_2(x, \mathsf{com}, \mathsf{ch}, \mathsf{rsp})$

Setup($1^\lambda$):
    $\mathsf{pp} \leftarrow$ public parameters for $\Pi$
    return pp

Sign($\mathsf{sk} = w, \mathsf{pk} = x, m$):
    $\mathsf{com} \xleftarrow{\$} \mathsf{P}_1(w, x)$
    $\mathsf{ch} \leftarrow \mathsf{H}(\mathsf{com}, m)$
    $\mathsf{rsp} \leftarrow \mathsf{P}_2(w, x, \mathsf{com}, \mathsf{ch})$
    $\sigma \leftarrow (\mathsf{com}, \mathsf{rsp})$
    return $\sigma$
```

4.1.3 Fiat-Shamir Transform

A relevant question is whether a zero-knowledge interactive proof (of knowledge) can be turned into a non-interactive proof, preserving the zero-knowledge property. Non-interactive proofs are useful in applications since they do not require any interaction between the parties. Moreover, the zero-knowledge property would allow for cryptographic applications such as digital signatures. In particular, given a non-interactive zero-knowledge proof of knowledge for a NP relation R, a signer can take a statement-witness pair $(x, w) \in R$ as their private key and publish x as their public key. To sign a message m, the signer can produce a proof of knowledge of x together with an *approval* of m, that can be then directly verified.

This was first solved for Σ-protocols by Fiat and Shamir in [96], that proposed a generic transformation to obtain a non-interactive zero-knowledge proof of knowledge that can be turned into a signature. The streamlined idea is that the sampling of the random challenge by the verifier can be replaced by the output of a random oracle H taking as input the commitment com. In practice, the role of the random oracle is taken by a collision-resistant hash function $\mathsf{H}\colon \{0,1\}^* \to \mathsf{Ch}$. To obtain a digital signature, the challenge computation includes the message to be signed, thus obtaining $\mathsf{ch} \leftarrow \mathsf{H}(\mathsf{com}, m)$. The resulting signature scheme is detailed in Algorithm 4.2.

Remark. In Algorithm 4.2, the signature is given by a partial transcript of the commitment and response $\sigma = (\mathsf{com}, \mathsf{rsp})$. During verification, the challenge is retrieved from the commitment and the message, and the full transcript can be verified. If Π satisfies an additional property for which a valid commitment is uniquely determined by a challenge and a response, Π is said to be *commitment recoverable* and the signature can be optimized. In fact, the signature can be replaced with the pair $(\mathsf{ch}, \mathsf{rsp})$ and, during verification, the commitment com is first recovered and then the challenge is compared against the output of $\mathsf{H}(\mathsf{com}, m)$.

An important result is that, if Π has high min-entropy, the digital signature obtained with the Fiat-Shamir transform is EUF-CMA secure.

Theorem 4.18 ([129])**.** *Let Π be a Σ-protocol with high min-entropy with respect to the security parameter λ. Then, the digital signature scheme* $\mathsf{FS}[\Pi]$ *obtained by applying the Fiat-Shamir transform on Π is* EUF-CMA *secure in the ROM.*

The security of the Fiat-Shamir transform in the QROM has been extensively studied in recent years [67, 189, 140, 82]. The main result that extends the previous theorem in the QROM is given in [82] with the additional requirement that the Σ-protocol has *quantum computationally unique responses*. A precise definition of this notion is provided in [188, 82], in the following we introduce a simplified notion that implies the more general one and is sufficient to prove the security of the signature scheme.

Definition 4.19 ([36])**.** A Σ-protocol for a NP relation $R \subseteq X \times W$ has *computationally unique responses* with respect to a security parameter λ if, for any adversary $\mathcal{A}$, the probability of producing two valid transcripts with the same commitment com and challenge ch but different responses $\mathsf{rsp}, \mathsf{rsp}'$ is negligible, i.e.,

$$\Pr_{(x,w)\leftarrow\$ R}\left[\begin{array}{c|c} \mathsf{V}_2(x, \mathsf{com}, \mathsf{ch}, \mathsf{rsp}) = 1 & \\ \mathsf{V}_2(x, \mathsf{com}, \mathsf{ch}, \mathsf{rsp}') = 1 & (\mathsf{com}, \mathsf{ch}, \mathsf{rsp}, \mathsf{rsp}') \leftarrow\$ \mathcal{A}(x) \\ \mathsf{rsp} \neq \mathsf{rsp}' & \end{array}\right] \leq \mathsf{negl}(\lambda).$$

If the above probability is zero, then the Σ-protocol is said to have *perfect unique responses*.

Theorem 4.20 ([82])**.** *Let Π be a Σ-protocol with high min-entropy and computationally unique responses with respect to the security parameter λ. Then, the digital signature scheme* $\mathsf{FS}[\Pi]$ *obtained by applying the Fiat-Shamir transform on Π is strong* EUF-CMA *secure in the QROM.*

4.2 Group Actions

In the following, we adopt the multiplicative notation for a group G.

Definition 4.21 (Group Action)**.** Let G be a group and X be a set. The *action* of G on X is a map $\star\colon G \times X \to X$ that is *compatible* with the group operation:

- Let e be the identity of G, then $e \star x = x$ for any $x \in X$.

- For any $g, h \in G$ and any $x \in X$, it holds that $(gh) \star x = g \star (h \star x)$.

The action of G on X is denoted with the triple $(G, X, \star)$.

A group action is only required to be compatible with the group operation. However, other properties are often considered for group actions.

Definition 4.22. Consider a group action $(G, X, \star)$, define the following properties:

- **Transitive**: if, for every $x, y \in X$, there exists $g \in G$ such that $y = g \star x$.
- **Faithful**: if $g \star x = x$ for all $x \in X$, then $g = e$.
- **Free**: for any $g \in G$, g is the identity element e if and only if there exist $x \in X$ such that $g \star x = x$.
- **Regular**: if it is free and transitive.

We also restate two classic definitions associated with group actions.

Definition 4.23 (Group Orbit)**.** Let $(G, X, \star)$ be a group action. The *orbit* $G_X(x)$ of an element $x \in X$ is the set of elements in X given by the action of G on x, i.e.,

$$G_X(x) = \{g \star x \mid g \in G\}.$$

Definition 4.24 (Group Stabilizer)**.** Let $(G, X, \star)$ be a group action. The *stabilizer* of G with respect to an element $x \in X$ is the set of group elements that fix x, i.e.,

$$\mathrm{Stab}_G(x) = \{g \in G \mid g \star x = x\}.$$

4.2.1 Effective Group Actions

To employ group actions in cryptographic applications, additional efficiency and security properties are required. In [6], Alamati et al. introduce the notion of *effective* group action, imposing the necessary requirements for the action to be computationally manageable and efficient.

Definition 4.25 (Effective Group Action)**.** A group action $(G, X, \star)$ is *effective* if the following properties are satisfied:

- G is finite, and there exists efficient probabilistic polynomial-time algorithms for the following operations:
  1. *Membership testing*: decide if a given string encoding is a valid group element in G.
  2. *Equality testing*: decide whether two given string encodings represent the same group element in G.

3. *Sampling*: efficiently sample a group element g from a distribution on G which is statistically close to the uniform distribution.
4. *Operation*: given $g, h \in G$, compute the product gh.
5. *Inversion*: given $g \in G$, compute the inverse g^{-1}.

- X is finite, and there exists efficient probabilistic polynomial-time algorithms for the following operations:
  1. *Membership testing*: decide if a given string encoding is a valid set element in X.
  2. *Unique representation*: given a set element $x \in X$, compute a canonical representation of x.
- There exists a base element $x_0 \in X$ for which a canonical representation exists and is known.
- Given a group element $g \in G$ and a set element $x \in X$, it is efficient to compute the action $g \star x$.

Remark. To capture group actions for which it is not possible to efficiently compute $g \star x$ for all group elements, [6] also introduces the notion of *restricted* effective group action. This requires the group action to be effective only on a small subset of G. In particular, the above properties are restricted to a generating set $(g_1, \dots, g_n)$ of G such that $n = \text{poly}(\log(|G|))$.

4.2.2 Computational Assumptions

The main computational assumption associated with group actions concerns the non-invertibility of the action. Informally, given $x \in X$, the map $f_x \colon G \to X, g \mapsto g \star x$ should be *one-way* (Definition 1.1), i.e., it should be efficient to compute $f_x(g)$ for any $g \in G$ while, given $x \in X$ and $y = f_x(g) = g \star x$ for some $g \in G$, it should be computationally infeasible to find g. The associated computational problem is known as *vectorization* problem, or, as we will refer to it in the following, as *Group Action Inverse Problem* (GAIP).

Definition 4.26 (GAIP)**.** Let $(G, X, \star)$ be a group action. Given $x, y \in X$ find, if any, an element $g \in G$ such that $y = g \star x$.

Remark. Notice that, when x, y are sampled from the same orbit, GAIP coincides with the one-wayness of the family of functions $f_x \colon G \to X$ defined above.

A generalization of GAIP is sometimes considered when it is only required to find a linking group element between multiple set elements. This problem is known as *multiple Group Action Inverse Problem* (mGAIP).

$$\begin{tikzcd}
x \arrow[rr, "g"] \arrow[dr, "\tilde{g}"'] & & y \arrow[dl, dashed] \\
& \tilde{x} &
\end{tikzcd}$$

Figure 4.1: High-level description of Protocol 4.30.

Definition 4.27 (mGAIP)**.** Let $(G, X, \star)$ be a group action. Given $x_1, \dots, x_k \in X$ find, if any, an element $g \in G$ and two different indices $i, j \in \{1, \dots, k\}$ such that $x_j = g \star x_i$.

Remark. While it is straightforward to reduce mGAIP to GAIP, the opposite direction can also be proved [17, 43], showing that the two problems are equivalent.

A related problem, which can be seen as the analogue of the Diffie-Hellman problem [76], asks to compute the action of the product of two unknown group elements, given the individual actions on a fixed set element. This problem is known as the *parallelization* problem, or the *computational Group Action Diffie-Hellman* (cGADH) problem.

Definition 4.28 (cGADH)**.** Let $(G, X, \star)$ be a group action. Given $x \in X$, $g \star x$ and $h \star x$ for two unknown elements $g, h \in G$, compute $(gh) \star x$.

When the action is not free, an additional problem is to find non-trivial stabilizer elements, i.e., group elements that fix elements of the set.

Definition 4.29 (Stabilizer Computation Problem)**.** Let $(G, X, \star)$ be a group action. Given $x \in X$, find, if any, $g \in G \setminus \{e\}$ such that $g \star x = x$.

Additional problems, like *weak unpredictability* and *weak pseudorandomness*, are often considered but are out of the scope of this thesis. We refer the reader to [65] and [6] for an in-depth description.

4.2.3 Digital Signatures

In the following, we show how to build a digital signature based on cryptographic group actions. Given a group action $(G, X, \star)$, we start by defining a Σ-protocol for the NP relation $R \subseteq X \times G$ associated with GAIP. In particular, given a fixed base element $x \in X$, a statement for R is a set element $y \in X$ and the witness is a group element $g \in G$ such that $y = g \star x$. The high-level idea for the Σ-protocol is to commit to a random set element $\tilde{x} \leftarrow \tilde{g} \star x$ by taking a uniformly random $\tilde{g} \leftarrow\!\!\$\ G$, and then reveal either a link between $(x, \tilde{x})$ or $(y, \tilde{x})$ (see Figure 4.1).

Protocol 4.30 (Group Action Σ-Protocol)**.** Given the public parameters $\mathrm{pp} = (G, X, \star, x)$, the protocol proceeds as follows:

- $(g, y) \leftarrow_{\$} \mathsf{Gen}(\mathsf{pp})$: the key-generation algorithm takes as input the public parameters pp. It uniformly samples $g \in G$ and computes $y \leftarrow g \star x$. It returns the witness-statement pair (g, y).

- $\mathsf{com} \leftarrow_{\$} \mathsf{P}_1(g, y)$: given a statement $y \in X$ and the corresponding witness $g \in G$, the prover uniformly samples $\tilde{g}$ and computes $\tilde{x} \leftarrow \tilde{g} \star x$. It returns $\mathsf{com} \leftarrow \tilde{x}$.

- $\mathsf{ch} \leftarrow_{\$} \mathsf{V}_1(\mathsf{com})$: given a commitment com, the Verifier returns a uniformly random challenge $\mathsf{ch} \in \{0,1\}$.

- $\mathsf{rsp} \leftarrow \mathsf{P}_2(g, y, \mathsf{com}, \mathsf{ch})$: given a statement x, the corresponding witness g, a commitment $\mathsf{com} = \tilde{x}$ and a challenge $\mathsf{ch} \in \{0,1\}$, the prover computes a response rsp as follows. If $\mathsf{ch} = 0$, then $\mathsf{rsp} \leftarrow \tilde{g}$. Else, if $\mathsf{ch} = 1$, then $\mathsf{rsp} \leftarrow \tilde{g}g^{-1}$. Finally, it outputs rsp.

- $\{0,1\} \leftarrow \mathsf{V}_2(y, \mathsf{com}, \mathsf{ch}, \mathsf{rsp})$: given a statement $y \in X$, a commitment com, a challenge $\mathsf{ch} \in \{0,1\}$ and a response rsp, the verifier proceeds as follows. If $\mathsf{ch} = 0$, computes $\mathsf{com}' \leftarrow \mathsf{rsp} \star x$. Else, if $\mathsf{ch} = 1$, computes $\mathsf{com}' \leftarrow \mathsf{rsp} \star y$. Finally, the verifier accepts (returns 1) if $\mathsf{com}' = \mathsf{com}$, otherwise rejects (returns 0).

The following result shows that the above is, in fact, a Σ-protocol.

Proposition 4.31. *Protocol 4.30 is perfectly complete, special-sound and honest-verifier zero-knowledge.*

Proof. We prove each property separately.

Completeness In an honest execution, the verifier receives $\mathsf{rsp}_0 = \tilde{g}$ when $\mathsf{ch} = 0$ and $\mathsf{rsp}_1 = \tilde{g}g^{-1}$ when $\mathsf{ch} = 1$. Correctness easily follows since $\mathsf{rsp}_0 \star x = \tilde{g} \star x = \mathsf{com}$ and $\mathsf{rsp}_1 \star y = \tilde{g} \star (g^{-1} \star y) = \tilde{g} \star x = \mathsf{com}$.

Special Soundness Suppose the extractor has access to two accepting transcripts $(\mathsf{com}, 0, \mathsf{rsp}_0)$ and $(\mathsf{com}, 1, \mathsf{rsp}_1)$. Then, $\mathsf{rsp}_0 \star x = \mathsf{rsp}_1 \star y = \mathsf{com}$. It follows that $(\mathsf{rsp}_1^{-1}\mathsf{rsp}_0) \star x = y$ and $\mathsf{rsp}_1^{-1}\mathsf{rsp}_0$ is a witness for the GAIP instance (x, y).

Honest-Verifier Zero-Knowledge A polynomial time simulator Sim can be obtained as follows. On input a public statement $y \in X$, Sim randomly samples $\mathsf{ch} \leftarrow_{\$} \{0,1\}$ and $\mathsf{rsp} \leftarrow_{\$} G$. Then, if $\mathsf{ch} = 0$ it computes $\mathsf{com} \leftarrow \mathsf{rsp} \star x$, otherwise if $\mathsf{ch} = 1$ it computes $\mathsf{com} \leftarrow \mathsf{rsp} \star y$. Finally, it returns the transcript $(\mathsf{com}, \mathsf{ch}, \mathsf{rsp})$. Notice that in the real distribution $\mathsf{com} = \tilde{g} \star x$ is uniformly distributed in the orbit of x, ch is uniformly distributed in $\{0,1\}$ and rsp is uniquely determined as either $\mathsf{rsp} = \tilde{g}$ or $\mathsf{rsp} = \tilde{g}g^{-1}$. When $\mathsf{ch} = 0$, the simulated distribution coincides with the real distribution. When $\mathsf{ch} = 1$,

Algorithm 4.3: Signature Scheme based on Group Actions

$\star\colon G \times X \to X$ is a cryptographic group action. $\mathsf{H}\colon \{0,1\}^* \to \{0,1\}^\lambda$ is a random oracle.

```
Setup(1^λ):
    $x_0 \leftarrow_\$ X$
    $\mathsf{pp} \leftarrow x_0$
    return pp
```

```
KGen(pp = $x_0$):
    $g \leftarrow_\$ G$
    $x_1 \leftarrow g \star x_0$
    return $(\mathsf{pk} = x_1, \mathsf{sk} = g)$
```

```
Vrfy(pk = $x_1$, $m$, $\sigma$):
    $(\mathsf{ch}, \mathsf{rsp}) \leftarrow \sigma$
    $(\mathsf{ch}_1, \dots, \mathsf{ch}_\lambda) \leftarrow \mathsf{ch}$
    $(\mathsf{rsp}_1, \dots, \mathsf{rsp}_\lambda) \leftarrow \mathsf{rsp}$
    for $j \leftarrow 1, \dots, \lambda$ do
        $\tilde{x}'_j \leftarrow \mathsf{rsp}_j \star x_{\mathsf{ch}_j}$
    $\mathsf{com}' \leftarrow (\tilde{x}'_1, \dots, \tilde{x}'_\lambda)$
    return $\mathsf{ch} = \mathsf{H}(\mathsf{com}', m)$
```

```
Sign(sk = $g$, pk = $x_1$, $m$):
    for $j \leftarrow 1, \dots, \lambda$ do
        $\tilde{g}_j \leftarrow_\$ G$
        $\tilde{x}_j \leftarrow \tilde{g}_j \star x_0$
    $\mathsf{com} \leftarrow (\tilde{x}_1, \dots, \tilde{x}_\lambda)$
    $\mathsf{ch} \leftarrow \mathsf{H}(\mathsf{com}, m)$
    $(\mathsf{ch}_1, \dots, \mathsf{ch}_\lambda) \leftarrow \mathsf{ch}$
    for $j \leftarrow 1, \dots, \lambda$ do
        $\mathsf{rsp}_j \leftarrow \tilde{g}_j g^{-\mathsf{ch}_j}$
    $\mathsf{rsp} \leftarrow (\mathsf{rsp}_1, \dots, \mathsf{rsp}_\lambda)$
    return $\sigma \leftarrow (\mathsf{ch}, \mathsf{rsp})$
```

the two distributions are equivalent. In fact, since $f_g\colon G \to G, h \mapsto gh$ is a bijective map from G to G, the real response $\tilde{g}g^{-1}$ is uniformly distributed in G as the simulated response. □

The Σ-protocol Π of Protocol 4.30 has a binary challenge space. The special soundness property and Proposition 4.14 implies that Π is knowledge-sound with knowledge error $1/2$. As mentioned in Section 4.1.2, the knowledge soundness can be amplified by parallel repetition. In particular, to achieve λ bit of security, it is enough to repeat the protocol $t = \lambda$ times. Moreover, it is straightforward to show that Π is commitment recoverable. It is now possible to apply the Fiat-Shamir transform (Algorithm 4.2) on Π^t to obtain an EUF-CMA secure digital signature in the random oracle model (Theorem 4.18). The resulting signature scheme is described in Algorithm 4.3.

To apply Theorem 4.20 and prove the security of the signature scheme in the quantum random oracle model, it is still necessary to prove that Π has computationally unique responses. In [36], the authors proved that, given $(y, g) \in$ such that $g \star x = y$, finding two valid responses $\mathsf{rsp}, \mathsf{rsp}'$ for the same commitment-challenge pair $(\mathsf{com}, \mathsf{ch})$ is equivalent to find a non-trivial element in the stabilizer of x. In particular, they proved the following result.

Lemma 4.32 ([36])**.** *Let Π be as in Protocol 4.30 with base element $x \in X$. Π has computationally unique responses if and only if no polynomial-time quantum adversary* *can solve the Stabilizer Computation Problem (Definition 4.29) for x except with a negligible probability.*

Corollary 4.33. *Let Π be as in Protocol 4.30 for a cryptographic group action $(G, X, \star)$ with base element x. If no polynomial-time quantum adversary can solve the GAIP (Definition 4.26) and the Stabilizer Computation Problem (Definition 4.29) for x except with a negligible probability, then $\mathsf{FS}[\Pi^\lambda]$ is strong* EUF-CMA *in the QROM.*

Notice that when the action is free, each element in X has trivial stabilizer and the protocol has perfect unique responses, leading to an immediate proof of strong EUF-CMA security of the signature scheme in the QROM.

4.3 Signature Optimizations

In this section, we describe some standard techniques for optimizing Protocol 4.30. These techniques are commonly employed in the digital signature schemes that will be presented in the next section. In what follows, we will focus primarily on the techniques employed to reduce the size of the signature, as they are most meaningful for comparison with aggregate signatures. In fact, the feasibility of transposing these optimizations into the constructions that will be presented in Chapter 5 is crucial for an honest comparison with the actual parameters of signature schemes based on group actions.

Consider a group action $(G, X, \star)$ and a security parameter λ. The non-optimized version of the Σ-protocol Π is described in Protocol 4.30. In the following, we assume that a group element can be represented with strings of ℓ_G bits, while a challenge for a single instance of the protocol can be represented with a single bit. Since Π is commitment-recoverable, we are only interested in the size of the restricted transcript $(\mathsf{ch}, \mathsf{rsp})$, where the response rsp is an element of the group. We already discussed that to achieve negligible knowledge error, it is required to parallel repeat Π for $t = \lambda$ times, obtaining $\Pi_1 = \Pi^t$. We present each optimization as a successive transformation applied on top of Π_1, analysing the updated parameters (e.g., the number of repetitions of the protocol) and the size of the signature obtained by applying Fiat-Shamir.

The bit size of the signature for $\mathsf{FS}[\Pi_1]$ is given by:

$$|\mathsf{ch}| + |\mathsf{rsp}| = \lambda + \lambda\ell_G.$$

4.3.1 Compression of Random Elements

A basic technique used to reduce the size of the signature is based on the following simple observation: when $\mathsf{ch}_i = 0$ the response for the i-th repetition is just the uniformly random group element $\tilde{g}_i \in G$ used to build the commitment. Therefore, in practice, $\tilde{g}_i$ can be replaced by a short random seed s_i of size λ which is used as the input of a Pseudorandom Number Generator PRNG to generate the group element. Then, every time $\mathsf{ch}_i = 0$, the signer can set $\mathsf{rsp}_i \leftarrow s_i$ saving $\ell_{\mathrm{G}} - \lambda$ bits for each 0 challenge. If the challenge array is uniformly sampled from $\{0,1\}^t$, then the expected number of bits saved using this optimization is $t(\ell_{\mathrm{G}} - \lambda)/2$.

Remark. When random responses are compressed using seed of length λ, it is required to also employ a random salt of length at least 2λ to prevent collision search attacks [53]. Otherwise, an attacker is able to find a collision after observation of $\mathcal{O}(\lambda/2)$ signatures, halving the security of the scheme. Using a random salt r, the i-th random element is computed by taking the output of the PRNG on input (seed, r, i). This corresponds to a slight increase in the signature size, which now includes the random salt.

Let Π_2 be the protocol obtained by applying the aforementioned optimization to Π_1, then the expected bit size of the signature for $\mathsf{FS}[\Pi_2]$ is given by

$$|r| + |\mathsf{ch}| + |\mathsf{rsp}| = 2\lambda + \lambda + t\left(\frac{1}{2}\lambda + \frac{1}{2}\ell_{\mathrm{G}}\right),$$

where the terms of $|\mathsf{rsp}|$ correspond to the size of the responses to non-zero and zero challenges, respectively. Notice that this holds only on average and that in the worst case the size of the response can grow up to $t\ell_{\mathrm{G}}$.

This technique was already adopted in the context of isogenies [183] and later employed in group action-based signatures.

4.3.2 Seed Trees

A binary tree of seeds (*seed tree*) can be used to reduce the communication cost of the seeds used to construct the random elements of the group [32]. The tree is computed by taking a master seed of length λ as the root of the tree. Then, from each node, two children are generated from the output of length 2λ of a PRNG taking as input the value of the node. To represent t seeds, this process is repeated for $\lceil \log(t) \rceil$ times so that the tree has $2^{\lceil \log(t) \rceil} \geq t$ leaves having the seeds as values. The seeds corresponding to a subset of the leaves can be revealed by sharing a suitable subset of parent nodes and computing the corresponding leaves. In particular, to communicate the value of all the t seeds except for those indexed by a subset of $\{1, \dots, t\}$ of size ω, it is enough to send the values of the following number of nodes:

$$2^{\lceil \log_2(\omega) \rceil} + \omega(\lceil \log_2(t) \rceil - \lceil \log_2(\omega) \rceil - 1).$$

Remark. The Seed Tree optimization is a particular instance of a generic construction employing *puncturable* pseudorandom functions [42], which can be used to compress a large number of pseudorandom information, like the list of seeds to represent group elements.

The communication cost of using a seed tree is advantageous when there are at least $\frac{t}{2}$ zero challenges. This can be enforced by sampling the challenges according to a fixed-weight distribution, as shown in the next optimization.

4.3.3 Fixed-Weight Challenges

When random responses corresponding to $\mathsf{ch}_i = 0$ are compressed with a seed as described above, the resulting size is much smaller than when the challenge is non-zero. A standard technique to exploit this imbalance is to modify the distribution of challenges to increase the number of zero challenges in ch [32, 173, 17]. More precisely, with this optimization, we choose parameters t, ω such that there are exactly ω non-zero challenges among t execution of the protocol. When the challenge space is binary, as in Protocol 4.30, the number of challenges in $\{0,1\}^t$ having exactly ω components equal to 1 is $\binom{t}{\omega}$. Therefore, the choice of t, ω must be made so that

$$\binom{t}{\omega}^{-1} \leq 2^{-\lambda}. \tag{4.1}$$

Remark. The security of this solution is well understood in the case of special-sound Σ-protocol since, as for parallel repetition, the resulting protocol is still special-sound with challenge space of cardinality $\binom{t}{\omega}$. However, this is not trivial to extend to generic k-special-sound Σ-protocol of multi-round protocols and was only recently proved secure in [18].

By applying this optimization on Π_2, for each response we send ω group elements corresponding to $\mathsf{ch}_i = 1$. The remaining $t - \omega$ group elements corresponding to $\mathsf{ch}_i = 0$ are replaced by random seeds that can be further compressed using the Seed Tree optimization. We obtain the protocol Π_3, the size of the signature associated with $\mathsf{FS}[\Pi_3]$ is given by:

$$|r| + |\mathsf{ch}| + |\mathsf{rsp}| = 2\lambda + \lambda + \left(2^{\lceil \log_2(\omega) \rceil} + \omega(\lceil \log_2(t) \rceil - \lceil \log_2(\omega) \rceil - 1)\right) \cdot \lambda + \omega \ell_{\mathrm{G}}.$$

Notice that when $\lambda \ll \ell_{\mathrm{G}}$, using this optimization with an appropriate choice of ω compresses the signature considerably. On the other hand, to maintain the same security level, t must be chosen according to Equation (4.1). This typically results in an increase in the number of parallel repetitions, leading to a trade-off between the size of the signature and the efficiency of the signing and verification process.

4.3.4 Multiple Public Keys

Finally, it is possible to consider multiple public keys for each user in order to reduce the size of the signature. This is a standard technique [70] employed in group action-based signatures to achieve a trade-off between signature size and public key size. Unlike previous optimizations, in this case the underlying security assumption is modified. The signer generates $s-1$ public keys associated to different private keys, and the challenge space is extended from $\{0,1\}$ to $\{0,\ldots,s-1\}$ so that a challenge can select one of the keys. The response is then generated using the relevant private key, exhibiting a group element that maps the selected public key to the commitment. Notice that the security assumption underlying the signature is modified from GAIP to mGAIP with s set elements. Since the challenge space of the single instance is extended from a binary space to one of s elements, the soundness error is reduced to $1/s$. Clearly, this also reduces the number of repetitions required to $t = \lceil \lambda / \log(s) \rceil$.

To obtain a soundness error negligible in λ with a single instance of the protocol would require to generate an exponential number of keys ($s = 2^{\lambda}$). For this reason, this approach is usually combined with the previous optimizations to reduce the size of the signature with a limited increase in the public-key size. Notice that when $\mathsf{ch}_i = 0$, the response is still a random group element that can be replaced with a short seed; while for $\mathsf{ch}_i \neq 0$ a full group element is required. We can then apply the Fixed-Weight optimization to send ω group elements corresponding to $\mathsf{ch}_i \neq 0$ and $t-\omega$ short seeds corresponding to $\mathsf{ch}_i = 0$. Therefore, the choice of t, ω must be made so that

$$\left[\binom{t}{\omega}(s-1)^{\omega}\right]^{-1} \leq 2^{-\lambda}. \tag{4.2}$$

By combining all previous optimizations, we obtain the protocol Π_4. The size of the signature associated with $\mathsf{FS}[\Pi_4]$ is given by:

$$|r| + |\mathsf{ch}| + |\mathsf{rsp}| = 2\lambda + \lambda + \left(2^{\lceil \log_2(\omega) \rceil} + \omega(\lceil \log_2(t) \rceil - \lceil \log_2(\omega) \rceil - 1)\right) \cdot \lambda + \omega \ell_{\mathrm{G}}. \tag{4.3}$$

Notice that this is the same as the fixed-weight case, but here the number of repetitions will be smaller due to the increased number of public keys, resulting in a more compact signature.

4.3.5 Further Optimizations

Recently, new techniques have been proposed to optimize group action-based signatures, as well as more peculiar combinations of existing ones. These include, in particular, techniques derived from the recent MPC-in-the-head paradigm [125, 127]. Since these optimizations are not as widely used as the previous ones, we omit their discussion and refer to the extensive survey of [43].

4.4 Post-Quantum Group Actions

In this section, we give a high-level description of some Σ-protocols based on post-quantum non-commutative group actions and their related signature schemes. In particular, we will focus on the protocols underlying the schemes submitted to the NIST call for additional post quantum signatures [161]: LESS [13], MEDS [60], and ALTEQ [37].

4.4.1 Code Equivalence

Recall that a $[n, k, q]$ *linear code* $\mathfrak{C}$ is a k-dimensional subspace of $\mathbb{F}_q^n$ endowed with a metric $\mathrm{d} : \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{N}$.

In Section 2.3.2, we introduced the computational problems underlying code-based Hash-and-Sign schemes. In this section, we will focus on isomorphism problems between codes, on which we can define cryptographic group actions and obtain signature schemes through the approach described in this chapter. Working in the setting of coding theory, the isomorphism problem considered is the so-called *code equivalence* problem. We begin by treating this problem in its highest generality, and then specialize it according to the metric chosen for the linear code.

The concept of equivalence between linear codes is formulated from maps, called *isometries*, that keep the metric of the code unchanged.

Definition 4.34 (Isometry)**.** Let $\mathbb{V}$ be a vector space and consider a metric $\mathrm{d} : \mathbb{V} \times \mathbb{V} \to \mathbb{N}$. A (linear) *isometry* for d is a (linear) map $\psi \colon \mathbb{V} \to \mathbb{V}$ such that, for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{V}$,

$$\mathrm{d}(\psi(\boldsymbol{x}), \psi(\boldsymbol{y})) = \mathrm{d}(\boldsymbol{x}, \boldsymbol{y}).$$

Equivalently, if ψ is linear and d is induced by a weight wt, it is sufficient that

$$\mathrm{wt}(\psi(\boldsymbol{x})) = \mathrm{wt}(\boldsymbol{x}).$$

In the following, we consider only linear isometries on $\mathbb{F}_q^n$, omitting the adjective linear.

Two linear codes $\mathfrak{C}$ and $\mathfrak{C}'$ are said to be *equivalent* if there exists an isometry between them. Without further specification regarding the class of codes and the family of isometries, it is possible to define the following generalized equivalence problem.

Definition 4.35 (General Code Equivalence)**.** Let $\mathrm{d} : \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{N}$ be a metric over $\mathbb{F}_q^n$. Given two equivalent linear codes $\mathfrak{C}$ and $\mathfrak{C}'$ with respect to d, find a linear isometry ψ such that $\psi(\mathfrak{C}) = \mathfrak{C}'$.

Notice that the set of isometries with respect to d forms a group under the composition of linear maps. In fact, the identity on $\mathbb{F}_q^n$ is clearly an isometry and the composition of two isometries is still an isometry. Moreover, notice that any isometry ψ is injective since d is a metric and $\mathrm{d}(\boldsymbol{x}, \boldsymbol{y}) = 0 \iff \boldsymbol{x} = \boldsymbol{y}$. Then, since $\mathbb{F}_q^n$ is finite dimensional, ψ is bijective and admits an inverse. Let G_d be the group defined above, and let X be the set of $[n, k, q]$ linear codes. Then, we can formulate the General Code Equivalence problem as the GAIP of the following group action:

$$\star\colon G_\mathrm{d} \times X \to X, (\psi, \mathfrak{C}) \mapsto \psi(\mathfrak{C}).$$

Systematic Form Recall that a $[n, k, q]$ linear code $\mathfrak{C}$ can be represented using a *generator matrix* $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ which has the code as image. The choice of $\mathbf{G}$ is not unique, since any change of basis obtained as $\mathbf{LG}$ with $\mathbf{L} \in \mathrm{GL}_k(q)$ generates the same code.

An *information set* for $\mathfrak{C}$ is a subset of indices $J \subset \{1, \dots, n\}$ such that the submatrix of $\mathbf{G}$ given by the columns indexed by J is of full rank, i.e., $\mathrm{rank}(\mathbf{G}_J) = k$. When $[k] := \{1, \dots, k\}$ is an information set, we can apply the change of basis $\mathbf{G}' = \mathbf{G}_{[k]}^{-1}\mathbf{G} = [\mathbf{I}_k \mid \mathbf{G}_{[k]}^{-1}\mathbf{G}_{\{k+1,\dots,n\}}]$. This is relevant because it allows for a special representation of the code.

Definition 4.36 (Systematic Form)**.** Let $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ be a generator matrix for a $[n, k, q]$ linear code $\mathfrak{C}$. $\mathbf{G}$ is said to be in *systematic form* if it has the following form

$$\mathbf{G} = [\mathbf{I}_k \mid \mathbf{A}], \quad \text{with } \mathbf{A} \in \mathbb{F}_q^{k\times n-k}.$$

When $[k]$ is an information set for $\mathfrak{C}$, a generator matrix $\mathbf{G}$ can always be put in systematic form, and we write $\mathrm{SF}(\mathbf{G}) = \mathbf{G}_{[k]}^{-1}\mathbf{G}$. Notice that the systematic form is a canonical representation of the code, since it is invariant under change of basis. In fact, for any $\mathbf{L} \in \mathrm{GL}_k(q)$, we have $\mathrm{SF}(\mathbf{LG}) = (\mathbf{LG}_{[k]})^{-1}\mathbf{LG} = \mathrm{SF}(\mathbf{G})$.

We can then extend the previous group action to canonical representatives of the codes, by taking:

$$\star\colon G_\mathrm{d} \times X \to X, (\psi, \mathbf{G}) \mapsto \mathrm{SF}(\psi(\mathbf{G})).$$

Or, equivalently, on any representative given by the set of full-rank matrices in $\mathbb{F}_q^{k\times n}$, by also considering the change of basis:

$$\star\colon (\mathrm{GL}_k(q) \times G_\mathrm{d}) \times \mathbb{F}_q^{k\times n} \to \mathbb{F}_q^{k\times n}, ((\mathbf{L}, \psi), \mathbf{G}) \mapsto \mathbf{L}\psi(\mathbf{G}).$$

The GAIP associated with the previous actions are all equivalent to the General Code Equivalence problem.

4.4.2 Linear Code Equivalence

In this section we introduce LESS [34, 17], a signature scheme based on the hardness of code equivalence induced by isometries for the *Hamming metric* (Definition 2.19).

Hardness Assumptions To study isometries in the Hamming metric, let $\mathcal{S}_n$ be the symmetric group on n elements. The elements of $\mathcal{S}_n$ are permutations of the form $\pi = (i_1, \dots, i_n)$. Given $\pi \in \mathcal{S}_n$ we can take the linear map $\psi_\pi \colon \mathbb{F}_q^n \to \mathbb{F}_q^n$ associated with π such that $\psi_\pi(\boldsymbol{x}) = (x_{\pi^{-1}(1)}, \dots, x_{\pi^{-1}(n)})$. It is easy to observe that changing the order of the coordinates does not change the Hamming weight of $\boldsymbol{x}$, so that ψ_π is an isometry.

When $q > 2$ we can also combine the permutation with a scaling factor for each coordinate. In more detail, let $\mathrm{Mon}(n, q)$ the group of monomial transformations, namely, of the form $\mu = (\pi, \boldsymbol{v})$ with $\pi \in \mathcal{S}_n$ and $\boldsymbol{v} \in (\mathbb{F}_q^*)^n$ such that $\mu(\boldsymbol{x}) = (v_1 x_{\pi^{-1}(1)}, \dots, v_n x_{\pi^{-1}(n)})$. The linear map associated with μ is still an isometry for the Hamming metric. Moreover, we can state the following result.

Theorem 4.37 ([146]). *Let ψ be a linear isometry for the Hamming metric over $\mathbb{F}_q^n$, then ψ is associated with a monomial transformation $\mu \in \mathrm{Mon}(n, q)$.*

We can then define the Code Equivalence problem for linear codes endowed with the Hamming metric.

Definition 4.38 (Linear Code Equivalence Problem (LEP))**.** Given two equivalent $[n, k, q]$ linear codes $\mathfrak{C}$ and $\mathfrak{C}'$, find a monomial transformation $\mu \in \mathrm{Mon}(n, q)$ such that $\mu(\mathfrak{C}) = \mathfrak{C}'$.

We can also consider a special case of LEP where we only consider the action of permutations.

Definition 4.39 (Permutation Code Equivalence (PEP))**.** Given two equivalent $[n, k, q]$ linear codes $\mathfrak{C}$ and $\mathfrak{C}'$, find a permutation $\pi \in \mathcal{S}_n$ such that $\pi(\mathfrak{C}) = \mathfrak{C}'$.

Group Action It is well known that $\mathrm{Mon}(n, q)$ is isomorphic to the semidirect product $\mathcal{S}_n \ltimes (\mathbb{F}_q^*)^n$, where $(\mathbb{F}_q^*)^n$ is isomorphic to the group of non-singular $n \times n$ diagonal matrices. The elements of $\mathrm{Mon}(n, q)$ can be represented by matrices $\mathbf{Q} \in \mathrm{GL}_n(q)$ of the form $\mathbf{Q} = \mathbf{PD}$, where $\mathbf{P}$ is a permutation matrix and $\mathbf{D}$ is a non-singular diagonal matrix. We can then consider the following action of $\mathrm{Mon}(n, q)$ on the set X of $[n, k, q]$ linear codes represented by generator matrices in systematic form:

$$\star_{\mathsf{LEP}} \colon \mathrm{Mon}(n, q) \times X \to X, (\mathbf{Q}, \mathbf{G}) \mapsto \mathrm{SF}(\mathbf{GQ}).$$

We can see that the GAIP for the above action is precisely the Linear Code Equivalence problem. Similarly, if we restrict the previous action to $\mathcal{S}_n$, the associated GAIP is the Permutation Code Equivalence problem. The signature scheme can be obtained by instantiating Protocol 4.30 with $\star_{\mathsf{LEP}}$ and applying the Fiat-Shamir transform as described in Algorithm 4.3.

Signature Optimizations The proposed parameterizations for LESS [13] include the Fixed-Weight (Section 4.3.3) and Seed Trees (Section 4.3.2) optimizations and, for some sets only, the Multiple Public Keys (Section 4.3.4) optimization. In particular, for each NIST security level I, III, and V, the LESS specification proposes three sets of parameters with a progressive trade-off between signature size and public key size, starting with the *balanced* set that does not use multiple keys, with a gradual increase in the *intermediate* and *short* sets.

LESS also includes a specific optimization presented in [169] with the introduction of the *Information Set* variant of LEP (IS-LEP), which requires two codes to be equivalent only on an information set. More in detail, two codes $\mathfrak{C}$ and $\mathfrak{C}'$ are *information set linearly equivalent* if there exists monomial transformations $\tilde{\mu} \in \mathrm{Mon}(n,q), \zeta \in \mathrm{Mon}(n-k,q)$ and an information set J for both $\mathfrak{C}'$ and $\tilde{\mathfrak{C}} = \tilde{\mu}(\mathfrak{C})$ such that, given generator matrices $\tilde{\mathbf{G}}, \mathbf{G}' \in \mathbb{F}_q^{k\times n}$ for $\tilde{\mathfrak{C}}$ and $\mathfrak{C}'$, it holds that

$$\tilde{\mathbf{G}}_J^{-1}\tilde{\mathbf{G}}_{\{1,\ldots,n\}\setminus J} = \zeta(\mathbf{G}_J'^{-1}\mathbf{G}'_{\{1,\ldots,n\}\setminus J}).$$

We can then define the *Information Set Linear Equivalence Problem* that, given $\mathfrak{C}$ and $\mathfrak{C}'$, requires finding the monomial transformations $\tilde{\mu}$ and ζ. In [169], Persichetti and Santini proved that two linear codes are linearly equivalent if and only if they are information set linearly equivalent, obtaining an equivalence between LEP and IS-LEP. With this result, it is possible to modify the Σ-protocol associated with LEP so that in the response, the prover only needs to transmit the part of the monomial transformation that corresponds to the action on the information set.

Recently, Chou, Persichetti, and Santini [62] introduced another notion of equivalence for linear codes and proved that it reduces to linear equivalence. This notion uses *canonical forms* to achieve further compression of group elements transmitted in the response. This optimization is not currently part of the LESS specification, and we refer to [62] for more details.

Concrete Parameters The elements of the monomial group $\mathrm{Mon}(n,q)$ can be represented with $n(\lceil\log_2(n)\rceil + \lceil\log_2(q-1)\rceil)$ bits. Thanks to the information set optimization, it is only required to transmit $\ell_{\mathrm{G}} = k(\lceil\log_2(n)\rceil + \lceil\log_2(q-1)\rceil)$. By using the Fixed-Weight optimization with t repetitions and ω non-zero challenges, and the Multiple Public Keys optimization with s keys, the size of the signature is given by Equation (4.3). The proposed parameters for LESS are shown in Table 4.1.

4.4.3 Matrix Code Equivalence

In this section we introduce MEDS [61], a signature scheme based on the hardness of code equivalence induced by isometries for the *rank metric*.

Hardness Assumptions A $[m \times n, k, q]$ *matrix code* $\mathfrak{C}$ is a k-dimensional subspace of the space of $m \times n$ matrices over $\mathbb{F}_q$. We refer to this vector space as $\mathbb{F}_q^{m\times n}$ which is isomorphic to $\mathbb{F}_q^{mn}$. The rank metric is induced by the following weight.

Definition 4.40 (Rank Weight)**.** Let $\mathbf{A} \in \mathbb{F}_q^{m\times n}$. The *rank weight* of $\mathbf{A}$ is given by the rank of $\mathbf{A}$, i.e., $\mathrm{wt}(\mathbf{A}) = \mathrm{rank}(\mathbf{A})$.

Notice that the rank of a matrix in $\mathbb{F}_q^{m\times n}$ is preserved by left and right multiplication by an invertible matrix and, for square matrices, by transposition. Let $\mathbf{L} \in \mathbb{F}_q^{n\times n}$ and $\mathbf{S} \in \mathbb{F}_q^{m\times m}$, we define the linear maps $\psi_{\mathbf{L},\mathbf{S}}, \psi^\top \colon \mathbb{F}_q^{m\times n} \to \mathbb{F}_q^{m\times n}$ as $\psi_{\mathbf{L},\mathbf{S}}(\mathbf{A}) = \mathbf{LAS}$ and $\psi^\top(\mathbf{A}) = \mathbf{A}^\top$. Indeed, in [155] it is shown that these linear transformations are all and only isometries in the rank metric.

Theorem 4.41 ([155])**.** *Let ψ be a linear isometry for the rank metric over $\mathbb{F}_q^{m\times n}$, then there exist invertible matrices $\mathbf{L} \in \mathbb{F}_q^{n\times n}$ and $\mathbf{S} \in \mathbb{F}_q^{m\times m}$ such that either $\psi = \psi_{\mathbf{L},\mathbf{S}}$ or $\psi = \psi_{\mathbf{L},\mathbf{S}} \circ \psi^\top$, where the latter case can occur only if $m = n$.*

In the following, even when $n \neq m$, we only consider isometries of the form $\psi_{\mathbf{L},\mathbf{S}}$. We can then define the Code Equivalence problem for linear codes endowed with the rank metric.

Definition 4.42 (Matrix Code Equivalence (MCE) Problem)**.** Given two equivalent $[m \times n, k, q]$ linear codes $\mathfrak{C}$ and $\mathfrak{C}'$, find invertible matrices $\mathbf{L} \in \mathbb{F}_q^{n\times n}$ and $\mathbf{S} \in \mathbb{F}_q^{m\times m}$ such that $\psi_{\mathbf{L},\mathbf{S}}(\mathfrak{C}) = \mathfrak{C}'$.

Group Action We can consider the following action of $\mathrm{GL}_n(q) \times \mathrm{GL}_m(q)$ on the set X of $[m \times n, k, q]$ matrix codes:

$$\star_{\mathsf{MCE}} \colon (\mathrm{GL}_n(q) \times \mathrm{GL}_m(q)) \times X \to X, ((\mathbf{L}, \mathbf{S}), \mathfrak{C}) \mapsto \psi_{\mathbf{L},\mathbf{S}}(\mathfrak{C}).$$

To represent the action as a linear map, we can consider the isomorphism $\mathrm{vec} \colon \mathbb{F}_q^{m\times n} \to \mathbb{F}_q^{mn}$ that maps a matrix $\mathbf{A}$ to a vector $\mathrm{vec}(\mathbf{A})$ obtained by *flattening* the matrix, i.e.:

$$\mathrm{vec} \colon \mathbf{A} = \begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{pmatrix} \mapsto \mathrm{vec}(\mathbf{A}) = (a_{1,1}, \dots, a_{1,n}, \dots, a_{m,1}, \dots, a_{m,n}).$$

The inverse map is denoted by mat. Using the isomorphism vec, a $[m \times n, k, q]$ matrix code is mapped into a $[mn, k, q]$ linear code, which can be represented using a generator matrix $\mathbf{G} \in \mathbb{F}_q^{k \times mn}$. Moreover, we can express the action using the Kronecker product of $\mathbf{L}^\top$ and $\mathbf{S}$: it holds that $\mathrm{vec}(\psi_{\mathbf{L},\mathbf{S}}(\mathbf{A})) = \mathrm{vec}(\mathbf{LAS}) = \mathrm{vec}(\mathbf{A}) \cdot (\mathbf{L}^\top \otimes \mathbf{S})$. Therefore, we can represent $\star_{\mathsf{MCE}}$ as the action of $\mathrm{GL}_n(q) \times \mathrm{GL}_m(q)$ on the set X of $[m \times n, k, q]$ matrix codes represented by generator matrices $\mathbf{G} \in \mathbb{F}_q^{k \times mn}$ in systematic form:

$$\star_{\mathsf{MCE}} \colon (\mathrm{GL}_n(q) \times \mathrm{GL}_m(q)) \times X \to X, ((\mathbf{L}, \mathbf{S}), \mathbf{G}) \mapsto \mathrm{SF}(\mathbf{G}(\mathbf{L}^\top \otimes \mathbf{S})).$$

We can see that the GAIP for the above action is the Matrix Code Equivalence problem. The signature scheme can be obtained by instantiating Protocol 4.30 with $\star_{\mathsf{MCE}}$ and applying the Fiat-Shamir transform as described in Algorithm 4.3.

Signature Optimizations The proposed parameterizations for MEDS [60] include the Fixed-Weight (Section 4.3.3) with Seed Trees (Section 4.3.2) optimizations and the Multiple Public Keys (Section 4.3.4) optimization. In particular, for each NIST security level I, III, and V, the MEDS specification proposes two sets of parameters with a strong trade-off between signature size and scheme efficiency due to heavy use of the Fixed-Weight optimization.

MEDS also proposes a technique to reduce the size of the signature by representing group elements by linear combination of a known base $(\mathbf{A}_1, \ldots, \mathbf{A}_k)$. In particular, a matrix $\mathbf{A} \in \mathbb{F}_q^{m \times n}$ can be written as $\mathbf{A} = \sum_i \lambda_i \mathbf{A}_i$, requiring only the $\lambda_i \in \mathbb{F}_q$ to be transmitted, obtaining a compression to $k\lceil \log_2 q \rceil$ bits. This optimization is not currently part of the MEDS proposed parameters, and we refer to [60] for more details.

Concrete Parameters The elements of the group $\mathrm{GL}_n(q) \times \mathrm{GL}_m(q)$ can be represented with $\ell_{\mathrm{G}} = (n^2 + m^2)\lceil \log_2 q \rceil$ bits. By using the Fixed-Weight optimization with t repetitions and ω non-zero challenges, and the Multiple Public Keys optimization with s keys, the size of the signature is given by Equation (4.3). The proposed parameters for MEDS are shown in Table 4.2.

4.4.4 Alternating Trilinear Form Equivalence

In this section we introduce ALTEQ [185], a signature scheme based on the hardness of the *Alternating Trilinear Form Equivalence* problem.

Hardness Assumptions A *trilinear form* is a function $\phi \colon \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$ that is linear in each of its three arguments. A trilinear form ϕ is *alternating* if ϕ evaluates to 0 whenever two arguments are the same, i.e., $\phi(\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{x}) = \phi(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{x}) = 0$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_q^n$.

Definition 4.43 (Alternating Trilinear Form Equivalence Problem)**.** Given two alternating trilinear forms $\phi, \psi \colon \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$, find, if any, an invertible matrix $\mathbf{A} \in \mathrm{GL}_n(q)$ such that, for any $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{F}_q^n$,

$$\psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \phi(\mathbf{A}\boldsymbol{x}, \mathbf{A}\boldsymbol{y}, \mathbf{A}\boldsymbol{z}).$$

In the following, we denote the alternating trilinear form defined by $\phi(\mathbf{A}\boldsymbol{x}, \mathbf{A}\boldsymbol{y}, \mathbf{A}\boldsymbol{z})$ as $\phi \circ \mathbf{A}$.

Group Action We can consider the following action of $\mathrm{GL}_n(q)$ on the set of alternating trilinear forms $\mathrm{ATF}(n, q) = \{\phi \colon \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q\}$:

$$\star_{\mathsf{ATFE}} \colon \mathrm{GL}_n(q) \times \mathrm{ATF}(n, q) \to \mathrm{ATF}(n, q), (\mathbf{A}, \phi) \mapsto \phi \circ \mathbf{A}.$$

We can see that the GAIP for the above action is the Alternating Trilinear Form Equivalence problem. The signature scheme can be obtained by instantiating Protocol 4.30 with $\star_{\mathsf{ATFE}}$ and applying the Fiat-Shamir transform as described in Algorithm 4.3.

Signature Optimizations The proposed parameterizations for ALTEQ [37] include the Fixed-Weight (Section 4.3.3) and the Multiple Public Keys (Section 4.3.4) optimizations. The Seed Tree (Section 4.3.2) optimization is considered in the ALTEQ specification, but is not currently employed, pending further analysis on the trade-offs by the authors. More in detail, for each NIST security level I, III, and V, the ALTEQ specification proposes two sets of parameters with a strong trade-off between signature size and public key size due to heavy use of the Multiple Public Keys optimization.

Concrete Parameters The elements of the group $\mathrm{GL}_n(q)$ can be represented with $\ell_{\mathsf{G}} = n^2 \lceil \log_2 q \rceil$ bits. By using the Fixed-Weight optimization with t repetitions and ω non-zero challenges without Seed Tree, and the Multiple Public Keys optimization with s keys, the size of the signature is given by Equation (4.3).

$$2\lambda + \lambda + (t - \omega)\lambda + \omega \ell_{\mathsf{G}}.$$

The proposed parameters for ALTEQ are shown in Table 4.3.

Table 4.1: Proposed parameters for LESS [13] with corresponding key/signature sizes.

Set	NIST Cat.	Action Params			Signature Params			|pk|	|sig|
		n	k	q	t	ω	s	(KiB)	(KiB)
LESS-1b	I	252	126	127	247	30	2	13.6	8.4
LESS-1i					244	20	4	40.8	5.8
LESS-1s					198	17	8	95.2	5.0
LESS-3b	III	400	200	127	759	33	2	34.2	16.8
LESS-3s					895	26	3	68.5	13.4
LESS-5b	V	548	274	127	1352	40	2	64.2	29.8
LESS-5b					907	37	3	128.5	26.6

Table 4.2: Proposed parameters for MEDS [60] with corresponding key/signature sizes.

Set	NIST Cat.	Action Params		Signature Params			|pk|	|sig|
		q	(n, m, k)	t	ω	s	(KiB)	(KiB)
MEDS-9923	I	4093	14	1152	14	4	9.7	9.7
MEDS-13320				192	20	5	12.9	12.7
MEDS-41711	III	4093	22	608	26	4	40.7	40.1
MEDS-69497				160	36	5	54.3	53.5
MEDS-134180	V	2039	30	192	52	5	131.0	129.4
MEDS-167717				112	66	6	163.8	161.6

Table 4.3: Proposed parameters for ALTEQ [37] with corresponding key/signature sizes.

Set	NIST Cat.	Action Params		Centr. Sig. Params			|pk|	|sig|
		n	q	t	ω	s	(KiB)	(KiB)
Balanced-I	I	13	$2^{32} - 5$	84	22	7	7.9	15.6
Short-I				16	14	458	511.7	9.3
Balanced-III	III	20	$2^{32} - 5$	201	28	7	31.2	47.9
Short-III				39	20	229	1019.8	31.8

Chapter 5

Aggregate and Multi-Signatures from Group Actions

In this chapter, we investigate the design of aggregate signatures based on cryptographic group actions. We begin by proposing a generic sequential aggregation framework, based on the half-aggregation approach introduced in [54, 58] for Schnorr signature and later adapted in [48] for lattice-based Fiat-Shamir signature aggregation. Recall that a signature in the Fiat-Shamir paradigm typically consists of a challenge-response pair, from which one can retrieve the commitment and verify the validity of the transcript. Equivalently, it is possible to construct the signature from the commitment-response pair, but this solution is typically disadvantageous in terms of signature size. Half-aggregation schemes consider the second approach and aggregate the commitment component of the individual signatures. In the case of Schnorr signatures, this still allows a good compression rate. On the other hand, when the length of challenges is much smaller than the length of commitments and responses, the resulting compression is lower and even negative for a small number of users, as in the case of the lattice-based scheme of [48]. Instead, our scheme proposes aggregation of the challenge component, achieving positive aggregation for any number of users. Unfortunately, the interest in this scheme is only theoretical, as the achievable compression rates are extremely limited for concrete signature schemes based on group actions.

To overcome the limitations of sequential aggregation, in Section 5.2 we investigate the design of an interactive multi-signature. The intuition behind the scheme is to construct a distributed Σ-protocol that applies the Multiple Public Key optimization (Section 4.3.4) by distributing public keys among protocol participants. Despite the interactive component, the size of the multi-signature still grows linearly with the number of participants. Nevertheless, we show that by adopting some further optimizations studied in Section 4.3, it is possible to achieve relevant compression rates. Finally, in Section 5.3, we apply the multi-signature scheme to the group action-based signatures introduced in Section 4.4,

Algorithm 5.1: Group-Action History-Free SAS (GA-HF-SAS)

Let $c_0 = \varepsilon, z_0 = \varepsilon$. The random oracle is $\mathsf{H}\colon \{0,1\}^* \to \{0,1\}^t$. ρ_i and ς_i are the partial description and the complementary information of the aggregate signature Σ_i, respectively.

```
Setup($1^\lambda$):
    $x_0 \leftarrow\!\!\$\; X$
    $\mathsf{pp} \leftarrow x_0$
    return pp
```

```
AggSign($(x_i, g_i), m_i, \rho_{i-1}$):
    $(c_{i-1}, z_{i-1}) \leftarrow \rho_{i-1}$
    for $j \leftarrow 1, \dots, t$ do
        $\tilde{g}_i^{(j)} \leftarrow\!\!\$\; G$
    $\tilde{x}_i \leftarrow [\tilde{g}_i^{(j)} \star x_0]_{j\in[t]}$
    $\tilde{c}_i \leftarrow \mathsf{H}(\tilde{x}_i, x_i, m_i, z_{i-1})$
    $c_i \leftarrow c_{i-1} \oplus \tilde{c}_i$
    $z_i \leftarrow [\tilde{g}_i^{(j)} g_i^{-c_i^{(j)}}]_{j\in[t]}$
    $\rho_i \leftarrow (c_i, z_i)$
    $\varsigma_i \leftarrow z_i$
    return $\rho_i, \varsigma_i$
```

```
KGen($\mathsf{pp} = x_0$):
    $g \leftarrow\!\!\$\; G$
    $x \leftarrow g \star x_0$
    return $\mathsf{pk} \leftarrow x, \mathsf{sk} \leftarrow g$
```

```
AggVrfy($L_n, \bar{\Sigma}_n$):
    $(x_1, m_1), \dots, (x_n, m_n) \leftarrow L_n$
    $(c_n, z_1, \dots, z_n) \leftarrow \bar{\Sigma}_n$
    for $i \leftarrow n, \dots, 1$ do
        $\tilde{x}_i \leftarrow [z_i^{(j)} \star x_{i\cdot c_i^{(j)}}]_{j\in[t]}$
        $\tilde{c}_i \leftarrow \mathsf{H}(\tilde{x}_i, x_i, m_i, z_{i-1})$
        $c_{i-1} \leftarrow c_i \oplus \tilde{c}_i$
    return $c_0 = \varepsilon$
```

```
Combine($\varsigma_1, \dots, \varsigma_{n-1}, \rho_n$):
    $(z_1, \dots, z_{n-1}) \leftarrow (\varsigma_1, \dots, \varsigma_{n-1})$
    $(c_n, z_n) \leftarrow \rho_n$
    return $\bar{\Sigma}_n \leftarrow (c_n, z_1, \dots, z_n)$
```

specifically LESS [13], MEDS [60] and ALTEQ [37]. For each scheme, we analyse the compression capabilities and the reduction in the number of protocol iterations compared to the concatenation of individual signatures.

5.1 Sequential Half-Aggregation of Group Action-Based Signatures

We present a PS-HF-SAS scheme based on cryptographic group action signature schemes. The properties of the underlying cryptographic group action $(G, X, \star)$ are described in Section 4.2, and the signature scheme is obtained by applying the Fiat-Shamir transform as described in Section 4.2.3. To obtain a reduction from EUF-CMA, we also require the action to be regular. Note that we can always consider a transitive action by restricting the action to the orbit of a single element. Also, in application to signatures, the action is typically required to be free to ensure the security of the scheme (Corollary 4.33).

Aggregation of signatures is accomplished by sequentially combining challenges and preserving the entire list of responses. In more detail, the first user's signature for message m_1 with public key $x_1 \in X$ is obtained as in Algorithm 4.3 and is given by the challenge-response pair (c_1, z_1). Then, if a second user with public key $x_2 \in X$ wants to add their own signature of a message m_2, the aggregation proceeds as follows:

1. The commitment is obtained as in the plain signature of Algorithm 4.3, by sampling $\tilde{g}_2^{(1)}, \dots, \tilde{g}_2^{(t)} \in G$ and computing $\tilde{x}_2 \leftarrow [\tilde{g}_2^{(j)} \star x_2]_{j \in [t]}$. Here, we are using the notation $\tilde{g}_i^{(j)}$ to denote the j-th parallel repetition of the i-th user.

2. The challenge is computed in two steps. First, by taking the output $\tilde{c}_2$ of a random oracle H on the commitment $\tilde{x}_2$, the public key x_2 and the message m_2 of the second user and on the response z_1 of the previous user. Then, the challenge is given by the aggregation of $\tilde{c}_2$ and the previous aggregated challenge c_1, i.e., $c_2 \leftarrow c_1 \oplus \tilde{c}_2$.

3. Finally, the response z_2 is computed as in Algorithm 4.3 on commitment $\tilde{x}_2$ with challenge c_2, i.e., $z_2 \leftarrow [\tilde{g}_2^{(j)} g_2^{-c_2^{(j)}}]_{j \in [t]}$, where g_2 is the secret key for x_2.

Since we are considering a partial-signature SAS, only partial information of the so-far aggregate signature is required for subsequent aggregation. Using the notation of Definition 3.6, the partial description of the signature ϱ_i produced by the i-th user is given by (c_i, z_i), while the complementary information ς_i required for the full description of the aggregate signature is given by z_i. The full description of the aggregate signature of n signers is then given by the final aggregated challenge c_n and the full list of responses $z_1, \dots, z_n$ from all signers. A detailed description of the scheme is given in Algorithm 5.1.

Notice that the trivial concatenation of n signatures is given by $(c_1, \dots, c_n, z_1, \dots, z_n)$. Compared with the construction of [48], where aggregation is done on the commitments $\tilde{x}$ and compression is positive once the size of $(c_1, \dots, c_n)$ is greater than $\tilde{x}$, in our scheme we always achieve positive compression. Unfortunately, in group action-based signatures, the size of responses is typically much larger than the size of challenges, so although the proposed construction provides a half-aggregation, the compression rate achieved is only about 1%.

5.1.1 Security Proof

In the following, we prove the PS-HF-UF-CMA security of Algorithm 5.1. We assume that $(G, X, \star)$ is a regular cryptographic group action, Π is the Σ-protocol of Protocol 4.30 for the relation associated with GAIP. We assume that Π is repeated t times before applying the Fiat-Shamir transform, and $\mathsf{Sig} = \mathsf{FS}[\Pi^t]$ is the signature scheme described in Algorithm 4.3. In the proof, M is the message space of Sig and Ch is the challenge space of Π^t.

Theorem 5.1 (PS-HF-UF-CMA Security). *Let $\mathcal{A}$ be a* PS-HF-UF-CMA *adversary against the* GA-HF-SAS *scheme on* Sig *in the random oracle model, which makes* $\mathsf{q_S}$ *signing queries to* OAggSign, $\mathsf{q_H}$ *queries to the random oracle* H. *Then, there exists an* EUF-CMA *adversary $\mathcal{B}$ against* Sig *issuing* $\mathsf{q_S}$ *signing queries to* OSign *and* $\mathsf{q_{H'}}$ *to the random oracle* H′, *such that*

$$\mathsf{Adv}^{\mathsf{PS\text{-}HF\text{-}UF\text{-}CMA}}_{\mathsf{GA\text{-}HF\text{-}SAS}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{Sig}}(\mathcal{B}) + \frac{\mathsf{q_S}}{|X|^t} + \frac{\mathsf{q_H^2}}{2^t} + \frac{\mathsf{q_H q_S}}{|\mathsf{M}|},$$

and the running time of $\mathcal{B}$ is about that of $\mathcal{A}$

In the following, we sketch the high-level idea of the proof; full details can be found after Lemmas 5.2 and 5.3 at the end of this section. We prove the reduction by showing that the PS-HF-UF-CMA game can be simulated by the EUF-CMA adversary $\mathcal{B}$. The structure of the proof is similar to that of Theorem 3.8, but is partially simplified because of the direct reduction to the signature rather than the underlying security assumption. In particular, we will again make use of a labelled tree HTree whose nodes will be populated by some of the queries to the random oracle H. The HTree is initialized with a root node with a single value $h_0 = \varepsilon$. Each subsequent node N_i is added following a query to the random oracle H with input $Q_i = (\tilde{x}_i, x_i, m_i, z_{i-1})$ and will store the following values:

- a reference to its parent node N_{i-1};
- the query Q_i to the random oracle H;
- the aggregated challenge $c_i \leftarrow c_{i-1} \oplus \tilde{c}_i$ obtained by combining the aggregated challenge c_{i-1} of the previous node N_{i-1} and the challenge sampled as output of the random oracle $\tilde{c}_i$. This value will be used to establish if future nodes can be added as children of N_i.

A node N_{i+1} can be added as a child of a node N_i if it satisfies the relation $[z_i^{(j)} \star x_{i \cdot c_i^{(j)}}]_{j \in [t]} = \tilde{x}_i$, where $x_i, \tilde{x}_i, c_i$ are stored in N_i, while z_i is stored in N_{i+1}. This relationship establishes that the query Q_{i+1} can be properly used by the signer with key x_{i+1} to aggregate their signature on message m_{i+1} with previous response z_i, produced by key x_i on commitment $\tilde{x}_i$ and challenge c_i, which in turn are stored in N_i. Whenever a query $Q_i = (\tilde{x}_i, x_i, m_i, z_{i-1})$, with $z_{i-1} \neq \varepsilon$, satisfies this relation with a node N_{i-1} we say that Q_i can be *tethered* to N_{i-1}. If $z_{i-1} = \varepsilon$, then Q_i can always be tethered to the root of the HTree.

Eventually, when $\mathcal{A}$ outputs a valid aggregate signature $\bar{\Sigma}_n = (c_n, z_1, \ldots, z_n)$ for the history $L_n = (x_1, m_1), \ldots, (x_n, m_n)$, if $\mathcal{A}$ is winning, there exists an index

$i^\star \in [n]$ such that $x_{i^\star} = x^\star$ and $m_{i^\star} \notin \mathcal{Q}$. Then, $\mathcal{B}$ recovers $\tilde{x}_{i^\star}$ and $c_{i^\star}$ by iterating the procedure of Lines 3 to 6 in AggVrfy for $n - i^\star$ steps. Since $\bar{\Sigma}_n$ is valid, $Q_{i^\star} = (\tilde{x}_{i^\star}, x^\star, m_{i^\star}, z_{i^\star-1})$ must have been queried to H. Then $\mathcal{B}$ checks if $Q_{i^\star}$ is stored in a node $N^\star \in$ HTree as a child of a node $N_{i^\star-1}$ storing $Q_{i^\star-1} = (\tilde{x}_{i^\star-1}, x_{i^\star-1}, m_{i^\star-1}, z_{i^\star-2})$, which in turn is a child of a node originating from N_1. If no such node exists, it aborts by raising $\mathsf{bad}_{\mathsf{teth}}$. Otherwise, the message $m^\star$, produced on Line 9 of H, is extracted from $N^\star$ and $\mathcal{B}$ wins the EUF-CMA game returning the signature $(c_{i^\star}, z_{i^\star})$ on message $m^\star$.

For the complete proof of Theorem 5.1 we need the following technical lemmas, which are the analogues of Lemmas 3.9 and 3.11, respectively.

Lemma 5.2. *When a new node is added to the* HTree *as a result of a call to* H, *the aggregated challenge c_i is uniformly distributed in* Ch.

Proof. When a new node is added to HTree, it is populated either on Line 12 or on Line 16 of H. Let $Q = (\tilde{x}_i, x_i, m_i, z_{i-1})$ be the input to H. If $x_i = x^\star$, then c_i is taken as the output of the outer random oracle H′ taking as input a fresh random value $m \leftarrow_\$ \mathsf{M}$. Otherwise, c_i is computed as $c_{i-1} \oplus \tilde{c}_i$, where $\tilde{c}_i$ is sampled uniformly at random from Ch. Therefore, c_i is chosen randomly and is independent of the view of $\mathcal{A}$. □

Lemma 5.3. *If an input Q has not been entered in the* HTree *after being queried to* H, *the probability that it will ever become tethered to a node in* HTree *is at most* $\mathsf{q}'/2^t$, *where* q' *is the number of queries made to* H *after* Q.

Proof. Suppose that $Q = (\tilde{x}, x_l, m, z)$ was queried to H and was not added to the HTree, i.e. $\mathsf{Lookup}(z) = \perp$. Now suppose that a query $Q' = (\tilde{x}', x_k, m', z')$ was subsequently sent to H and was added to HTree as part of a node N' with aggregated challenge c'. For Q to be tethered to N', it must hold that $[z^{(j)} \star x_{k \cdot c'^{(j)}}]_{j \in [t]} = \tilde{x}'$. Following Lemma 5.2, when a new node is added to the HTree as a result of a call to H, the aggregated challenge c' is uniformly distributed in Ch. In particular, c' is random and independent of x_k and z. Therefore, the probability of having $[z^{(j)} \star x_{k \cdot c'^{(j)}}]_{j \in [t]} = \tilde{x}'$ is at most 2^{-t}. Since there are at most q' queries to H after Q and each query can add at most one node to the HTree, the desired probability follows by the union bound. □

Proof for PS-HF-UF-CMA security (Theorem 5.1)

Proof. We now prove the reduction by presenting a sequence of hybrid games, modifying the PS-HF-UF-CMA game (Experiment 3.2) until it can be simulated by the EUF-CMA adversary $\mathcal{B}$. The complete reduction is described in Algorithm 5.2. In the following, we use the notation $\Pr[\mathsf{Game}_\mathsf{n}(\mathcal{A}) = 1]$ to denote the probability that Game_n returns 1 when played by $\mathcal{A}$. The game sequence Game_0-Game_3 for OAggSign is detailed in Argument 5.1. The game sequence Game_0-Game_3 for H is detailed in Argument 5.2.

Algorithm 5.2: Full Reduction EUF-CMA $\Longrightarrow$ PS-HF-UF-CMA

```
$\mathcal{B}(x^\star)$:
$\mathcal{Q} \leftarrow \varnothing$
$(L_n, \bar{\Sigma}_n) \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{H},\mathsf{OAggSign}}(x^\star)$
$(x_1, m_1), \ldots, (x_n, m_n) \leftarrow L_n$
$(c_n, z_1, \ldots, z_n) \leftarrow \bar{\Sigma}_n$
if $\mathsf{AggVrfy}(L_n, \bar{\Sigma}_n) \wedge \exists i^\star : (x_{i^\star} = x^\star \wedge$
  $m_{i^\star} \notin \mathcal{Q})$ then
    Recover $c_{i^\star}$ as in AggVrfy
    $N^\star \leftarrow \mathsf{Lookup}(z_{i^\star})$
    if $N^\star = \perp$ then
        raise $\mathsf{bad}_{\mathsf{teth}}$
    Retrieve $m^\star$ from $N^\star$
    if $m^\star \in \mathcal{Q}$ then
        raise $\mathsf{bad}_{\mathsf{mcol}}$
    Return $(m^\star, (c_{i^\star}, z_{i^\star}))$
```

```
$\mathsf{OAggSign}(m_i, \rho_{i-1})$:
$(c_{i-1}, z_{i-1}) \leftarrow \rho_{i-1}$
$x_i \leftarrow x^\star$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{m_i\}$
$(c_i, z_i) \leftarrow\!\!\$\ \mathsf{OSign}(m_i)$
$\tilde{x}_i \leftarrow [z_i^{(j)} \star x_{i \cdot c^{(j)}}]_{j \in [t]}$
$\tilde{c}_i \leftarrow c_{i-1} \oplus c_i$
$Q \leftarrow (\tilde{x}_i, x_i, m_i, z_{i-1})$
if $\mathsf{HT}[Q] \neq \perp$ then
    raise $\mathsf{bad}_{\mathsf{ctcol}}$
$\mathsf{HT}[Q] \leftarrow \tilde{c}_i$
return $(c_i, z_i), z_i$
```

```
$\mathsf{H}(\tilde{x}_i, x_i, m_i, z_{i-1})$:
$Q \leftarrow (\tilde{x}_i, x_i, m_i, z_{i-1})$
if $\mathsf{HT}[Q] \neq \perp$ then
    return $\mathsf{HT}[Q]$
$N_{i-1} \leftarrow \mathsf{Lookup}(z_{i-1})$
if $N_{i-1} \neq \perp$ then
    $N_i \leftarrow$ new node with parent $N_{i-1}$
    Retrieve $c_{i-1}$ from $N_{i-1}$
    if $x_i = x^\star$ then
        $m \leftarrow\!\!\$\ \mathsf{M}$
        $c_i \leftarrow \mathsf{H}'(\tilde{x}_i, x_i, m)$
        $\tilde{c}_i \leftarrow c_{i-1} \oplus c_i$
        Populate node $N_i$ with $Q, c_i, m$
    else
        $\tilde{c}_i \leftarrow\!\!\$\ \mathsf{Ch}$
        $c_i \leftarrow c_{i-1} \oplus \tilde{c}_i$
        Populate node $N_i$ with $Q, c_i$
else
    $\tilde{c}_i \leftarrow\!\!\$\ \mathsf{Ch}$
$\mathsf{HT}[Q] \leftarrow \tilde{c}_i$
return $\tilde{c}_i$
```

```
$\mathsf{Lookup}(z)$:
if $z = \varepsilon$ then
    return Root of HTree
$\mathsf{NList} \leftarrow \{N \in \mathsf{HTree} : (x_i, \tilde{x}_i, c_i) \in$
  $N \wedge [z^{(j)} \star x_{i \cdot c_i^{(j)}}]_{j \in [t]} = \tilde{x}_i\}$
if $|\mathsf{NList}| > 1$ then
    raise $\mathsf{bad}_{\mathsf{tcol}}$
else if $|\mathsf{NList}| = 0$ then
    return $\perp$
else
    return node in NList
```

Hybrid Argument 5.1: Games for OAggSign($m_i, \rho_{i-1} = (c_{i-1}, x_{i-1})$)
Let $x_i = x^\star$. The oracle returns ρ_i, ς_i.

```
Game$_0$:
  $Q \leftarrow Q \cup \{m_i\}$
  $\tilde{g}_i \leftarrow\!\!\$\; G^t$
  $\tilde{x}_i \leftarrow [\tilde{g}_i^{(j)} \star x_0]_{j\in[t]}$
  $\tilde{c}_i \leftarrow \mathsf{H}(\tilde{x}_i, x_i, m_i, z_{i-1})$
  $c_i \leftarrow c_{i-1} \oplus \tilde{c}_i$
  $z_i \leftarrow [\tilde{g}_i^{(j)} g_i^{-c_i^{(j)}}]_{j\in[t]}$
  return $(c_i, z_i), z_i$
```

```
Game$_1$-Game$_3$:
  $Q \leftarrow Q \cup \{m_i\}$
  $\tilde{g}_i \leftarrow\!\!\$\; G^t$
  $\tilde{x}_i \leftarrow [\tilde{g}_i^{(j)} \star x_0]_{j\in[t]}$
  $Q \leftarrow (\tilde{x}_i, x_i, m_i, z_{i-1})$
  if $\mathsf{HT}[Q] \neq \perp$ then
      raise bad$_{\mathsf{ctcol}}$
  $\tilde{c}_i \leftarrow\!\!\$\; \mathsf{Ch}$
  $\mathsf{HT}[Q] \leftarrow \tilde{c}_i$
  $c_i \leftarrow c_{i-1} \oplus \tilde{c}_i$
  $z_i \leftarrow [\tilde{g}_i^{(j)} g_i^{-c_i^{(j)}}]_{j\in[t]}$
  return $(c_i, z_i), z_i$
```

Game$_0$ This is the original PS-HF-UF-CMA game against the GA-HF-SAS scheme, except that it uses programmable random oracles. At the start of the game, the challenger initializes a table HT for H. When a query Q for H is received, if $\mathsf{HT}[Q] = \perp$ it uniformly samples $\tilde{c} \leftarrow\!\!\$\; \mathsf{Ch}$ and stores $\mathsf{HT}[Q] \leftarrow \tilde{c}$, finally it returns $\mathsf{HT}[Q]$. It follows that $\Pr[\mathsf{Game}_0(\mathcal{A}) = 1] = \mathsf{Adv}^{\mathsf{PS\text{-}HF\text{-}UF\text{-}CMA}}_{\mathsf{GA\text{-}HF\text{-}SAS}}(\mathcal{A})$.

Game$_1$ This game is identical to Game$_0$ except that OAggSign aborts by raising bad$_{\mathsf{ctcol}}$ if on query $(m_i, \rho_{i-1} = (c_{i-1}, z_{i-1}))$, after sampling $\tilde{g}_i \leftarrow\!\!\$\; G^t$ it computes $\tilde{x}_i \leftarrow [\tilde{g}_i^{(j)} \star x_0]_{j\in[t]}$ such that the random oracle H was already queried at input $Q = (\tilde{x}_i, x^\star, m_i, z_{i-1})$, i.e. $\mathsf{HT}[Q] \neq \perp$. Otherwise, it samples $\tilde{c}_i \leftarrow\!\!\$\; \mathsf{Ch}$ and programs $\mathsf{HT}[Q] \leftarrow \tilde{c}_i$. It follows that $|\Pr[\mathsf{Game}_0(\mathcal{A}) = 1] - \Pr[\mathsf{Game}_1(\mathcal{A}) = 1]| \leq \Pr[\mathsf{bad}_{\mathsf{ctcol}}]$.

Game$_2$ This game is identical to Game$_1$ except that the random oracle H is simulated as follows. At the start of the game, the challenger initializes a labelled tree HTree, as described at the beginning of the proof. When a query $Q = (\tilde{x}_i, x_i, m_i, z_{i-1})$ for H is received and $\mathsf{HT}[Q] = \perp$, it samples a uniformly random challenge $\tilde{c}_i \leftarrow\!\!\$\; \mathsf{Ch}$. Then, it checks if Q can be added as a child node of existing nodes in HTree. To determine whether this is the case, it uses the Lookup function (see Algorithm 5.2) on input z_{i-1}. The Lookup function determines whether there exists a node $N_{i-1} \in$ HTree storing a public key x_{i-1}, a commitment $\tilde{x}_{i-1}$ and an aggregated challenge c_{i-1} such that z_{i-1} is a valid response, i.e. $[z_{i-1}^{(j)} \star x_{(i-1)\cdot c_{i-1}^{(j)}}]_{j\in[t]} = \tilde{x}_{i-1}$. If Q can be tethered to more than one node, the game aborts by raising bad$_{\mathsf{tcol}}$. Otherwise, H add a new node N_i with parent N_{i-1} returned by Lookup(z_{i-1}). N_i contains the original query

Hybrid Argument 5.2: Games for $\mathsf{H}(\tilde{x}_i, x_i, m_i, z_{i-1})$

```
Game_0-Game_1:
$Q \leftarrow (\tilde{x}_i, x_i, m_i, z_{i-1})$
if $\mathsf{HT}[Q] \neq \perp$ then
    return $\mathsf{HT}[Q]$
$\tilde{c}_i \leftarrow\$ \mathsf{Ch}$
$\mathsf{HT}[Q] \leftarrow \tilde{c}_i$
return $\tilde{c}_i$
```

```
Game_2:
$Q \leftarrow (\tilde{x}_i, x_i, m_i, z_{i-1})$
if $\mathsf{HT}[Q] \neq \perp$ then
    return $\mathsf{HT}[Q]$
$\tilde{c}_i \leftarrow\$ \mathsf{Ch}$
$N_{i-1} \leftarrow \mathsf{Lookup}(z_{i-1})$
if $N_{i-1} \neq \perp$ then
    $N_i \leftarrow$ new node with parent $N_{i-1}$
    Retrieve $c_{i-1}$ from $N_{i-1}$
    $c_i \leftarrow c_{i-1} \oplus \tilde{c}_i$
    Populate node $N_i$ with $Q, c_i$
$\mathsf{HT}[Q] \leftarrow \tilde{c}_i$
return $\tilde{c}_i$
```

```
Game_3:
$Q \leftarrow (\tilde{x}_i, x_i, m_i, z_{i-1})$
if $\mathsf{HT}[Q] \neq \perp$ then
    return $\mathsf{HT}[Q]$
$N_{i-1} \leftarrow \mathsf{Lookup}(z_{i-1})$
if $N_{i-1} \neq \perp$ then
    $N_i \leftarrow$ new node with parent $N_{i-1}$
    Retrieve $c_{i-1}$ from $N_{i-1}$
    if $x_i = x^\star$ then
        $m \leftarrow\$ \mathsf{M}$
        if $m \in \mathcal{Q}$ then
            raise $\mathsf{bad}_{\mathsf{mcol}}$
        $c_i \leftarrow\$ \mathsf{Ch}$
        $\tilde{c}_i \leftarrow c_{i-1} \oplus c_i$
        Populate node $N_i$ with $Q, c_i, m$
    else
        $\tilde{c}_i \leftarrow\$ \mathsf{Ch}$
        $c_i \leftarrow c_{i-1} \oplus \tilde{c}_i$
        Populate node $N_i$ with $Q, c_i$
else
    $\tilde{c}_i \leftarrow\$ \mathsf{Ch}$
$\mathsf{HT}[Q] \leftarrow \tilde{c}_i$
return $\tilde{c}_i$
```

Q and the aggregated challenge $c_i \leftarrow c_{i-1} \oplus \tilde{c}_i$, where c_{i-1} is the previous aggregated challenge stored in N_{i-1}. Finally, H programs $\mathsf{HT}[Q] \leftarrow \tilde{c}_i$ and returns $\tilde{c}_i$. It holds that $|\Pr[\mathsf{Game}_1(\mathcal{A}) = 1] - \Pr[\mathsf{Game}_2(\mathcal{A}) = 1]| \leq \Pr[\mathsf{bad}_{\mathsf{tcol}}]$.

Game_3 This game is identical to Game_2 except that the random oracle H is simulated as follows. When H receives a query $Q = (\tilde{x}_i, x_i, m_i, z_{i-1})$ with $x_i = x^\star$, if Q can be tethered to a node N_i it samples a uniformly random message $m \leftarrow\$ \mathsf{M}$ and aborts by raising $\mathsf{bad}_{\mathsf{mcol}}$ if the message has been queried to OAggSign, i.e., $m \in \mathcal{Q}$. Otherwise, it samples the aggregated challenged at random $c_i \leftarrow\$ \mathsf{Ch}$ and sets the random oracle response as $\tilde{c}_i \leftarrow c_{i-1} \oplus c_i$, where c_{i-1} is the previous aggregated challenge stored in N_{i-1}. Finally, it stores the random message m to the new node N_i. Notice that if the game does not abort, it always produces a uniformly random value in Ch. It holds that $|\Pr[\mathsf{Game}_2(\mathcal{A}) = 1] - \Pr[\mathsf{Game}_3(\mathcal{A}) = 1]| \leq \Pr[\mathsf{bad}_{\mathsf{mcol}}]$.

In the following, we show that the EUF-CMA adversary $\mathcal{B}$ can simulate Game_3 using the outer signing oracle OSign and the outer random oracle H'.

- When OAggSign receives a query $(m_i, (c_{i-1}, z_{i-1}))$, $\mathcal{B}$ queries the outer signing oracle OSign on m_i and receives a challenge c_i and a response z_i. From (c_i, z_i) it recovers the commitment $\tilde{x}_i$ computed by OSign and programs the random oracle H on query $(\tilde{x}_i, x_i, m_i, z_{i-1})$ with the aggregated challenge $\tilde{c}_i \leftarrow c_{i-1} \oplus c_i$. Finally, $\mathcal{B}$ returns $\rho_i \leftarrow (c_i, z_i)$ and $\varsigma_i \leftarrow z_i$.
- When H receives a query $Q = (\tilde{x}_i, x_i, m_i, z_{i-1})$ with $x_i = x^\star$, if Q can be tethered to a node $N_i \in$ HTree, $\mathcal{B}$ samples a uniformly random message $m \leftarrow_\$ \mathsf{M}$, queries the outer random oracle H′ on $(\tilde{x}_i, x_i, m)$ and uses the response c_i as the aggregated challenge described in Game$_3$.

If none of the bad events happen, $\mathcal{B}$ wins the EUF-CMA game if and only if $\mathcal{A}$ wins Game$_3$. Therefore, it holds that

$$\begin{aligned}\mathsf{Adv}^{\text{EUF-CMA}}_{\text{CGA-S}}(\mathcal{B}) &= \Pr[\text{Game}_3(\mathcal{A}) = 1] \\ &\geq \mathsf{Adv}^{\text{PS-HF-UF-CMA}}_{\text{GA-HF-SAS}}(\mathcal{A}) - \Pr[\text{bad}_{\text{ctcol}}] - \Pr[\text{bad}_{\text{tcol}}] \\ &\quad - \Pr[\text{bad}_{\text{mcol}}] - \Pr[\text{bad}_{\text{teth}}].\end{aligned}$$

$\mathcal{B}$ can simulate Game$_3$ with at most the same running time of $\mathcal{A}$ plus the time required for running AggVrfy and simulating the queries to the random oracle H, and to the signing oracle OAggSign. These operations are polynomial in the security parameter and are repeated at most $\mathsf{q_H} + \mathsf{q_S} + 1$ times.

In the following, we bound the probability of each bad event happening.

Probability of bad$_{\text{ctcol}}$ The event bad$_{\text{ctcol}}$ occurs on Line 9 of OAggSign in input $(m_i, \rho_i = (c_{i-1}, z_{i-1}))$ when, after querying the outer signing oracle OSign on m_i and receiving (c_i, z_i), it recovers a commitment $\tilde{x}_i$ such that a value for $Q = (\tilde{x}_i, x_i, m_i, z_{i-1})$ was already assigned in HT. Since $\tilde{x}_i$ is generated by OSign, for each query to OAggSign, this happens with probability at most

$$\max_{\tilde{x}=(\tilde{x}^{(1)},\ldots,\tilde{x}^{(t)})} \Pr\left[\tilde{x}^{(j)} = \tilde{g}^{(j)} \star x_0, \forall j \in [t] \;\middle|\; \tilde{g}^{(j)} \leftarrow_\$ G\right].$$

If the action is regular, the map $g \mapsto g \star x_0$ is a bijection and for any fixed $\tilde{x} \in X$ it holds that

$$\Pr\left[\tilde{x} = \tilde{g} \star x_0 \;\middle|\; \tilde{g} \leftarrow_\$ G\right] = \Pr[\tilde{x} = x \mid x \leftarrow_\$ X] = \frac{1}{|X|}.$$

Since at most $\mathsf{q_S}$ are made to OAggSign, then $\Pr[\text{bad}_{\text{ctcol}}] \leq \mathsf{q_S}|X|^{-t}$.

Probability of bad$_{\text{tcol}}$ The event bad$_{\text{tcol}}$ occurs on Line 5 of Lookup on input z when the HTree contains more than one tethered node. This happens if there are two nodes $N_i, N_j \in \mathsf{HTree}$, storing $(x_i, \tilde{x}_i, c_i)$ and $(x_j, \tilde{x}_j, c_j)$, such that z is a common response, i.e.

$$[z^{(k)} \star x_{i \cdot c_i^{(k)}}]_{k \in [t]} = \tilde{x}_i \quad \text{and} \quad [z^{(k)} \star x_{j \cdot c_j^{(k)}}]_{k \in [t]} = \tilde{x}_j.$$

Each query to H add at most one node to HTree, so the tree has a maximum of $\mathsf{q_H}$ nodes. Therefore, we need to bound the probability that any pair of nodes in HTree has a common response z.

When a new node is added to HTree, the public key x_i and the commitment $\tilde{x}_i$ are part of the input to H and may be adversarially chosen, while, from Lemma 5.2, the aggregated challenge c_i is a uniformly random value in Ch. If the action is regular, then the response z is uniquely determined by $x_i, \tilde{x}_i$ and c_i, and the map $x \mapsto z^{(j)} \star x$ is a bijection for all $j \in [t]$. Therefore, assuming $|G| \geq 2^\lambda$, the best strategy for an adversary is to choose $x_j = x_i$ and $\tilde{x}_j = \tilde{x}_i$. Since c_j is uniformly distributed and independent of c_i, it follows that $[z^{(k)} \star x_{j \cdot c_j^{(k)}}] = \tilde{x}_j$ happens with probability $1/2$ for all $j \in [t]$.

Since the number of node pairs in the HTree is at most $\mathsf{q_H^2}/2$, by the union bound we obtain $\Pr[\mathsf{bad_{tcol}}] \leq \mathsf{q_H^2}/2^{t+1}$.

Probability of $\mathsf{bad_{mcol}}$ The event $\mathsf{bad_{mcol}}$ occurs on Line 12 of the simulation of $\mathcal{B}$ if the message $m^\star$ sampled on Line 9 of H was queried to OAggSign. There are at most $\mathsf{q_S}$ queries to OAggSign, thus the probability that a uniformly random message produces a collision with one of the queries is at most $\mathsf{q_S}/|\mathsf{M}|$. Since at most $\mathsf{q_H}$ queries are made to H, then $\Pr[\mathsf{bad_{mcol}}] \leq \mathsf{q_H q_S}/|\mathsf{M}|$.

Probability of $\mathsf{bad_{teth}}$ The event $\mathsf{bad_{teth}}$ occurs on Line 9 of the simulation of $\mathcal{B}$ when, after the adversary $\mathcal{A}$ outputs a valid aggregate signature $\bar{\Sigma}_n$ for the history $L_n = (\mathsf{pk}_1, m_1), \ldots, (\mathsf{pk}_n, m_n)$, the simulator recovers $x_{i^\star}$, with $i^\star \in [n]$ such that $\mathsf{pk}_{i^\star} = \mathsf{pk}^\star$ and $m_{i^\star} \notin \mathcal{Q}$, but $x_{i^\star}$ cannot be tethered to any node in the HTree.

When $\mathsf{bad_{teth}}$ happens, the aggregate signature $\bar{\Sigma}_n$ must be valid on L_n. In particular, the inputs $Q_1 = (\mathsf{F}_1, m_1, r_1, \varepsilon), Q_2 = (\mathsf{F}_2, m_2, r_2, x_1), \ldots, Q_{i^\star} = (\mathsf{F}_{i^\star}, m_{i^\star}, r_{i^\star}, x_{i^\star - 1})$ have been queried to H in OAggVrfy. Let $\tilde{c}_1, \ldots, \tilde{c}_{i^\star}$ be the outputs of these queries, so that $\mathsf{HT}[Q_j] = \tilde{c}_j$. Each of these entries has been populated by H. In fact, the only exception could occur if $(m_{i^\star}, z_{i^\star - 1})$ was queried to OAggSign. But this cannot happen since otherwise $m_{i^\star} \in \mathcal{Q}$ and the aggregate signature would not be a valid forgery.

Each step of OAggVrfy also recovers a value $c_i \leftarrow c_{i+1} \oplus \tilde{c}_{i+1}$ which is the aggregated challenge at the i-th step. Since the aggregate signature is correct, we obtain that $c_i = \tilde{c}_1$. Observe that since Q_1 was queried to H, it must be tethered to the root of HTree and was therefore inserted as a node of HTree with aggregated challenge $c_i = c_{i-1} \oplus \tilde{c}_i$. Then, since $[z_i^{(j)} \star x_{i\cdot c_i^{(j)}}]_{j\in[t]} = \tilde{x}_i$, the query Q_2 is tethered to N_1. Now we prove that either all $Q_1, \ldots, Q_{i^\star}$ are part of a path of nodes in HTree, or there exists an input Q_k that was queried to H, is tethered to a node in HTree and is not itself in a node of HTree. We proceed by induction on $k \leq i^\star$: we have already shown that Q_1 is in HTree; suppose that Q_k is in the HTree, then, since $[z_k^{(j)} \star x_{k\cdot c_k^{(j)}}]_{j\in[t]} = \tilde{x}_k$, the query Q_{k+1} is tethered to Q_k and it may or may not be part of HTree. To conclude, we prove that if an input Q has not been entered in the HTree after being queried to H, the probability that it will ever become tethered to a node in HTree is at most $\mathsf{q}'/2^t$, where q' is the number of queries made to H after Q (Lemma 5.3). Since there are at most q_H queries that add new nodes to HTree, we obtain, by the union bound, that $\Pr[\mathsf{bad}_\mathsf{teth}] \leq \mathsf{q}_\mathsf{H}^2/2^{t+1}$.

Combining the previous bound on bad events, we obtain the claimed estimate of $\mathsf{Adv}^{\text{PS-HF-UF-CMA}}_{\text{GA-HF-SAS}}(\mathcal{A})$. □

5.1.2 Support for Standard Optimizations

In Section 4.3 we described the main techniques used to optimize group action-based signature schemes. All the techniques mentioned are easily extended to the sequential aggregation scheme described in this section. Note that the aggregation procedure in Algorithm 5.1, only modifies the way challenges are computed by aggregating them with the previous signer's challenge. In particular, the computation of the responses is not changed during aggregation; thus, any signature optimization that only affects the responses is immediately applicable to the aggregated signature. This is the case with the Compression of Random Elements (Section 4.3.1) and the Seed Trees (Section 4.3.2) optimizations.

On the other hand, Fixed-Weight (Section 4.3.3) and Multiple Public Keys (Equation (4.3)) optimizations modify the challenge structure, in the former case by unbalancing the distribution of the challenge space, while in the latter by expanding the binary space of the single instance to a multi-bit one. In both cases, the simple technique of aggregating challenges via bit-by-bit XOR described in Algorithm 5.1 is no longer applicable. To recover correctness, however, it is sufficient to modify the protocol so that the output of the random oracle is a seed of length 2λ, which can be expanded into the challenge space by using an appropriate Pseudorandom Number Generator PRNG. Aggregation then takes place on the seeds in the same manner as described in the original scheme. More in detail, if we consider an arbitrary challenge space Ch, we use a random oracle $\mathsf{H}\colon \{0,1\}^* \to \{0,1\}^{2\lambda}$ and $\mathsf{PRNG}\colon \{0,1\}^{2\lambda} \to \mathsf{Ch}$, and the challenge is computed as follows. Suppose the i-th signer with public key x_i wants to aggregate their signature on message m_i on a previous partial aggregation given by (c_{i-1}, z_{i-1}).

After computing the commitment $\tilde{x}_i$, as in the original protocol, the signer takes $\tilde{c}_i \leftarrow \mathsf{H}(\tilde{x}_i, x_i, m_i, z_{i-1})$ and computes $c_i \leftarrow c_{i-1} \oplus \tilde{c}_i$. Now, instead of using c_i as a challenge, the signers take $c_i' \leftarrow \mathsf{PRNG}(c_i)$ to compute the response. Notice that the partial description of the aggregate signature is now given by (c_i, z_i), so that the expanded challenge c_i' is only used in computation of the responses. The same expansion procedure is used at the verification stage to derive intermediate commitments.

5.2 Multi-Signature from Cryptographic Group Action

In this section we propose a new multi-signature based on cryptographic group actions. The intuition behind the scheme is to adapt the Multi Public Keys technique, described in Section 4.3.4 in the context of group action-based Σ-protocols, to an interactive protocol in which the individual signer's public keys are distributed among multiple parties. To prove the security of the multi-signature, we consider a variant of the original Σ-protocol. Intuitively, this variant allows the prover to artificially enlarge the challenge space using ephemeral keys in the commitment phase. The resulting signature is inefficient for a direct application, but it more accurately captures the perspective of the individual signer in the multi-signature scheme. We show that from this variant it is possible to obtain a secure signature by applying the Fiat-Shamir transform. This allows us to describe a multi-signature scheme whose security can be formally reduced to the hardness of the Group Action Inverse Problem.

In the following, we assume that in an interactive protocol between n parties $P_1, \dots, P_n$ each party has access to point-to-point communication channels. Moreover, when the interactive protocol is run by a party P_i, we write $x \rightarrowtriangle P_j$ to denote the transmission of x from P_i to P_j. Similarly, we write $x \leftarrowtriangle P_j$ to denote a transmission from P_j to P_i and the subsequent assignment to x.

5.2.1 Multi-Signatures

A multi-signature scheme MS is a tuple of four algorithms $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{MuSign}, \mathsf{MuVrfy})$.

- $\mathsf{Setup}(1^\lambda)$: takes as input a security parameter 1^λ and outputs a public parameter pp.
- $\mathsf{KGen}(\mathsf{pp})$: takes as input a public parameter pp and generates a key pair $(\mathsf{pk}, \mathsf{sk})$.

Experiment 5.1: $\mathsf{MS\text{-}UF\text{-}CMA}_{\mathsf{MS}}$

$\mathcal{M}_{\mathsf{sid}}$ is a machine running the instruction of the party P_i in the multi-signature protocol $\mathsf{MuSign}(\mathsf{sid}, \mathsf{sk}^\star, \mathsf{pk}^\star, m, L)$, where $L = (\mathsf{pk}_1, \dots, \mathsf{pk}_n)$ such that $\mathsf{pk}_i = \mathsf{pk}^\star$.

```
$\mathcal{Q} \leftarrow \varnothing$; $\mathcal{S} \leftarrow \varnothing$
$\mathsf{pp} \leftarrow_\$ \mathsf{Setup}(\lambda)$
$(\mathsf{pk}^\star, \mathsf{sk}^\star) \leftarrow_\$ \mathsf{KGen}(\mathsf{pp})$
$(L, m, \sigma) \leftarrow_\$ \mathcal{A}^{\mathsf{O}, \mathsf{OMuSign}}(\mathsf{pk}^\star)$
if $\mathsf{pk}^\star \notin L \lor (m, L) \in \mathcal{Q}$ then
    return $\bot$
return $\mathsf{MuVrfy}(L, m, \sigma)$
```

```
$\mathsf{OMuSign}(\mathsf{sid}, msg)$:
if $\mathsf{sid} \notin S$ then
    $(m, L) \leftarrow msg$
    if $\mathsf{pk}^\star \notin L$ then
        return $\bot$
    $\mathcal{M}_{\mathsf{sid}} \leftarrow_\$ \mathsf{MuSign}(\mathsf{sid}, \mathsf{sk}^\star, \mathsf{pk}^\star, m, L)$
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, L)\}$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathsf{sid}\}$
    return $\mathcal{M}_{\mathsf{sid}}()$
return $\mathcal{M}_{\mathsf{sid}}(msg)$
```

- $\mathsf{MuSign}(\mathsf{sid}, \mathsf{sk}, \mathsf{pk}, m, L)$: is an interactive protocol that is run by a party P_i taking as input a session ID sid, a key pair $(\mathsf{pk}, \mathsf{sk})$, a message to be signed m and an ordered set of co-signers' public keys $L = (\mathsf{pk}_1, \dots, \mathsf{pk}_n)$ such that $\mathsf{pk}_i = \mathsf{pk}$. The protocol terminates with each party obtaining a signature σ as output.

- $\mathsf{MuVrfy}(L, m, \sigma)$: takes as input an ordered set of public keys L, a message m and a signature σ and returns 1 for acceptance or 0 for rejection.

Below, we show the definition of Multi-Signature existential Unforgeability against Chosen-Message Attacks (MS-UF-CMA). In this model, the forger controls all signers' private keys except for at least one honest signer. The forger can choose the keys of the rogue signers and adaptively query an aggregate signature oracle. Finally, to win the experiment, the forger must produce a valid, non-trivial multi-signature involving the public key of the honest signer. The security notion is adapted from [69].

Definition 5.4 (MS-UF-CMA Security)**.** Let O be a random oracle, let $\mathsf{MS} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{MuSign}, \mathsf{MuVrfy})$ be a multi-signature scheme, and let $\mathcal{A}$ be an adversary. We define the advantage of $\mathcal{A}$ playing the MS-UF-CMA game (Experiment 5.1) against MS in the random oracle model as:

$$\mathsf{Adv}^{\mathsf{MS\text{-}UF\text{-}CMA}}_{\mathsf{MS}}(\mathcal{A}) = \Pr[\mathsf{MS\text{-}UF\text{-}CMA}_{\mathsf{MS}}(\mathcal{A}) = 1].$$

We say that MS is *existential unforgeable against chosen-message* attacks if the advantage $\mathsf{Adv}^{\mathsf{MS\text{-}UF\text{-}CMA}}_{\mathsf{MS}}(\mathcal{A})$ is negligible for any adversary $\mathcal{A}$.

5.2.2 Sigma Protocol Variant

In the following we present a variant of the base Σ-protocol for cryptographic group action (Protocol 4.30), and we prove the EUF-CMA security of the associated signature. The variant allows the signer to use ephemeral keys during signature creation. Although this has no impact on security, the modified signature allows the behaviour of an adversary to be abstracted more accurately in the multi-signature protocol, and is therefore a useful tool in the security proof of the upcoming scheme.

To prove the security of the modified signature we show that the underlying Σ-protocol is a (quantum) proof of knowledge. We start by taking a simplified variant $\Pi[n]$, with a fixed number of $n-1$ ephemeral keys, showing that it is a proof of knowledge roughly equivalent to the basic protocol Π, with a higher knowledge error. Next, we consider the full variant Π' where the number of ephemeral keys is chosen by the prover, and we show that we can use the knowledge extractor for $\Pi[n]$ to extract a witness from a dishonest prover against the full variant, moreover we show that this does not change the knowledge error of the protocol. Finally, we show that if the base protocol Π is complete, has high min-entropy, and is HVZK, then Π' also has the same properties. We observe that these properties are required in concrete applications to construct digital signatures from group actions, so we make the same assumptions in the construction of the variant. Finally, we obtain a secure signature scheme by applying the Fiat-Shamir transform.

Variant with fixed ephemeral keys

We start with a slight modification of the base Σ-protocol for a cryptographic group action $(G, X, \star)$, as detailed in Protocol 4.30. In the base protocol, to prove the knowledge of a secret $g_1 \in G$ such that $x_1 = g_1 \star x_0$, the prover proceeds as follows: first, it samples a random $\tilde{g} \in G$ and computes the commitment as $\tilde{x} \leftarrow \tilde{g} \star x_0$. Then, they exhibit a path from either x_0 or x_1 to $\tilde{x}$ based on the challenge of the verifier. Recall from Proposition 4.31, that this protocol is complete, special-sound and honest-verifier zero-knowledge.

Given $n > 1$, we denote with $\Pi[n]$ the variant Σ-protocol where the prover additionally samples $n-1$ ephemeral keys $\hat{x}_2, \dots, \hat{x}_n \in X$ during the commitment phase. The verifier can then request a path to $\tilde{x}$ from x_0, x_1 or one of the ephemeral keys. The protocol is detailed in Algorithm 5.3. This procedure artificially increases the challenge space to the set $\{0, \dots, n\}$ and increases the knowledge error to $(n-1)/n$. Clearly, this is a step back from the basic protocol, but it effectively represents a protocol where all but one key is potentially under the control of an adversary.

Following the proof of Proposition 4.31, it is straightforward to prove that $\Pi[n]$ is a Σ-protocol.

Algorithm 5.3: Group Action Σ-Protocol with n ephemeral keys

```
Setup: Group $G$ acting on $X$ via $\star$. Choose $x_0 \in X, n \in \mathbb{N}$. Set $\hat{x}_0 = x_0, \hat{g}_0 = e$.
Private key: $g_1 \in G$.
Public key: $x_1 \leftarrow g_1 \star x_0$. Set $\hat{x}_1 = x_1$.

    Prover($g_1, x_1$)                                                Verifier($x_1$)
    for $j \leftarrow 2, \dots, n$ do
        $\hat{g}_j \xleftarrow{\$} G$
        $\hat{x}_j \leftarrow \hat{g}_j \star x_0$
    $\tilde{g} \xleftarrow{\$} G$
    $\tilde{x} \leftarrow \tilde{g} \star x_0$     $\xrightarrow{\hat{x}_2, \dots, \hat{x}_n, \tilde{x}}$
                                  $\xleftarrow{\text{ch}}$          ch $\xleftarrow{\$} [n]$
    $z \leftarrow \tilde{g}\hat{g}_{\text{ch}}^{-1}$               $\xrightarrow{z}$
                                                               Accept if $z \star \hat{x}_{\text{ch}} = \tilde{x}$
```

Proposition 5.5. $\Pi[n]$ *is perfectly complete,* $(n+1)$*-special-sound and honest-verifier zero-knowledge.*

Proof. We prove each property separately.

Completeness In an honest execution, the verifier receives $z = \tilde{g}\hat{g}_{\text{ch}}^{-1}$. Completeness easily follows since $z \star \hat{x}_{\text{ch}} = \tilde{g}\hat{g}_{\text{ch}}^{-1}\hat{g}_{\text{ch}} \star x_0 = \tilde{x}$.

$(n+1)$-Special Soundness Suppose the extractor has access to $n+1$ accepting transcript $(\text{com}, \text{ch}_1, z_1), \dots, (\text{com}, \text{ch}_{n+1}, z_{n+1})$ with $\text{com} = (\hat{x}_2, \dots, \hat{x}_n, \tilde{x})$ and pairwise distinct challenges $\text{ch}_1, \dots, \text{ch}_{n+1}$. It follows that there exists $i, j \in [n+1]$ such that $\text{ch}_i = 1$ and $\text{ch}_j = 0$. Then it can run the special-soundness extractor from Proposition 4.31 on $(\text{com}, \text{ch}_i, z_i), (\text{com}, \text{ch}_j, z_j)$ to extract the witness g_1.

Honest-Verifier Zero-Knowledge A polynomial time simulator can be obtained as follows. On input a public key $x_1 \in X$, the simulator randomly samples $\text{ch} \xleftarrow{\$} \{0, \dots, n\}, z \xleftarrow{\$} G$, and $\hat{g}_2, \dots, \hat{g}_n \xleftarrow{\$} G$. Then it computes $\hat{x}_j \leftarrow \hat{g}_j \star x_0$ and $\tilde{x} \leftarrow z \star \hat{x}_{\text{ch}}$. Finally, it returns the transcript $(\text{com} = (\hat{x}_2, \dots, \hat{x}_n, \tilde{x}), \text{ch}, z)$. Notice that in the real distribution, $\hat{x}_2, \dots, \hat{x}_n$ and $\tilde{x}$ are uniformly distributed in the orbit of x_0, ch is uniformly distributed in $\{0, \dots, n\}$, and $z \in G$ is uniquely determined by $z \star \hat{x}_{\text{ch}} = \tilde{x}$. This is the same in the simulated distribution. □

Since $\Pi[n]$ is $(n+1)$-out-of-$(n+1)$ special-sound Σ-protocol, we can take its parallel repetition and still obtain a proof of knowledge. In particular, by applying Theorem 4.15, we obtain that the t-fold parallel repetition of $\Pi[n]$ is knowledge-sound with knowledge error $n^t/(n+1)^t$. Moreover, the same approach can be applied to show that the t-fold parallel repetition of $\Pi[n]$, with a single sample of the ephemeral keys $\hat{x}_2, \dots, \hat{x}_n$, is still knowledge-sound with knowledge error $n^t/(n+1)^t$.

Variant with variable ephemeral keys

Next, we further modify the protocol by allowing the prover to choose the number of ephemeral keys to be generated. In the following, given a security parameter λ, x_0 is a fixed element in X, $g_0 = e_G$ is the identity of G, and N is a fixed positive integer. Given $n \in \mathbb{N}$, let $t(n)$ be the minimum positive integer such that $(n/(n+1))^{t(n)} \leq 2^{-\lambda}$. Let $\mathsf{Ch} \subseteq [0,N]^{t(N)}$ and $f_n \colon \mathsf{Ch} \to [0,n]^{t(n)}$ be a family of maps such that $f_n^{-1}\left(\mathcal{U}([0,n]^{t(n)})\right) \sim \mathcal{U}(\mathsf{Ch})$ for any $n \in [1,N]$.

Protocol 5.6 (Group Action Σ-protocol with Ephemeral Keys)**.** Given the public parameters $\mathsf{pp} = (G, X, \star, x_0, g_0, N, \mathsf{Ch}, \{f_n\})$, the protocol proceeds as follows:

- $(g_1, x_1) \leftarrow\$ \mathsf{Gen}(\mathsf{pp})$: the key-generation algorithm takes as input the public parameters pp. It uniformly samples $g_1 \in G$ and computes $x_1 \leftarrow g_1 \star x_0$. It returns the witness-statement pair (x_1, g_1).
- $\mathsf{com} \leftarrow\$ \mathsf{P}_1(g_1, x_1)$: given a statement $x_1 \in X$ and the corresponding witness $g_1 \in G$, the prover chooses $n \in [1,N]$. Then, it uniformly samples $\hat{g}_k$ and computes $\hat{x}_k \leftarrow \hat{g}_k \star x_0$ for $k \in [2,n]$. Then, it uniformly samples $\tilde{g}^{(j)}$ and computes $\tilde{x}^{(j)} \leftarrow \tilde{g}^{(j)} \star x_0$ for $j \in [1, t(n)]$. Finally, it returns $\mathsf{com} \leftarrow (\hat{x}_2, \dots, \hat{x}_n, \tilde{x}^{(1)}, \dots, \tilde{x}^{(t(n))})$.
- $\mathsf{ch} \leftarrow\$ \mathsf{V}_1(\mathsf{com})$: given a commitment com, the verifier returns a uniformly random challenge $\mathsf{ch} \in \mathsf{Ch}$.
- $\mathsf{rsp} \leftarrow \mathsf{P}_2(g_1, x_1, \mathsf{com}, \mathsf{ch})$: given a statement x_1, the corresponding witness g_1, a commitment $\mathsf{com} = (\hat{x}, \tilde{x})$ and a challenge $\mathsf{ch} \in \mathsf{Ch}$, the prover sets $\hat{g}_1 = g_1, \hat{x}_1 = x_1$ and computes $\mathsf{ch}' \leftarrow f_n(\mathsf{ch})$. Then, for each component $\mathsf{ch}'_j \in [0,n]$ of ch', they compute a response $z_j \leftarrow \tilde{g}^{(j)}\hat{g}^{-1}_{\mathsf{ch}'_j}$ for $j \in [1, t(n)]$. Finally, they output $\mathsf{rsp} \leftarrow (z_1, \dots, z_{t(n)})$.
- $\{0,1\} \leftarrow \mathsf{V}_2(x_1, \mathsf{com}, \mathsf{ch}, \mathsf{rsp})$: given a statement $x_1 \in X$, a commitment $\mathsf{com} = (\hat{x}, \tilde{x})$, a challenge ch and a response $\mathsf{rsp} = (z_1, \dots, z_{t(n)})$, the verifier proceeds as follows. They compute $\mathsf{ch}' \leftarrow f_n(\mathsf{ch})$ and set $\hat{x}_1 = x_1$. Then, for each $j \in [1, t(n)]$, compute $\tilde{y}^{(j)} \leftarrow z_j \star \hat{x}_{\mathsf{ch}'_j}$. The verifier accepts (returns 1) if $\tilde{y} = \tilde{x}$, otherwise rejects (returns 0).

We denote Protocol 5.6 with Π'. Following the proof of Proposition 5.5, it is straightforward to prove that Π' is complete and HVZK, where the latter follows by modifying the simulator of $\Pi[n]$ so that it randomly samples $n \leftarrow_{\$} \{1, \dots, N\}$. Additionally, we show that Π' is a proof of knowledge. On a high level, we show that any dishonest prover $\mathcal{P}^*$ attacking Π' can be used to build a prover $\mathcal{P}_n^*$ against $\Pi[n]^{t(n)}$ with the same success probability of $\mathcal{P}^*$. Since $\Pi[n]^{t(n)}$ is a proof of knowledge, it is possible to extract the witness from $\mathcal{P}_n^*$.

In Π', the challenge space Ch is taken together with a family of surjective maps $f_n \colon \mathsf{Ch} \to [0,n]^{t(n)}$ such that $f_n^{-1}(\mathcal{U}([0,n]^{t(n)})) \sim \mathcal{U}(\mathsf{Ch})$. Let $\mathcal{P}^*$ be a deterministic prover[1] attacking Π' on input x_1. More precisely, on input $c \in \mathsf{Ch}$, $\mathcal{P}^*$ outputs a fixed first message a and its response z. We define $V \colon \mathsf{Ch} \times \{0,1\}^* \to \{0,1\}$ the function that runs the verification check that the verifier performs on the transcript (a, c, z). $\mathcal{P}^*$ is successful on input c if $V(c, \mathcal{P}^*(c)) = 1$. Since $\mathcal{P}^*$ is deterministic, the number of ephemeral keys n is fixed and known from the first message a. Then, such a prover naturally induces a (probabilistic) prover $\mathcal{P}_n^*$ attacking $\Pi[n]^{t(n)}$. More precisely, on input $c' \in [0,n]^{t(n)}$, $\mathcal{P}_n^*$ randomly samples $c \leftarrow_{\$} f_n^{-1}(c')$, runs $z \leftarrow \mathcal{P}^*(c)$, and outputs z. Since $\Pi[n]^{t(n)}$ is knowledge-sound, we can use the protocol extractor on $\mathcal{P}_n^*$ and estimate its success probability with respect to the success probability of $\mathcal{P}^*$.

We know that the knowledge error for $\Pi[n]^{t(n)}$ is $\kappa_n^{t(n)}$ with $\kappa_n = n/(n+1)$. Then, given access to $\mathcal{P}_n^*$, the extractor for $\Pi[n]^{t(n)}$ succeeds with probability at least $(\varepsilon(x_1, \mathcal{P}_n^*) - \kappa^{t(n)})/\mathsf{poly}(|x_1|)$. To conclude, it is enough to show that the success probability of $\mathcal{P}_n^*$ is the same as for $\mathcal{P}^*$:

$$\begin{aligned}
\varepsilon(x_1, \mathcal{P}_n^*) &= \Pr_{c' \leftarrow \$[0,n]^{t(n)}}[V(c, \mathcal{P}_n^*(c')) = 1] \\
&= \Pr_{c' \leftarrow \$[0,n]^{t(n)}}\left[V(c, \mathcal{P}^*(c)) = 1 \mid c \leftarrow_{\$} f_n^{-1}(c')\right] \\
&= \Pr_{c \leftarrow \$\mathsf{Ch}}[V(c, \mathcal{P}^*(c)) = 1] = \varepsilon(x_1, \mathcal{P}^*).
\end{aligned}$$

Therefore, the probability that the extractor on $\mathcal{P}^*$ is successful is at least

$$\frac{\varepsilon(x_1, \mathcal{P}^*) - \kappa_n^{t(n)}}{\mathsf{poly}(|x_1|)}.$$

The previous holds for any prover $\mathcal{P}^*$ with n ephemeral keys. Moreover, by taking $n = N$, it holds for any dishonest prover against Π'. Therefore, Π' is a knowledge-sound Σ-protocol.

[1] In the proof of Proposition 4.14, we have observed how it is always possible to reduce to a deterministic prover.

Algorithm 5.4: Variant Signature Scheme based on Group Actions

$\star\colon G \times X \to X$ is a cryptographic group action. $\mathsf{H}^{(n)}\colon \{0,1\}^* \to [0,n]^{t(n)}$ is a random oracle.

```
Setup($1^\lambda$):
  $x_0 \leftarrow_{\$} X$
  $\mathsf{pp} \leftarrow x_0$
  return $\mathsf{pp}$

KGen($\mathsf{pp} = x_0$):
  $g_1 \leftarrow_{\$} G$
  $x_1 \leftarrow g_1 \star x_0$
  return $(\mathsf{pk} = x_1, \mathsf{sk} = g_1)$

Vrfy($\mathsf{pk} = \hat{x}_1, m, \sigma$):
  $(L, \tilde{x}, z_1, \dots, z_{t(n)}) \leftarrow \sigma$
  $(\hat{x}_2, \dots, \hat{x}_n) \leftarrow L$
  $\mathsf{ch} \leftarrow \mathsf{H}^{(n)}(L, \tilde{x}, m)$
  for $j \leftarrow 1, \dots, t(n)$ do
      $\tilde{x}'_j \leftarrow z_j \star \hat{x}_{\mathsf{ch}_j}$
  $\tilde{x}' \leftarrow (\tilde{x}'_1, \dots, \tilde{x}'_{t(n)})$
  return $\tilde{x}' = \tilde{x}$
```

```
Sign($\mathsf{sk}, \mathsf{pk}, m, n$):
  $\hat{x}_0 \leftarrow x_0$; $\hat{g}_0 \leftarrow e$
  $\hat{x}_1 \leftarrow \mathsf{pk}$; $\hat{g}_1 \leftarrow \mathsf{sk}$
  for $k \leftarrow 2, \dots, n$ do
      $\hat{g}_k \leftarrow_{\$} G$
      $\hat{x}_k \leftarrow \hat{g}_k \star x_0$
  $L \leftarrow (\hat{x}_2, \dots, \hat{x}_n)$
  for $j \leftarrow 1, \dots, t(n)$ do
      $\tilde{g}^{(j)} \leftarrow_{\$} G$
      $\tilde{x}^{(j)} \leftarrow \tilde{g}^{(j)} \star x_0$
  $\tilde{x} \leftarrow (\tilde{x}^{(1)}, \dots, \tilde{x}^{t(n)})$
  $\mathsf{ch} \leftarrow \mathsf{H}^{(n)}(L, \tilde{x}, m)$
  for $j \leftarrow 1, \dots, t(n)$ do
      $z_j \leftarrow \tilde{g}^{(j)} \hat{g}_{\mathsf{ch}_j}^{-1}$
  return $\sigma \leftarrow (L, \tilde{x}, z_1, \dots, z_{t(n)})$
```

Application of the Fiat-Shamir transform

By applying the Fiat-Shamir transform to the protocol Π', we obtain a digital signature scheme $\mathsf{FS}[\Pi']$. Notice that Π' is not commitment-recoverable, so that the signature is obtained by taking the transcript of Π' without the challenge. The challenge can be recovered as the digests of a hash function H on the commitment com and the message m. In the signature scheme, instead of computing f_n on the output of H, we can consider an additional argument for the hash function and write $\mathsf{H}^{(n)} = \mathsf{H}(n, \cdot)\colon \{0,1\}^* \to [0,n]^{t(n)}$. Notice that this description still falls within the random oracle model and can be instantiated, for instance, by using an extendable-output function (XOF). The full description of the signature scheme can be found in Algorithm 5.4.

Theorem 5.7. *Let Π' be as in Protocol 5.6 for a cryptographic group action $(G, X, \star)$ with base element x_0. If no polynomial-time (quantum) adversary can solve the GAIP (Definition 4.26) and the Stabilizer Computation Problem (Definition 4.29) for x_0 except with a negligible probability, then* $\mathsf{FS}[\Pi']$ *(Algorithm 5.4) is strong* EUF-CMA *in the (quantum) random oracle model.*

Proof. To prove the (strong) EUF-CMA security of the signature scheme, we apply

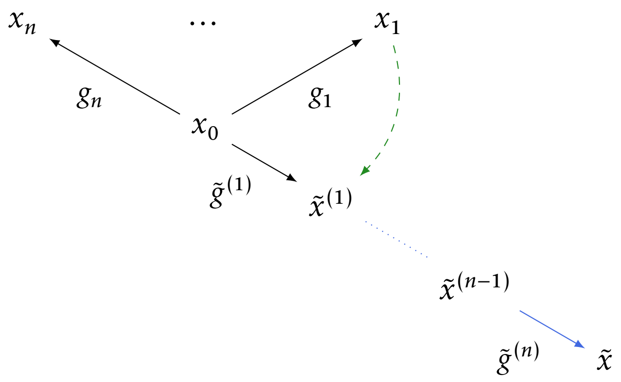


Figure 5.1: High-level description of MS-GA scheme of Algorithm 5.5, answering on ch = 1. P_1 reveals the map from x_1 to $\tilde{x}^{(1)}$, while all other parties reveal the ephemeral group element $\tilde{g}^{(j)}$.

Theorem 4.20 on FS[Π']. We already proved that the Σ-protocol Π' is complete, HVZK, and knowledge-sound. It remains to prove that it has computational unique responses. From Lemma 4.32, we know that the base protocol Π (Protocol 4.30) has computational unique responses if and only if the Stabilized Computation Problem for x_0 is computationally hard. It is easy to see that since the sigma protocol Protocol 4.30 has computational unique responses, also the sigma protocol variants Π' has computational unique responses. The proof follows as in Corollary 4.33. □

5.2.3 The Multi-Signature Scheme

In this section, we present a multi-signature scheme based on cryptographic group actions, for which the key pair used by the signing parties are compatible with the key pairs of standard digital signatures based on group actions presented in Section 4.2.3.

The multi-signature scheme is designed in a way that it closely resembles the centralized digital signature scheme described in the previous section. In fact, in the multi-signature that we present in this section, the ephemeral signing keys used in the centralized signature in Algorithm 4.3 are replaced by the signing keys of the other parties taking part in the signing process. We recall that the centralized signature in Algorithm 4.3 is not efficient in any way, but it is unforgeable under chosen message attacks. In the security analysis, we will reduce the security of the multi-signature scheme, according to Definition 5.4, to the unforgeability of the centralized digital signature.

At a high level, the multi-signature signing algorithm that we present instructs each party in the signing set to perform the following operations. Suppose L is an ordered signing set of n users $P_1, \ldots, P_n$. Each party P_i in L randomly generates a salt r_i, which will be used to generate a shared randomness associated to the signing session, and contributes to the creation of the sigma protocol commitment $\tilde{x}$. In particular, the parties in L perform the following operations in a round-robin fashion:

1. P_i commits to a random salt r_i, by computing com_i, a commitment which binds r_i also to the commitments of the parties acting before P_i in the ordered set L[2];

2. P_i contributes in the generation of the sigma protocol commitment $\tilde{x}$ by generating $t(n)$ group elements $\tilde{g}^{(i)} = (\tilde{g}_1^{(i)}, \dots, \tilde{g}_{t(n)}^{(i)})$ and using them to compute the partial commitment in $\tilde{x}^{(i)}$ starting from the partial commitment it received from the party acting before it, namely $\tilde{x}^{(i-1)}$.

Then P_i sends to P_{i+1} the cryptographic commitment com_i and the sigma protocol partial commitments $\tilde{x}^{(i)}$. The same operations are repeated by each party, until the last signing party P_n, who broadcasts its commitment com_n and $\tilde{x} = \tilde{x}^{(n)}$. Then, all the parties reveal their randomness r_i and check that the cryptographic commitments have been honestly computed. If this is the case, each party computes the shared randomness $r \leftarrow \mathsf{H}_1(r_1, \dots, r_n)$ which acts as a session identifier. Using the shared randomness r and the commitment $\tilde{x}$, the parties generate the challenge $\mathsf{ch} \leftarrow \mathsf{H}_2^{(n)}(\tilde{x}, r, L, m)$, a string of $t(n)$ elements in $\{0,1,\dots,n\}$. In the response phase, a challenge $\mathsf{ch}_j = i \neq 0$ requires revealing a map from x_i, the public key of P_i to $\tilde{x}_j = \tilde{x}_j^{(n)}$. Each party P_k, for $k \neq i$, reveals the ephemeral group element $\tilde{g}_j^{(k)}$. P_i then computes the response as

$$z_j = \Big(\prod_{k=0}^{n-1} \tilde{g}_j^{(n-k)}\Big) g_i^{-1}.$$

Otherwise, if $\mathsf{ch}_j = 0$ then the response is the group element mapping the base point x_0 to $\tilde{x}_i$. Each party reveals the ephemeral group element $\tilde{g}_j^{(k)}$ and the response is computed as

$$z_j = \Big(\prod_{k=0}^{n-1} \tilde{g}_j^{(n-k)}\Big). \tag{5.1}$$

In the latter case, it is agreed that the calculation of z_j is entrusted to the last user P_n. A high-level description of the multi-signature scheme is shown in Figure 5.1 for $\mathsf{ch} = 1$. The full description of the protocol is given in Algorithm 5.5.

[2] Binding the commitment to r_i to the commitments of the previous salts is useful to avoid broadcasting each commitment to each party. The only commitment that must be broadcast and seen by every party is com_n, which is a commitment to all the salts $r_1, \dots, r_n$.

Algorithm 5.5: Multi-Signature from Group Action (MS-GA[$\star$])

$\star\colon G \times X \to X$ is a cryptographic group action. The random oracles are $\mathsf{H}_0\colon \{0,1\}^* \to \{0,1\}^{2\lambda}$, $\mathsf{H}_1\colon \{0,1\}^* \to \{0,1\}^{\ell_{\text{salt}}}$ and $\mathsf{H}_2^{(n)}\colon \{0,1\}^* \to [0,n]^{t(n)}$. During the execution of MuSign, each party maintains a list of active session identifiers in a list $\mathcal{S}$.

```
Setup($1^\lambda$):
  $x_0 \leftarrow_\$ X$
  $\mathsf{pp} \leftarrow x_0$
  return $\mathsf{pp}$

KGen($\mathsf{pp} = x_0$):
  $g \leftarrow_\$ G$
  $x \leftarrow g \star x_0$
  return $(\mathsf{pk} = x, \mathsf{sk} = g)$

MuVrfy($L, m, \sigma$):
  $(x_1, \dots, x_n) \leftarrow L$
  $(\mathsf{ch}, r, z_1, \dots, z_{t(n)}) \leftarrow \sigma$
  for $j \leftarrow 1, \dots, t(n)$ do
      $\tilde{x}_j \leftarrow z_j \star x_{\mathsf{ch}_j}$
  $\tilde{x} \leftarrow (\tilde{x}_1, \dots, \tilde{x}_{t(n)})$
  if $\mathsf{H}_2^{(n)}(\tilde{x}, r, L, m) = \mathsf{ch}$ then
      return 1
  else
      return 0

MuSign(sid, sk, pk, $m$, $L$):
  $(\mathsf{pk}_1, \dots, \mathsf{pk}_n) \leftarrow L$
  if $\mathsf{sid} \in \mathcal{S} \vee \nexists i : \mathsf{pk}_i = \mathsf{pk}$ then
      return $\perp$
  Set $i$ such that $\mathsf{pk}_i = \mathsf{pk}$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathsf{sid}\}$
  $x_i \leftarrow \mathsf{pk}$; $g_i \leftarrow \mathsf{sk}$
  $r_i \leftarrow_\$ \{0,1\}^{\ell_{\text{salt}}}$
  $\mathsf{com}_{i-1}, \tilde{x}^{(i-1)} \leftarrowtriangle P_{i-1}$    ▹ $\mathsf{com}_0 = \varepsilon, \tilde{x}_j^{(0)} = x_0$
  $\mathsf{com}_i \leftarrow \mathsf{H}_0(\mathsf{com}_{i-1}, r_i)$
  $\tilde{g}^{(i)} \leftarrow_\$ G^{t(n)}$
  $\tilde{x}^{(i)} \leftarrow [\tilde{g}_j^{(i)} \star \tilde{x}_j^{(i-1)}]_{j \in [t(n)]}$
  $\tilde{x}^{(i)}, \mathsf{com}_i \rightarrowtriangle P_{i+1}$
  $\tilde{x} \leftarrow (\tilde{x}_1^{(n)}, \dots, \tilde{x}_{t(n)}^{(n)})$    ▹ If $i = n$ send to each party
  $r_i \rightarrowtriangle P_k, r_k \leftarrowtriangle P_k, \forall k \neq i$
  if $\exists j : \mathsf{com}_j \neq \mathsf{H}_0(\mathsf{com}_{j-1}, r_j)$ then
      return $\perp$
  $r \leftarrow \mathsf{H}_1(r_1, \dots, r_n)$
  $\mathsf{ch} \leftarrow \mathsf{H}_2^{(n)}(\tilde{x}, r, L, m)$
  for $j \leftarrow 1, \dots, t(n)$ do
      if $\mathsf{ch}_j = i \vee (\mathsf{ch}_j = 0 \wedge i = n)$ then
          $\tilde{g}_j^{(k)} \leftarrowtriangle P_k, \forall k \neq i$
          $z_j \leftarrow (\prod_{k=0}^{n-1} \tilde{g}_j^{(n-k)}) g_{\mathsf{ch}_j}^{-1}$    ▹ $g_0 = e_G$
          $z_j \rightarrowtriangle P_k, \forall k \neq i$
      else
          $\tilde{g}_j^{(i)} \rightarrowtriangle P_{\mathsf{ch}_j}$
          $z_j \leftarrowtriangle P_{\mathsf{ch}_j}$
  $\sigma \leftarrow (\mathsf{ch}, r, z_1, \dots, z_{t(n)})$
```

5.2.4 Security Proof

In the following, we prove the MS-UF-CMA security of the protocol in Algorithm 5.5. In particular, we reduce security to the EUF-CMA of the centralized signature variant described in Section 5.2.2.

Theorem 5.8. *Let $\star\colon G \times X \to X$ be a cryptographic group action and let Π' be as in Protocol 5.6. Let $\mathcal{A}$ be a* MS-UF-CMA *adversary against* MS-GA[$\star$] *in the random oracle model which makes $\mathsf{q_S}$ signing queries, $\mathsf{q_H}$ queries to the random oracles $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_2$. Then, there exists a* EUF-CMA *adversary $\mathcal{B}$ against* FS[Π'] *issuing $\mathsf{q_S}$*

signing queries and $\mathsf{q_H}$ *queries to the random oracle* H'*, such that*

$$\mathsf{Adv}^{\text{MS-UF-CMA}}_{\text{MS-GA}[\star]}(\mathcal{A}) \leq \mathsf{Adv}^{\text{EUF-CMA}}_{\text{FS}[\Pi']}(\mathcal{B}) + \frac{\mathsf{q_S}(\mathsf{q_H} + \mathsf{q_S})}{2^{\ell_{\text{salt}}}} + \frac{2\mathsf{q_S}\mathsf{q_H}}{2^{\ell_{\text{M}}}},$$

and the running time of $\mathcal{B}$ *is about that of* $\mathcal{A}$*.*

Proof. In the following, we denote the random oracles and the signing oracle in the MS-UF-CMA game as $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_2$ and OMuSign. The EUF-CMA adversary $\mathcal{B}$ has access to a signing oracle OSign of the $\mathsf{FS}[\Pi']$ and an outer random oracle H'. After receiving the target public key $\mathsf{pk}^\star$ in the EUF-CMA game, $\mathcal{B}$ forwards $\mathsf{pk}^\star$ to $\mathcal{A}$.

At a high level, we show that controlling $n-1$ users in the multi-signature is no better than choosing $n-1$ ephemeral keys in the centralized signature. We show that $\mathcal{B}$ can simulate OMuSign by querying OSign and programming the random oracle H_2 with the challenges provided by the outer random oracle H'. During a query to OMuSign, the adversary may choose the value of the commitment $\tilde{x}$ of the sigma protocol by controlling the last user. However, the adversary can not control the value of the shared salt r, and $\mathcal{B}$ is able to program the random oracle H_2 before the adversary learns the value of r. In this way, the manipulation of the final commitment cannot influence the challenge and be exploited in parallel sessions.

The main focus of the reduction concerns the simulation of the random oracle H_2. After receiving a query containing the target public key among the participants' keys, $\mathcal{B}$ queries the outer random oracle H' and reprograms H_2 with a permutation of the received challenge. When $\mathcal{A}$ produces a valid signature, $\mathcal{B}$ will be able to map it into a signature for the centralized scheme using the keys of the users controlled by $\mathcal{A}$ as ephemeral keys.

In the following, we make the simplified assumption that before $\mathcal{A}$ outputs a forged signature, it makes a query on H_2, as would be done during signature verification. Moreover, we assume that $\mathcal{A}$ always outputs a valid signature, and halts by returning $\perp$ otherwise. Notice that we can always modify $\mathcal{A}$ to behave this way by running the verification algorithm on the provided signature, and checking that the message provided was not queried to the signing oracle with the same set of signers.

More in detail, we prove the reduction by presenting a sequence of hybrid games, modifying the MS-UF-CMA game (Experiment 5.1) until it can be simulated by the EUF-CMA adversary $\mathcal{B}$ against the centralized signature $\mathsf{FS}[\Pi']$. In the following, we use the notation $\Pr[\mathsf{Game_n}(\mathcal{A}) = 1]$ to denote the probability that $\mathsf{Game_n}$ returns 1 when played by $\mathcal{A}$. The complete reduction is described in Algorithm 5.6.

Algorithm 5.6: Full Reduction EUF-CMA $\Longrightarrow$ MS-UF-CMA

```
$\mathcal{B}(\mathsf{pk}^\star)$:
  $\mathcal{Q} \leftarrow \varnothing$; $\mathcal{S} \leftarrow \varnothing$; $\mathcal{M} \leftarrow \varnothing$
  $(L, m, \sigma) \xleftarrow{\$} \mathcal{A}^{\mathsf{O},\mathsf{OMuSign}}(\mathsf{pk}^\star)$
  $(x_1, \dots, x_n) \leftarrow L$
  $(\mathsf{ch}, r, z_1, \dots, z_{t(n)}) \leftarrow \sigma$
  if $\mathsf{MuVrfy}(L, m, \sigma) \wedge \exists i : (x_i = x^\star \wedge (m, L) \notin \mathcal{Q})$ then
      Recover $\tilde{x}$ as in MuVrfy
      $m' \leftarrow \mathsf{MT}[\tilde{x}, r, L, m]$
      if $m' \in \mathcal{M}$ then
          raise $\mathsf{bad}_{\mathsf{mcol}}$
      $L' \leftarrow \pi_{1,i}(L)[2:]$
      $\sigma' \leftarrow (L', \tilde{x}, z_1, \dots, z_{t(n)})$
      return $(m', \sigma')$

$\mathsf{H}_0(r)$:
  if $\mathsf{HT}_0[r] = \perp$ then
      $\mathsf{com} \xleftarrow{\$} \{0,1\}^{2\lambda}$
      $\mathsf{HT}_0[r] \leftarrow \mathsf{com}$
  return $\mathsf{HT}_0[r]$

$\mathsf{H}_1(Q)$:
  if $\mathsf{HT}_1[Q] = \perp$ then
      $r \xleftarrow{\$} \{0,1\}^{2\lambda}$
      $\mathsf{HT}_1[Q] \leftarrow r$
  return $\mathsf{HT}_1[Q]$

$\mathsf{H}_2(Q = (\tilde{x}, r, L, m))$:
  if $\mathsf{HT}_2[Q] \neq \perp$ then
      return $\mathsf{HT}_2[Q]$
  $L \leftarrow (x_1, \dots, x_n)$
  if $\nexists i$ such that $x_i = \mathsf{pk}^\star$ then
      $\mathsf{ch} \xleftarrow{\$} \mathsf{Ch}$
  else
      $L' \leftarrow \pi_{1,i}(L)[2:]$
      $m' \xleftarrow{\$} \mathsf{M}$
      $\mathsf{ch}^{\mathcal{B}} \leftarrow \mathsf{H}'(L', \tilde{x}, m')$
      $\mathsf{ch} \leftarrow \pi_{1,i}(\mathsf{ch}^{\mathcal{B}})$
      $\mathsf{MT}[Q] \leftarrow m'$
  $\mathsf{HT}_2[Q] \leftarrow \mathsf{ch}$
  return ch

$\mathsf{OMuSign}(\mathsf{sid}, (m, L))$:
  $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, L)\}$
  $(x_1, \dots, x_n) \leftarrow L$
  if $\mathsf{sid} \in S \vee \nexists i : x_i = x^\star$ then
      return $\perp$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathsf{sid}\}$
  $m' \xleftarrow{\$} \mathsf{M}$
  $\mathcal{M} \leftarrow \mathcal{M} \cup \{m'\}$
  $(\mathsf{com}^{\mathcal{B}}, z_1^{\mathcal{B}}, \dots, z_{t(n)}^{\mathcal{B}}) \xleftarrow{\$} \mathsf{OSign}(m')$
  $(\hat{x}_2^{\mathcal{B}}, \dots, \hat{x}_n^{\mathcal{B}}, \tilde{x}_1^{\mathcal{B}}, \dots, \tilde{x}_{t(n)}^{\mathcal{B}}) \leftarrow \mathsf{com}^{\mathcal{B}}$
  $\mathsf{ch}^{\mathcal{B}} \leftarrow \pi_{1,i}(\mathsf{H}'(\mathsf{com}^{\mathcal{B}}, m'))$
  $r_i \xleftarrow{\$} \{0,1\}^{\lambda}$
  $\mathsf{com}_{i-1}, \tilde{x}^{(i-1)} \leftarrowtriangle P_{i-1}$        ▷ $\mathsf{com}_0 = \varepsilon, \tilde{x}_j^{(0)} = x_0$
  $\mathsf{com}_i \leftarrow \mathsf{H}_0(\mathsf{com}_{i-1}, r_i)$
  for $j \leftarrow 1, \dots, t(n)$ do
      if $\mathsf{ch}_j^{\mathcal{B}} \neq i$ then
          $\tilde{g}_j^{(i)} \xleftarrow{\$} G$
          $\tilde{x}_j^{(i)} \leftarrow \tilde{g}_j^{(i)} \star \tilde{x}_j^{(i-1)}$
      else
          $\tilde{x}_j^{(i)} \leftarrow \tilde{x}_j^{\mathcal{B}}$
  $\tilde{x}^{(i)}, \mathsf{com}_i \rightarrowtriangle P_{i+1}$
  $\tilde{x} \leftarrow (\tilde{x}_1^{(n)}, \dots, \tilde{x}_{t(n)}^{(n)})$
  Retrieve $r_k$ such that $\mathsf{HT}_0[\mathsf{com}_{k-1}, r_k] = \mathsf{com}_k, \forall k \neq i$
  $r \leftarrow \mathsf{H}_1(r_1, \dots, r_n)$
  if $\mathsf{HT}_2[\tilde{x}, r, L, m] \neq \perp$ then
      raise $\mathsf{bad}_{\mathsf{hcol}}$
  $\mathsf{HT}_2[\tilde{x}, r, L, m] \leftarrow \mathsf{ch}^{\mathcal{B}}$
  $r_i \rightarrowtriangle P_k, \bar{r}_k \leftarrowtriangle P_k, \forall k \neq i$
  if $\exists j : \bar{r}_j \neq r_j$ then
      return $\perp$
  for $j \leftarrow 1, \dots t(n)$ do
      if $\mathsf{ch}_j = i$ then
          $\tilde{g}_j^{(k)} \leftarrowtriangle P_k, \forall k \neq i$
          $z_j \leftarrow (\prod_{k=0}^{n-(i+1)} \tilde{g}_j^{(n-k)}) z_j^{\mathcal{B}}$
          $z_j \rightarrowtriangle P_k, \forall k \neq i$
      else if $\mathsf{ch}_j = 0 \wedge i = n$ then
          $\tilde{g}_j^{(k)} \leftarrowtriangle P_k, \forall k \neq i$
          $z_j \leftarrow (\prod_{k=0}^{n-1} \tilde{g}_j^{(n-k)})$
          $z_j \rightarrowtriangle P_k, \forall k \neq i$
      else
          $\tilde{g}_j^{(i)} \rightarrowtriangle P_{\mathsf{ch}_j}$
```

Game_0 This is the initial strong MS-UF-CMA game against the MS-GA[$\star$] scheme, except that it uses programmable random oracles. At the start of the game, the challenger initializes three tables, $\mathsf{HT}_0, \mathsf{HT}_1, \mathsf{HT}_2$ for $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_2$, respectively. When a query Q for H_0 is received, if $\mathsf{HT}_0[Q] = \perp$ it uniformly samples $\mathsf{com} \leftarrow_\$ \{0,1\}^{2\lambda}$ and stores $\mathsf{HT}_0[Q] \leftarrow \mathsf{com}$, finally it returns $\mathsf{HT}_0[Q]$ (similarly for H_1 and H_2). It follows that $\Pr[\text{Game}_0(\mathcal{A}) = 1] = \mathsf{Adv}^{\text{MS-UF-CMA}}_{\text{MS-GA}[\star]}(\mathcal{A})$.

Game_1 This game is identical to Game_0, except that OMuSign aborts by raising $\mathsf{bad}_{\mathsf{hcol}}$ when the following happens: being $\tilde{x}$ the commitment and $r \leftarrow \mathsf{H}_1(r_1, \dots, r_n)$ the salt generated by $\mathcal{A}$ and OMuSign during the sign query $(\mathsf{sid}, (m, L))$, the challenger aborts the game if the random oracle H_2 was already queried at input $Q = (\tilde{x}, r, L, m)$, i.e. $\mathsf{HT}_2[Q] \neq \perp$. Otherwise, OMuSign samples $\mathsf{ch} \leftarrow_\$ [0, n]^{t(n)}$ and programs $\mathsf{HT}_2[Q] \leftarrow \mathsf{ch}$. It follows that $|\Pr[\text{Game}_0(\mathcal{A}) = 1] - \Pr[\text{Game}_1(\mathcal{A}) = 1]| \leq \Pr[\mathsf{bad}_{\mathsf{hcol}}]$.

Game_2 This game is identical to Game_1, except that OMuSign and H_2 are simulated as follows. At the start of the game, the challenger initializes an empty set $\mathcal{M}$, that will be used to track the messages queried by the simulator to OSign, and a look-up table MT used to map the queries to H_2 to the messages included in the queries to H'. In particular, when OMuSign receives a query, it samples a random message $m' \in \mathsf{M}$ and adds it to $\mathcal{M}$ before sending a sign query to OSign for m'. When H_2 receives a query $Q = (\tilde{x}, r, L, m)$ such that $\mathsf{pk}^\star \in L$, it samples a random message $m' \in \mathsf{M}$ and sets $\mathsf{MT}[Q] \leftarrow m'$ before querying H' on $(L', \tilde{x}, m')$. After the adversary outputs a valid signature $\sigma = (\mathsf{ch}, r, z_1, \dots, z_{t(n)})$ on message m with users public keys $L = (x_1, \dots, x_n)$, the challenger derives $\tilde{x}$ as in the execution of MuVrfy, and retrieves $m' \leftarrow \mathsf{MT}[\tilde{x}, r, L, m]$. If $m' \in \mathcal{M}$, the game aborts by raising $\mathsf{bad}_{\mathsf{mcol}}$. It follows that $|\Pr[\text{Game}_1(\mathcal{A}) = 1] - \Pr[\text{Game}_2(\mathcal{A}) = 1]| \leq \Pr[\mathsf{bad}_{\mathsf{mcol}}]$.

The reason why in Game_2, for the sign query to OMuSign and hash query to H_2, the simulator $\mathcal{B}$ is instructed to sample a random message m' resides in the definition of forgery in the multi-signature scheme and in the centralized signature scheme. In particular, in the multi-signature game $\mathcal{A}$ can produce a forgery on a message m signed on behalf of the public keys in L even if during the training with the oracle OMuSign it previously queried a signature for the same m but with a different set of signers L'. This does not hold for the centralized signature, where the forgery must be associated to a message that has never been queried to H'.

We now show that the EUF-CMA adversary $\mathcal{B}$ can simulate Game_2 as described in Algorithm 5.6. At the start of the game, $\mathcal{B}$ initializes an empty set $\mathcal{M} \leftarrow \emptyset$ that will store the queries to OSign. In the following, given a vector x, we will denote with $\pi_{i,j}(x)$ the permutation of elements with index i and j in x.

Random oracles queries. When a query Q_0 for H_0 is received, if $\mathsf{HT}_0[Q_0] = \perp$, $\mathcal{B}$ uniformly samples $\mathsf{com} \leftarrow_{\$} \{0,1\}^{2\lambda}$, stores $\mathsf{HT}_0[Q_0] \leftarrow \mathsf{com}$, and returns $\mathsf{HT}_0[Q_0]$. Similarly, when a query Q_1 for H_1 is received if $\mathsf{HT}_1[Q_1] = \perp$, $\mathcal{B}$ uniformly samples $\mathsf{com} \leftarrow_{\$} \{0,1\}^{2\lambda}$, stores $\mathsf{HT}_1[Q_1] \leftarrow \mathsf{com}$, and returns $\mathsf{HT}_1[Q_1]$. Instead H_2, the random oracle employed to generate the challenge ch, is simulated as follows. Suppose a query $Q_2 = (\tilde{x}, r, L, m)$ for H_2 is received and $\mathsf{HT}_2[Q_2] = \perp$. Let $n = |L|$, if $\mathsf{pk}^\star \notin L$, i.e. the random oracle query do not refer to the public key that the forger must impersonate, $\mathcal{B}$ uniformly samples $\mathsf{ch} \leftarrow_{\$} [0, n]^{t(n)}$ and returns it. Otherwise, suppose $L = (x_1, \dots, x_n)$ such that $x_i = \mathsf{pk}^\star$ and let L' be the set of public keys in L after permuting the order of x_1 and x_i and subsequently removing x_i, i.e. $L' \leftarrow \pi_{1,i}(L)[2{:}]$. This way, L' can be used as a set of ephemeral keys for the centralized signature scheme. Then, $\mathcal{B}$ samples a uniformly random message m' and queries $(L', \tilde{x}, m')$ to $\mathsf{H}'^{(n)}$, obtaining $\mathsf{ch}^{\mathcal{B}}$, which acts as the challenge of the centralized signature scheme. Next, it computes ch as the permutation $\pi_{1,i}(\mathsf{ch}^{\mathcal{B}})$, making it compatible with the public keys in L), and stores $\mathsf{MT}[Q_2] \leftarrow m'$ and $\mathsf{HT}_2[Q_2] \leftarrow \mathsf{ch}$. Finally, it returns $\mathsf{HT}_2[Q_2]$.

Signing queries. On a new query $Q = (\mathsf{sid}, (m, L))$, $\mathcal{B}$ runs MuSign up to Line 9. Suppose $L = (x_1, \dots, x_n)$ such that $x_i = \mathsf{pk}^\star$. Then, it samples a uniformly random message $m' \leftarrow_{\$} \mathsf{M}$, adds m' to $\mathcal{M}$, and queries OSign on m'. The signing oracle response is $(\mathsf{com}^{\mathcal{B}}, z_1^{\mathcal{B}}, \dots, z_{t(n)}^{\mathcal{B}})$, with $\mathsf{com}^{\mathcal{B}} = (\hat{x}_2^{\mathcal{B}}, \dots, \hat{x}_n^{\mathcal{B}}, \tilde{x}_1^{\mathcal{B}}, \dots, \tilde{x}_{t(n)}^{\mathcal{B}})$. $\mathcal{B}$ will only use responses $z_j^{\mathcal{B}}$ from OSign that link x_i to $\tilde{x}_j^{\mathcal{B}}$, which correspond to the values 1 of the challenges sampled by the oracle. Hence, $\mathcal{B}$ computes the permutation of the challenge of the centralized signature, obtaining $\mathsf{ch}^{\mathcal{B}} \leftarrow \pi_{1,i}(\mathsf{H}'^{(n)}(\mathsf{com}^{\mathcal{B}}, m'))$, that will be used to program the answer of H_2. Then, MuSign is simulated up to Line 13 as follows: for each $j \leftarrow 1, \dots, t(n)$ it receives $\tilde{x}_j^{(i-1)}$. Then, if $\mathsf{ch}_j^{\mathcal{B}} \neq i$, it samples $\tilde{g}_j^{(i)} \leftarrow_{\$} G$ and sets $\tilde{x}_j^{(i)} \leftarrow \tilde{g}_j^{(i)} \star \tilde{x}_j^{(i-1)}$. Otherwise, if $\mathsf{ch}_j^{\mathcal{B}} = i$, it sets $\tilde{x}_j^{(i)} \leftarrow \tilde{x}_j^{\mathcal{B}}$, since it knows the group element mapping the public key x_i to $\tilde{x}_j^{\mathcal{B}}$ from the centralized signature previously queried. Subsequently, before revealing r_i, $\mathcal{B}$ retrieves $r_1, \dots, r_n$ from HT_0. If some com_j's were not obtained after a query to H_0 for $(\mathsf{com}_{j-1}, r_j)$, it follows the execution of MuSign and returns $\perp$ on Line 16. Next, it computes $r \leftarrow \mathsf{H}_1(r_1, \dots, r_n)$ and programs $\mathsf{HT}_2[\tilde{x}, r, L, m] \leftarrow \mathsf{ch}^{\mathcal{B}}$. Finally, the remainder of MuSign's execution is simulated as follows: for each $j \leftarrow 1, \dots, t(n)$, if $\mathsf{ch}_j \neq i$, $\mathcal{B}$ reveals $\tilde{g}_j^{(i)}$. Otherwise, it receives $\tilde{g}_j^{(k)}$ for $k \neq i$ and computes $z_j \leftarrow \tilde{g}_j^{(n)} \cdot \ldots \cdot \tilde{g}_j^{(i+1)} z_j^{\mathcal{B}}$.

Eventually, $\mathcal{A}$ will output a valid signature $\sigma = (\mathsf{ch}, r, z_1, \dots, z_{t(n)})$ for a message m under public keys $L = (x_1, \dots, x_n)$. If $\mathcal{A}$ is winning the MS-UF-CMA game, then there exists an index $i \in [n]$ such that $\mathsf{pk}^\star = x_i$ and $(m, L) \notin \mathcal{Q}$. $\mathcal{B}$ can run MuVrfy up to Line 5 to recover $\tilde{x} = (\tilde{x}_1, \dots, \tilde{x}_{t(n)})$. From our simplifying assumption on $\mathcal{A}$, σ is valid and $Q = (\tilde{x}, r, L, m)$ must have been queried to H_2. Then, $\mathcal{B}$ assigns to L' the set of public keys in L after permuting the order of x_1 and x_i and subsequently removing x_i, i.e. $L' = \pi_{1,i}(L)[2{:}]$. Next, it retrieves $m' \leftarrow \mathsf{MT}[\tilde{x}, r, L, m]$ and aborts by raising $\mathsf{bad}_{\mathsf{mcol}}$ if $m' \in \mathcal{M}$. Otherwise, $\mathcal{B}$ wins the EUF-CMA game, returning the signature $(L', \tilde{x}, z_1, \dots, z_{t(n)})$ on message m'. In fact, ch was obtained in the simulation of H_2 as $\mathsf{ch} = \pi_{1,i}(\mathsf{ch}')$, where $\mathsf{ch}' = \mathsf{H}'^{(n)}(L', \tilde{x}, m')$. Let $\hat{x}_k \leftarrow x_k$ for all $k \in [n], k \neq 1, i$ and let $\hat{x}_i \leftarrow x_1, \hat{x}_1 \leftarrow x_i$. For any $j \leftarrow 1, \dots, t(n)$, it follows that:

$$z_j \star \hat{x}_{\mathsf{ch}'_j} = z_j \star x_{\mathsf{ch}_j} = \tilde{x}_j.$$

If none of the bad events happen, $\mathcal{B}$ perfectly simulate Game_2, and we obtain

$$\begin{aligned}\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{FS}[\Pi']}(\mathcal{B}) &= \Pr[\mathsf{Game}_2(\mathcal{A}) = 1] \\ &\geq \mathsf{Adv}^{\mathsf{MS\text{-}UF\text{-}CMA}}_{\mathsf{MS\text{-}GA}[\star]}(\mathcal{A}) - \Pr[\mathsf{bad}_{\mathsf{hcol}}] - \Pr[\mathsf{bad}_{\mathsf{mcol}}].\end{aligned}$$

$\mathcal{B}$ can simulate Game_2 with at most the same running time of $\mathcal{A}$ plus the time required for running MuVrfy.

In the following, we bound the probability of each bad event happening.

Probability of $\mathsf{bad}_{\mathsf{hcol}}$ The event $\mathsf{bad}_{\mathsf{hcol}}$ occurs on Line 25 of $\mathsf{OMuSign}$ on input $(\mathsf{sid}, (m, L))$ when, after obtaining the commitment $\tilde{x}$ and the salt $r \leftarrow \mathsf{H}_1(r_1, \dots, r_n)$, a value for $Q = (\tilde{x}, r, L, m)$ was already assigned in HT_2. The table HT_2 is populated by either $\mathsf{OMuSign}$ or H_2, so that its entries are at most $\mathsf{q_S} + \mathsf{q_H}$. The salt r is obtained from H_1 on inputs $r_1, \dots, r_n$, where $r_k, k \neq i$ is provided by $\mathcal{A}$ and r_i is sampled uniformly random from $\{0,1\}^{\ell_{\mathsf{salt}}}$. The probability that a uniformly random r_i produces a collision with one of the entries is then at most $(\mathsf{q_S} + \mathsf{q_H})2^{-\ell_{\mathsf{salt}}}$. Since at most $\mathsf{q_S}$ are made to $\mathsf{OMuSign}$, then $\Pr[\mathsf{bad}_{\mathsf{hcol}}] \leq \mathsf{q_S}(\mathsf{q_S} + \mathsf{q_H})2^{-\ell_{\mathsf{salt}}}$.

Probability of $\mathsf{bad}_{\mathsf{mcol}}$ The event $\mathsf{bad}_{\mathsf{mcol}}$ occurs on Line 9 of the simulation of $\mathcal{B}$ when, after the adversary $\mathcal{A}$ outputs a valid signature $\sigma = (\mathsf{ch}, r, z_1, \dots, z_{t(n)})$ on message m with users public keys $L = (x_1, \dots, x_n)$, $\mathcal{B}$ derives $\tilde{x}$ as in the execution of MuVrfy, and retrieves $m' \leftarrow \mathsf{MT}[\tilde{x}, r, L, m]$, the message sampled during a random oracle query to H_2, such that $m' \in \mathcal{M}$. There are two possibilities that can cause the $\mathsf{bad}_{\mathsf{mcol}}$ event: either in H_2 (Line 8) if it samples $m' \in \mathsf{M}$ that is already in $\mathcal{M}$, or in $\mathsf{OMuSign}$ (Line 6) if it samples $m' \in \mathsf{M}$ such that it is already a value in MT. The set $\mathcal{M}$ is populated by $\mathsf{OMuSign}$, so that its entries are at most $\mathsf{q_S}$. Since $|\mathsf{M}| = 2^{\ell_{\mathsf{M}}}$, the probability that a uniformly random $m' \in \mathsf{M}$ produces a collision with one of the entries of $\mathcal{M}$ is then at most $\mathsf{q_S}/|\mathsf{M}| \leq \mathsf{q_S}/2^{\ell_{\mathsf{M}}}$. Since at most $\mathsf{q_H}$ queries are made to H_2, the probability of the first occurrence is at most $\mathsf{q_H}\mathsf{q_S}/2^{\ell_{\mathsf{M}}}$. Similarly, the table MT is populated by H_2 with at most $\mathsf{q_H}$ entries, and the probability that a uniformly random $m' \in \mathsf{M}$ produces a collision with one of the entries of MT is at most $\mathsf{q_H}/2^{\ell_\mathsf{M}}$. Since at most $\mathsf{q_S}$ queries are made to OMuSign, the probability of the second occurrence is at most $\mathsf{q_H}\mathsf{q_S}/2^{\ell_\mathsf{M}}$. Therefore, we obtain $\Pr[\mathsf{bad}_{\mathsf{mcol}}] \leq 2\mathsf{q_S}\mathsf{q_H}/2^{\ell_\mathsf{M}}$.

Combining the previous bound on bad events, we obtain the claimed estimate of $\mathsf{Adv}^{\mathsf{MS\text{-}UF\text{-}CMA}}_{\mathsf{MS\text{-}GA}[\star]}(\mathcal{A})$. □

Concurrent executions Note that the security proof reduces the security of the multi-signature to the security of a centralized signature, which is concurrently secure since it does not require interactions between parties. Also, the simulation does not require $\mathcal{B}$ to rewind the adversary during the execution of the training phase, when the adversary queries the sign oracle and builds signatures of chosen messages with its support. The rewinding is executed only once when the adversary of the multi-signature produces its forgery. This means that the execution time of the simulator $\mathcal{B}$ is polynomial in the execution time of the adversary, which is polynomial in lambda.

Moreover, since the signing parties commit to the r_i, for all i in the signing set, and these determine the value of r which is used to compute the challenge ch, the protocol is resistant to the practical attacks that can occur in the concurrent setting [25].

Identifiable abort Our protocol can be adapted to allow the signers to identify when a party misbehaves. If during the construction of the commitment, each party broadcasts their partial commitment, then in an eventual failure of the signature protocol the honest party will always be able to identify at least one malicious party for each signing protocol execution. In fact, when the commitments are opened, one can check that the group element $\tilde{g}_i$ of the parties P_i which are not selected by the challenge actually maps the previous partial commitment $\tilde{x}_{i-1}$ to $\tilde{x}_i$.

5.2.5 Signature Optimizations

In this section we discuss the applicability to the multi-signature scheme of the standard optimizations discussed for group action-based Σ-protocol and centralized signature scheme in Section 4.3. Note that when defining the multi-signature protocol, it is necessary to ensure that the centralized signature keys are compatible with the interactive protocol. Therefore, all optimizations that do not intervene directly on the keys are potentially applicable.

As in the case of aggregation, one of the main efficiency measures for the multi-signature scheme is the *compression rate*, i.e., the reduction in the length of the signature aggregation of n users compared to the trivial concatenation of

n individual signatures. Let Σ_n be the multi-signature of n users and let σ be the individual signature of the centralized scheme. The compression rate of n signatures is defined as $\tau(n) = 1 - \frac{|\Sigma_n|}{n \cdot |\sigma|}$. In order to optimize the compression rate, we reduce the size of Σ_n without affecting the security of the scheme.

Consider a group action $(G, X, \star)$ and a security parameter λ. In the non-optimized version of the multi-signature, the signature produced by n users is given by $\Sigma_n = (\mathsf{ch}, r, z_1, \ldots, z_{t(n)})$. The expected sizes (in bits) of the elements of Σ_n are expressed by

$$|\mathsf{ch}| = \log_2(n+1)t(n), \quad |r| = 2\lambda, \quad |z| = t(n)\ell_{\mathrm{G}},$$

where ℓ_{G} is the bit size of elements in G.

Compression of Random Elements

The compression of random elements for the base Σ-protocol Π is described in Section 4.3.1. Recall that in Π, about half of the responses are expected to be random elements of the group, which can be compressed using short seeds.

When we consider the multi-signature described in Algorithm 5.5, things get more complicated because the signers $P_1, \ldots, P_n$ build the responses $z_i, i \in [t(n)]$, in a round-robin fashion by multiplying the group elements that they have generated as described in Equation (5.1). Therefore, when the challenge $\mathsf{ch}_i = 0$, no contribution is required from the secret keys of the participants, and the response to the i-th repetition of the sigma protocol can be encoded in two possible ways:

- with the full list of seeds $(s_1, \ldots, s_n)$ corresponding to the group element of each party, which are multiplied to retrieve the response z_i;
- with the direct encoding of the group element z_i.

This means that the use of seeds is convenient as long as the representation of an element in G is heavier than the concatenation of n seeds, i.e. $n\lambda < \ell_{\mathrm{G}}$, and if the expected number of zeros in a $t(n)$-bit long challenge is $\frac{t(n)}{n+1}$, then the expected number of bits saved by using this technique is $\frac{t(n)(\ell_{\mathrm{G}} - n\lambda)}{n+1}$.

Similarly, the challenge ch can be expanded from a digest $d \in \{0,1\}^{2\lambda}$ obtained from H_2 in Algorithm 5.5.

Applying the aforementioned optimizations, the expected sizes (in bits) of the challenge and the response array are approximated by

$$|\mathsf{ch}| = 2\lambda, \quad |r| = 2\lambda, \quad |z| = \frac{t(n)}{n+1}(n\ell_{\mathrm{G}} + \ell_{\mathrm{seeds}}), \quad \ell_{\mathrm{seeds}} = \min\{n\lambda, \ell_{\mathrm{G}}\}$$

where the terms of $|z|$ correspond to the size of the responses to non-zero and zero challenges, respectively.

Seed Trees

The use of seed tree for the base Σ-protocol is described in Section 4.3.2. After building a binary tree to represent t seeds, recall that, to communicate the value of all the t seeds except for those indexed by a subset of $\{1, \dots, t\}$ of size ω, it is enough to send the values of the following number of nodes:

$$2^{\lceil \log_2(\omega) \rceil} + \omega(\lceil \log_2(t) \rceil - \lceil \log_2(\omega) \rceil - 1).$$

It follows that the communication cost of using a seed tree is advantageous when there are at least $\frac{t}{2}$ zero challenges. In the multi-signature, the expected number of zero challenges is $\frac{t(n)}{n+1}$. Therefore, the use of a seed tree is already ineffective for $n = 2$ users.

Fixed-Weight Challenges

The use of fixed-weight challenges for the base Σ-protocol is described in Section 4.3.3. In the proposed multi-signature, the use of fixed-weight challenges can still be useful to decrease the cheating probability of the adversary. In fact, the best strategy for an adversary is to control all parties except the target user P_i and to have a challenge with few components $\mathsf{ch}_j = i$. To make this possibility negligible, a large number of parallel repetitions $t(n)$ must be chosen, making the signature inefficient. As a countermeasure, we can consider challenges where each value $i \in [1, n]$ appears the same number of times. More in detail, for each $n \in \mathbb{N}$, we choose t, ω such that the challenges are elements of $[0, n]^t$ with exactly ω components equal to i, for each $i \in [1, n]$, and the remaining $t - n\omega$ components are equal to 0. Let $\mathsf{Ch}_n^{t,\omega}$ denote the challenge set mentioned above. The number of challenges in $\mathsf{Ch}_n^{t,\omega}$ is

$$\frac{t!}{(t-n\omega)!(\omega!)^n}.$$

Once a commitment is fixed, let $\eta_{t,\omega}$ be the maximum number of challenges in $\mathsf{Ch}_n^{t,\omega}$ an adversary can answer to without knowing the private key (i.e. from the responses to such challenges it would not be possible to extract the witness). Then t, ω must be chosen such that

$$\frac{\eta_{t,\omega}}{|\mathsf{Ch}_n^{t,\omega}|} \leq 2^{-\lambda}. \tag{5.2}$$

Lemma 5.9. *Given $n \in \mathbb{N}$, the value $\eta_{t,\omega}$ can be expressed as*

$$\max_{0 \leq k \leq n-1} \frac{(t-(n-k)\omega)!}{(t-n\omega)!(\omega!)^k} \frac{((n-k)\omega)!}{(\omega!)^{n-k}}. \tag{5.3}$$

Proof. Suppose w.l.o.g. that the target user is P_1, so that, without knowing their private key, an adversary cannot answer two challenges with 0 and 1 in the same component. In the following, let $\mathrm{Ch}_n = \{0, \dots, n\}$ and let Ch_n^t be the set of challenge strings of length t. For a subset $C \subset \mathrm{Ch}_n^t$, let $\mathcal{H}_C$ be the undirected graph whose vertices are the elements of Ch_n and in which, for any $x, y \in \mathrm{Ch}_n$, there is a link between x and y in $\mathcal{H}_C$ if and only if there exist two challenge strings $\mathrm{ch}, \mathrm{ch}' \in C$ such that $\mathrm{ch}_i = x$ and $\mathrm{ch}'_i = y$ for some index $1 \leq i \leq t$.

Let $\mathcal{C}$ be the set of all subsets C of $\mathrm{Ch}_n^{t,\omega}$ such that 0 and 1 are not connected in $\mathcal{H}_C$. It follows that $\eta_{t,\omega}$ is the maximum cardinality among the sets in $\mathcal{C}$. Given a set $C \in \mathcal{C}$, let k be the number of challenges $\alpha_1, \dots, \alpha_k \in \mathrm{Ch}_n \setminus \{1\}$ for which there is a path between 0 and α_i in $\mathcal{H}_C$. The remaining $n - k - 1$ challenges $\beta_1, \dots, \beta_{n-k-1} \in \mathrm{Ch}_n \setminus \{0,1\}$ can either be all connected to 1 or form smaller connected components. For any $\mathrm{ch} \in \mathrm{Ch}_n^{t,\omega}$, there are exactly ω components of ch equal to α_i or β_j, and $t - n\omega$ components equal to 0. Therefore, in C we can have at most $\frac{(t-(n-k)\omega)!}{(t-n\omega)!(\omega!)^k}$ choices for the entries that have a path to 0. The remaining $(n-k)\omega$ entries, can have at most $\frac{((n-k)\omega)!}{(\omega!)^{n-k}}$ choices when $\beta_1, \dots, \beta_{n-k-1}$ are all connected to 1. Therefore, the maximal size of a set $C \in \mathcal{C}$ is

$$\max_{0 \leq k \leq n-1} \frac{(t-(n-k)\omega)!}{(t-n\omega)!(\omega!)^k} \frac{((n-k)\omega)!}{(\omega!)^{n-k}}.$$ □

In the following, we choose $t = (n+1)\omega$, so that each value in $\{0, \dots, n\}$ appears exactly ω times.

Lemma 5.10. *Given $n \in \mathbb{N}$, let $\eta_\omega = \eta_{(n+1)\omega,\omega}$. Then*

$$\eta_\omega = \frac{(n\omega)!}{(\omega!)^n}.$$

Proof. Substituting $t = (n+1)\omega$ in Equation (5.3), we obtain

$$\eta_\omega = \eta_{(n+1)\omega,\omega} = \max_{0 \leq k \leq n-1} \frac{((k+1)\omega)!}{(\omega!)^{k+1}} \frac{((n-k)\omega)!}{(\omega!)^{n-k}}.$$

Consider the discrete function $f(k)$ taking values in $\{0, \dots, n-1\}$, defined by

$$f(k) = \frac{((k+1)\omega)!}{(\omega!)^{k+1}} \frac{((n-k)\omega)!}{(\omega!)^{n-k}}.$$

Notice that $f(k) = f(n-1-k)$, it is then sufficient to prove that f is decreasing for $k \leq \lfloor (n-1)/2 \rfloor$. In fact, for any k, it holds that

$$\frac{f(k)}{f(k+1)} = \frac{((k+1)\omega)!}{((k+2)\omega)!} \frac{((n-k)\omega)!}{((n-k-1)\omega)!} = \prod_{i=1}^{\omega-1} \frac{(n-k)\omega - i}{(k+2)\omega - i}.$$

Notice that for any term in the product, it holds that

$$(n-k)\omega - i \geq (k+2)\omega - i \iff k \leq \left\lfloor \frac{n-2}{2} \right\rfloor .$$

Therefore,

$$\eta_\omega = f(0) = f(n-1) = \frac{(n\omega)!}{(\omega!)^n}. \qquad \square$$

If we substitute the value of η_ω from previous lemma in Equation (5.2), then the choice of ω should be made such that

$$2^{-\lambda} \geq \frac{\eta_\omega}{|\mathsf{Ch}_n^{(n+1)\omega,\omega}|} = \binom{(n+1)\omega}{\omega}^{-1}.$$

The choice of ω is made with the aim of minimizing the size of the response array z, where

$$|z| = n\omega\ell_{\mathrm{G}} + \omega\ell_{\mathsf{seeds}}. \tag{5.4}$$

In Section 5.3 we provide a concrete analysis of the optimal values for ω for the selected signature schemes.

Multiple Public Keys

The use of multiple public keys for the base Σ-protocol is described in Section 4.3.4. In the centralized scheme, each user generates s public keys, and the challenge space is extended so that a challenge selects one of the keys. The response is then generated using the relevant private key, exhibiting a group element that maps the selected public key to the commitment. The challenge space of the single instance is then extended from a binary space to one of $s+1$ elements, thereby reducing the soundness error.

Using this optimization in the multi-signature requires minor modifications to the MS protocol described in Algorithm 5.5. In fact, the protocol already allows multi-bit challenges to select a specific user's key. It is, therefore, sufficient to extend the challenge space so that one of the user's keys can be selected. Notice, however, that this optimization modifies the public keys of the underlying signature, and is therefore applicable only if the signature scheme provides for it. The changes described below will then be used in Section 5.3 for the parameterization of signatures using multiple public keys.

In the following, we combine the use of multiple public keys with the unbalanced challenge optimization of the previous section, evaluating its impact on soundness error and signature size. Concretely, suppose each user P_i has s public keys $x_i^{(0)}, \ldots, x_i^{(s-1)} \in X$, where $x_i^{(0)} = x_0$. For a fixed $n \in \mathbb{N}$, let $\mathsf{Ch}_{n,s} = \{0, \ldots, n(s-1)\}$ be the challenge space of the single instance, where 0 identifies x_0

and $k = (i-1)(s-1) + j$ identifies $x_i^{(j)}$ of user P_i, with $1 \leq i \leq n$ and $1 \leq j \leq s-1$. Similarly to the single key case, we choose t, ω such that the challenges are elements of $\mathsf{Ch}_{n,s}^{t}$ with exactly ω components corresponding to the i-th user, and the remaining $t - n\omega$ components are equal to 0. Let $\mathsf{Ch}_{n,s}^{t,\omega}$ denote the challenge set described above. The number of challenges in $\mathsf{Ch}_{n,s}^{t,\omega}$ is

$$\frac{t!}{(t-n\omega)!(\omega!)^n}(s-1)^{n\omega}.$$

Let $\eta_{t,\omega}$ be the maximum number of challenges in $\mathsf{Ch}_{n,s}^{t,\omega}$ an adversary can answer to without knowledge of the private key. Then t, ω must be chosen such that

$$\frac{\eta_{t,\omega}}{|\mathsf{Ch}_{n,s}^{t,\omega}|} \leq 2^{-\lambda}. \tag{5.5}$$

As in the case of the single key, we simplify by choosing $t = (n+1)\omega$. By adapting *Lemma* 5.9 and *Lemma* 5.10, we obtain that

$$\eta_\omega = \max_{k_1+\ldots+k_s=n-1} \prod_{i=1}^{s} \frac{((k_i+1)\omega)!}{(\omega!)^{k_i+1}}(s-1)^{k_i\omega} = \frac{(n\omega)!}{(\omega!)^n}(s-1)^{(n-1)\omega}.$$

If we substitute the value of η_ω in Equation (5.5), then the choice of ω should be made such that

$$2^{-\lambda} \geq \frac{\eta_\omega}{|\mathsf{Ch}_{n,s}^{(n+1)\omega,\omega}|} = \binom{(n+1)\omega}{\omega}^{-1}(s-1)^{-\omega}. \tag{5.6}$$

With respect to Equation (5.4), the expression for the size of the response size remains unchanged. On the other hand, as s increases, we can choose a smaller ω in order to obtain a more compact signature.

5.3 Instantiation and Evaluation

In this section, we will provide concrete applications of the multi-signature scheme described in Section 5.2.3 to the digital signature schemes based on group actions described in Section 4.4, namely LESS, MEDS, and ALTEQ. We evaluate the efficiency of the scheme by measuring the compression rate, as defined in Section 5.2.5. We will not provide a similar analysis of the aggregated sequential signature proposed in Section 5.1 since, as previously discussed, challenge aggregation achieves a compression rate of less than 1%.

Consider a group action $(G, X, \star)$ and a security parameter λ. Using the optimizations described in the previous section, in Table 5.1 we summarize the bit length of N centralized signatures and the multi-signature of N users[3] associated with the group action. In more detail, we assume that for both centralized and multi-signature, we have the same size for random salts $\ell_{\text{salt}} = 2\lambda$, outputs of the random oracle $\ell_{\text{digest}} = 2\lambda$, seeds for random elements λ, and group elements ℓ_{G}. For the centralized signature, the number of repetitions t and fixed-weight parameter of the challenges ω are chosen according to Equation (4.2), and are reported in the parameter sets of each scheme. In the multi-signature, the fixed-weight parameter ω' is chosen according to Equation (5.6) and depends on the number of signers. Notice that, as discussed in Section 5.2.5, only the centralized signature can exploit the use of seed trees. It follows that the length of N concatenated signatures is given by:

$$\begin{aligned} N \cdot |\sigma| &= N \cdot (\ell_{\text{salt}} + \ell_{\text{digest}} + \omega\ell_{\mathrm{G}} + \ell_{\text{seeds}}) \\ &= N \cdot \left(4\lambda + \omega\ell_{\mathrm{G}} + (2^{\lceil \log_2(\omega)\rceil} + \omega(\lceil \log_2(t)\rceil - \lceil \log_2(\omega)\rceil - 1)) \cdot \ell_{\text{tree\_seed}}\right). \end{aligned}$$

On the other hand, compression of random elements can also be exploited in the case of multi-signatures if the number of signers N is less than $\lfloor \ell_{\mathrm{G}}/\lambda \rfloor$. It follows that the length of a multi-signature of N users is given by:

$$|\Sigma_N| = \ell_{\text{salt}} + \ell_{\text{digest}} + N\omega'\ell_{\mathrm{G}} + \ell_{\text{seeds}} = 4\lambda + N\omega'\ell_{\mathrm{G}} + \min\{\omega'\ell_{\mathrm{G}}, N\omega'\lambda\}.$$

We observe that in both cases, the length of the signatures is dominated by $N\omega\ell_{\mathrm{G}}$ (resp. $N\omega'\ell_{\mathrm{G}}$). The fixed-weight parameter ω for the centralized signature is fixed for each set of parameters, so that the term $N\omega\ell_{\mathrm{G}}$ grows linearly with N. On the other hand, the fixed-weight parameter ω' for the multi-signature depends on the number of signers, and is given by:

$$\omega' = \arg\min_{\omega} \left\{ \binom{(N+1)\omega}{\omega}^{-1} (s-1)^{-\omega} \leq 2^{-\lambda} \right\}.$$

We can use the following bound from [66, Theorem 11.1.3] to find a simplified expression of ω':

$$\binom{n}{k} \leq 2^{nH_b(k/n)},$$

where $H_b(n) = -n\log_2(n) - (1-n)\log_2(1-n)$ is the *binary entropy function*. Therefore, for any $N > 1$,

$$2^{(N+1)\omega' H_b(1/(N+1))}(s-1)^{\omega'} \geq \binom{(N+1)\omega'}{\omega'}(s-1)^{\omega'} \geq 2^{\lambda},$$

[3] In this section we use N to denote the number of signers as n is often used as an internal parameter for the underlying signature scheme.

Table 5.1: Data sizes (in bits) and choice of parameters comparison between single-signature and multi-signature schemes with N users.

	Centralized Signature	**Multi-signature**
ℓ_{salt}	2λ	
ℓ_{digest}	2λ	
$\ell_{\text{tree\_seed}}$	λ	
ℓ_{G}	Byte size of elements in G	
s	Number of public keys	
ω	$\min\left\{\omega \mid \binom{t}{\omega}^{-1}(s-1)^{-\omega} \leq 2^{-\lambda}\right\}$	$\min\left\{\omega \mid \binom{(N+1)\omega}{\omega}^{-1}(s-1)^{-\omega} \leq 2^{-\lambda}\right\}$
ℓ_{seeds}	$(2^{\lceil\log_2(\omega)\rceil} + \omega(\lceil\log_2(t)\rceil - \lceil\log_2(\omega)\rceil - 1)) \cdot \ell_{\text{tree\_seed}}$	$\min\{\omega\ell_{\text{G}}, N\omega\lambda\}$
ℓ_{sig}	$N \cdot (\ell_{\text{salt}} + \ell_{\text{digest}} + \omega\ell_{\text{G}} + \ell_{\text{seeds}})$	$\ell_{\text{salt}} + \ell_{\text{digest}} + N\omega\ell_{\text{G}} + \ell_{\text{seeds}}$

so that

$$\begin{aligned}(N+1)\omega' H_b(1/(N+1)) + \omega' \log_2(s-1) \geq \lambda \\ \implies \omega' = \left\lceil \frac{\lambda}{(N+1)H_b(1/(N+1)) + \log_2(s-1)} \right\rceil.\end{aligned}$$

It follows that a rough approximation for the compression rate $\tau(N)$ is given by

$$\tau(N) = 1 - \frac{|\Sigma_N|}{N \cdot |\sigma|} \approx 1 - \frac{N\omega'\ell_{\text{G}}}{N\omega\ell_{\text{G}}} = 1 - \frac{1}{\omega}\left\lceil \frac{\lambda}{(N+1)H_b(1/(N+1)) + \log_2(s-1)} \right\rceil.$$

Remark. The above expression provides an initial estimate of the effectiveness of using multi-signature. In particular, we will have higher compression as the number of signers increases, and higher maximum compression for parameter sets using a fixed-weight parameter that is not too low.

A second metric relevant to the analysis of multi-signature efficiency concerns the computational costs of producing a signature. In the case of group action-based signatures, the main parameter affecting the performance of the signing and verification process is the number t of protocol repetitions. In particular, the computational cost of producing a signature linearly increase with t. Similarly to the fixed-weight parameter, the number of repetitions for the centralized signature is fixed for each set of parameters, and when N signatures are concatenated, the single instance of the protocol is repeated $N \cdot t$ times distributed among N users. The number of repetitions for the multi-signature is determined by the number of signers and is given by $(N+1)\omega'$. Applying a similar analysis as above, the repetition rate, i.e., the rate of reduction in the number of iterations between centralized and multi-signature, is given by

$$1 - \frac{(N+1)\omega'}{N \cdot t} \approx 1 - \frac{\omega'}{t} = 1 - \frac{1}{t}\left\lceil \frac{\lambda}{(N+1)H_b(1/(N+1)) + \log_2(s-1)} \right\rceil.$$

Remark. The previous expression compares performance only in terms of iterations of the underlying protocol. In the case of an interactive multi-signature, however, it is also necessary to consider the cost associated with the communication rounds between the parties during the signing process. The analysis of this cost is beyond the scope of this work and is not further analysed.

5.3.1 LESS

We consider the signature scheme described in Section 4.4.2 and the parameters proposed in [13] with respect to NIST security levels I, III, and V as reported in Table 4.1.

Recall that the LESS group action is $\star_{\mathsf{LEP}}$, which is given by the action of the monomial group $\mathrm{Mon}(n,q)$ on the set X of $[n,k,q]$ linear codes represented by generator matrices in systematic form:

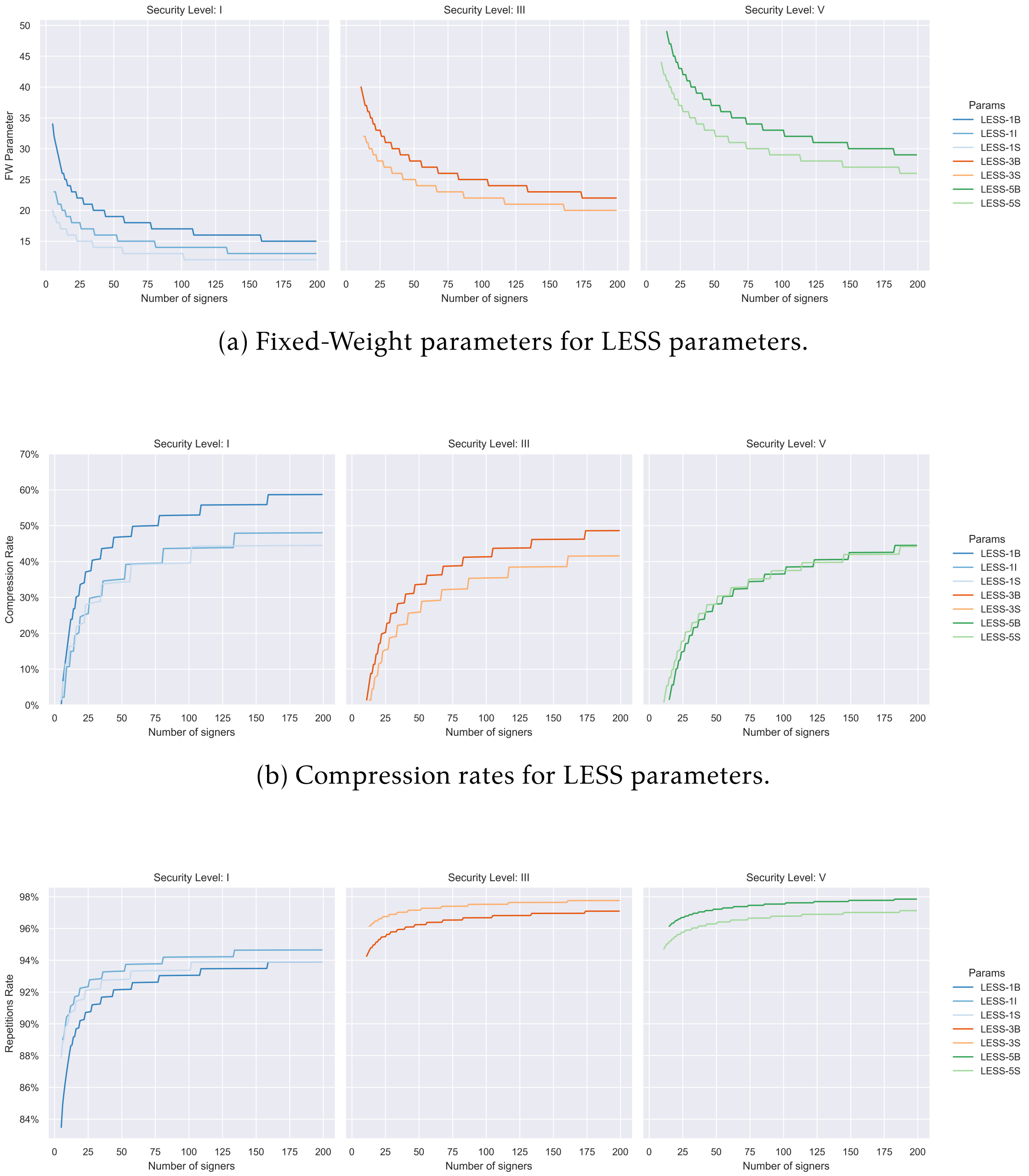
$$\star_{\mathsf{LEP}} \colon \mathrm{Mon}(n,q) \times X \to X, (\mathbf{Q}, \mathbf{G}) \mapsto \mathrm{SF}(\mathbf{GQ}).$$

The elements of the group $\mathrm{Mon}(n,q)$ are represented using the information set optimization with a string of length $\ell_{\mathrm{G}} = k(\lceil \log_2(n) \rceil + \lceil \log_2(q-1) \rceil)$. Since in LESS, the parameter k is approximately equal to the security parameter λ, the compression of random elements in the multi-signature can be applied when the number of users N is less than

$$\left\lfloor \frac{\ell_{\mathrm{G}}}{\lambda} \right\rfloor \approx \lceil \log_2(n) \rceil + \lceil \log_2(q-1) \rceil.$$

For each NIST security level I, III, and V, the LESS specification proposes three sets of parameters with a progressive trade-off between signature size and public key size, starting with the *balanced* set that does not use multiple keys, with a gradual increase in the *intermediate* and *short* sets. The increase in the number of public keys also corresponds to a lower fixed-weight parameter ω, leading to lower multi-signature compression for “shorter” parameters. The trend of the fixed-weight parameter ω' for the multi-signature as the number of users changes is shown in Figure 5.2a.

Figure 5.2b shows the compression rates as the number of users increases. The best aggregation rates are obtained with the balanced parameter of security level I, with a compression rate greater than 30% for $N > 15$ and greater than 50%



(a) Fixed-Weight parameters for LESS parameters.

(b) Compression rates for LESS parameters.

(c) Repetitions rates for LESS parameters.

Figure 5.2: Multi-signature rates and parameters for LESS.

for $N > 75$. Figure 5.2c similarly shows the repetitions rates upon varying the number of users. Here the best results are obtained for level III and V parameters with reductions in the number of repetitions exceeding 90% for any number of users.

5.3.2 MEDS

We consider the signature scheme described in Section 4.4.3 and the parameters proposed in [60] with respect to NIST security levels I, III, and V as reported in Table 4.2.

Recall that the MEDS group action is $\star_{\mathsf{MCE}}$, which is given by the action of $\mathrm{GL}_n(q) \times \mathrm{GL}_m(q)$ on the set X of $[m \times n, k, q]$ matrix codes:

$$\star_{\mathsf{MCE}} \colon (\mathrm{GL}_n(q) \times \mathrm{GL}_m(q)) \times X \to X, ((\mathbf{L}, \mathbf{S}), \mathfrak{C}) \mapsto \psi_{\mathbf{L},\mathbf{S}}(\mathfrak{C}).$$

The elements of the group $\mathrm{GL}_n(q) \times \mathrm{GL}_m(q)$ are represented with a string of length $\ell_{\mathrm{G}} = (n^2 + m^2)\lceil \log_2 q \rceil$.

For each NIST security level I, III, and V, the MEDS specification proposes two sets of parameters with a strong trade-off between signature size and scheme efficiency due to heavy use of the Fixed-Weight optimization. For the shorter parameterization, the high number of repetitions allows for an extremely low fixed-weight parameter ω, reducing the compression of the multi-signature. This is particularly observable in the short parameterization of security level I, where there is positive aggregation only from $N = 36$. The trend of the fixed-weight parameter ω' for the multi-signature as the number of users changes is shown in Figure 5.3a.

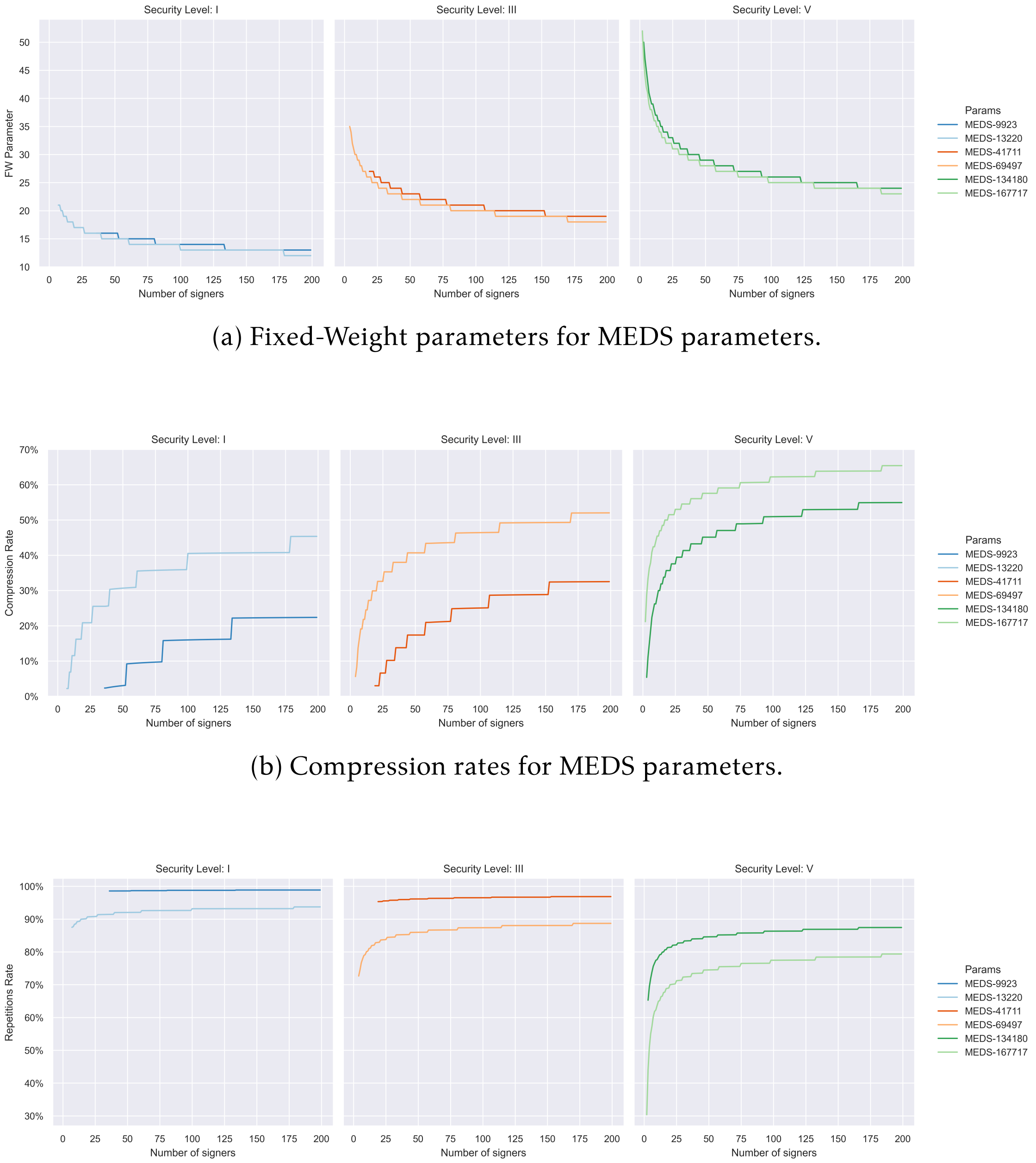
Figure 5.3b shows the compression rates as the number of users increases. The best aggregation rates are obtained with the balanced parameter of security level V, with a compression rate greater than 30% already for $N > 2$ and greater than 50% for $N > 19$. Figure 5.3c similarly shows the repetitions rates upon varying the number of users. Here the best results are obtained for level I and III balanced parameters with reductions in the number of repetitions exceeding 95% for any number of users.

5.3.3 ALTEQ

We consider the signature scheme described in Section 4.4.4 and the parameters proposed in [37] with respect to NIST security levels I and III as reported in Table 4.3.

Recall that the ALTEQ group action is $\star_{\mathsf{ATFE}}$, which is given by the action of $\mathrm{GL}_n(q)$ on the set of alternating trilinear forms $\mathrm{ATF}(n, q) = \{\phi \colon \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q\}$:

$$\star_{\mathsf{ATFE}} \colon \mathrm{GL}_n(q) \times \mathrm{ATF}(n, q) \to \mathrm{ATF}(n, q), (\mathbf{A}, \phi) \mapsto \phi \circ \mathbf{A}.$$

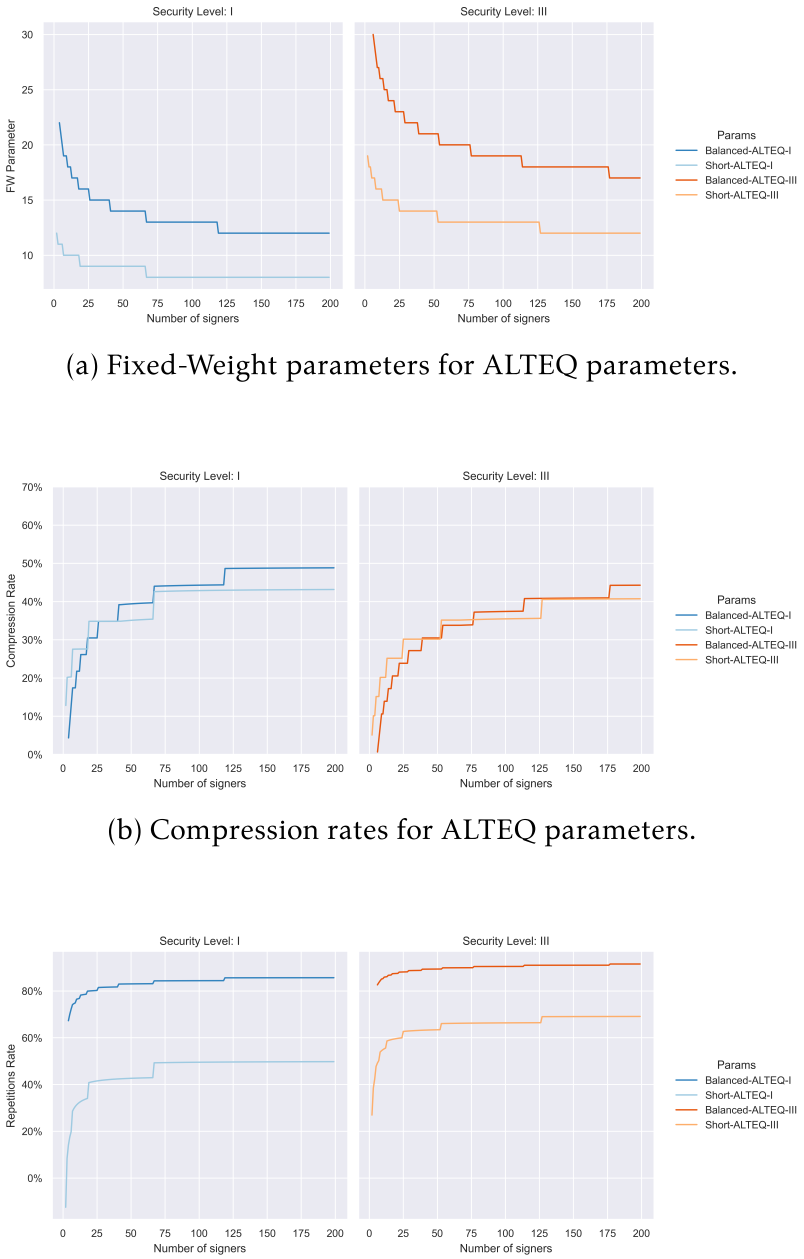


(a) Fixed-Weight parameters for MEDS parameters.

(b) Compression rates for MEDS parameters.

(c) Repetitions rates for MEDS parameters.

Figure 5.3: Multi-signature rates and parameters for MEDS.



(a) Fixed-Weight parameters for ALTEQ parameters.

(b) Compression rates for ALTEQ parameters.

(c) Repetitions rates for ALTEQ parameters.

Figure 5.4: Multi-signature rates and parameters for ALTEQ.

The elements of the group $\mathrm{GL}_n(q)$ are represented with a string of length $\ell_{\mathrm{G}} = n^2\lceil\log_2 q\rceil$.

For each NIST security level I and III the ALTEQ specification proposes two sets of parameters with a strong trade-off between signature size and public key size due to heavy use of the Multiple Public Keys optimization. The increase in the number of public keys also corresponds to a lower fixed-weight parameter ω, leading to slightly lower multi-signature compression for "shorter" parameters. On the other hand, the large number of public keys also benefits the multi-signature, with little reduction in compression compared to the MEDS case. The trend of the fixed-weight parameter ω' for the multi-signature as the number of users changes is shown in Figure 5.4a.

Figure 5.4b shows the compression rates as the number of users increases. Similar results are achieved by all parameterizations, with Level I sets having slightly higher aggregation for fewer users. Considering the balanced parameters of security level I, the compression rate is greater than 30% for $N > 17$ and greater than 50% for $N > 236$. Figure 5.4c similarly shows the repetitions rates upon varying the number of users. Here the best results are obtained for the balanced parameters of level III with reductions in the number of repetitions exceeding 80% for any number of users.

Conclusions

In this thesis, we investigated sequential and interactive aggregation techniques for two classes of digital signatures, with applications to recent proposals of post-quantum signatures. Specifically, we designed a sequential aggregation framework for Hash-and-Sign schemes based on generic trapdoor functions, and an interactive multi-signature framework for group action-based signatures in the Fiat-Shamir paradigm.

During NIST's first call for standardization of post-quantum schemes, the digital signature landscape was mainly dominated by solutions based on structured lattices. The significant activity in the development of lattice-based solutions, has led the research to focus on advanced protocols based on the same assumptions, including aggregate signatures. The additional properties of lattices, however, make these constructions often incompatible with other post-quantum assumptions, even when the signature schemes share the same paradigms. The call for additional post-quantum signatures launched by NIST in 2023 greatly expanded the post-quantum signature landscape, differentiating security assumptions with numerous proposals in both the Hash-and-Sign and Fiat-Shamir paradigms.

For signature aggregation in the Hash-and-Sign paradigm, there is a long track record of works describing sequential aggregation frameworks based on trapdoor permutations, such as RSA. Later works have attempted to extend these approaches to trapdoor functions from post-quantum assumptions. In Chapter 3, we showed that these attempts turn out to be insecure when the trapdoor function lacks some advanced properties available, for instance, for lattice-based PSFs. To obtain a generic framework in the absence of strong assumptions on the underlying functions, we adopted the probabilistic Hash-and-Sign with retry paradigm in the context of sequential aggregation. In particular, we proposed a partial-signature history-free sequential aggregate signature scheme, generalizing previous results and minimizing the requirements for the underlying trapdoor functions. We proved the security of our scheme in the random oracle model, assuming only the non-invertibility of the underlying TDF and an additional notion of indistinguishability on preimages, known as preimage sampleability. This additional property is provable or implicitly assumed for numerous post-quantum TDFs to achieve the security of the associated signature schemes. We explicitly instantiated our framework with three multivariate-based signature schemes, namely UOV, PROV and MAYO, and one code-based scheme, Wave. For each scheme, we have shown how it is possible to reduce the security of sequential aggregation to the same underlying assumptions of the digital signature. Furthermore, although the aggregate signature size is linear with the number of users, we achieved a compression rate between 5% and 34%, also for a small number of signers.

In the second part of the thesis we investigated the aggregation of Fiat-Shamir signatures based on non-abelian group actions. Previous attempts at aggregation in the Fiat-Shamir paradigm mainly involve the Schnorr signature and signatures based on lattices such as Dilithium. Unfortunately, these approaches are not extendable to non-Abelian group actions that lack the necessary algebraic structure. In Chapter 5, we first investigated the design of a sequential aggregation framework based on partial-signature aggregation involving challenges. The security of this construction can be reduced to that of the underlying signature, however it achieves extremely low compression rates and is of little use in practical applications. Next, we propose an interactive multi-signature protocol, achieving a trade-off between aggregation capability and the need for party interaction. The underlying protocol simulates a group action Σ-protocol, where public keys are distributed among participants. The resulting multi-signature is provably secure and can be reduced to the inverse problem of the group action. Finally, we show how the scheme can be instantiated with the three main proposals submitted to NIST competition, namely LESS, MEDS, and ALTEQ, achieving relevant compression rates even for a small number of participants.

Abbreviations

APSF Average Preimage Sampleable Function.

AS Aggregate Signature.

CA Certificate Authority.

CR Collision Resistance.

EUF-CMA Existential Unforgeability against Chosen-Message Attacks.

FDH Full Domain Hash.

FH-SAS Full-History Sequential Aggregate Signature.

FH-UF-CMA Full-History existential Unforgeability against Chosen-Message Attacks.

HaS Hash-and-Sign.

HF-SAS History-Free Sequential Aggregate Signature.

INV Non-Invertibility.

MPC Multi-Party Computation.

MQ Multivariate Quadratic.

MS-UF-CMA Multi-Signature existential Unforgeability against Chosen-Message Attacks.

NIST National Institute of Standards and Technology.

OW One-Wayness.

PKI Public Key Infrastructure.

PQC Post-Quantum Cryptography.

PS Preimage Sampling.

PS-HF-SAS Partial-Signature History-Free Sequential Aggregate Signature.

PS-HF-UF-CMA Partial-Signature History-Free existential Unforgeability against Chosen-Message Attacks.

PSF Preimage Sampleable Function.

QROM Quantum Random Oracle Model.

ROM Random Oracle Model.

SAS Sequential Aggregated Signature.

SAS-EUF-CMA Sequential Existential Unforgeability against Chosen-Message Attacks.

SUF-CMA Strong existential Unforgeability against Chosen-Message Attacks.

TDF Trapdoor Function.

TDP Trapdoor Permutation.

UOV Unbalanced Oil and Vinegar.

List of Tables

1.1 Categories and number of digital signature proposals between the first and second NIST calls. 19

2.1 Summary of existing security proof for Hash-and-Sign schemes in the ROM. 29
2.2 Proposed parameters for Falcon [172] with corresponding key/signature sizes. 39
2.3 Proposed parameters for Wave [15] with corresponding key/signature sizes. 41
2.4 Proposed parameters for UOV [31] with corresponding key/signature sizes. 48
2.5 Proposed parameters for PROV [115] with corresponding key/signature sizes. 48
2.6 Proposed parameters for MAYO [30] with corresponding key/signature sizes. 48

3.1 Aggregate signature sizes and compression rates for UOV [31]. . . 82
3.2 Aggregate signature sizes and compression rates for PROV [115]. . 82
3.3 Aggregate signature sizes and compression rates for MAYO [30]. . 82
3.4 Aggregate signature sizes and compression rates for Wave [71]. . . 86

4.1 Proposed parameters for LESS [13] with corresponding key/signature sizes. 117
4.2 Proposed parameters for MEDS [60] with corresponding key/signature sizes. 117
4.3 Proposed parameters for ALTEQ [37] with corresponding key/signature sizes. 117

5.1 Data sizes (in bits) and choice of parameters comparison between single-signature and multi-signature schemes with N users. 152

List of Figures

3.1 Simplified description of SAS scheme from [143] 51
3.2 Simplified description of FH-SAS 54
3.3 High-level description of HaS-HF-SAS scheme 65

4.1 High-level description of Protocol 4.30. 103

5.1 High-level description of MS-GA scheme of Algorithm 5.5. 137
5.2 Multi-signature rates and parameters for LESS. 154
5.3 Multi-signature rates and parameters for MEDS. 156
5.4 Multi-signature rates and parameters for ALTEQ. 157

List of Algorithms

2.1 Hash-and-Sign with retry 28

2.2 Falcon Signature Scheme 37

2.3 Wave Signature Scheme 40

2.4 UOV Signature Scheme 44

2.5 Provable UOV Signature Scheme 45

2.6 MAYO Signature Scheme 46

3.1 FH-SAS$_\mathsf{T}$ Scheme for Generic TDF 54

3.2 Hash-and-Sign History-Free SAS (HaS-HF-SAS) 65

3.3 Full Reduction OW $\implies$ PS-HF-UF-CMA 68

4.1 3-Round Protocol 96

4.2 Fiat-Shamir Transformation of a Σ-protocol 99

4.3 Signature Scheme based on Group Actions 105

5.1 Group-Action History-Free SAS (GA-HF-SAS) 120

5.2 Full Reduction EUF-CMA $\implies$ PS-HF-UF-CMA 124

5.3 Group Action Σ-Protocol with n ephemeral keys 133

5.4 Variant Signature Scheme based on Group Actions 136

5.5 Multi-Signature from Group Action (MS-GA[$\star$]) 139

5.6 Full Reduction EUF-CMA $\implies$ MS-UF-CMA 141

List of Experiments

1.1 Experiment template 10
1.2 One-way function 12
1.3 (Strong) Existential Unforgeability against Chosen-Message Attack 14
2.1 Trapdoor Function OW, INV, and CR 24
2.2 Trapdoor Function Preimage Sampleability 27
3.1 SAS Full-History EUF-CMA 53
3.2 SAS (Strong) Partial-Signature History-Free EUF-CMA 63
5.1 Multi-Signature EUF-CMA 131

Bibliography

[1] Marius A. Aardal, Diego F. Aranha, Katharina Boudgoust, Sebastian Kolby, and Akira Takahashi. *Aggregating Falcon Signatures with LaBRADOR*. Cryptology ePrint Archive, Paper 2024/311. https://eprint.iacr.org/2024/311. 2024 (cit. on p. 3).

[2] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. "From Identification to Signatures via the Fiat-Shamir Transform: Minimizing Assumptions for Security and Forward-Security." In: *EUROCRYPT 2002*. Ed. by Lars R. Knudsen. Vol. 2332. LNCS. Springer, Heidelberg, Apr. 2002, pp. 418–433. DOI: 10.1007/3-540-46035-7_28 (cit. on p. 98).

[3] Jae Hyun Ahn, Matthew Green, and Susan Hohenberger. "Synchronized aggregate signatures: new definitions, constructions and applications." In: *ACM CCS 2010*. Ed. by Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov. ACM Press, Oct. 2010, pp. 473–484. DOI: 10.1145/1866307.1866360 (cit. on p. 2).

[4] Miklós Ajtai. "Generating Hard Instances of Lattice Problems (Extended Abstract)." In: *28th ACM STOC*. ACM Press, May 1996, pp. 99–108. DOI: 10.1145/237814.237838 (cit. on p. 36).

[5] Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, and Daniel Smith-Tone. *Status report on the third round of the NIST Post-Quantum Cryptography Standardization process*. Tech. rep. NIST IR 8413. Gaithersburg, MD: National Institute of Standards and Technology, Sept. 2022. DOI: 10.6028/nist.ir.8413-upd1 (cit. on pp. 1, 19).

[6] Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. "Cryptographic Group Actions and Applications." In: *ASIACRYPT 2020, Part II*. Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12492. LNCS. Springer, Heidelberg, Dec. 2020, pp. 411–439. DOI: 10.1007/978-3-030-64834-3_14 (cit. on pp. 101, 102, 103).

[7] Martin R. Albrecht, Valerio Cini, Russell W. F. Lai, Giulio Malavolta, and Sri Aravinda Krishnan Thyagarajan. "Lattice-Based SNARKs: Publicly Verifiable, Preprocessing, and Recursively Composable - (Extended Abstract)." In: *CRYPTO 2022, Part II*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Vol. 13508. LNCS. Springer, Heidelberg, Aug. 2022, pp. 102–132. DOI: 10.1007/978-3-031-15979-4_4 (cit. on pp. 2, 3).

[8] Handan Kilinç Alper and Jeffrey Burdges. "Two-Round Trip Schnorr Multisignatures via Delinearized Witnesses." In: *CRYPTO 2021, Part I*. Ed. by Tal Malkin and Chris Peikert. Vol. 12825. LNCS. Virtual Event: Springer, Heidelberg, Aug. 2021, pp. 157–188. DOI: 10.1007/978-3-030-84242-0_7 (cit. on p. 3).

[9] Andris Ambainis, Mike Hamburg, and Dominique Unruh. "Quantum Security Proofs Using Semi-classical Oracles." In: *CRYPTO 2019, Part II*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11693. LNCS. Springer, Heidelberg, Aug. 2019, pp. 269–295. DOI: 10.1007/978-3-030-26951-7_10 (cit. on p. 33).

[10] Thomas Attema and Serge Fehr. "Parallel Repetition of $(k_1, \dots, k_\mu)$-Special-Sound Multi-round Interactive Proofs." In: *CRYPTO 2022, Part I*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Vol. 13507. LNCS. Springer, Heidelberg, Aug. 2022, pp. 415–443. DOI: 10.1007/978-3-031-15802-5_15 (cit. on pp. 97, 98).

[11] László Babai. "Trading Group Theory for Randomness." In: *17th ACM STOC*. ACM Press, May 1985, pp. 421–429. DOI: 10.1145/22145.22192 (cit. on pp. 92, 93).

[12] Ali Bagherzandi, Jung Hee Cheon, and Stanislaw Jarecki. "Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma." In: *ACM CCS 2008*. Ed. by Peng Ning, Paul F. Syverson, and Somesh Jha. ACM Press, Oct. 2008, pp. 449–458. DOI: 10.1145/1455770.1455827 (cit. on p. 3).

[13] Marco Baldi, Alessandro Barenghi, Luke Beckwith, Jean-François Biasse, Andre Esser, Kris Gaj, Kamyar Mohajerani, Gerardo Pelosi, Edoardo Persichetti, Markku-Juhani O. Saarinen, Paolo Santini, and Robert Wallace. *LESS — Linear Equivalence Signature Scheme*. Tech. rep. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. National Institute of Standards and Technology, 2023 (cit. on pp. 4, 91, 110, 113, 117, 120, 153).

[14] Marco Baldi, Marco Bianchi, Franco Chiaraluce, Joachim Rosenthal, and Davide Schipani. "Using LDGM Codes and Sparse Syndromes to Achieve Digital Signatures." In: *Post-Quantum Cryptography - 5th International*

Workshop, PQCrypto 2013. Ed. by Philippe Gaborit. Springer, Heidelberg, June 2013, pp. 1–15. DOI: 10.1007/978-3-642-38616-9_1 (cit. on p. 34).

[15] Gustavo Banegas, Kévin Carrier, André Chailloux, Alain Couvreur, Thomas Debris-Alazard, Philippe Gaborit, Pierre Karpman, Johanna Loyer, Ruben Niederhagen, Nicolas Sendrier, Benjamin Smith, and Jean-Pierre Tillich. *Wave*. Tech. rep. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. National Institute of Standards and Technology, 2023 (cit. on p. 41).

[16] Gustavo Banegas, Thomas Debris-Alazard, Milena Nedeljković, and Benjamin Smith. *Wavelet: Code-based postquantum signatures with fast verification on microcontrollers*. Cryptology ePrint Archive, Report 2021/1432. https://eprint.iacr.org/2021/1432. 2021 (cit. on p. 86).

[17] Alessandro Barenghi, Jean-François Biasse, Edoardo Persichetti, and Paolo Santini. "LESS-FM: Fine-Tuning Signatures from the Code Equivalence Problem." In: *Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021*. Ed. by Jung Hee Cheon and Jean-Pierre Tillich. Springer, Heidelberg, 2021, pp. 23–43. DOI: 10.1007/978-3-030-81293-5_2 (cit. on pp. 4, 91, 103, 108, 112).

[18] Michele Battagliola, Riccardo Longo, Federico Pintore, Edoardo Signorini, and Giovanni Tognolini. *Security of Fixed-Weight Repetitions of Special-Sound Multi-Round Proofs*. Cryptology ePrint Archive, Paper 2024/884. https://eprint.iacr.org/2024/884. 2024 (cit. on p. 108).

[19] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. "Unrestricted Aggregate Signatures." In: *ICALP 2007*. Ed. by Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki. Vol. 4596. LNCS. Springer, Heidelberg, July 2007, pp. 411–422. DOI: 10.1007/978-3-540-73420-8_37 (cit. on p. 2).

[20] Mihir Bellare and Gregory Neven. "Multi-signatures in the plain public-Key model and a general forking lemma." In: *ACM CCS 2006*. Ed. by Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati. ACM Press, Oct. 2006, pp. 390–399. DOI: 10.1145/1180405.1180453 (cit. on p. 3).

[21] Mihir Bellare and Phillip Rogaway. "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols." In: *ACM CCS 93*. Ed. by Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby. ACM Press, Nov. 1993, pp. 62–73. DOI: 10.1145/168588.168596 (cit. on pp. 4, 11, 23, 24, 27).

[22] Mihir Bellare and Phillip Rogaway. “The Exact Security of Digital Signatures: How to Sign with RSA and Rabin.” In: *EUROCRYPT'96*. Ed. by Ueli M. Maurer. Vol. 1070. LNCS. Springer, Heidelberg, May 1996, pp. 399–416. DOI: 10.1007/3-540-68339-9_34 (cit. on pp. 4, 23, 27, 29).

[23] Mihir Bellare and Moti Yung. “Certifying Cryptographic Tools: The Case of Trapdoor Permutations.” In: *CRYPTO'92*. Ed. by Ernest F. Brickell. Vol. 740. LNCS. Springer, Heidelberg, Aug. 1993, pp. 442–460. DOI: 10.1007/3-540-48071-4_31 (cit. on p. 51).

[24] Emanuele Bellini, Rusydi H. Makarim, Carlo Sanna, and Javier A. Verbel. “An Estimator for the Hardness of the MQ Problem.” In: *AFRICACRYPT 22*. Ed. by Lejla Batina and Joan Daemen. Vol. 2022. LNCS. Springer Nature, July 2022, pp. 323–347. DOI: 10.1007/978-3-031-17433-9_14 (cit. on p. 42).

[25] Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. “On the (in)Security of ROS.” In: *Journal of Cryptology* 35.4 (Oct. 2022), p. 25. DOI: 10.1007/s00145-022-09436-0 (cit. on p. 145).

[26] Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. A. van Tilborg. “On the inherent intractability of certain coding problems (Corresp.)” In: *IEEE Trans. Inf. Theory* 24.3 (1978), pp. 384–386. DOI: 10.1109/TIT.1978.1055873 (cit. on p. 40).

[27] Ward Beullens. “Breaking Rainbow Takes a Weekend on a Laptop.” In: *CRYPTO 2022, Part II*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Vol. 13508. LNCS. Springer, Heidelberg, Aug. 2022, pp. 464–479. DOI: 10.1007/978-3-031-15979-4_16 (cit. on p. 34).

[28] Ward Beullens. “Improved Cryptanalysis of UOV and Rainbow.” In: *EUROCRYPT 2021, Part I*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12696. LNCS. Springer, Heidelberg, Oct. 2021, pp. 348–373. DOI: 10.1007/978-3-030-77870-5_13 (cit. on p. 43).

[29] Ward Beullens. “MAYO: Practical Post-quantum Signatures from Oil-and-Vinegar Maps.” In: *SAC 2021*. Ed. by Riham AlTawy and Andreas Hülsing. Vol. 13203. LNCS. Springer, Heidelberg, Sept. 2022, pp. 355–376. DOI: 10.1007/978-3-030-99277-4_17 (cit. on pp. 46, 47, 50, 85).

[30] Ward Beullens, Fabio Campos, Sofía Celi, Basil Hess, and Matthias J. Kannwischer. *MAYO*. Tech. rep. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. National Institute of Standards and Technology, 2023 (cit. on pp. 4, 34, 47, 48, 82, 84, 85).

[31] Ward Beullens, Ming-Shing Chen, Jintai Ding, Boru Gong, Matthias J. Kannwischer, Jacques Patarin, Bo-Yuan Peng, Dieter Schmidt, Cheng-Jhih Shih, Chengdong Tao, and Bo-Yin Yang. *UOV — Unbalanced Oil and Vinegar*. Tech. rep. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. National Institute of Standards and Technology, 2023 (cit. on pp. 4, 34, 44, 48, 81, 82).

[32] Ward Beullens, Shuichi Katsumata, and Federico Pintore. “Calamari and Falafl: Logarithmic (Linkable) Ring Signatures from Isogenies and Lattices.” In: *ASIACRYPT 2020, Part II*. Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12492. LNCS. Springer, Heidelberg, Dec. 2020, pp. 464–492. DOI: 10.1007/978-3-030-64834-3_16 (cit. on pp. 107, 108).

[33] Ward Beullens and Bart Preneel. “Field Lifting for Smaller UOV Public Keys.” In: *INDOCRYPT 2017*. Ed. by Arpita Patra and Nigel P. Smart. Vol. 10698. LNCS. Springer, Heidelberg, Dec. 2017, pp. 227–246 (cit. on p. 34).

[34] Jean-François Biasse, Giacomo Micheli, Edoardo Persichetti, and Paolo Santini. “LESS is More: Code-Based Signatures Without Syndromes.” In: *AFRICACRYPT 20*. Ed. by Abderrahmane Nitaj and Amr M. Youssef. Vol. 12174. LNCS. Springer, Heidelberg, July 2020, pp. 45–65. DOI: 10.1007/978-3-030-51938-4_3 (cit. on p. 112).

[35] Nina Bindel, Mike Hamburg, Kathrin Hövelmanns, Andreas Hülsing, and Edoardo Persichetti. “Tighter Proofs of CCA Security in the Quantum Random Oracle Model.” In: *TCC 2019, Part II*. Ed. by Dennis Hofheinz and Alon Rosen. Vol. 11892. LNCS. Springer, Heidelberg, Dec. 2019, pp. 61–90. DOI: 10.1007/978-3-030-36033-7_3 (cit. on p. 33).

[36] Markus Bläser, Zhili Chen, Dung Hoang Duong, Antoine Joux, Tuong Nguyen, Thomas Plantard, Youming Qiao, Willy Susilo, and Gang Tang. “On Digital Signatures Based on Group Actions: QROM Security and Ring Signatures.” In: *Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024*. Ed. by Markku-Juhani Saarinen and Daniel Smith-Tone. Springer, Heidelberg, June 2024, pp. 227–261. DOI: 10.1007/978-3-031-62743-9_8 (cit. on pp. 100, 105).

[37] Markus Bläser, Dung Hoang Duong, Anand Kumar Narayanan, Thomas Plantard, Youming Qiao, Arnaud Sipasseuth, and Gang Tang. *ALTEQ*. Tech. rep. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. National Institute of Standards and Technology, 2023 (cit. on pp. 4, 91, 110, 116, 117, 120, 155).

[38] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. "Random Oracles in a Quantum World." In: *ASIACRYPT 2011*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Vol. 7073. LNCS. Springer, Heidelberg, Dec. 2011, pp. 41–69. DOI: 10.1007/978-3-642-25385-0_3 (cit. on pp. 12, 30).

[39] Dan Boneh, Manu Drijvers, and Gregory Neven. "Compact Multi-signatures for Smaller Blockchains." In: *ASIACRYPT 2018, Part II*. Ed. by Thomas Peyrin and Steven Galbraith. Vol. 11273. LNCS. Springer, Heidelberg, Dec. 2018, pp. 435–464. DOI: 10.1007/978-3-030-03329-3_15 (cit. on p. 2).

[40] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps." In: *EUROCRYPT 2003*. Ed. by Eli Biham. Vol. 2656. LNCS. Springer, Heidelberg, May 2003, pp. 416–432. DOI: 10.1007/3-540-39200-9_26 (cit. on p. 2).

[41] Dan Boneh and Sam Kim. "One-time and interactive aggregate signatures from lattices." In: *preprint* 4 (2020) (cit. on p. 3).

[42] Dan Boneh and Brent Waters. "Constrained Pseudorandom Functions and Their Applications." In: *ASIACRYPT 2013, Part II*. Ed. by Kazue Sako and Palash Sarkar. Vol. 8270. LNCS. Springer, Heidelberg, Dec. 2013, pp. 280–300. DOI: 10.1007/978-3-642-42045-0_15 (cit. on p. 107).

[43] Giacomo Borin, Edoardo Persichetti, Paolo Santini, Federico Pintore, and Krijn Reijnders. *A Guide to the Design of Digital Signatures based on Cryptographic Group Actions*. Cryptology ePrint Archive, Paper 2023/718. https://eprint.iacr.org/2023/718. 2023 (cit. on pp. 103, 109).

[44] Joppe W. Bos, Olivier Bronchain, Léo Ducas, Serge Fehr, Yu-Hsuan Huang, Thomas Pornin, Eamonn W. Postlethwaite, Thomas Prest, Ludo N. Pulles, and Wessel van Woerden. *HAWK*. Tech. rep. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. National Institute of Standards and Technology, 2023 (cit. on p. 34).

[45] Cecilia Boschini, Akira Takahashi, and Mehdi Tibouchi. "MuSig-L: Lattice-Based Multi-signature with Single-Round Online Phase." In: *CRYPTO 2022, Part II*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Vol. 13508. LNCS. Springer, Heidelberg, Aug. 2022, pp. 276–305. DOI: 10.1007/978-3-031-15979-4_10 (cit. on p. 3).

[46] Leif Both and Alexander May. "Decoding Linear Codes with High Error Rate and Its Impact for LPN Security." In: *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018*. Ed. by Tanja Lange and Rainer Steinwandt. Springer, Heidelberg, 2018, pp. 25–46. DOI: 10.1007/978-3-319-79063-3_2 (cit. on p. 40).

[47] Katharina Boudgoust and Adeline Roux-Langlois. "Overfull: Too Large Aggregate Signatures Based on Lattices." In: *The Computer Journal* 67.2 (2024), pp. 719–727. DOI: 10.1093/COMJNL/BXAD013 (cit. on p. 3).

[48] Katharina Boudgoust and Akira Takahashi. "Sequential Half-Aggregation of Lattice-Based Signatures." In: *ESORICS 2023, Part I*. Ed. by Gene Tsudik, Mauro Conti, Kaitai Liang, and Georgios Smaragdakis. Vol. 14344. LNCS. Springer, Heidelberg, Sept. 2023, pp. 270–289. DOI: 10.1007/978-3-031-50594-2_14 (cit. on pp. 3, 5, 63, 64, 80, 87, 119, 121).

[49] Kyle Brogle, Sharon Goldberg, and Leonid Reyzin. "Sequential Aggregate Signatures with Lazy Verification from Trapdoor Permutations - (Extended Abstract)." In: *ASIACRYPT 2012*. Ed. by Xiaoyun Wang and Kazue Sako. Vol. 7658. LNCS. Springer, Heidelberg, Dec. 2012, pp. 644–662. DOI: 10.1007/978-3-642-34961-4_39 (cit. on pp. 2, 49, 51, 57, 62, 64, 66).

[50] Antoine Casanova, Jean-Charles Faugère, Gilles Macario-Rat, Jacques Patarin, Ludovic Perret, and Jocelyn Ryckeghem. *GeMSS*. Tech. rep. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions. National Institute of Standards and Technology, 2020 (cit. on p. 34).

[51] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. "CSIDH: An Efficient Post-Quantum Commutative Group Action." In: *ASIACRYPT 2018, Part III*. Ed. by Thomas Peyrin and Steven Galbraith. Vol. 11274. LNCS. Springer, Heidelberg, Dec. 2018, pp. 395–427. DOI: 10.1007/978-3-030-03332-3_15 (cit. on pp. 4, 91).

[52] André Chailloux and Thomas Debris-Alazard. "Tight and Optimal Reductions for Signatures Based on Average Trapdoor Preimage Sampleable Functions and Applications to Code-Based Signatures." In: *PKC 2020, Part II*. Ed. by Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas. Vol. 12111. LNCS. Springer, Heidelberg, May 2020, pp. 453–479. DOI: 10.1007/978-3-030-45388-6_16 (cit. on pp. 23, 25, 26, 29, 31, 34, 41, 79).

[53] André Chailloux and Simona Etinski. "On the (in)security of optimized Stern-like signature schemes." In: *Designs, Codes and Cryptography* 92.3 (2024), pp. 803–832. DOI: 10.1007/S10623-023-01329-Y (cit. on p. 107).

[54] Konstantinos Chalkias, François Garillot, Yashvanth Kondi, and Valeria Nikolaenko. "Non-interactive Half-Aggregation of EdDSA and Variants of Schnorr Signatures." In: *CT-RSA 2021*. Ed. by Kenneth G. Paterson. Vol. 12704. LNCS. Springer, Heidelberg, May 2021, pp. 577–608. DOI: 10.1007/978-3-030-75539-3_24 (cit. on p. 119).

[55] Sanjit Chatterjee, M. Prem Laxman Das, and Tapas Pandit. "Revisiting the Security of Salted UOV Signature." In: *Progress in Cryptology - INDOCRYPT 2022 - 23rd International Conference on Cryptology in India, Kolkata, India, December 11-14, 2022, Proceedings*. Ed. by Takanori Isobe and Santanu Sarkar. Vol. 13774. Lecture Notes in Computer Science. 2022, pp. 697–719. DOI: 10.1007/978-3-031-22912-1_31 (cit. on pp. 32, 34).

[56] Jiahui Chen, Jie Ling, Jianting Ning, Zhiniang Peng, and Yang Tan. "MQ Aggregate Signature Schemes with Exact Security Based on UOV Signature." In: *Information Security and Cryptology - 15th International Conference, Inscrypt 2019, Nanjing, China, December 6-8, 2019, Revised Selected Papers*. Ed. by Zhe Liu and Moti Yung. Vol. 12020. Lecture Notes in Computer Science. 2019, pp. 443–451. DOI: 10.1007/978-3-030-42921-8_26 (cit. on pp. 4, 49, 51, 52, 53, 54, 58, 60, 80).

[57] Yanbo Chen. "DualMS: Efficient Lattice-Based Two-Round Multi-signature with Trapdoor-Free Simulation." In: *CRYPTO 2023, Part V*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14085. LNCS. Springer, Heidelberg, Aug. 2023, pp. 716–747. DOI: 10.1007/978-3-031-38554-4_23 (cit. on p. 3).

[58] Yanbo Chen and Yunlei Zhao. "Half-Aggregation of Schnorr Signatures with Tight Reductions." In: *ESORICS 2022, Part II*. Ed. by Vijayalakshmi Atluri, Roberto Di Pietro, Christian Damsgaard Jensen, and Weizhi Meng. Vol. 13555. LNCS. Springer, Heidelberg, Sept. 2022, pp. 385–404. DOI: 10.1007/978-3-031-17146-8_19 (cit. on pp. 49, 119).

[59] Yilei Chen, Nicholas Genise, and Pratyay Mukherjee. "Approximate Trapdoors for Lattices and Smaller Hash-and-Sign Signatures." In: *ASIACRYPT 2019, Part III*. Ed. by Steven D. Galbraith and Shiho Moriai. Vol. 11923. LNCS. Springer, Heidelberg, Dec. 2019, pp. 3–32. DOI: 10.1007/978-3-030-34618-8_1 (cit. on p. 26).

[60] Tung Chou, Ruben Niederhagen, Edoardo Persichetti, Lars Ran, Tovohery Hajatiana Randrianarisoa, Krijn Reijnders, Simona Samardjiska, and Monika Trimoska. *MEDS — Matrix Equivalence Digital Signature*. Tech. rep. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. National Institute of Standards and Technology, 2023 (cit. on pp. 4, 91, 110, 115, 117, 120, 155).

[61] Tung Chou, Ruben Niederhagen, Edoardo Persichetti, Tovohery Hajatiana Randrianarisoa, Krijn Reijnders, Simona Samardjiska, and Monika Trimoska. "Take Your MEDS: Digital Signatures from Matrix Code Equivalence." In: *AFRICACRYPT 23*. Ed. by Nadia El Mrabet, Luca De Feo, and Sylvain Duquesne. Vol. 14064. LNCS. Springer Nature, July 2023, pp. 28–52. DOI: 10.1007/978-3-031-37679-5_2 (cit. on pp. 4, 91, 114).

[62] Tung Chou, Edoardo Persichetti, and Paolo Santini. *On Linear Equivalence, Canonical Forms, and Digital Signatures*. Cryptology ePrint Archive, Paper 2023/1533. https://eprint.iacr.org/2023/1533. 2023 (cit. on p. 113).

[63] Benoît Cogliati, Pierre-Alain Fouque, Louis Goubin, and Brice Minaud. *New Security Proofs and Techniques for Hash-and-Sign with Retry Signature Schemes*. Cryptology ePrint Archive, Paper 2024/609. https://eprint.iacr.org/2024/609. 2024 (cit. on pp. 12, 32).

[64] Nicolas Courtois, Matthieu Finiasz, and Nicolas Sendrier. "How to Achieve a McEliece-Based Digital Signature Scheme." In: *ASIACRYPT 2001*. Ed. by Colin Boyd. Vol. 2248. LNCS. Springer, Heidelberg, Dec. 2001, pp. 157–174. DOI: 10.1007/3-540-45682-1_10 (cit. on p. 34).

[65] Jean-Marc Couveignes. *Hard Homogeneous Spaces*. Cryptology ePrint Archive, Report 2006/291. https://eprint.iacr.org/2006/291. 2006 (cit. on p. 103).

[66] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Hoboken, NJ: Wiley-Interscience, 2006 (cit. on p. 151).

[67] Özgür Dagdelen, Marc Fischlin, and Tommaso Gagliardoni. "The Fiat-Shamir Transformation in a Quantum World." In: *ASIACRYPT 2013, Part II*. Ed. by Kazue Sako and Palash Sarkar. Vol. 8270. LNCS. Springer, Heidelberg, Dec. 2013, pp. 62–81. DOI: 10.1007/978-3-642-42045-0_4 (cit. on pp. 12, 100).

[68] Ivan Damgård, Oded Goldreich, Tatsuaki Okamoto, and Avi Wigderson. "Honest Verifier vs Dishonest Verifier in Public Cain Zero-Knowledge Proofs." In: *CRYPTO'95*. Ed. by Don Coppersmith. Vol. 963. LNCS. Springer, Heidelberg, Aug. 1995, pp. 325–338. DOI: 10.1007/3-540-44750-4_26 (cit. on p. 95).

[69] Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. "Two-Round n-out-of-n and Multi-signatures and Trapdoor Commitment from Lattices." In: *PKC 2021, Part I*. Ed. by Juan Garay. Vol. 12710. LNCS. Springer, Heidelberg, May 2021, pp. 99–130. DOI: 10.1007/978-3-030-75245-3_5 (cit. on pp. 3, 131).

[70] Luca De Feo and Steven D. Galbraith. "SeaSign: Compact Isogeny Signatures from Class Group Actions." In: *EUROCRYPT 2019, Part III*. Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11478. LNCS. Springer, Heidelberg, May 2019, pp. 759–789. DOI: 10.1007/978-3-030-17659-4_26 (cit. on pp. 4, 91, 108).

[71] Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. "Wave: A New Family of Trapdoor One-Way Preimage Sampleable Functions Based on Codes." In: *ASIACRYPT 2019, Part I*. Ed. by Steven D. Galbraith and Shiho Moriai. Vol. 11921. LNCS. Springer, Heidelberg, Dec. 2019, pp. 21–51. DOI: 10.1007/978-3-030-34578-5_2 (cit. on pp. 34, 41, 50, 86).

[72] Thomas Debris-Alazard and Jean-Pierre Tillich. "Two Attacks on Rank Metric Code-Based Schemes: RankSign and an IBE Scheme." In: *ASIACRYPT 2018, Part I*. Ed. by Thomas Peyrin and Steven Galbraith. Vol. 11272. LNCS. Springer, Heidelberg, Dec. 2018, pp. 62–92. DOI: 10.1007/978-3-030-03326-2_3 (cit. on p. 34).

[73] Lalita Devadas, Rishab Goyal, Yael Kalai, and Vinod Vaikuntanathan. "Rate-1 Non-Interactive Arguments for Batch-NP and Applications." In: *63rd FOCS*. IEEE Computer Society Press, Oct. 2022, pp. 1057–1068. DOI: 10.1109/FOCS54457.2022.00103 (cit. on p. 2).

[74] Antonio J. Di Scala, Carlo Sanna, and Edoardo Signorini. "On the condition number of the Vandermonde matrix of the nth cyclotomic polynomial." In: *Journal of Mathematical Cryptology* 15.1 (Nov. 2021), pp. 174–178. DOI: 10.1515/JMC-2020-0009 (cit. on p. 5).

[75] Antonio J. Di Scala, Carlo Sanna, and Edoardo Signorini. "RLWE and PLWE over cyclotomic fields are not equivalent." In: *Applicable Algebra in Engineering, Communication and Computing* 35.3 (2022), pp. 351–358. DOI: 10.1007/S00200-022-00552-9 (cit. on p. 5).

[76] Whitfield Diffie and Martin E. Hellman. "New Directions in Cryptography." In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654. DOI: 10.1109/TIT.1976.1055638 (cit. on p. 103).

[77] Jintai Ding, Joshua Deaton, Kurt Schmidt, Vishakha, and Zheng Zhang. "Cryptanalysis of the Lifted Unbalanced Oil Vinegar Signature Scheme." In: *CRYPTO 2020, Part III*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12172. LNCS. Springer, Heidelberg, Aug. 2020, pp. 279–298. DOI: 10.1007/978-3-030-56877-1_10 (cit. on p. 34).

[78] Jintai Ding, Boru Gong, Hao Guo, Xiaoou He, Yi Jin, Yuansheng Pan, Dieter Schmidt, Chengdong Tao, Danli Xie, Bo-Yin Yang, and Ziyu Zhao. *TUOV — Triangular Unbalanced Oil and Vinegar*. Tech. rep. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. National Institute of Standards and Technology, 2023 (cit. on pp. 4, 34).

[79] Jintai Ding and Dieter Schmidt. "Rainbow, a New Multivariable Polynomial Signature Scheme." In: *ACNS 05*. Ed. by John Ioannidis, Angelos Keromytis, and Moti Yung. Vol. 3531. LNCS. Springer, Heidelberg, June 2005, pp. 164–175. DOI: 10.1007/11496137_12 (cit. on p. 34).

[80] Jintai Ding and Bo-Yin Yang. "Degree of Regularity for HFEv and HFEv-." In: *Post-Quantum Cryptography - 5th International Workshop, PQCrypto 2013*. Ed. by Philippe Gaborit. Springer, Heidelberg, June 2013, pp. 52–66. DOI: 10.1007/978-3-642-38616-9_4 (cit. on p. 34).

[81] Jelle Don, Serge Fehr, and Christian Majenz. "The Measure-and-Reprogram Technique 2.0: Multi-round Fiat-Shamir and More." In: *CRYPTO 2020, Part III*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12172. LNCS. Springer, Heidelberg, Aug. 2020, pp. 602–631. DOI: 10.1007/978-3-030-56877-1_21 (cit. on p. 33).

[82] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. "Security of the Fiat-Shamir Transformation in the Quantum Random-Oracle Model." In: *CRYPTO 2019, Part II*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11693. LNCS. Springer, Heidelberg, Aug. 2019, pp. 356–383. DOI: 10.1007/978-3-030-26951-7_13 (cit. on pp. 12, 100).

[83] Yarkın Doröz, Jeffrey Hoffstein, Joseph H. Silverman, and Berk Sunar. *MMSAT: A Scheme for Multimessage Multiuser Signature Aggregation*. Cryptology ePrint Archive, Report 2020/520. https://eprint.iacr.org/2020/520. 2020 (cit. on p. 3).

[84] Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs. "On the Security of Two-Round Multi-Signatures." In: *2019 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2019, pp. 1084–1101. DOI: 10.1109/SP.2019.00050 (cit. on p. 3).

[85] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. "Efficient Identity-Based Encryption over NTRU Lattices." In: *ASIACRYPT 2014, Part II*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8874. LNCS. Springer, Heidelberg, Dec. 2014, pp. 22–41. DOI: 10.1007/978-3-662-45608-8_2 (cit. on p. 34).

[86] Léo Ducas and Phong Q. Nguyen. "Learning a Zonotope and More: Cryptanalysis of NTRUSign Countermeasures." In: *ASIACRYPT 2012*. Ed. by Xiaoyun Wang and Kazue Sako. Vol. 7658. LNCS. Springer, Heidelberg, Dec. 2012, pp. 433–450. DOI: 10.1007/978-3-642-34961-4_27 (cit. on p. 33).

[87] Léo Ducas, Eamonn W. Postlethwaite, Ludo N. Pulles, and Wessel P. J. van Woerden. "Hawk: Module LIP Makes Lattice Signatures Fast, Compact and Simple." In: *ASIACRYPT 2022, Part IV*. Ed. by Shweta Agrawal and Dongdai Lin. Vol. 13794. LNCS. Springer, Heidelberg, Dec. 2022, pp. 65–94. DOI: 10.1007/978-3-031-22972-5_3 (cit. on p. 34).

[88] Léo Ducas and Thomas Prest. “Fast Fourier Orthogonalization.” In: *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC 2016, Waterloo, ON, Canada, July 19-22, 2016*. Ed. by Sergei A. Abramov, Eugene V. Zima, and Xiao-Shan Gao. 2016, pp. 191–198. doi: 10.1145/2930889.2930923 (cit. on p. 34).

[89] Léo Ducas and Wessel P. J. van Woerden. “On the Lattice Isomorphism Problem, Quadratic Forms, Remarkable Lattices, and Cryptography.” In: *EUROCRYPT 2022, Part III*. Ed. by Orr Dunkelman and Stefan Dziembowski. Vol. 13277. LNCS. Springer, Heidelberg, May 2022, pp. 643–673. doi: 10.1007/978-3-031-07082-2_23 (cit. on pp. 4, 91).

[90] Oliver Eikemeier, Marc Fischlin, Jens-Fabian Götzmann, Anja Lehmann, Dominique Schröder, Peter Schröder, and Daniel Wagner. “History-Free Aggregate Message Authentication Codes.” In: *SCN 10*. Ed. by Juan A. Garay and Roberto De Prisco. Vol. 6280. LNCS. Springer, Heidelberg, Sept. 2010, pp. 309–328. doi: 10.1007/978-3-642-15317-4_20 (cit. on p. 51).

[91] Rachid El Bansarkhani and Johannes Buchmann. “Towards Lattice Based Aggregate Signatures.” In: *AFRICACRYPT 14*. Ed. by David Pointcheval and Damien Vergnaud. Vol. 8469. LNCS. Springer, Heidelberg, May 2014, pp. 336–355. doi: 10.1007/978-3-319-06734-6_21 (cit. on pp. 3, 49, 51, 52, 62).

[92] Rachid El Bansarkhani, Mohamed Saied Emam Mohamed, and Albrecht Petzoldt. “MQSAS - A Multivariate Sequential Aggregate Signature Scheme.” In: *ISC 2016*. Ed. by Matt Bishop and Anderson C. A. Nascimento. Vol. 9866. LNCS. Springer, Heidelberg, Sept. 2016, pp. 426–439. doi: 10.1007/978-3-319-45871-7_25 (cit. on pp. 3, 4, 49, 51, 52, 53, 54, 58, 60, 80).

[93] Rachid El Bansarkhani and Jan Sturm. “An Efficient Lattice-Based Multisignature Scheme with Applications to Bitcoins.” In: *CANS 16*. Ed. by Sara Foresti and Giuseppe Persiano. Vol. 10052. LNCS. Springer, Heidelberg, Nov. 2016, pp. 140–155. doi: 10.1007/978-3-319-48965-0_9 (cit. on p. 3).

[94] Andre Esser and Emanuele Bellini. “Syndrome Decoding Estimator.” In: *PKC 2022, Part I*. Ed. by Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe. Vol. 13177. LNCS. Springer, Heidelberg, Mar. 2022, pp. 112–141. doi: 10.1007/978-3-030-97121-2_5 (cit. on p. 40).

[95] Jean-Charles Faugère, Valérie Gauthier-Umaña, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. “A distinguisher for high rate McEliece cryptosystems.” In: *2011 IEEE Information Theory Workshop, ITW 2011, Paraty, Brazil, October 16-20, 2011*. IEEE, 2011, pp. 282–286. doi: 10.1109/ITW.2011.6089437 (cit. on p. 34).

[96] Amos Fiat and Adi Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems." In: *CRYPTO'86*. Ed. by Andrew M. Odlyzko. Vol. 263. LNCS. Springer, Heidelberg, Aug. 1987, pp. 186–194. DOI: 10.1007/3-540-47721-7_12 (cit. on pp. 91, 95, 99).

[97] Matthieu Finiasz. "Parallel-CFS - Strengthening the CFS McEliece-Based Signature Scheme." In: *SAC 2010*. Ed. by Alex Biryukov, Guang Gong, and Douglas R. Stinson. Vol. 6544. LNCS. Springer, Heidelberg, Aug. 2011, pp. 159–170. DOI: 10.1007/978-3-642-19574-7_11 (cit. on p. 34).

[98] Marc Fischlin, Anja Lehmann, and Dominique Schröder. "History-Free Sequential Aggregate Signatures." In: *SCN 12*. Ed. by Ivan Visconti and Roberto De Prisco. Vol. 7485. LNCS. Springer, Heidelberg, Sept. 2012, pp. 113–130. DOI: 10.1007/978-3-642-32928-9_7 (cit. on pp. 2, 51, 62).

[99] Nils Fleischhacker, Gottfried Herold, Mark Simkin, and Zhenfei Zhang. "Chipmunk: Better Synchronized Multi-Signatures from Lattices." In: *ACM CCS 2023*. Ed. by Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda. ACM Press, Nov. 2023, pp. 386–400. DOI: 10.1145/3576915.3623219 (cit. on p. 3).

[100] Nils Fleischhacker, Mark Simkin, and Zhenfei Zhang. "Squirrel: Efficient Synchronized Multi-Signatures from Lattices." In: *ACM CCS 2022*. Ed. by Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi. ACM Press, Nov. 2022, pp. 1109–1123. DOI: 10.1145/3548606.3560655 (cit. on p. 3).

[101] Scott Fluhrer. *Reassessing Grover's Algorithm*. Cryptology ePrint Archive, Report 2017/811. https://eprint.iacr.org/2017/811. 2017 (cit. on p. 18).

[102] Giacomo Fregona, Claudia De Lazzari, Damiano Giani, Fernando Chirici, Francesco Stocco, Edoardo Signorini, Guglielmo Morgari, Tommaso Occhipinti, Alessandro Zavatta, and Davide Bacco. "Authentication Methods for Quantum Key Distribution: Challenges and Perspectives." In: *Toward a Quantum-Safe Communication Infrastructure*. Ed. by André Xuereb Rainer Steinwandt. Vol. 64. NATO Science for Peace and Security Series - D: Information and Communication Security. IOS Press, Apr. 2024, pp. 54–66. DOI: 10.3233/NICSP240007 (cit. on p. 5).

[103] Masayuki Fukumitsu and Shingo Hasegawa. "A Lattice-Based Provably Secure Multisignature Scheme in Quantum Random Oracle Model." In: *ProvSec 2020*. Ed. by Khoa Nguyen, Wenling Wu, Kwok-Yan Lam, and Huaxiong Wang. Vol. 12505. LNCS. Springer, Heidelberg, Nov. 2020, pp. 45–64. DOI: 10.1007/978-3-030-62576-4_3 (cit. on p. 3).

[104] Hiroki Furue, Yasuhiko Ikematsu, Fumitaka Hoshino, Tsuyoshi Takagi, Kan Yasuda, Toshiyuki Miyazawa, Tsunekazu Saito, and Akira Nagai. *QR-UOV*. Tech. rep. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. National Institute of Standards and Technology, 2023 (cit. on pp. 4, 34).

[105] Philippe Gaborit, Olivier Ruatta, Julien Schrek, and Gilles Zémor. "New Results for Rank-Based Cryptography." In: *AFRICACRYPT 14*. Ed. by David Pointcheval and Damien Vergnaud. Vol. 8469. LNCS. Springer, Heidelberg, May 2014, pp. 1–12. DOI: 10.1007/978-3-319-06734-6_1 (cit. on p. 34).

[106] Juan A. Garay, Philip D. MacKenzie, and Ke Yang. "Strengthening Zero-Knowledge Protocols Using Signatures." In: *Journal of Cryptology* 19.2 (Apr. 2006), pp. 169–209. DOI: 10.1007/s00145-005-0307-3 (cit. on p. 95).

[107] Craig Gentry, Adam O'Neill, and Leonid Reyzin. "A Unified Framework for Trapdoor-Permutation-Based Sequential Aggregate Signatures." In: *PKC 2018, Part II*. Ed. by Michel Abdalla and Ricardo Dahab. Vol. 10770. LNCS. Springer, Heidelberg, Mar. 2018, pp. 34–57. DOI: 10.1007/978-3-319-76581-5_2 (cit. on pp. 2, 3).

[108] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. "Trapdoors for hard lattices and new cryptographic constructions." In: *40th ACM STOC*. Ed. by Richard E. Ladner and Cynthia Dwork. ACM Press, May 2008, pp. 197–206. DOI: 10.1145/1374376.1374407 (cit. on pp. 3, 23, 25, 26, 29, 30, 33, 36, 37, 60, 80).

[109] Craig Gentry and Zulfikar Ramzan. "Identity-Based Aggregate Signatures." In: *PKC 2006*. Ed. by Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin. Vol. 3958. LNCS. Springer, Heidelberg, Apr. 2006, pp. 257–273. DOI: 10.1007/11745853_17 (cit. on p. 2).

[110] Danilo Gligoroski, Simona Samardjiska, Håkon Jacobsen, and Sergey Bezzateev. *McEliece in the world of Escher*. Cryptology ePrint Archive, Report 2014/360. https://eprint.iacr.org/2014/360. 2014 (cit. on p. 34).

[111] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. "Public-Key Cryptosystems from Lattice Reduction Problems." In: *CRYPTO'97*. Ed. by Burton S. Kaliski Jr. Vol. 1294. LNCS. Springer, Heidelberg, Aug. 1997, pp. 112–131. DOI: 10.1007/BFb0052231 (cit. on pp. 33, 37).

[112] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The Knowledge Complexity of Interactive Proof Systems." In: *SIAM Journal on Computing* 18.1 (1989), pp. 186–208 (cit. on pp. 94, 95).

[113] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract)." In: *17th ACM STOC*. ACM Press, May 1985, pp. 291–304. DOI: 10.1145/22145.22178 (cit. on p. 92).

[114] Shafi Goldwasser and Michael Sipser. "Private Coins versus Public Coins in Interactive Proof Systems." In: *18th ACM STOC*. ACM Press, May 1986, pp. 59–68. DOI: 10.1145/12130.12137 (cit. on p. 93).

[115] Louis Goubin, Benoît Cogliati, Jean-Charles Faugère, Pierre-Alain Fouque, Robin Larrieu, Gilles Macario-Rat, Brice Minaud, and Jacques Patarin. *PROV — PRovable unbalanced Oil and Vinegar*. Tech. rep. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. National Institute of Standards and Technology, 2023 (cit. on pp. 4, 34, 46, 48, 82, 83, 84).

[116] Alex B. Grilo, Kathrin Hövelmanns, Andreas Hülsing, and Christian Majenz. "Tight Adaptive Reprogramming in the QROM." In: *ASIACRYPT 2021, Part I*. Ed. by Mehdi Tibouchi and Huaxiong Wang. Vol. 13090. LNCS. Springer, Heidelberg, Dec. 2021, pp. 637–667. DOI: 10.1007/978-3-030-92062-3_22 (cit. on p. 33).

[117] Lov K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search." In: *28th ACM STOC*. ACM Press, May 1996, pp. 212–219. DOI: 10.1145/237814.237866 (cit. on p. 17).

[118] Gunnar Hartung, Björn Kaidel, Alexander Koch, Jessica Koch, and Andy Rupp. "Fault-Tolerant Aggregate Signatures." In: *PKC 2016, Part I*. Ed. by Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang. Vol. 9614. LNCS. Springer, Heidelberg, Mar. 2016, pp. 331–356. DOI: 10.1007/978-3-662-49384-7_13 (cit. on p. 2).

[119] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. "NTRUSIGN: Digital Signatures Using the NTRU Lattice." In: *CT-RSA 2003*. Ed. by Marc Joye. Vol. 2612. LNCS. Springer, Heidelberg, Apr. 2003, pp. 122–140. DOI: 10.1007/3-540-36563-X_9 (cit. on pp. 33, 37).

[120] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. "NTRU: A Ring-Based Public Key Cryptosystem." In: *Third Algorithmic Number Theory Symposium (ANTS)*. Vol. 1423. LNCS. Springer, Heidelberg, June 1998, pp. 267–288 (cit. on p. 38).

[121] Susan Hohenberger, Venkata Koppula, and Brent Waters. "Universal Signature Aggregators." In: *EUROCRYPT 2015, Part II*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9057. LNCS. Springer, Heidelberg, Apr. 2015, pp. 3–34. DOI: 10.1007/978-3-662-46803-6_1 (cit. on p. 2).

[122] Akinori Hosoyamada and Kan Yasuda. “Building Quantum-One-Way Functions from Block Ciphers: Davies-Meyer and Merkle-Damgård Constructions.” In: *ASIACRYPT 2018, Part I*. Ed. by Thomas Peyrin and Steven Galbraith. Vol. 11272. LNCS. Springer, Heidelberg, Dec. 2018, pp. 275–304. DOI: 10.1007/978-3-030-03326-2_10 (cit. on p. 25).

[123] James Howe, Thomas Prest, and Daniel Apon. “SoK: How (not) to Design and Implement Post-quantum Cryptography.” In: *CT-RSA 2021*. Ed. by Kenneth G. Paterson. Vol. 12704. LNCS. Springer, Heidelberg, May 2021, pp. 444–477. DOI: 10.1007/978-3-030-75539-3_19 (cit. on p. 26).

[124] Andreas Hülsing, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, Jean-Philippe Aumasson, Bas Westerbaan, and Ward Beullens. *SPHINCS$^+$*. Tech. rep. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022. National Institute of Standards and Technology, 2022 (cit. on pp. 2, 19).

[125] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. “Zero-knowledge from secure multiparty computation.” In: *39th ACM STOC*. Ed. by David S. Johnson and Uriel Feige. ACM Press, June 2007, pp. 21–30. DOI: 10.1145/1250790.1250794 (cit. on p. 109).

[126] K. Itakura and K. Nakamura. “A public-key cryptosystem suitable for digital multisignature.” In: *NEC research and development* 71 (1983), pp. 1–8 (cit. on p. 3).

[127] Antoine Joux. *MPC in the head for isomorphisms and group actions*. Cryptology ePrint Archive, Paper 2023/664. https://eprint.iacr.org/2023/664. 2023 (cit. on p. 109).

[128] Saqib A. Kakvi, Eike Kiltz, and Alexander May. “Certifying RSA.” In: *ASIACRYPT 2012*. Ed. by Xiaoyun Wang and Kazue Sako. Vol. 7658. LNCS. Springer, Heidelberg, Dec. 2012, pp. 404–414. DOI: 10.1007/978-3-642-34961-4_25 (cit. on p. 51).

[129] Eike Kiltz, Daniel Masny, and Jiaxin Pan. “Optimal Security Proofs for Signatures from Identification Schemes.” In: *CRYPTO 2016, Part II*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9815. LNCS. Springer, Heidelberg, Aug. 2016, pp. 33–61. DOI: 10.1007/978-3-662-53008-5_2 (cit. on p. 100).

[130] Aviad Kipnis, Jacques Patarin, and Louis Goubin. “Unbalanced Oil and Vinegar Signature Schemes.” In: *EUROCRYPT’99*. Ed. by Jacques Stern. Vol. 1592. LNCS. Springer, Heidelberg, May 1999, pp. 206–222. DOI: 10.1007/3-540-48910-X_15 (cit. on pp. 34, 44, 50).

[131] Aviad Kipnis and Adi Shamir. "Cryptanalysis of the Oil & Vinegar Signature Scheme." In: *CRYPTO'98*. Ed. by Hugo Krawczyk. Vol. 1462. LNCS. Springer, Heidelberg, Aug. 1998, pp. 257–266. DOI: 10.1007/BFb0055733 (cit. on pp. 34, 43).

[132] Philip N. Klein. "Finding the closest lattice vector when it's unusually close." In: *11th SODA*. Ed. by David B. Shmoys. ACM-SIAM, Jan. 2000, pp. 937–941 (cit. on p. 37).

[133] Neal Koblitz and Alfred J. Menezes. "The random oracle model: a twenty-year retrospective." In: *DCC* 77.2-3 (2015), pp. 587–610. DOI: 10.1007/s10623-015-0094-2 (cit. on p. 11).

[134] Haruhisa Kosuge and Keita Xagawa. "Probabilistic Hash-and-Sign with Retry in the Quantum Random Oracle Model." In: *PKC 2024, Part I*. Ed. by Qiang Tang and Vanessa Teague. Vol. 14601. LNCS. Springer, Heidelberg, Apr. 2024, pp. 259–288. DOI: 10.1007/978-3-031-57718-5_9 (cit. on pp. 12, 23, 27, 29, 32, 33, 35, 43, 45, 49).

[135] Gregory Landais and Nicolas Sendrier. "Implementing CFS." In: *INDOCRYPT 2012*. Ed. by Steven D. Galbraith and Mridul Nandi. Vol. 7668. LNCS. Springer, Heidelberg, Dec. 2012, pp. 474–488. DOI: 10.1007/978-3-642-34931-7_27 (cit. on p. 34).

[136] Kwangsu Lee, Dong Hoon Lee, and Moti Yung. "Sequential Aggregate Signatures Made Shorter." In: *ACNS 13*. Ed. by Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini. Vol. 7954. LNCS. Springer, Heidelberg, June 2013, pp. 202–217. DOI: 10.1007/978-3-642-38980-1_13 (cit. on p. 2).

[137] Kwangsu Lee, Dong Hoon Lee, and Moti Yung. "Sequential Aggregate Signatures with Short Public Keys: Design, Analysis and Implementation Studies." In: *PKC 2013*. Ed. by Kaoru Kurosawa and Goichiro Hanaoka. Vol. 7778. LNCS. Springer, Heidelberg, Feb. 2013, pp. 423–442. DOI: 10.1007/978-3-642-36362-7_26 (cit. on p. 2).

[138] Wijik Lee, Young-Sik Kim, Yong-Woo Lee, and Jong-Seon No. *pqsigRM*. Tech. rep. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions. National Institute of Standards and Technology, 2017 (cit. on p. 34).

[139] A.A. Levitskaya. "Systems of random equations over finite algebraic structures." In: *Cybernetics and Systems Analysis* 41 (2005), pp. 67–93 (cit. on p. 84).

[140] Qipeng Liu and Mark Zhandry. "Revisiting Post-quantum Fiat-Shamir." In: *CRYPTO 2019, Part II*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11693. LNCS. Springer, Heidelberg, Aug. 2019, pp. 326–355. DOI: 10.1007/978-3-030-26951-7_12 (cit. on pp. 12, 100).

[141] Yu Liu, Haodong Jiang, and Yunlei Zhao. *Tighter Post-quantum Proof for Plain FDH, PFDH and GPV-IBE*. Cryptology ePrint Archive, Report 2022/1441. https://eprint.iacr.org/2022/1441. 2022 (cit. on p. 12).

[142] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. "Sequential Aggregate Signatures and Multisignatures Without Random Oracles." In: *EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. LNCS. Springer, Heidelberg, May 2006, pp. 465–485. DOI: 10.1007/11761679_28 (cit. on p. 2).

[143] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. "Sequential Aggregate Signatures from Trapdoor Permutations." In: *EUROCRYPT 2004*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. LNCS. Springer, Heidelberg, May 2004, pp. 74–90. DOI: 10.1007/978-3-540-24676-3_5 (cit. on pp. 2, 4, 49, 50, 51, 52, 54, 55, 60, 61, 62).

[144] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. *CRYSTALS-DILITHIUM*. Tech. rep. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022. National Institute of Standards and Technology, 2022 (cit. on pp. 2, 19).

[145] Changshe Ma, Jian Weng, Yingjiu Li, and Robert H. Deng. "Efficient discrete logarithm based multi-signature scheme in the plain public key model." In: *DCC* 54.2 (2010), pp. 121–133. DOI: 10.1007/s10623-009-9313-z (cit. on p. 3).

[146] Florence Jessie MacWilliams. "Combinatorial problems of elementary abelian groups." PhD thesis. Radcliffe College, 1962 (cit. on p. 112).

[147] Tsutomu Matsumoto and Hideki Imai. "Public Quadratic Polynominal-Tuples for Efficient Signature-Verification and Message-Encryption." In: *EUROCRYPT'88*. Ed. by C. G. Günther. Vol. 330. LNCS. Springer, Heidelberg, May 1988, pp. 419–453. DOI: 10.1007/3-540-45961-8_39 (cit. on p. 42).

[148] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. "Simple Schnorr multi-signatures with applications to Bitcoin." In: *DCC* 87.9 (2019), pp. 2139–2164. DOI: 10.1007/s10623-019-00608-x (cit. on p. 3).

[149] Alessio Meneghetti and Edoardo Signorini. "History-Free Sequential Aggregation of Hash-and-Sign Signatures." In: *CT-RSA 2024*. Ed. by Elisabeth Oswald. Vol. 14643. LNCS. Springer, Heidelberg, May 2024, pp. 187–223. DOI: 10.1007/978-3-031-58868-6_8 (cit. on p. 50).

[150] Daniele Micciancio and Chris Peikert. "Hardness of SIS and LWE with Small Parameters." In: *CRYPTO 2013, Part I*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8042. LNCS. Springer, Heidelberg, Aug. 2013, pp. 21–39. DOI: 10.1007/978-3-642-40041-4_2 (cit. on p. 36).

[151] Daniele Micciancio and Chris Peikert. "Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller." In: *EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. LNCS. Springer, Heidelberg, Apr. 2012, pp. 700–718. DOI: 10.1007/978-3-642-29011-4_41 (cit. on p. 34).

[152] Daniele Micciancio and Oded Regev. "Worst-Case to Average-Case Reductions Based on Gaussian Measures." In: *45th FOCS*. IEEE Computer Society Press, Oct. 2004, pp. 372–381. DOI: 10.1109/FOCS.2004.72 (cit. on p. 36).

[153] Dustin Moody and Ray A. Perlner. "Vulnerabilities of "McEliece in the World of Escher"." In: *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016*. Ed. by Tsuyoshi Takagi. Springer, Heidelberg, 2016, pp. 104–117. DOI: 10.1007/978-3-319-29360-8_8 (cit. on p. 34).

[154] Guglielmo Morgari, Edoardo Signorini, and Francesco Stocco. "On the classical authentication in Quantum Key Distribution." In: *CrypTOrino 2021*. Vol. 4. Collectio Ciphrarum. Aracne, May 2021 (cit. on p. 5).

[155] Katherine Morrison. "Equivalence for Rank-Metric and Matrix Codes and Automorphism Groups of Gabidulin Codes." In: *IEEE Transactions on Information Theory* 60.11 (2014), pp. 7035–7046. DOI: 10.1109/TIT.2014.2359198 (cit. on p. 114).

[156] Gregory Neven. "Efficient Sequential Aggregate Signed Data." In: *EUROCRYPT 2008*. Ed. by Nigel P. Smart. Vol. 4965. LNCS. Springer, Heidelberg, Apr. 2008, pp. 52–69. DOI: 10.1007/978-3-540-78967-3_4 (cit. on pp. 2, 3, 4, 49, 51, 52, 62, 64).

[157] Phong Q. Nguyen and Oded Regev. "Learning a Parallelepiped: Cryptanalysis of GGH and NTRU Signatures." In: *EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. LNCS. Springer, Heidelberg, May 2006, pp. 271–288. DOI: 10.1007/11761679_17 (cit. on p. 33).

[158] Jonas Nick, Tim Ruffing, and Yannick Seurin. "MuSig2: Simple Two-Round Schnorr Multi-signatures." In: *CRYPTO 2021, Part I*. Ed. by Tal Malkin and Chris Peikert. Vol. 12825. LNCS. Virtual Event: Springer, Heidelberg, Aug. 2021, pp. 189–221. DOI: 10.1007/978-3-030-84242-0_8 (cit. on p. 3).

[159] Jonas Nick, Tim Ruffing, Yannick Seurin, and Pieter Wuille. “MuSig-DN: Schnorr Multi-Signatures with Verifiably Deterministic Nonces.” In: *ACM CCS 2020*. Ed. by Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna. ACM Press, Nov. 2020, pp. 1717–1731. DOI: 10.1145/3372297.3417236 (cit. on p. 3).

[160] Antonio Nicolosi, Maxwell N. Krohn, Yevgeniy Dodis, and David Mazières. “Proactive Two-Party Signatures for User Authentication.” In: *NDSS 2003*. The Internet Society, Feb. 2003 (cit. on p. 3).

[161] NIST. *Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process*. URL: https://csrc.nist.gov/projects/pqc-dig-sig/standardization/call-for-proposals. 2022 (cit. on pp. 2, 19, 91, 110).

[162] NIST. *Post-Quantum Cryptography Standardization*. URL: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography. 2017 (cit. on pp. 1, 19).

[163] Kazuo Ohta and Tatsuaki Okamoto. “Multi-signature schemes secure against active insider attacks.” In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 82.1 (1999), pp. 21–31 (cit. on p. 3).

[164] Tatsuaki Okamoto. “A digital multisignature scheme using bijective public-key cryptosystems.” In: *ACM Transactions on Computer Systems (TOCS)* 6.4 (1988), pp. 432–441 (cit. on p. 3).

[165] Jacques Patarin. “Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms.” In: *EUROCRYPT’96*. Ed. by Ueli M. Maurer. Vol. 1070. LNCS. Springer, Heidelberg, May 1996, pp. 33–48. DOI: 10.1007/3-540-68339-9_4 (cit. on p. 34).

[166] Jacques Patarin. “The oil and vinegar algorithm for signatures.” In: *Dagstuhl Workshop on Cryptography, 1997*. 1997 (cit. on pp. 34, 43).

[167] Jacques Patarin, Benoît Cogliati, Jean-Charles Faugère, Pierre-Alain Fouque, Louis Goubin, Robin Larrieu, Gilles Macario-Rat, and Brice Minaud. *VOX*. Tech. rep. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. National Institute of Standards and Technology, 2023 (cit. on pp. 4, 34).

[168] Jacques Patarin, Nicolas Courtois, and Louis Goubin. “QUARTZ, 128–Bit Long Digital Signatures.” In: *CT-RSA 2001*. Ed. by David Naccache. Vol. 2020. LNCS. Springer, Heidelberg, Apr. 2001, pp. 282–297. DOI: 10.1007/3-540-45353-9_21 (cit. on p. 34).

[169] Edoardo Persichetti and Paolo Santini. “A New Formulation of the Linear Equivalence Problem and Shorter LESS Signatures.” In: *ASIACRYPT 2023, Part VII*. Ed. by Jian Guo and Ron Steinfeld. Vol. 14444. LNCS. Springer, Heidelberg, Dec. 2023, pp. 351–378. DOI: 10.1007/978-981-99-8739-9_12 (cit. on p. 113).

[170] Albrecht Petzoldt, Enrico Thomae, Stanislav Bulygin, and Christopher Wolf. “Small Public Keys and Fast Verification for $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic Public Key Systems.” In: *CHES 2011*. Ed. by Bart Preneel and Tsuyoshi Takagi. Vol. 6917. LNCS. Springer, Heidelberg, Sept. 2011, pp. 475–490. DOI: 10.1007/978-3-642-23951-9_31 (cit. on p. 43).

[171] Aurélie Phesso and Jean-Pierre Tillich. “An Efficient Attack on a Code-Based Signature Scheme.” In: *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016*. Ed. by Tsuyoshi Takagi. Springer, Heidelberg, 2016, pp. 86–103. DOI: 10.1007/978-3-319-29360-8_7 (cit. on p. 34).

[172] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. *FALCON*. Tech. rep. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022. National Institute of Standards and Technology, 2022 (cit. on pp. 2, 19, 34, 38, 39, 80).

[173] Robert Ransom. *Constant-time verification for cut-and-choose-based signatures*. Cryptology ePrint Archive, Report 2020/1184. https://eprint.iacr.org/2020/1184. 2020 (cit. on p. 108).

[174] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.” In: *Communications of the Association for Computing Machinery* 21.2 (Feb. 1978), pp. 120–126. DOI: 10.1145/359340.359342 (cit. on pp. 1, 23, 33).

[175] Koichi Sakumoto, Taizo Shirai, and Harunaga Hiwatari. “On Provable Security of UOV and HFE Signature Schemes against Chosen-Message Attack.” In: *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011*. Ed. by Bo-Yin Yang. Springer, Heidelberg, Nov. 2011, pp. 68–82. DOI: 10.1007/978-3-642-25405-5_5 (cit. on pp. 23, 25, 32, 34, 43, 45, 84).

[176] Claus-Peter Schnorr. “Efficient Identification and Signatures for Smart Cards.” In: *CRYPTO’89*. Ed. by Gilles Brassard. Vol. 435. LNCS. Springer, Heidelberg, Aug. 1990, pp. 239–252. DOI: 10.1007/0-387-34805-0_22 (cit. on p. 1).

[177] Claus-Peter Schnorr. "Efficient Signature Generation by Smart Cards." In: *Journal of Cryptology* 4.3 (Jan. 1991), pp. 161–174. DOI: 10.1007/BF00196725 (cit. on p. 91).

[178] Travis L. Scholten, Carl J. Williams, Dustin Moody, Michele Mosca, William Hurley, William J. Zeng, Matthias Troyer, and Jay M. Gambetta. *Assessing the Benefits and Risks of Quantum Computers*. 2024. arXiv: 2401.16317 (cit. on p. 18).

[179] Adi Shamir. "IP=PSPACE." In: *31st FOCS*. IEEE Computer Society Press, Oct. 1990, pp. 11–15. DOI: 10.1109/FSCS.1990.89519 (cit. on p. 93).

[180] Peter W. Shor. "Algorithms for Quantum Computation: Discrete Logarithms and Factoring." In: *35th FOCS*. IEEE Computer Society Press, Nov. 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700 (cit. on p. 18).

[181] Victor Shoup. *Sequences of games: a tool for taming complexity in security proofs*. Cryptology ePrint Archive, Report 2004/332. https://eprint.iacr.org/2004/332. 2004 (cit. on p. 11).

[182] Damien Stehlé and Ron Steinfeld. "Making NTRU as Secure as Worst-Case Problems over Ideal Lattices." In: *EUROCRYPT 2011*. Ed. by Kenneth G. Paterson. Vol. 6632. LNCS. Springer, Heidelberg, May 2011, pp. 27–47. DOI: 10.1007/978-3-642-20465-4_4 (cit. on p. 34).

[183] Anton Stolbunov. "Cryptographic schemes based on isogenies." PhD thesis. Norwegian University of Science and Technology, 2012 (cit. on p. 107).

[184] Ewa Syta, Iulia Tamas, Dylan Visher, David Isaac Wolinsky, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ismail Khoffi, and Bryan Ford. "Keeping Authorities "Honest or Bust" with Decentralized Witness Cosigning." In: *2016 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2016, pp. 526–545. DOI: 10.1109/SP.2016.38 (cit. on p. 3).

[185] Gang Tang, Dung Hoang Duong, Antoine Joux, Thomas Plantard, Youming Qiao, and Willy Susilo. "Practical Post-Quantum Signature Schemes from Isomorphism Problems of Trilinear Forms." In: *EUROCRYPT 2022, Part III*. Ed. by Orr Dunkelman and Stefan Dziembowski. Vol. 13277. LNCS. Springer, Heidelberg, May 2022, pp. 582–612. DOI: 10.1007/978-3-031-07082-2_21 (cit. on pp. 4, 91, 115).

[186] Chengdong Tao, Albrecht Petzoldt, and Jintai Ding. "Efficient Key Recovery for All HFE Signature Variants." In: *CRYPTO 2021, Part I*. Ed. by Tal Malkin and Chris Peikert. Vol. 12825. LNCS. Virtual Event: Springer, Heidelberg, Aug. 2021, pp. 70–93. DOI: 10.1007/978-3-030-84242-0_4 (cit. on p. 34).

[187] Rodolfo Canto Torres and Nicolas Sendrier. "Analysis of Information Set Decoding for a Sub-linear Error Weight." In: *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016*. Ed. by Tsuyoshi Takagi. Springer, Heidelberg, 2016, pp. 144–161. DOI: 10.1007/978-3-319-29360-8_10 (cit. on p. 40).

[188] Dominique Unruh. "Computationally Binding Quantum Commitments." In: *EUROCRYPT 2016, Part II*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. LNCS. Springer, Heidelberg, May 2016, pp. 497–527. DOI: 10.1007/978-3-662-49896-5_18 (cit. on p. 100).

[189] Dominique Unruh. "Post-quantum Security of Fiat-Shamir." In: *ASIACRYPT 2017, Part I*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Vol. 10624. LNCS. Springer, Heidelberg, Dec. 2017, pp. 65–95. DOI: 10.1007/978-3-319-70694-8_3 (cit. on pp. 12, 100).

[190] Lih-Chung Wang, Chun-Yen Chou, Jintai Ding, Yen-Liang Kuan, Ming-Siou Li, Bo-Shu Tseng, Po-En Tseng, and Chia-Chun Wang. *SNOVA*. Tech. rep. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. National Institute of Standards and Technology, 2023 (cit. on pp. 4, 34).

[191] Zhipeng Wang and Qianhong Wu. "A Practical Lattice-Based Sequential Aggregate Signature." In: *ProvSec 2019*. Ed. by Ron Steinfeld and Tsz Hon Yuen. Vol. 11821. LNCS. Springer, Heidelberg, Oct. 2019, pp. 94–109. DOI: 10.1007/978-3-030-31919-9_6 (cit. on pp. 3, 80).

[192] Brent Waters and David J. Wu. "Batch Arguments for NP and More from Standard Bilinear Group Assumptions." In: *CRYPTO 2022, Part II*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Vol. 13508. LNCS. Springer, Heidelberg, Aug. 2022, pp. 433–463. DOI: 10.1007/978-3-031-15979-4_15 (cit. on p. 2).

[193] Takashi Yamakawa and Mark Zhandry. "Classical vs Quantum Random Oracles." In: *EUROCRYPT 2021, Part II*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12697. LNCS. Springer, Heidelberg, Oct. 2021, pp. 568–597. DOI: 10.1007/978-3-030-77886-6_20 (cit. on p. 29).

[194] Mark Zhandry. "Secure Identity-Based Encryption in the Quantum Random Oracle Model." In: *CRYPTO 2012*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Vol. 7417. LNCS. Springer, Heidelberg, Aug. 2012, pp. 758–775. DOI: 10.1007/978-3-642-32009-5_44 (cit. on pp. 12, 29).