



École polytechnique de Louvain
Institut ICTEAM

Privacy Enhancing Cryptographic Mechanisms with Public Verifiability

Thomas Peters

Thèse soutenue en vue de l'obtention du grade de
docteur en sciences de l'ingénieur

Membres du jury:

Pr. Olivier Pereira (UCL, Belgium) - promoteur
Dr. Benoît Libert (ENS Lyon, France)
Dr. David Pointcheval (ENS Paris, France)
Pr. Jean-Jacques Quisquater (UCL, Belgium)
Pr. François-Xavier Standaert (UCL, Belgium) - secrétaire
Dr. Bogdan Warinschi (Bristol University, UK)
Dr. Moti Yung (Columbia University & Google Inc., USA)
Pr. Christophe De Vleeschouwer (UCL, Belgium) - président

Louvain-la-Neuve, Belgique
2014

Preface

Cryptography is an intriguing discipline. It has grown from art to science, drawing strength in mathematics to protect its users and bringing solutions in concrete situations with inherent elegance. I had the chance to hear about cryptography for the first time at university from Jean-Jacques Quisquater, one of the founders of academic research in the field in the French-speaking world. I was first amazed to learn about things like the birthday paradox – several students among the 50-strong audience having the same birthday – and the fact that cryptography is capable of proving a statement in such a way that the elements of proof are not revealed and that “any theorem can be changed into a labyrinth.” The mix of mystery, paradox, mathematics and ingenuity was particularly attractive for a first year mathematical science student. I got immersed into the number theory underpinning the RSA cryptosystem but without yet realizing the full implication of that. At that time, I did not imagine doing a doctorate in cryptography. So much for naivety.

Acknowledgements

My first thanks go to my supervisor, Olivier Pereira. I would like to thank him for all the clear and useful advice I received from him and also for his warm welcome all the times I passed through his door, perhaps sometimes with too much enthusiasm. Unfortunately, I will not have enough time nor space to thank here all the other people who helped and encouraged me during these three and a half years. So I will simply say, thank you Olivier for having given me the opportunity to join the Crypto Group and for all the freedom of research you allowed me, without depriving me of your valuable guidance.

I must also thank Sylvie for her pleasant and unconditional (English) support. She may have already received many thanks from other people of the group in their theses, but she deserves it all. I also thank all the members of the Crypto Group that I had the opportunity to meet during my thesis. I think that we are very lucky to be able to enjoy such a nice working environment.

I must not forget to thank my co-authors. It was very enriching to collaborate together, learn different approaches and understand the various feelings. . . I also had the chance to present our work and to meet some other people who made the conferences even more enjoyable for me. Special thanks to Benoît, Iwen, Julien, Damien, . . . and apologies to those I forget.

Ending a thesis is a life accomplishment. I would certainly not be here without the support of my family, and more especially my mother and Bernard of course. My final special thanks go to two lovely people (the loveliest people in the world). Thank you, my little Nathalie, for having helped me every day, for believing in me and pushing me further. Thank you also, my little Hugo. You did not really help me write this thesis, but your presence simply makes my life beautiful and adds sense to everything I do.

*To my father
& for my son*

Contents

Preface	i
Acknowledgements	iii
I Preliminaries	3
1 Introduction	5
1.1 Modern Cryptography and Challenges	6
1.1.1 Provable Security	6
1.1.2 Public-Key Cryptography	7
1.1.3 Composability	7
1.1.4 Distributed Computing and Resilience	8
1.2 Motivation and Scope	8
1.2.1 Enhancing Privacy in Signatures	9
1.2.2 Enhancing Privacy in Commitments and Proofs	11
1.2.3 Enhancing Privacy in Encryption	13
1.3 The Core Equation	14
2 Cryptographic Background	15
2.1 Basic Concepts	16
2.1.1 Definitions and Generalities	16
2.1.2 Security Assumptions	19
2.1.3 Security Model	21
2.1.4 Secret Sharing	23
2.2 Signature Schemes	23
2.2.1 Definition and Security Notions	23

2.2.2	Instantiations	25
2.3	Encryption Schemes	28
2.3.1	Definitions and Security Notions	29
2.3.2	Additional Features	30
2.3.3	Instantiations	31
2.4	Commitment Schemes	33
2.4.1	Definition and Security Notions	33
2.4.2	Additional Features	35
2.4.3	Instantiations	37
2.4.4	Schemes with Improved Features	41
2.5	Cryptographic Proof Systems	42
2.5.1	Definition and Security Notions	42
2.5.2	Non Interactive Sigma Protocol	44
2.5.3	The Groth-Sahai Methodology	45
2.6	Advanced Tools	56
2.6.1	Tag-Based Encryption	56
2.6.2	Hierarchical Identity-Based Encryption	57
2.6.3	The NNL Framework for Broadcast Encryption	60
II	Signature-Based Primitives	63
3	Revocable Group Signatures	65
3.1	Definition and Security Notions	66
3.1.1	Definition of Group Signatures with Revocation	66
3.1.2	Security Model for Revocable Group Signatures	68
3.2	The First Scalable Group Signature with Revocation	72
3.2.1	Intuition and Technical Lemma	72
3.2.2	Scalable Construction	75
3.2.3	Security Analysis	80
3.3	Group Signature with Almost-for-free Revocation	87
3.3.1	Intuition behind our Scheme	88
3.3.2	Construction with Compact Keys	89
3.3.3	Security Analysis	95
3.3.4	Efficiency of this second Scheme	100

3.4	Efficiency Comparison and Comments	100
4	P-Homomorphic Signatures	103
4.1	Definition and Security Notions	104
4.1.1	Unified Framework	105
4.1.2	Variations on Context-Hiding	106
4.1.3	Separation Results	108
4.2	Subset Predicate Signatures	110
4.2.1	Intuition behind our Subset Predicate Signatures	111
4.2.2	Construction with Short Keys for Subset Predicates	112
4.2.3	Security of our Subset Predicate Signatures	115
4.3	Quotable Signatures	117
4.3.1	Intuition behind our Substring Predicate Signatures	117
4.3.2	Construction of Linear-Size Quotable Signatures	118
4.3.3	Security of our Substring Predicate Signatures	121
4.4	Linearly Homomorphic Signatures	123
4.4.1	Intuition behind our Subspace Predicate Signatures	124
4.4.2	A One-Time Linearly Homomorphic Construction	127
4.4.3	A Full-Fledged Construction for Linear Subspaces	129
4.4.4	A Completely Context-Hiding Construction	135
III	Commitment & Proof System Primitives	143
5	Non-Malleable Trapdoor Commitments to Group Elements	145
5.1	Definitions and Known Results	146
5.2	A Generic Constant-Size Construction	148
5.2.1	Intuition of the Generic Transformation	149
5.2.2	Template of Linearly Homomorphic SPS Scheme	149
5.2.3	Construction of Simulation-Sound Structure-Preserving Trapdoor Commitments	150
5.2.4	Security Proof of the Generic Transformation	152
6	Quasi-Adaptive Non Interactive Zero Knowledge Proofs	155
6.1	Definition and Security Notions	156
6.1.1	Quasi-Adaptive NIZK Proofs	157

6.1.2	Simulation Soundness and Relative Soundness	158
6.1.3	Quasi-Adaptive Language for Linear Subspaces	159
6.2	Unbounded Simulation-Sound Arguments	159
6.2.1	Realization for Linear Subspace Membership	159
6.2.2	Proof of the Simulation-Sound Argument	162
6.3	Single-Theorem Relatively Sound Arguments	167
6.3.1	Single-Theorem Linear Subspace Membership Proof	167
6.3.2	Security of the Relatively Sound Argument	168
6.4	Extensions for the k -Linear Assumption	172
6.5	Comparison of Linear Subspace Arguments	174
IV	Encryption-Based Primitives	175
7	Publicly Verifiable Threshold Encryptions	177
7.1	Definition and Security Notions	178
7.1.1	Threshold Keyed-Homomorphic Encryptions	178
7.2	Adaptively Secure Threshold Cryptosystems	180
7.2.1	Intuition behind our Publicly-Verifiable Scheme	180
7.2.2	DLIN-Based CCA Construction	181
7.2.3	Efficiency and k -Linear Extention.	183
7.2.4	Security of the CCA Threshold Cryptosystem	184
7.3	Keyed Homomorphic Threshold Cryptosystems	189
7.3.1	Intuition behind our Keyed-Homomorphic Scheme	189
7.3.2	DLIN-Based KH-CCA Construction	190
7.3.3	Security of the Keyed-Homomorphic cryptosystem	193
	Conclusion	201
	References	216

Only the key should be secret
[Auguste Kerckhoffs, 1883]

Part I

Preliminaries

Chapter 1

Introduction

Contents

1.1 Modern Cryptography and Challenges	6
1.1.1 Provable Security	6
1.1.2 Public-Key Cryptography	7
1.1.3 Composability	7
1.1.4 Distributed Computing and Resilience	8
1.2 Motivation and Scope	8
1.2.1 Enhancing Privacy in Signatures	9
1.2.2 Enhancing Privacy in Commitments and Proofs . .	11
1.2.3 Enhancing Privacy in Encryption	13
1.3 The Core Equation	14

Technology is linking the slightest of our actions to the virtual world. It brings together IT resource-poor users and groups of people or institutions that may be unknown or whose intentions are. In nowadays' hyper-connected life, where we may be online even without knowing it, it is no longer sufficient to protect oneself from the outside environment as the ennemy could potentially lie within the entity we are interfacing with.

Since behind computers there are humans, we believe that there is no interest in a new technology breakthrough if the price to pay is a dilution of the rights to privacy, integrity or anonymity. The users are not always aware of their vulnerability while engaging into electronic exchanges – sometimes unbeknownst to them. The cryptographer's mission is to put forth concrete solutions to protect all these essential aspects.

In this context where the unequal balance of power is exploited at any level, our work proposes such mechanisms to enhance the privacy in three of the most fundamental cryptographic primitives, namely, digital signatures, encryption schemes and zero-knowledge proof systems. More precisely, we either extend their ordinary building blocks with necessary functionalities for today's application so as to achieve resiliency in the sense of distributed computing or

improve their security models by taking into account realistic threat – or even both.

Furthermore, without compromising on the security level regarding this requirement, this thesis focuses on making the verification mechanisms both accessible and effective so that they can be actually used in practice and thus force the honesty of all participants. In these three main areas we then build publicly verifiable advanced protocols while proposing the best performance to date. Efficiency is central when claiming an interest in a user-centric approach and parts of our constructions are even (asymptotically) optimal.

The contribution of the thesis is to offer forms of security that can be guaranteed both mathematically – by reduction to computational hard problems – and structurally, identify their robustness in time and control the effect of potential flaws in order to re-balance forces virtually.

1.1 Modern Cryptography and Challenges

Modern cryptography transformed the art of secret into science. What distinguishes today’s methodology from previous techniques is the fact that cryptography now relies on the reductionist paradigm. This formal framework provides the discipline with the necessary rigour to *prove* security. In that sense, this thesis is part of *provable security*.

1.1.1 Provable Security

Guaranteeing security is made possible thanks to the so-called *proofs by reduction*. The reduction principle is as follows: First we should ponder which features we want, and from which risk we need to protect ourselves. That means we have to formally define the objectives of the cryptographic primitive (if we are to achieve it) and model the potential failures we want to avoid regarding their use, *i.e.* the power of the adversaries posing a threat. Secondly, we need to clearly identify the computational hard problems on which we could rely in order to guarantee the security goals while building schemes. The assumptions that these problems are indeed efficiently intractable should be widely believed. Finally, the scheme is proven secure if we can show that any well-defined but general adversary in the security model can be “easily” adapted to a solver of one of the underlying hard problems. We say that the security of the scheme in the considered model is reduced to solving this problem. Consequently, no real-world attack fitting the model could exist without contradicting the assumption that the considered problem is hard.

Understanding the different levels of security of a given cryptographic primitive is fundamental since a proof by reduction of a scheme (realizing the functionality of the primitive) says nothing about an adversary out of the model.

1.1.2 Public-Key Cryptography

The science of cryptography consists in *building* secure schemes. While a single approach exists for proving computational security, cryptography is divided into two subfields.

On the one hand there is *secret-key cryptography* where the entities should share a common private key before being able to securely communicate. On the other hand there is *public-key cryptography* where each entity makes available to the others a public key while retaining exclusive possession of the associated secret key. The public key alone is necessary to engage in secure communication with such an entity.

These two categories are sometimes referred to as symmetric cryptography and asymmetric cryptography to reflect the fact that the same key is/is not used on both sides of the communication. This thesis deals with the second category and secret-key cryptography will no longer be mentioned here.

In almost 40 years of public-key cryptographic research, with initial advances going back to the publications of Diffie-Hellman [58] first and Rivest-Shamir-Adleman [127] later, a fair number of cryptographic primitives and secure protocols have been developed. In the beginning, the objective was simply to protect oneself against the rest of the world. Communications through *private and authenticated channels* were then made possible thanks to encryption-signatures. We clearly have valuable standards for that purpose with OAEP-RSA, DSA,... With the evolution of IT and the advent of Internet, the available solutions were no longer able to secure exchanges that became more exposed in nature. Previously, it was necessary to be protected from the network. Nowadays we are connected to it nearly all the time and we are engaging in more complex procedures. Consequently we are potentially more vulnerable to cyber-threats.

1.1.3 Composability

The number of players within a protocol and the variability of their roles increase its potential exposure to attacks against which it needs to be guarded. We know that a new kind of protocol needs an overall security approach, its own security model. In order to secure protocols which use several functionalities of different primitives, beyond the usual signature schemes and encryption schemes, the juxtaposition of secure solutions is not sufficient. Public key cryptography in general and this thesis in particular deal with such composability problems.

Composability is an important cryptographic design notion for building systems and protocols. Inside protocols, cryptographic tools need to compose well with each other in order to be used in combination. Structure-preserving cryptography, in turn, is a recent paradigm that takes care of composing algebraic tools, and primarily within groups supporting bilinear maps – *a.k.a.* pairings – to allow smooth composition with efficient known proof systems. The notion

allows for modular and simplified designs of various cryptographic protocols and primitives.

In this thesis we propose new *pairing-based* mechanisms from structure-preserving schemes. Some of our solutions keep this feature and we also show that new combinations lead to very efficient schemes with high levels of privacy.

1.1.4 Distributed Computing and Resilience

While ordinary encryption or signature schemes only allow pairwise communications, nowadays computers, mobile phones and a lot of other connected electronic devices are able to delegate or run algorithms on confidential or authenticated data that are subsequently processed by other protocols very often managed by more powerful servers.

The terms “distributed computing” or “distributed protocol” are often used to describe systems where data and computations are delegated or shared, thereby necessitating or authorizing the intervention of multiple players to execute a predetermined task. This thesis discusses to some extent group signatures, P-homomorphic signatures that authorize some form of malleability and threshold cryptosystems. All these primitives are fully included in this field of distributed protocols as detailed in the next section.

However, as shown by recent reports, the confidence given to service providers – as protocol managers – should not be unconditional or automatic. What can be reasonably expected from security when the parties involved in the process are themselves dishonest and liable to collude against a single user?

When the secret key associated to the public key of a secure encryption scheme or signature scheme is stolen, nothing can be made to “patch” the system except its complete resetting from scratch with collateral problems linked to identity and certificates. The situation is even worse for distributed and hierarchical protocols where users should not be affected by a failure occurring elsewhere in a subroutine.

Resilience is the ability of a system to be fault-tolerant. It measures how robust the system is in order to preserve the security of its different functionalities when faced with a precise threat. The security of the above cryptographic primitives is defined in that sense.

1.2 Motivation and Scope

This thesis aims at improving the security in some of the most fundamental aspects of cryptography through new privacy enhancing mechanisms providing a public verifiability.

By public verifiability we mean that the public parameters of our cryptographic schemes are sufficient to check the validity of the outputs of some

algorithms. All our schemes feature such a form of verification even though it concerns different algorithms.

It is to be noted that verifiability is strongly linked to security since we only consider as valid those results that have been obtained honestly. Verifiability provides proof that the procedure has been correctly followed and should not be liable to falsification. On the other hand, the elements provided by verifiability in order to achieve verification should not diminish the security of other properties of the scheme.

In spite of this, all our constructions are equipped with proofs in the *standard model* and in adaptive security models that we have strengthened or for which we have been first in providing solutions. It is also worth mentioning that all our constructions are based on the existence of efficient cryptographic bilinear maps. Consequently, our realizations fall into the so-called field of pairing-based cryptography.

Our research made it possible to offer robust and effective solutions to enhance the privacy of cryptographic primitives. In these achievements, we have privileged a user-centric vision while guaranteeing the highest security level to users. Our contributions in the various fields described below are optimal and/or state-of-the-art.

1.2.1 Enhancing Privacy in Signatures

Digital signature schemes allow a signer who has established a public key $\mathbf{pk}$ to sign a message in such a way that any other party can verify that the message originated from him/her and has not been modified in any way.

Revocable Group Signatures

In Chapter 3, our first contributions deal with the primitive of *group signature schemes*. This notion was originally introduced by Chaum and van Heyst in [46].

MOTIVATION. This approach relies on a group manager, who distributes credentials to a group of signers, who in turn can sign documents on behalf of the group. All individual signers are guaranteed to remain anonymous, unless an opening authority decides to trace them. While this primitive is highly desirable in various contexts (resilient signature services, anonymous digital credentials and keycard access...), all known constructions remained unusable despite 20 years of research, due to the lack of efficient revocation mechanisms enabling group managers to add and *remove* group members, depending on the needs or on the compromise of some network components.

CONTRIBUTIONS. We propose the first constructions of group signature with a scalable revocation mechanism, that is, all operations have a cost that is at most polylogarithmic in the group size. Furthermore, our solution is optimal in various respects: for instance, the size of signatures and the verification

cost are independent of the number of revocations and the maximal cardinality of the group. Relying on a stronger assumption, we also provide solutions with constant-size secret keys. This result stands in strong contrast with the previous state-of-the-art which only proposed solutions with public key size of linear complexity and makes our construction competitive with group signatures without revocation in the standard model.

Part of these contributions were published in

Scalable Group Signatures with Revocation [109]
Group Signatures with Almost-for-Free Revocation [108]

Malleability in Signatures

In Chapter 4, we improve the model for *computing on authenticated data* proposed by Ahn *et al.* [6] and propose efficient structure-preserving realizations.

MOTIVATION. While unforgeability of ordinary signatures prevent them from being modified, signature schemes allowing some kind of authorized malleability have shown their utility in different contexts. The recent concept of *P-Homomorphic Signatures* is the cryptographic primitive unifying a wide range of signatures that support such *public* modifications. In this general framework, where validity of signatures is only preserved for messages that are derivable according to the predicate *P* from other signed ones, a special privacy notion might be required. This new security property is named *context hiding* and it ensures that nobody can tell whether a signature is freshly generated or derived from existing ones. While the *strong context-hiding property* even requires statistical indistinguishability, we show concrete situations where this definition fails to prevent efficient attacks.

CONTRIBUTIONS. In the framework of *P-Homomorphic Signatures*, we first improved the context-hiding privacy notions. We start with a context-hiding notion that deals with a computational indistinguishability for adversarially-chosen signatures. We prove that this notion is incomparable with the strong context-hiding property. While previous definitions only take into account honestly-generated signatures, here, the adversary is only restricted to provide valid signatures no matter how he/she computes it. This modification is justified by the existence of randomizable signatures. Then, we gradually strengthen our models up to the *complete context-hiding* property which also implies the strong definition of Ahn *et al.*. Secondly, we build efficient structure-preserving *P*-homomorphic signatures for subset predicates, quotable predicates and linear subspace predicates that all satisfy our highest level of privacy. Moreover, the resulting subset signature scheme and quotable signature schemes enjoy performances that have not been achieved yet in the standard model even for a relaxed level of context hiding. Also, our linearly homomorphic signature scheme demonstrates unexpected applications so far. Some of these applications are detailed below.

Part of these contributions were published in

**Computing on Authenticated Data: New Privacy Definitions
and Constructions** [10]

**Efficient Completely Context-Hiding Quotable and Linearly
Homomorphic Signatures** [11]

**Linearly Homomorphic Structure-Preserving Signatures and
Their Applications** [106]

1.2.2 Enhancing Privacy in Commitments and Proofs

Commitment schemes allow any party who knows the public parameters of a scheme to commit to some values that he/she can reveal later on. Nobody else seeing the commitment is able to find out information on the hidden committed value, *a.k.a.* the *hiding property*. In turn, the committer cannot change his/her mind when he/she reveals the value, *a.k.a.* the *binding property*. Commitment schemes are often associated with zero-knowledge proof systems. The latter allows a prover to efficiently convince a verifier that a given statement is true without leaking any other information.

Usually, these primitives are used to improve other cryptographic building blocks and their security. Therefore, in order to enhance the privacy of efficient protocols in practice, they should not have to increase too much the overall complexity. In that sense, our contribution to these primitives enables their use in practice.

Non-Malleable Commitments

In Chapter 5, we put forward the first short non-malleable commitments on vectors of group elements while keeping the ability of proving statements relative to their openings.

MOTIVATION. Simulation-sound trapdoor commitments (SSTC) are commitments schemes where a trapdoor key allows equivocating openings. This means that the owner of this key is able to get around the binding property by computing “fake” – though valid – openings. However the binding property must be unchanged for other parties even if they see equivocations; this is the simulation soundness. As for the applications of SSTC schemes, they can be used to build universally composable zero-knowledge proof. Furthermore, known techniques [73, 75] also enable a generic transformation of SSTC into non-malleable commitments – with respect to opening. Non-malleable (NM) commitments schemes generate commitments that cannot be used to compute new commitments whose openings are in relation with unknown committed values. This feature is clearly mandatory for on-line auctions as well as other real situations. As far as composability is concerned, structure-preserving commitments to vectors of group elements cannot be length-reducing [5] and neither is the only NM commitment scheme [64] whose openings only consist of group elements.

CONTRIBUTIONS. We build the first *constant-size* (semi-) structure-preserving simulation-sound trapdoor commitments for vectors of group elements in $\mathbb{G}$, for a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. We bypass the impossibility result of [5] by allowing the elements of the commitments to lie in the target group $\mathbb{G}_T$, hence the word “semi”. Then, we obtain the first *constant-size* non-malleable commitments enjoying the same features by slightly adapting the generic transformation and by relying on a one-time structure-preserving signature. Finally we generalize our construction to provide a novel generic transformation from any structure-preserving linearly homomorphic signatures into non-malleable commitments.

Part of these contributions were published in

Linearly Homomorphic Structure-Preserving Signatures and Their Applications [106]

Simulation-Sound Quasi-Adaptive Arguments

In Chapter 6, we build a constant-size arguments for membership in a spanned vector space where the common reference string may depend on the given vector.

MOTIVATION. Zero-knowledge proof systems [19] allow a prover who has some witness to convince a verifier that a given string belongs to a predefined language. Additionally, the proof should leak no information about the witness. If the string is indeed in the language, the statement is true and if the prover accepts the proof, the proof is valid. These systems need to satisfy completeness and soundness. This respectively means that if the statement is true, the prover is always able to compute a valid proof and if the statement is false it is unfeasible for the prover to compute a valid proof. A non-interactive zero-knowledge (NIZK) proof simply means that the prover does not need the verifier to perform a proof which is publicly verifiable. Non-interactive proofs can be efficiently computed from a common reference string (CRS) for relations in groups equipped with a bilinear map. In the quasi-adaptive model, the CRS may depend on the specific language. Despite this relaxation, it is a strong enough model to achieve adaptive security for encryption schemes and it gives the advantage of providing smaller proofs. Indeed, in this system, proofs may support a one-time simulation-sound property, where a trapdoor key makes it possible to prove false statements that are accepted by provers. The property says that even if someone else sees one simulated proof he/she is still not able to break the soundness of the system. The state-of-the art solutions do not seem to readily extend to situations where several simulated proofs are needed except by a huge amount of additional computations.

CONTRIBUTIONS. For the case of proving that a vector belongs to a linear subspace generated by some given vectors, we build quasi-adaptive proof systems where the size of the proof is independent of the dimension of the linear subspace and of the size of the vectors. First we achieve this result for the general case of unbounded simulation soundness under a simple assumption. Secondly,

we further decrease the constant-size number of elements in the proof for the particular case of relative simulation soundness where only one simulated proof is given in the security model. Linear subspace membership proofs are of great interest since they make it possible to prove several linear algebraic relations as shown below.

Part of these contributions will be published in

**Non-Malleability from Malleability: Simulation-Sound
Quasi-Adaptive NIZK Proofs and CCA2-Secure
Encryption from Homomorphic Signatures [107]**

1.2.3 Enhancing Privacy in Encryption

Public-key encryption schemes allow any party to encrypt a message under the public key pk established by the designated receiver in such a way that no other party can tell anything about the encrypted message. Only the receiver can decrypt the ciphertext to recover the message using his/her secret key.

Verifiable Threshold Encryptions

In Chapter 7, we concentrate on encryption schemes allowing any party to verify that ciphertexts are well-formed and their decryption needs the collaboration of a number of receivers exceeding a given threshold among those who have a secret key share.

MOTIVATION. Adaptive security of threshold encryptions requires that the privacy of the encrypted messages is not altered when an adversary is able to corrupt part of the secret-key share owners on-the-fly as long as their number does not exceed the threshold. Adaptively-secure cryptosystems against chosen-ciphertext attacks (CCA) further require that the scheme remains secure if the adversary can also obtain decryption of other ciphertexts of his/her choice. While this level of privacy may seem quite strong such schemes have already been designed and are mandatory in some real-world situations. In particular, these schemes are non-malleable, which is an important feature when integrity is concerned. Nevertheless, last year Emura *et al.* proposed some reconciliation with homomorphic properties supported by the encryption primitive of *keyed-homomorphic* CCA encryptions where homomorphic operations can only be carried out using a dedicated evaluation key. Unfortunately their schemes only satisfy a relaxed notion of their stronger model and these do not readily extend to public verifiability.

CONTRIBUTIONS. As applications of our quasi-adaptive proof systems described in Chapter 6, we first build a fully secure publicly verifiable threshold encryption scheme using our quasi-adaptive proof supporting a one-time form of simulation-soundness since it is sufficient to perform the security reduction. This new technique leads to the most efficient scheme to date for that primitive with its 8-group elements and also enjoys the property of being

structure-preserving. In order to build the first fully-secure keyed homomorphic encryptions while also providing a publicly verifiable mechanism to check the validity of the ciphertexts and threshold decryption – which are both necessary in electronic voting schemes – we rely on our unbounded simulation-sound quasi-adaptive arguments. Indeed, post-challenge evaluation queries involving the challenge ciphertext need to be simulated in the reduction and the returned ciphertexts should not help the adversary to force decryption of invalid ciphertext.

Part of these contributions will be published in

**Non-Malleability from Malleability: Simulation-Sound
Quasi-Adaptive NIZK Proofs and CCA2-Secure
Encryption from Homomorphic Signatures [107]**

1.3 The Core Equation

All along the thesis, a single relation underlies all the constructions. For a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ defined over groups $\mathbb{G}, \mathbb{G}_T$ whose order is a same “huge” prime p , here is the relation you should find again and again:

$$1_{\mathbb{G}_T} = e(G_z, Z) \cdot e(G_r, R) \qquad 1_{\mathbb{G}_T} = e(H_z, Z) \cdot e(H_u, U)$$

Informally, for given parameters G_z, H_z, G_r, H_u , the security of all the presented schemes supposes that you are not able to compute a non-trivial triple (Z, R, U) satisfying the above relation.

Chapter 2

Cryptographic Background

Contents

2.1 Basic Concepts	16
2.1.1 Definitions and Generalities	16
2.1.2 Security Assumptions	19
2.1.3 Security Model	21
2.1.4 Secret Sharing	23
2.2 Signature Schemes	23
2.2.1 Definition and Security Notions	23
2.2.2 Instantiations	25
2.3 Encryption Schemes	28
2.3.1 Definitions and Security Notions	29
2.3.2 Additional Features	30
2.3.3 Instantiations	31
2.4 Commitment Schemes	33
2.4.1 Definition and Security Notions	33
2.4.2 Additional Features	35
2.4.3 Instantiations	37
2.4.4 Schemes with Improved Features	41
2.5 Cryptographic Proof Systems	42
2.5.1 Definition and Security Notions	42
2.5.2 Non Interactive Sigma Protocol	44
2.5.3 The Groth-Sahai Methodology	45
2.6 Advanced Tools	56
2.6.1 Tag-Based Encryption	56
2.6.2 Hierarchical Identity-Based Encryption	57
2.6.3 The NNL Framework for Broadcast Encryption	60

2.1 Basic Concepts

This section recalls some fundamental notions of public-key cryptography.

2.1.1 Definitions and Generalities

An algorithm can be seen as a finite set of instructions that precisely defines a sequence of operations in order to specify the computational process to evaluate a given function. This process can be represented by a Turing machine.

A PPT algorithm is the shortened form of a probabilistic algorithm that runs in polynomial time:

- An algorithm whose input length is at least $\lambda \in \mathbb{N}$ is *efficient* if it runs in polynomial-time in λ . That means that the number of basic operations needed during the execution of the algorithm is bounded by $p(\lambda)$ for some polynomial p .
- When the random coins r of a probabilistic algorithm Alg running on inputs x should be specified we write $\text{Alg}[x; r]$ using a semicolon.

Negligible functions are positive functions that decrease faster than the inverse of any polynomial:

Definition 1 (Negligible Function) *A function f is negligible if for every polynomial $p(\cdot)$ there exists an N such that for all integers $n \geq N$ it holds that $f(n) \leq \frac{1}{|p(n)|}$*

A computational problem is said to be *hard* or *intractable* if there is no efficient algorithm solving it with non-negligible probability. More precisely, for random instances of the problem of input length λ , efficiency and probability are measured in term of λ . Sometimes it may even be hard to check if an output is a solution. In general we may be interested in *decisional* problems.

Definition 2 (Indistinguishability) *Given two probabilistic algorithms $U = (U_\lambda)_\lambda$ and $V = (V_\lambda)_\lambda$ (or families of finite distributions), U and V are computationally indistinguishable, noted $U \approx V$, if for any PPT algorithm $\mathcal{D}$ the following holds: for any $\lambda \in \mathbb{N}$ there exists a negligible function negl such that*

$$|\Pr[1 \leftarrow \mathcal{D}^U(\lambda)] - \Pr[1 \leftarrow \mathcal{D}^V(\lambda)]| \leq \text{negl}(\lambda), \quad (2.1)$$

where $\mathcal{D}^U$ (resp. $\mathcal{D}^V$) means that $\mathcal{D}$ has an oracle access to U (resp. V), i.e., for given parameter λ , $\mathcal{D}$ may query samples according to the probability distribution defined by U_λ (resp. V_λ). If equation (2.1) still holds for any unbounded distinguisher $\mathcal{D}$, then U and V are statistically indistinguishable, and even perfectly indistinguishable if $\text{negl}(\lambda) = 0$.

Perfect indistinguishability means that whichever way U and V generate their outputs, from possibly different inputs, they occur with *exactly* the same probability. From a distribution standpoint, statistical indistinguishability means that U and V are statistically closed. Indeed, in that case the above definition is equivalent to requiring

$$\sum_{s \in S_\lambda} |\Pr[s \leftarrow U_\lambda] - \Pr[s \leftarrow V_\lambda]| \leq \text{negl}'(\lambda),$$

where S_λ is the common finite sample set, with $|S_\lambda| \geq 2^\lambda$. By extension, we will say that U and V are computationally closed in the case of computational indistinguishability.

As an example, a family of pseudo-random functions (PRF) are efficient algorithms whose outputs look like “real” uniform random numbers in the sense that it is hard to distinguish if a set of outputs has been generated by the PRF or is “truly” random: PRFs and random functions are computationally indistinguishable. More exactly, a PRF family U defines for every λ a distribution of a polynomial-size set of functions so that given an oracle access to $f \leftarrow U_\lambda$ whose output length is λ it is hard to distinguish it from V_λ which returns equally-likely strings from $\{0, 1\}^\lambda$.

One-Way Functions

Informally, a function f is one-way if it is easy to compute but hard to invert.

Definition 3 (One-Way Function) *Let λ be a security parameter, then a function $f_\lambda : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda$ is one-way if, (i) there exists an efficient algorithm that on any input $x \in \{0, 1\}^\lambda$ returns $f(x)$, (ii) for all PPT adversary A , there exists a negligible function negl such that*

$$\Pr[x \xleftarrow{\$} \{0, 1\}^\lambda; y = f(x); x' \leftarrow A(y) : f(x') = y] \leq \text{negl}(\lambda).$$

Hash Functions

Informally, a hash function takes in an element of $\{0, 1\}^*$ and efficiently returns a fixed-length bit string.

Definition 4 (Hash Function Family) *Let λ be a security parameter, then a family $\mathcal{H}$ of hash functions parametrized by λ samples a hash function $H : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$ according to an internal distribution.*

While the deterministic function H cannot be injective it should be hard to compute pre-images or to guess images without computing H . We recall two flavours of this property.

Definition 5 (Universal One-Way) A family $\mathcal{H}$ of hash functions is universal one-way if for any PPT adversary $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr[x \leftarrow \mathcal{A}(\mathcal{H}); H \leftarrow \mathcal{H}(\lambda); x' \leftarrow \mathcal{A}(H) : x' \neq x \wedge H(x') = H(x)] \leq \text{negl}(\lambda).$$

The following stronger notion requires the same intractability even if the adversary may choose both x, x' after receiving the hash function.

Definition 6 (Collision Resistant) A family $\mathcal{H}$ of hash functions is collision resistant if for any PPT adversary $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr[H \leftarrow \mathcal{H}(\lambda); (x, x') \leftarrow \mathcal{A}(H) : x' \neq x \wedge H(x') = H(x)] \leq \text{negl}(\lambda).$$

Bilinear Groups

A cyclic group $(p, \mathbb{G}, g)$ is a group $\mathbb{G}$ entirely generated by g where $g^p = 1_{\mathbb{G}}$.

Definition 7 (Bilinear Group) A tuple $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ is a bilinear group where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of prime order p , generated respectively by g_1, g_2 and $e(g_1, g_2)$, if $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a non-degenerated bilinear form: for all $X \in \mathbb{G}_1$, for all $Y \in \mathbb{G}_2$, for all $a, b \in \mathbb{Z}_p$,

$$e(X^a, Y^b) = e(X, Y)^{ab}. \quad (2.2)$$

For a security parameter λ , it is assumed that bilinear groups are efficiently samplable so that $p > 2^\lambda$. Mainly, there are three types of elliptic-curve instantiations [71]:

Type 1: where $\mathbb{G}_1 = \mathbb{G}_2$ and $g_1 = g_2$. We usually refer to Type-1 instances as symmetric pairings. We denote by $(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \Lambda(\lambda)$ the generation of this setting.

Type 2: where an additional efficient homomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ is given but none exists from $\mathbb{G}_1$ to $\mathbb{G}_2$.

Type 3: where presumably no efficient homomorphism exists in either way.

Type-3 elliptic curves have the most efficient instantiations and admit a smaller representation of $\mathbb{G}_1$ -elements than those of $\mathbb{G}_2$ -elements. At a same bit-security level, $\mathbb{G}$ -elements of type-1 elliptic curves have an intermediate size relatively to type-3 curves. In the thesis we will only use type-1 curves because of their symmetric property.

2.1.2 Security Assumptions

All the schemes proposed in the thesis have their security based on one or several of the following assumptions. To simplify their descriptions we will say “a problem is hard in a group $\mathbb{G}$ ” for “a problem is hard relatively to the generation of $\mathbb{G}$ ”, which means that the probability to efficiently solve the problem is negligible in the security parameter λ where the random coins are taken over the distribution of the λ -bit length instance of the problem and the distribution that generates the group $\mathbb{G}$ whose cardinality is at least 2^λ . In symmetric bilinear groups, the latter distribution is that of $\Lambda(\lambda)$ such that $(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \Lambda(\lambda)$.

As a warm-up, we start with the weakest assumption of the thesis. Breaking this assumption means breaking all the other ones since the underlying problem is the hardest to solve. For a set S , $s \xleftarrow{\$} S$ means that s is equally-likely sampled from S .

Assumption 1 (DLOG) *The Discrete Logarithm Problem (DLOG) in a cyclic group $(p, \mathbb{G}, g)$, is to compute $a \in \mathbb{Z}_p$ such that $h = g^a$ for some $h \xleftarrow{\$} \mathbb{G}$. The **Discrete Logarithm Assumption** asserts that the DLOG problem is hard in $\mathbb{G}$.*

Under the DLOG assumption in $\mathbb{G}$, the exponentiation is a one-way function in the sense of Definition 3.

Assumption 2 (CDH) *The Computational Diffie-Hellman Problem (CDH) in a cyclic group $(p, \mathbb{G}, g)$ is, given (g, g^a, g^b) , for some $a, b \xleftarrow{\$} \mathbb{Z}_p^*$, to find $g^{ab} \in \mathbb{G}$. The **Computational Diffie-Hellman Assumption** asserts that the CDH problem is hard in $\mathbb{G}$.*

Sometimes, reductions find it difficult to compute a CDH solution. However, in some cases the challenger can extract a relaxed solution which motivates the following assumption.

Assumption 3 (Flex-CDH [101]) *The Flexible Diffie-Hellman Assumption (Flex-CDH) in $\mathbb{G}$ asserts the hardness of finding a non-trivial triple $(g^\mu, g^{a \cdot \mu}, g^{ab \cdot \mu})$, for some non-zero $\mu \in \mathbb{Z}_p^*$, given $(g, g^a, g^b) \xleftarrow{\$} \mathbb{G}$.*

For indistinguishability-based security we have to rely on hard decision problems. A first decisional assumption is the DDH assumption where it should be hard to verify whether a group element is a CDH solution.

Assumption 4 (DDH) *The Decision Diffie-Hellman Problem (DDH) in a cyclic group $(p, \mathbb{G}, g)$, is to distinguish the following two distributions (g, g^a, g^b, g^{ab}) and (g, g^a, g^b, g^z) , with $a, b \xleftarrow{\$} \mathbb{Z}_p^*$, $z \xleftarrow{\$} \mathbb{Z}_p^*$. The **Decision Diffie-Hellman Assumption** asserts that DDH is hard in $\mathbb{G}$.*

The DDH assumption in $(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \Lambda(\lambda)$ does not hold. Indeed, it is even easy to solve *any* instance of the problem since, given $(g, h, f, Z) \in \mathbb{G}^4$, for some $h, f \xleftarrow{\$} \mathbb{G}$, checking whether $e(g, Z) \stackrel{?}{=} e(h, f)$ is efficient.

In symmetric pairings the DLIN assumption seems to be the weakest decisional assumption on which security of efficient schemes may rely.

Assumption 5 (DLIN [24]) *The Decision Linear Problem (DLIN) in a cyclic group $(p, \mathbb{G}, g)$, is to distinguish the following two distributions $(g^a, g^b, g^{ac}, g^{bd}, g^{c+d})$ and $(g^a, g^b, g^{ac}, g^{bd}, g^z)$, with $a, b, c, d \xleftarrow{\$} \mathbb{Z}_p^*$, $z \xleftarrow{\$} \mathbb{Z}_p^*$. The **Decision Linear Assumption** is the intractability of DLIN.*

Equivalently, for random generators $(g, h, f) \leftarrow \mathbb{G}^3$ and random scalars $(c, d) \leftarrow \mathbb{Z}_p^*$, it is hard to check whether an element $Z \in \mathbb{G}$ satisfies $(h^c, f^d, Z) \in \text{span}\langle (h, 1, g), (1, f, g) \rangle$ for unknown (c, d) , where span is the notation for the linear span.

Assumption 6 (DBP [3]) *The Double Pairing Problem (DBP) in a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ is, given $g_z, g_r \xleftarrow{\$} \mathbb{G}_1^2$, to find $(z, r) \in \mathbb{G}_2^2$ satisfying $1_{\mathbb{G}_T} = e(g_z, z)e(g_r, r)$. The **Double Pairing Assumption** asserts that the DBP problem is hard in $\mathbb{G}$.*

In symmetric pairings, it is easy to compute a solution of any DBP instances. Then, in type-1 pairings, we must not rely on the above assumption. Instead, we may use another hardness assumption, called SDP, which can be seen as the analogue of DBP in the symmetric case. This assumption is implied [45] by the DLIN assumption.

Assumption 7 (SDP [45]) *The Simultaneous Double Pairing Problem (SDP) in a symmetric bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \Lambda(\lambda)$ is, given $g_z, g_r, h_z, h_u \xleftarrow{\$} \mathbb{G}^4$, to find $(z, r, u) \in \mathbb{G}^3$ satisfying*

$$1_{\mathbb{G}_T} = e(g_z, z)e(g_r, r), \quad 1_{\mathbb{G}_T} = e(h_z, z)e(h_u, u). \quad (2.3)$$

The Simultaneous Double Pairing Assumption asserts that the SDP problem is hard in $\mathbb{G}$.

q -type assumptions We also rely on assumptions that remain true if the challenger is allowed to see a multiple of a polynomial number q of elements related to the solutions of the underlying problem. Sometimes such hard problems are called “one-more” problems when the goal is to find a new solution from given q solutions. The length of the assumption is then a function of that q -value, which makes it harder to challenge.

Assumption 8 (q -SDH [21]) *The q -Strong Diffie-Hellman problem (q -SDH) in a cyclic group $(p, \mathbb{G}, g)$ is, given $(g, g^a, \dots, g^{(a^q)})$, for some $a \xleftarrow{\$} \mathbb{Z}_p$, to find a pair $(g^{1/(a+s)}, s) \in \mathbb{G} \times \mathbb{Z}_p$. The **q -Strong Diffie-Hellman Assumption** asserts that the q -SDH problem is hard in $\mathbb{G}$.*

The following assumption has been used to prove the security of a constant-size structure-preserving signature scheme that allows signing vectors of group elements. We assume this assumption to hold since some of our schemes use this signature scheme as a building block.

Assumption 9 (q -SFP [4]) *The q -Simultaneous Flexible Pairing Problem (q -SFP) in a symmetric bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ is, given $g_z, h_z, g_r, h_r, a, \tilde{a}, b, \tilde{b} \in \mathbb{G}$ and $q \in \text{poly}(\lambda)$ tuples $(z_j, r_j, s_j, t_j, u_j, v_j, w_j) \in \mathbb{G}^7$ such that*

$$\begin{aligned} e(a, \tilde{a}) &= e(g_z, z_j) \cdot e(g_r, r_j) \cdot e(s_j, t_j) \\ e(b, \tilde{b}) &= e(h_z, z_j) \cdot e(h_r, u_j) \cdot e(v_j, w_j), \end{aligned} \quad (2.4)$$

to find a new tuple $(z^*, r^*, s^*, t^*, u^*, v^*, w^*) \in \mathbb{G}^7$ satisfying relation (2.4) and such that $z^* \neq 1_{\mathbb{G}}$ and $z^* \neq z_j$ for $j \in \{1, \dots, q\}$. The q -Simultaneous Flexible Pairing Assumption asserts that the q -SFP problem is hard in $\mathbb{G}$.

Assumption 10 (q -DHE [26]) *The q -Diffie-Hellman Exponent Problem (q -DHE) in a cyclic group $(p, \mathbb{G}, g)$ is, given $(g, g_1, \dots, g_q, g_{q+2}, \dots, g_{2q}) \in \mathbb{G}^{2q}$ such that $g_i = g^{(\alpha^i)}$ for each i and where $\alpha \xleftarrow{\$} \mathbb{Z}_p^*$, to compute the missing element $g_{q+1} = g^{(\alpha^{q+1})}$. The q -Diffie-Hellman Exponent Assumption asserts that the q -DHE problem is hard in $\mathbb{G}$.*

As noted in [35], this problem is not easier than the one used in [26], which comes down to computing $e(g, h)^{(\alpha^{q+1})}$ on input of the same values and the additional element $h \in \mathbb{G}$. The generic hardness of q -DHE is thus implied by the generic security of the family of assumptions described in [22].

Nevertheless, we need a stronger variant of Assumption 10, used in [90]. While the Flex-CDH assumption relaxes the resolution of the CDH problem, the following assumption relaxes the q -DHE problem in a similar way.

Assumption 11 (q -Flex-DHE) *The Flexible q -Diffie-Hellman Exponent Problem (q -Flex-DHE) in a cyclic group $(p, \mathbb{G}, g)$ of prime order is, given $(g, g_1, \dots, g_q, g_{q+2}, \dots, g_{2q}) \in \mathbb{G}^{2q}$ such that $g_i = g^{(\alpha^i)}$ for each i and where $\alpha \xleftarrow{\$} \mathbb{Z}_p^*$, to compute a non-trivial triple $(g^\mu, g_{q+1}^\mu, g_{2q}^\mu) \in \mathbb{G}^3$, for some $\mu \in \mathbb{Z}_p^*$ and where $g_{q+1} = g^{(\alpha^{q+1})}$. The Flexible q -Diffie-Hellman Exponent Assumption asserts that the q -Flex-DHE problem is hard in $\mathbb{G}$.*

2.1.3 Security Model

Most commonly used protocols have their security based on the idealization of cryptographic hash functions. Informally, a hash function is modelled as a random oracle if its outputs are considered truly random. However, in practice there is no way to *efficiently* sample elements from a uniform distribution but a lot of efficient schemes can be proved under this assumption.

Assumption 12 (Random Oracle Model [15]) *Imagine a world where all protocol participants have access to a random oracle. This is an entity that initially chooses (in private) a random function $R : \{0, 1\}^l \rightarrow \{0, 1\}^t$ for some l, t . Then any player can send any bit string a of length l to the oracle which will then return $R(a)$. Since R was completely random, $R(a)$ is a uniformly chosen string of length t , and is independent of a . Moreover, seeing the value of $R(a)$ gives no advantage whatsoever in predicting the value $R(b)$ for $b \neq a$. However, R is fixed, so every time someone sends a to the oracle, the answer will be the same value $R(a)$. This is known as the random oracle model (ROM).*

Apparently, it might not be clear where there is such a strong assumption in this affirmation. However, a reduction in the ROM can choose *in advance* some values at random before receiving any request of evaluation of R from an adversary. In some sense, the challenger (*i.e.* the theoretical entity which tries to break the underlying computational assumption from an adversary of the system) can bypass the unpredictability of R outputs in the reduction as long as they remain independent of the adversary's view when they are revealed to him/her. And this is why this random oracle model facilitates the construction of proofs. Actually, the challenger can program some random outputs, usually of a hash function, and “forces” the result of the hash function on adversarially-chosen inputs. These programmed values can be used during the setting of the scheme and may really ease the security proof.

This model has some limitations. It was shown [41] that some trivially insecure schemes can be proved secure in the ROM. However, these examples are not natural.

In the thesis we will not assume that hash functions can be modelled as a random oracle. The security proofs are in the *standard model*. Nevertheless, we suppose the generation of common reference strings (CRS) that are not under the control of the adversary. This means that the generation of the CRS is honest. This model allows studying the generation of some “trusted” CRS independently and somehow postpones the problem.

However, in practice, when secure schemes in the CRS model are instantiated we do not want to rely on a trusted party to generate sensible parameters. Some of them are then generated by computing hash values to provide the certainty that no trapdoor is created during the outset of the process. Consequently, everybody can check that these parameters were generated independently because of the unpredictability of the output of the hash function on freshly evaluated inputs. But the only way to keep the “crs-proof” true in such a concrete design if the reduction needs such a trapdoor is to assume that the hash function can be modelled as a random oracle. Indeed, it makes it possible to reprogram the hash when the proof needs to modify the crs in an indistinguishable manner. Otherwise we still need to trust something about the generation of the CRS.

2.1.4 Secret Sharing

How to share a secret? Threshold cryptography requires that the knowledge of a single secret should not break all the components of a secure system. More precisely, if there was a sensitive data whose knowledge could break the security of a scheme, a threshold version of the scheme would share this data among several entities. In that way, no single entity is able to break the threshold scheme using his/her share.

More precisely, a (n, t) -threshold secret sharing splits a data into n pieces such that any t pieces allow the complete reconstruction of the data and any $t - 1$ pieces should not reveal anything about it. An efficient solution [132] to realize a (n, t) secret sharing is based on the Lagrange interpolation.

The process is as follows. Take a random polynomial P of degree $t - 1$ subject to $P(0) = s$ in $\mathbb{Z}_p$, for some prime p . A simple way to realize this is to take a random polynomial Q of degree $t - 2$ and set $P(x) = x \cdot Q(x) + s$. In order to share s into $n \geq t$ pieces return n shares of the form $(i, P(i))$. Then, for any t distinct (x_i, y_i) 's in $\{(1, P(1)), \dots, (n, P(n))\}$ use the well-known formula

$$P(x) = \sum_{i=1}^t y_i \prod_{j \neq i}^t \frac{x - x_j}{x_i - x_j} \quad (2.5)$$

to recover $s = P(0)$. In particular $s = \sum_{i \in S} P(i) \lambda_S^i$ for $\lambda_S^i = \prod_{j \in S \setminus \{i\}} \frac{j}{j-i}$ for any $S \subset \{1, \dots, n\}$ of size t .

Clearly, any $t - 1$ shares leave one degree of freedom to recover $P(x)$ and s since with overwhelming probability $1/p$ the degree of P is $t - 1$. Knowing these $t - 1$ shares leaves p possibilities for $P(0)$.

Finally, note that this technique also works for $h = g^s$ in a group $\mathbb{G}$ of prime order where DLOG assumption holds. Then, for public $h_i = g^{P(i)}$ we have $h = \prod_{i \in S} h_i^{\lambda_S^i}$ for public λ_S^i 's that are independent of s . Actually we should avoid division in $\mathbb{Q}$ and take care to stay in $\mathbb{Z}$ to “import” the property into $\mathbb{Z}_p$ (by ring homomorphism). This can be done by multiplying equation 2.5 with $\Delta = n!$ and remove it afterwards using $\Delta^{-1} \bmod p$. While the security is no more perfect but only computational when the h_i 's are public, it is sufficient for building secure schemes.

2.2 Signature Schemes

First we recall the definition of a signature scheme together with its usual security notions. We also illustrate the primitive with useful instantiations.

2.2.1 Definition and Security Notions

A common setup may be used to generate different signing keys. Then, we split the Setup_S from the KeyGen_S algorithms. To avoid confusion when several

schemes are used together with their own **Setup**, **KeyGen** or **Verify**, we index the algorithms of a signature scheme with an $\mathcal{S}$.

Definition 8 (Signature Scheme) An encryption scheme $\Pi_{\mathcal{S}}$ consists of a quadruple of algorithm $(\text{Setup}_{\mathcal{S}}, \text{KeyGen}_{\mathcal{S}}, \text{Sign}, \text{Verify}_{\mathcal{S}})$:

Setup $_{\mathcal{S}}(1^\lambda)$: from the security parameter λ , generate the public parameters pp ;

KeyGen $_{\mathcal{S}}(\text{pp})$: output a pair of keys, a (public) verifying key vk and a (secret) signing key sk ;

Sign (sk, M, ρ) : output a signature σ , on the message M , under the signing key sk with the randomness ρ ;

Verify $_{\mathcal{S}}(\text{vk}, M, \sigma)$: output a bit b , according to the validity of the signature σ on M with respect to the verification key vk .

When we use signature schemes, the following property will always hold.

Correctness of $\Pi_{\mathcal{S}}$. A signature scheme satisfies the *correctness* property if for any pair of keys (vk, sk) generated by **KeyGen $_{\mathcal{S}}$** , any message M , and any random coins ρ , we have $\text{Verify}_{\mathcal{S}}(\text{vk}, M, \text{Sign}(\text{sk}, M; \rho)) = 1$ with overwhelming probability in λ .

We now recall the security one expects for a signature scheme which is named *unforgeability*. Two slightly different definitions are recalled here. Both authorize an adversary to obtain (valid) signatures on chosen messages. This power is formalized by providing the adversary with a *signing oracle* Q_{sign} . This ability should not help it to compute a new valid signature as specified now.

Definition 9 (Security of Signatures) A signature scheme $\Pi_{\mathcal{S}}$ is existentially unforgeable (resp. strongly unforgeable) under chosen message attacks if, for any PPT adversary $\mathcal{A}$ involved in the experiment hereafter, where $\text{att} = \text{euf}$ (resp. $\text{att} = \text{suf}$), we have $\text{Adv}_{\mathcal{A}}^{\text{att-cma}}(\lambda) = \Pr[\text{Exp}_{\mathcal{A}}^{\text{att-cma}}(\lambda) = 1] \in \text{negl}(\lambda)$.

Experiment $\text{Exp}_{\mathcal{A}}^{\text{att-cma}}(\lambda)$

1. $(\text{ek}, \text{dk}) \leftarrow \text{KeyGen}_{\mathcal{E}} \circ \text{Setup}_{\mathcal{E}}(1^\lambda)$;
2. $(M^*, \sigma^*) \leftarrow \mathcal{A}^{Q_{\text{sign}}}(\text{vk})$;
3. If $M^* \in \text{SM}_{\text{euf}}$ (resp. $(M^*, \sigma^*) \in \text{SM}_{\text{suf}}$), return 0;
4. Return $b = \text{Verify}_{\mathcal{S}}(\text{vk}, M^*, \sigma^*)$;

In step 2, the adversary may adaptively query signatures on any message. On input a message M , the signing oracle Q_{sign} returns a valid signature σ on M and maintains an up-to-date initially empty list SM_{euf} (resp. SM_{suf}) by adding it M (resp. (M, σ)).

Note that from a strongly unforgeable signature it is hard to compute another signature even on a same signed message. Nothing prevents existentially unforgeable signatures from this threat, the **EUFCMA** experiment only says that it is infeasible to obtain signatures on other messages. In particular, randomizable signatures could be **EUFCMA** secure.

A signature scheme is *randomizable* if it further provides the algorithm ReRand_S . Given a signature σ and using random coins ρ' , $\text{ReRand}_S(\text{pp}, \sigma; \rho')$ outputs a (randomized) signature σ' whose distributions are the same as original signatures.

Sometimes a relaxed notion of security is sufficient. For example, a signature scheme is *weakly unforgeable* if, in the above experiment, the adversary must choose the messages that he/she will send to the signing oracle before receiving the public verifying key. Another security notion is the *one-time* unforgeability. In the one-time experiment, the adversary is only allowed to query a signature on a single message. The one-time security may be existentially or strongly unforgeable.

In a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ a signature scheme is *structure-preserving* if the public verifying key, the messages and the signatures are composed of elements in $\mathbb{G}$. As we will see, this feature allows us to efficiently prove that a committed signature on a committed message is valid.

2.2.2 Instantiations

Weak Boneh-Boyen Signature

The weak Boneh-Boyen signature scheme [21] is defined by the following four algorithms:

Setup_S(1^λ): generates a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \Lambda_{\text{sym}}(\lambda)$.

KeyGen_S(**pp**): picks $x \xleftarrow{\$} \mathbb{Z}_p$ and outputs public key $\text{vk} = (X = g^x)$, and the secret key $\text{sk} = (x)$.

Sign(**sk**, m): outputs a signature $\sigma = g^{1/(x+m)} \in \mathbb{G}$ on the message $m \in \mathbb{Z}_p$.

Verify_S(**vk**, σ): returns 1 if $e(\sigma, \text{vk} \cdot g^m) = e(g, g)$, and 0 otherwise.

The signature is deterministic and could not be EUFCMA secure. Instead, this scheme is provably weakly unforgeable assuming that q-SDH is hard (cf. Assumption 8) relatively to Λ_{sym} .

Waters Signature

The Waters signature scheme [138] is defined by the following four algorithms:

Setup_S(1^λ): generates a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \Lambda_{\text{sym}}(\lambda)$ with an additional generator $h \in \mathbb{G}$, and for $k \in \text{poly}(\lambda)$, generates $k+1$ generators

$u_0, u_1, \dots, u_k$ for the Waters function $\mathcal{F}(m) = u_0 \prod_{i=1}^k u_i^{m_i}$, where $m = (m_1, \dots, m_k) \in \{0, 1\}^k$.

KeyGen_S(pp): picks a random $x \xleftarrow{\$} \mathbb{Z}_p$ and outputs public key $\mathbf{vk} = X = g^x$, and the secret key $\mathbf{sk} = Y = h^x$.

Sign_S(sk = Y, m, ρ): outputs a signature $\sigma = (Y \cdot \mathcal{F}(m)^\rho, g^{-\rho})$, on the message $m \in \{0, 1\}^k$.

Verify_S(vk = X, m, σ): parses σ as (σ_1, σ_2) and returns 1 if the following equality $e(g, \sigma_1) \cdot e(\mathcal{F}(m), \sigma_2) = e(X, h)$ holds, and 0 otherwise.

This scheme is EUF-CMA under the CDH assumption.

Theorem 1 *This signature is randomizable:*

- **ReRand_S(vk, σ, m; ρ')** outputs $\sigma' = (\sigma_1 \cdot \mathcal{F}(m)^{\rho'}, \sigma_2 \cdot g^{-\rho'})$, for random $\rho' \xleftarrow{\$} \mathbb{Z}_q$. The randomized signature is equivalent to a fresh one using randomness $(\rho + \rho') \bmod p$.

Actually only $\mathcal{F}(m)$ must be known to randomize a signature on m .

One-Time AHO Signature

In [4], Abe, Haralambiev and Ohkubo (AHO) showed how to sign *one* vector of n group elements with a constant-size structure-preserving signature. The one-time AHO signature scheme is defined by the following four algorithms:

Setup_S(1^λ): generates a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \Lambda_{\text{sym}}(\lambda)$ with four extra generators $g_z, h_z, g_r, h_u \xleftarrow{\$} \mathbb{G}$.

KeyGen_S(pp, n): given $n \in \text{poly}(\lambda)$, chooses $\chi_i, \gamma_i, \delta_i \xleftarrow{\$} \mathbb{Z}_p$, for $i = 1$ to n and computes $g_i = g_z^{\chi_i} g_r^{\gamma_i}$, $h_i = h_z^{\chi_i} h_u^{\delta_i}$ for each $i \in \{1, \dots, n\}$. Finally, chooses exponents $\zeta, \rho, \varphi \xleftarrow{\$} \mathbb{Z}_p$ and sets $a = g_z^\zeta g_r^\rho$ and $b = h_z^\zeta h_u^\varphi$. Sets $\mathbf{vk} = (g_z, h_z, g_r, h_u, \{g_i, h_i\}_{i=1}^n, a, b)$ and $\mathbf{sk} = (\zeta, \rho, \varphi, \{\chi_i, \gamma_i, \delta_i\}_{i=1}^n)$.

Sign^{OT}_S(sk, (m₁, ..., m_n)): computes and outputs $\sigma = (z, r, u)$

$$z = g^\zeta \prod_{i=1}^n m_i^{-\chi_i}, \quad r = g^\rho \prod_{i=1}^n m_i^{-\gamma_i}, \quad u = g^\varphi \prod_{i=1}^n m_i^{-\delta_i}.$$

Verify_S(vk, σ, (m₁, ..., m_n)): parses σ into $(z, r, u) \in \mathbb{G}^3$ and returns 1 if the following equations hold, and 0 otherwise.

$$\begin{aligned} e(a, g) &= e(g_z, z) e(g_r, r) \prod_{i=1}^n e(g_i, m_i), \\ e(b, g) &= e(g_z, z) e(g_r, r) \prod_{i=1}^n e(h_i, m_i). \end{aligned} \tag{2.6}$$

In [4], the scheme was proved to be strongly unforgeable against one-time chosen-message attacks under the SDP assumption.

AHO Signature

In [4, 3], Abe, Haralambiev and Ohkubo (AHO) showed how to conveniently sign n group elements at once using signatures consisting of $O(1)$ group elements. Furthermore, their signatures are structure-preserving.

The AHO signature scheme is defined by the following four algorithms:

Setup_S(1^λ): generates a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \Lambda_{\text{sym}}(\lambda)$ with two extra generators $G_r, H_r \in \mathbb{G}$.

KeyGen_S(pp, n): given an upper bound $n \in \mathbb{N}$ on the number of group elements that can be signed altogether, picks $\gamma_z, \delta_z \xleftarrow{\$} \mathbb{Z}_p$ and $\gamma_i, \delta_i \xleftarrow{\$} \mathbb{Z}_p$, for $i = 1$ to n . Then, computes $G_z = G_r^{\gamma_z}$, $H_z = H_r^{\delta_z}$ and $G_i = G_r^{\gamma_i}$, $H_i = H_r^{\delta_i}$ for each $i \in \{1, \dots, n\}$. Finally, chooses exponents $\alpha_a, \alpha_b \xleftarrow{\$} \mathbb{Z}_p$ and defines $A = e(G_r, g^{\alpha_a})$ and $B = e(H_r, g^{\alpha_b})$. Sets the public key as

$$\text{vk} = (G_r, H_r, G_z, H_z, \{G_i, H_i\}_{i=1}^n, A, B) \in \mathbb{G}^{2n+4} \times \mathbb{G}_T^2$$

while the private key consists of $\text{sk} = (\alpha_a, \alpha_b, \gamma_z, \delta_z, \{\gamma_i, \delta_i\}_{i=1}^n)$.

Sign(sk, $(M_1, \dots, M_n)$): to sign a vector $(M_1, \dots, M_n) \in \mathbb{G}^n$ using the signing key $\text{sk} = (\alpha_a, \alpha_b, \gamma_z, \delta_z, \{\gamma_i, \delta_i\}_{i=1}^n)$, chooses $\zeta, \rho, \tau, \nu, \omega \xleftarrow{\$} \mathbb{Z}_p$ and computes $\theta_1 = g^\zeta$ as well as

$$\begin{aligned} \theta_2 &= g^{\rho - \gamma_z \zeta} \cdot \prod_{i=1}^n M_i^{-\gamma_i}, & \theta_3 &= G_r^\tau, & \theta_4 &= g^{(\alpha_a - \rho)/\tau}, \\ \theta_5 &= g^{\nu - \delta_z \zeta} \cdot \prod_{i=1}^n M_i^{-\delta_i}, & \theta_6 &= H_r^\omega, & \theta_7 &= g^{(\alpha_b - \nu)/\omega}. \end{aligned}$$

The signature consists of $\sigma = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7)$.

Verify_S(vk, $\sigma, (M_1, \dots, M_n)$): parses σ as $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7) \in \mathbb{G}^7$ and returns 1 if and only if these equalities hold:

$$A = e(G_z, \theta_1) \cdot e(G_r, \theta_2) \cdot e(\theta_3, \theta_4) \cdot \prod_{i=1}^n e(G_i, M_i), \quad (2.7)$$

$$B = e(H_z, \theta_1) \cdot e(H_r, \theta_5) \cdot e(\theta_6, \theta_7) \cdot \prod_{i=1}^n e(H_i, M_i). \quad (2.8)$$

In [4, 3], the scheme was proved to be existentially unforgeable under chosen-message attacks under the q -SFP assumption, where q is the maximal number of signing queries.

When it comes to prove that a committed signature is indeed a valid signature on committed elements, a partial rerandomization makes it possible to give part of the signature in clear. After rerandomization, we have $\theta'_1 = \theta_1$ while $\{\theta'_i\}_{i=2}^7$ are uniformly distributed among the values satisfying the equalities $e(G_r, \theta'_2) \cdot e(\theta'_3, \theta'_4) = e(G_r, \theta_2) \cdot e(\theta_3, \theta_4)$ and $e(H_r, \theta'_5) \cdot e(\theta'_6, \theta'_7) = e(H_r, \theta_5) \cdot e(\theta_6, \theta_7)$. Moreover, $\{\theta'_i\}_{i \in \{3,4,6,7\}}$ are statistically independent of $(M_1, \dots, M_n)$ and the rest of the signature. This implies that, in anonymity-related protocols, re-randomized $\{\theta'_i\}_{i \in \{3,4,6,7\}}$ can be safely revealed as long as $(M_1, \dots, M_n)$ and $\{\theta'_i\}_{i \in \{1,2,5\}}$ are given in a committed form. See the second example of Section 2.5.3.

It was also observed [4, 3] that signature components (θ_3, θ_6) can be used as a commitment to the message. Under the q -SFP assumption, it is infeasible to find signatures $\sigma = (\theta_1, \dots, \theta_7)$, $\sigma' = (\theta'_1, \dots, \theta'_7)$ on two distinct messages M, M' such that $(\theta_3, \theta_6) = (\theta'_3, \theta'_6)$. This is true even if the adversary has access to a signing oracle and obtains signatures on both M and M' . While the parameter $n \in \text{poly}(\lambda)$ is fixed, this allows building a signature on unbounded (though of polynomial-size) vectors of messages. We will take advantage of that property in the design of quotable signatures.

A less efficient constant-size structure preserving signature was proposed later [1] which is based on the *simple* (not " q -type") assumption DLIN. For efficiency reason we use the signature presented here.

Groth One-Time Signature

Groth signature scheme [80] is defined by the following four algorithms:

Setup_S(1^λ): generates a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \Lambda_{\text{sym}}(\lambda)$ and picks a collision-free hash function $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p$.

KeyGen_S(pp): picks $x, y, z, t \xleftarrow{\$} \mathbb{Z}_p$ and outputs public key $\text{vk} = (f = g^x, h = g^y, d = f^z h^t)$, and the secret key $\text{sk} = (x, y, z, t)$.

Sign(sk, m ; ρ): outputs a signature $\sigma = (\sigma_1, \sigma_2) \in \mathbb{Z}_p^2$ with random $\rho \xleftarrow{\$} \mathbb{Z}_p$ such that $\sigma_1 = z - H(m)/x + \rho$ and $\sigma_2 = t - \rho \cdot x/y$.

Verify_S(vk, σ): parses σ as (σ_1, σ_2) and returns 1 if $d = g^{H(m)} f^{\sigma_1} h^{\sigma_2}$, and 0 otherwise.

This scheme is SUF-OT-CMA under the DLog assumption (see that the challenger does not need to know y for $\rho := 0$). The scheme also works without pairing.

2.3 Encryption Schemes

First we recall the definition of an encryption scheme together with its most well-known security notions. We also put forward some practical properties.

Then, we illustrate the primitive with useful examples. More advanced encryption schemes empowered with improved features are included in a later section.

2.3.1 Definitions and Security Notions

A common setup may be used to generate different key pairs. Then, we split the Setup_S from the KeyGen_S algorithms.

Definition 10 (Encryption Scheme) *An encryption scheme Π_E consists of a quadruple of efficient algorithms $(\text{Setup}_E, \text{KeyGen}_E, \text{Encrypt}, \text{Decrypt})$:*

Setup $_E$ (1^λ): *from the security parameter λ , generates the public parameters pp ;*

KeyGen $_E$ (pp): *outputs a pair of keys, a (public) encryption key ek and a (secret) decryption key dk ;*

Encrypt(ek, M, ρ): *given the message M , outputs a ciphertext $\mathbf{c}$, under the encryption key ek with the randomness ρ ;*

Decrypt($\text{dk}, \mathbf{c}$): *outputs a plaintext M , encrypted in the ciphertext $\mathbf{c}$ using the decryption key dk or $\perp$.*

When we use encryption schemes, the following property will always hold.

Correctness of Π_E . An encryption scheme satisfies the *correctness* property if for any pair of keys (ek, dk) generated by KeyGen_E , any message M , and any random ρ , we have $\text{Decrypt}(\text{dk}, \text{Encrypt}(\text{ek}, M, \rho)) = M$ at least with overwhelming probability in λ .

At a high level, security of encryption scheme requires that the distributions of ciphertexts over different messages are computationally closed.

In the thesis, we consider several notions of security modelled under a parametrized indistinguishability experiment. The different flavours of the security notions consist in giving the adversary the opportunity to invoke a decryption oracle Q_{decrypt} subject to some restrictions. If the adversary $\mathcal{A}$ makes a decryption query for a ciphertext of his/her choice, the oracle returns the corresponding plaintext.

Definition 11 (Security of Encryptions) *An encryption scheme Π_E is indistinguishable against the attack scenario $\text{att} \in \{\text{cpa}, 1\text{-cpa}, \text{cca}1, \text{cca}2\}$ if, for any PPT adversary $\mathcal{A}$ involved in the experiment hereafter, his/her advantage must satisfy $\text{Adv}_{\mathcal{A}}^{\text{ind-att}}(\lambda) = |\Pr[\text{Exp}_{\mathcal{A}}^{\text{ind-att}}(\lambda) = 1] - 1/2| \in \text{negl}(\lambda)$.*

Experiment $\mathbf{Exp}_{\mathcal{A}}^{\text{ind-att}}(\lambda)$

1. $\text{pp} \leftarrow \text{Setup}_{\mathcal{E}}(1^\lambda)$; $(\text{ek}, \text{dk}) \leftarrow \text{KeyGen}_{\mathcal{E}}(\text{pp})$;
2. $(M_0, M_1) \leftarrow \mathcal{A}(\text{find} : \text{ek}, Q_{\text{pre}}^{\text{att}})$;
3. Picks random $b \xleftarrow{\$} \{0, 1\}$;
4. $\mathbf{c}^* \leftarrow \text{Encrypt}(\text{ek}, M_b)$;
5. $b' \leftarrow \mathcal{A}(\text{guess} : \mathbf{c}^*, Q_{\text{post}}^{\text{att}})$;
6. If $b' = b$ then returns 1; Else returns 0;

According to att, $\mathcal{A}$ has access to a pre-challenge oracle $Q_{\text{pre}}^{\text{att}}$ and to a post-challenge oracle $Q_{\text{post}}^{\text{att}}$ with the following specifications. When chosen-plaintext-attack (CPA) is considered, i.e. att = cpa, $Q_{\text{pre}}^{\text{cpa}}$ and $Q_{\text{post}}^{\text{cpa}}$ always return $\perp$; no additional power is given to $\mathcal{A}$ in the indistinguishability game. For the chosen-ciphertext-attacks (CCA), in the find phase, $Q_{\text{pre}}^{\text{cca}1} = Q_{\text{pre}}^{\text{cca}2} = Q_{\text{dec}}$ returns the decrypton of any ciphertext. In the guess phase, $Q_{\text{post}}^{\text{cca}1}$ always returns $\perp$ and $Q_{\text{post}}^{\text{cca}2} = Q_{\text{decrypt}}^{-\mathbf{c}^*}$ is like Q_{decrypt} but returns $\perp$ if the adversary tries to decrypt the challenge ciphertext.

The IND-CPA security means that no adversary who chose the two messages in step 2 should be able to efficiently guess which message has been encrypted in step 4. The encryption key ek allows encrypting messages of his/her choice.

When an adversary $\mathcal{A}$ is entitled to only query for decryption before seeing the challenge ciphertext $\mathbf{c}^*$ (step 4.) we talk about non-adaptive CCA (IND-CCA1) security [120]. Here the PPT adversary can query the number of decryptions he/she wants even after receiving the corresponding plaintext of previous queries. So he/she can make his/her choices adaptively to the Q_{decrypt} 's outputs. However, his/her choice is non adaptive with respect to $\mathbf{c}^*$. If the adversary is in addition authorized to query decryptions after seeing the challenge ciphertext we talk about adaptive CCA (IND-CCA2) security [126]. For simplicity we refer to this notion as the CCA security.

2.3.2 Additional Features

In addition to security properties of cryptosystems some other features are usually recommended.

Randomizable Encryptions. An encryption scheme is randomizable if it further provides the algorithm $\text{ReRand}_{\mathcal{E}}$ such that for a ciphertext $\mathbf{c}$ and random coins ρ' , $\text{ReRand}_{\mathcal{E}}(\text{ek}, \mathbf{c}, \rho')$ outputs a (randomized) ciphertext $\mathbf{c}'$ whose distributions are the same as original ciphertexts.

Homomorphic Encryptions. An encryption scheme is homomorphic if it further provides the algorithm $\text{Homom}_{\mathcal{E}}$ such that for a couple of ciphertexts $(\mathbf{c}_1 \leftarrow \text{Encrypt}(m_1), \mathbf{c}_2 \leftarrow \text{Encrypt}(m_2))$, $\text{Homom}_{\mathcal{E}}(\text{ek}, \mathbf{c}_1, \mathbf{c}_2)$ outputs a ciphertext $\mathbf{c}$ of $m_1 m_2$ (with multiplicative notation).

Some consider $\text{Homom}_{\mathcal{E}}$ to be probabilistic. Here we clearly split the two properties even if we will use both most of the time. It suffices to apply $\text{ReRand}_{\mathcal{E}}$ after $\text{Homom}_{\mathcal{E}}$.

Security with these additional algorithm is unchanged. It is easy to see that homomorphic or randomizable encryptions are never CCA-secure because of their malleability. Nevertheless we will show that CCA-secure cryptosystems may support some kind of homomorphic properties.

The literature also displays more complex encryption primitives whose specificities require to reconsider the security model. Some *advanced* encryption schemes are discussed later (in subsection 2.6) as their instantiations are usually based on other primitives.

Now, we recall some instantiations of ordinary encryption schemes.

2.3.3 Instantiations

ElGamal Encryption

The ElGamal encryption scheme [72] is defined by the following four algorithms:

- Setup $_{\mathcal{E}}(1^\lambda)$** : generates a cyclic multiplicative group $\mathbb{G}$ of order q of size λ , together with a generator g and set $\text{pp} = (q, \mathbb{G}, g)$.
- KeyGen $_{\mathcal{E}}(\text{pp})$** : chooses a random scalar $x \xleftarrow{\$} \mathbb{Z}_q$ and computes $X = g^x$. It returns $(\text{ek}, \text{dk}) = (X, x)$.
- Encrypt $(\text{ek} = X, M, \rho)$** : for a message $M \in \mathbb{G}$ and a random scalar $\rho \xleftarrow{\$} \mathbb{Z}_q$ computes and outputs the ciphertext $\mathbf{c} = (c_1 = g^\rho, c_2 = M \cdot X^\rho) \in \mathbb{G}^2$.
- Decrypt $(\text{dk} = x, \mathbf{c})$** : parses $\mathbf{c}$ as a couple in $\mathbb{G}$. If it is not possible returns $\perp$ otherwise returns $c_1^{-x} c_2$.

Under the DDH assumption (cf. Assumption 4), the ElGamal encryption is CPA-secure [135].

Theorem 2 *This encryption is randomizable and homomorphic:*

- **ReRand $_{\mathcal{E}}(\text{ek}, \mathbf{c} = (c_1, c_2); \rho')$** outputs $\mathbf{c}' = (c_1 \cdot g^{\rho'}, c_2 \cdot X^{\rho'})$, for random $\rho' \xleftarrow{\$} \mathbb{Z}_q$. The randomized ciphertext is equivalent to a fresh encryption using randomness $(\rho + \rho') \bmod q$.
- **Homom $_{\mathcal{E}}(\text{ek}, \mathbf{c}_1, \mathbf{c}_2)$** outputs $\mathbf{c} = \mathbf{c}_1 \cdot \mathbf{c}_2$ for componentwise multiplication. The resulting ciphertext is an encryption of $M_1 \cdot M_2$ with randomness $(\rho_1 + \rho_2) \bmod q$ if $\mathbf{c}_i$ is an encryption of M_i with randomness ρ_i , for $i = 1, 2$.

BBS Encryption

The Boneh-Boyen-Shacham (BBS) encryption scheme [24] is defined by the following four algorithms:

Setup_E(1^λ): generates a cyclic multiplicative group $\mathbb{G}$ of order q of size λ , together with a generator g and set $\mathbf{pp} = (q, \mathbb{G}, g)$.

KeyGen_E($\mathbf{pp}$): chooses two random scalars $\alpha_1, \alpha_2 \xleftarrow{\$} \mathbb{Z}_q$ which define the secret key $\mathbf{dk} = (\alpha_1, \alpha_2)$ and the public key $\mathbf{ek} = (f_1 = g^{\alpha_1}, f_2 = g^{\alpha_2})$.

Encrypt($\mathbf{ek} = (f_1, f_2), M, (\rho_1, \rho_2)$): for a message $M \in \mathbb{G}$ and two random scalars $\rho_1, \rho_2 \xleftarrow{\$} \mathbb{Z}_q$ computes and outputs the ciphertext $\mathbf{c} = (c_1, c_2, c_3) = (f_1^{\rho_1}, f_2^{\rho_2}, M \cdot g^{\rho_1 + \rho_2}) \in \mathbb{G}^3$.

Decrypt($\mathbf{dk} = (\alpha_1, \alpha_2), \mathbf{c}$): parses $\mathbf{c}$ as a triple in $\mathbb{G}$. If it is not possible returns $\perp$ otherwise returns $c_1^{-1/\alpha_1} \cdot c_2^{-1/\alpha_2} \cdot c_3$.

Under the DLIN assumption (cf. Assumption 5), the same authors show that the BBS encryption is CPA-secure. The advantage of this encryption relies on the fact that it can be secure in groups where the ElGamal encryption is not, *e.g.* in groups supporting symmetric pairings.

Theorem 3 *This encryption is randomizable and homomorphic:*

- **ReRand_E**($\mathbf{ek}, \mathbf{c} = (c_1, c_2, c_3); \rho'$) outputs $\mathbf{c}' = (c_1 \cdot f_1^{\rho'_1}, c_2 \cdot f_2^{\rho'_2}, c_3 \cdot g^{\rho'_1 + \rho'_2})$, for random $\rho'_1, \rho'_2 \xleftarrow{\$} \mathbb{Z}_q$. The randomized ciphertext is equivalent to a fresh encryption using randomness $(\rho_1 + \rho'_1, \rho_1 + \rho'_1) \bmod q$.
- **Homom_E**($\mathbf{ek}, \mathbf{c}_1, \mathbf{c}_2$) outputs $\mathbf{c} = \mathbf{c}_1 \cdot \mathbf{c}_2$ for componentwise multiplication. The resulting ciphertext is an encryption of $M_1 \cdot M_2$ with randomness $(\rho_{11} + \rho_{21}, \rho_{12} + \rho_{22}) \bmod q$ if $\mathbf{c}_i$ is an encryption of M_i with randomness (ρ_{i1}, ρ_{i2}) , for $i = 1, 2$.

These two encryption schemes can be adapted to give an additive group as message space. It suffices to consider messages of the form $M := g^m$ for the fixed generator g . If needed, another fixed generator can be used which will be produced by **Setup_E**. However the decryption algorithm is no longer efficient since after recovering g^m it has to compute $m \in \mathbb{Z}_q$ from an element of $M \in \mathbb{G}$, that means computing discrete logarithms (in basis g) in groups where DDH or DLIN are supposed to be hard. Computing m can only be easy for messages in a pre-defined set of size in $\text{poly}(\lambda)$ (of negligible size *w.r.t.* the one of $\mathbb{G}$ and $\mathbb{Z}_q$) known by **Decrypt**. When an additive group is required for large space the next construction provides an outstanding example where the DLog problem is easy for special elements.

For completeness we give an encryption scheme satisfying the highest level of security among those recalled at the beginning of the section. We will not

explicitely use this CCA secure scheme in the thesis but it is the first scheme to use the so-called *hash proof system* that was introduced later by the same authors [50]. This principle has been widely used afterwards.

Cramer-Shoup Encryption

The Cramer-Shoup encryption scheme [49] is defined by the following four algorithms:

Setup_E(1^λ): generates a cyclic multiplicative group $\mathbb{G}$ of order q of size λ , together with two generators g_1, g_2 . It also picks a random hash function $H : \{0, 1\}^* \rightarrow \mathbb{Z}_q$ among a Collision-Resistant hash family. Then it sets $\text{pp} = (q, \mathbb{G}, g_1, g_2, H)$.

KeyGen_E(pp): chooses a random tuple $\text{dk} = (x_1, x_2, y_1, y_2, z) \xleftarrow{\$} \mathbb{Z}_q^5$ and computes $\text{ek} = (u, v, h)$ with $u = g_1^{x_1} g_2^{x_2}$, $v = g_1^{y_1} g_2^{y_2}$ and $h = g_1^z$.

Encrypt(ek, M, r): for a message $M \in \mathbb{G}$ and a random scalar $r \xleftarrow{\$} \mathbb{Z}_q$ computes $c_1 = g_1^r$, $c_2 = g_2^r$, $e = M \cdot h^r$ and $d = (u \cdot v^\alpha)^r$ for $\alpha = H(c_1, c_2, e)$, and outputs the ciphertext $\mathbf{c} = (c_1, c_2, e, d) \in \mathbb{G}^4$.

Decrypt($\text{dk}, \mathbf{c}$): parses $\mathbf{c}$ as a tuple (c_1, c_2, e, d) in $\mathbb{G}^4$ and checks whether $d \stackrel{?}{=} c_1^{x_1 + \alpha y_1} \cdot c_2^{x_2 + \alpha y_2}$, where $\alpha = H(c_1, c_2, e)$. If this holds, returns e/c_1^z and $\perp$ otherwise.

Under the DDH assumption (cf. Assumption 4), the Cramer-Shoup encryption is CCA-secure when the hash function is Universal One-Way (cf. Definition 5).

This scheme is the first practical encryption proven CCA secure without assuming the existence of a random oracle (cf. Definition 12). There exists a BBS variant [131] of this scheme based on the DLIN assumption. Notice that (c_1, e) corresponds to an ElGamal ciphertext.

2.4 Commitment Schemes

A commitment scheme is the digital analogue of a safe. From the outside we cannot guess what value is inside. Furthermore, when we open it we know that the revealed value was not altered.

2.4.1 Definition and Security Notions

We still make a distinction between the setup of a scheme which generates common parameters and the generation of the keys.

Definition 12 (Commitment Scheme) *A commitment scheme Π_C is given by the following four algorithms ($\text{Setup}_C, \text{KeyGen}_C, \text{Com}, \text{Verify}_C$) such that :*

Setup_C(1^λ): from the security parameter λ , generates the public parameters pp ;

KeyGen_C(pp): takes in pp and outputs a public committing key ck .

Com($\text{ck}, m; \rho$): using the random ρ outputs a pair (d, a) made of a commitment and an opening value.

Verify_C(ck, d, a, m): returns either 1 or 0 if a is a valid opening of d for the message m .

These algorithms are all efficient in λ .

Note that the verification algorithm is public. For all commitment schemes the following property must always hold.

Correctness of Π_C . A commitment scheme satisfies the *correctness* property if for every committing keys ck generated by **KeyGen_C** on pp , every message M , and every random coins ρ , we have $\text{Verify}_C(\text{ck}, \text{Com}(\text{ck}, m; \rho), m) = 1$ with overwhelming probability in λ .

There are two security guarantees expected for commitment schemes. On the one hand, we want that a commitment does not leak information about the committed message, which is called the *hiding* property. On the other hand, we want that the committer cannot open its commitment on another message, which is called the *binding* property. In this way, commitments can be viewed as the digital analogue of sealed boxes or safes.

Definition 13 (Hiding property) A commitment scheme is computationally (resp. statistically or perfectly) hiding if, for any PPT adversary (resp. unbounded adversary) $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ involved in the experiment hereafter, we have $\text{Adv}_{\mathcal{A}}^{\text{hide}}(\lambda) = |\Pr[\text{Exp}_{\mathcal{A}}^{\text{hide}}(\lambda) = 1] - 1/2| \in \text{negl}(\lambda)$.

- Experiment $\text{Exp}_{\mathcal{A}}^{\text{hide}}(\lambda)$
1. $\text{pp} \leftarrow \text{Setup}_C(1^\lambda)$; $\text{ck} \leftarrow \text{KeyGen}_C(\text{pp})$;
 2. $(m_0, m_1) \leftarrow \mathcal{A}_1(\text{ck})$;
 3. Picks random $b \xleftarrow{\$} \{0, 1\}$;
 4. $(d^*, a^*) \leftarrow \text{Com}(\text{ck}, m_b)$;
 5. $b' \leftarrow \mathcal{A}_2(d^*)$;
 6. If $b' = b$ then returns 1; Else returns 0;

For the perfectly hiding case the negligible function is the null function so that there is no advantage in winning the hiding experiment.

Definition 14 (Binding property) A commitment scheme is computationally (resp. statistically or perfectly) binding if, for any PPT adversary (resp. unbounded adversary) $\mathcal{A}$ involved in the experiment hereafter, his/her advantage must satisfy $\text{Adv}_{\mathcal{A}}^{\text{bind}}(\lambda) = |\Pr[\text{Exp}_{\mathcal{A}}^{\text{bind}}(\lambda) = 1] - 1/2| \in \text{negl}(\lambda)$.

Experiment $\mathbf{Exp}_A^{\text{bind}}(\lambda)$

1. $\text{pp} \leftarrow \text{Setup}_C(1^\lambda)$; $\text{ck} \leftarrow \text{KeyGen}_C(\text{pp})$;
2. $(d, m_0, m_1, a_0, a_1) \leftarrow \mathcal{A}_1(\text{ck})$;
3. If $\text{Verify}_C(\text{ck}, d, a_0, m_0) = 1 = \text{Verify}_C(\text{ck}, d, a_1, m_1)$
and $m_0 \neq m_1$ returns 1; Else returns 0;

For the perfectly binding case the negligible function is the null function so that there is no advantage in winning the binding experiment.

A known result states that there exists no commitment scheme which is both perfectly hiding and perfectly binding. The two perfect notions are in conflict.

Like for the encryption schemes, commitment schemes can be randomizable and homomorphic.

2.4.2 Additional Features

As encryption schemes, commitments may support rerandomization and homomorphism but other interesting properties show that both notions are not equivalent.

Randomizable Commitments. A commitment scheme is randomizable if it contains an additional public algorithm ReRand_C such that for a commitment d and randomness ρ' , $\text{ReRand}_C(\text{ck}, d, \rho')$ outputs a (randomized) commitment d' that is completely redistributed among the ciphertexts of the same plaintext. Moreover decommitments or openings should also follow the homomorphism while keeping the correctness of the scheme.

Homomorphic Commitments. A commitment scheme is homomorphic if it contains an additional public algorithm Homom_C with the following property: for any couple of commitments (d_1, d_2) such that $(d_1, a_1) \leftarrow \text{Com}(\text{ek}, m_1)$ and $(d_2, a_2) \leftarrow \text{Com}(\text{ek}, m_2)$, $\text{Homom}_C(\text{ek}, d_1, d_2)$ outputs a valid commitment d on $m_1 m_2$ relative to the opening value $a_1 a_2$ (where both operations are noted multiplicatively).

When a commitment scheme is perfectly binding it means that for any commitment d it exists only one message m for which there is at least one opening value a such that $\text{Verify}_C(\text{ck}, d, a, m) = 1$. The hiding property implies the infeasibility of computing such message m . However, using a special private key it could be possible to efficiently output this m . In that case, the commitment scheme is said *extractable*.

Until now, encryption schemes seem to fulfill all the above requirements. The private key used for decryption also allows *extracting* committed values. However perfectly hiding commitment schemes are necessary in some protocols. They may also support other useful features such as *equivocability*, which

means that it is feasible to compute several openings on a same commitment to different messages. To stay compatible with the computationally binding property, such commitments need another dedicated (private) key in order to allow “equivocations”. This kind of commitment is called *trapdoor commitment*. Despite their incompatibility, these two new aspects may coexist. A key-generation algorithm should, indistinguishably, either define an extractable setting or an equivocable setting.

Definition 15 (Flippable Commitment) *A flippable commitment scheme $\Pi_{\mathcal{FC}}$ is given by a tuple of efficient algorithms $(\text{Setup}_{\mathcal{C}}, \text{KeyGen}_{\mathcal{MC}}, \text{Com}, \text{Verify}_{\mathcal{C}}, \text{Extract}, \text{Equiv})$ with the following specifications:*

Setup $_{\mathcal{C}}$ (1^λ): *given the security parameter λ , generates the public parameters pp ;*

KeyGen $_{\mathcal{MC}}$ (pp, b): *takes in the public parameters pp and outputs a public committing key ck_b and the corresponding private trapdoor key tk_b using one of the next algorithms according to the bit b :*

- **ExKey $_{\mathcal{C}}$** (pp): *outputs an extractable key-pair $(\text{ck}_0, \text{tk}_0)$.*
- **TrKey $_{\mathcal{C}}$** (pp): *outputs an equivocable key-pair $(\text{ck}_1, \text{tk}_1)$.*

Com($\text{ck}, m; \rho$): *using the random ρ outputs a pair (d, a) made of a commitment and an opening value.*

Verify $_{\mathcal{C}}$ (ck, d, a, m): *returns either 1 or 0 if a is a valid opening of d for the message m .*

Extract(tk, d): *using the trapdoor (extratable) key $\text{tk} = \text{tk}_0$ returns the (supposedly committed) message m' from d .*

Equiv(tk, d, a, m, m'): *if $\text{Verify}_{\mathcal{C}}(\text{ck}, d, a, m) = 0$ returns $\perp$, otherwise using the trapdoor (equivocable) key $\text{tk} = \text{tk}_1$ computes an opening a' (supposedly valid) of d on the message m' from (a, m) and returns (d, a') .*

Such commitments appeared in the literature without a specific name. They play a fundamental role in the efficient proof system of the next section. Now we recall the two correctness properties depending on the bit b . Thereafter, we give the whole correctness of flippable commitments.

Extractable Commitments. A commitment scheme defined by the algorithms $(\text{Setup}_{\mathcal{C}}, \text{KeyGen}_{\mathcal{MC}}(\cdot, 0), \text{Com}, \text{Verify}_{\mathcal{C}}, \text{Extract})$ is called an *extractable commitment*. We may also find the notation **ExSetup** in the literature for $\text{ExSetup}(\lambda) = \text{KeyGen}_{\mathcal{MC}}(\text{Setup}_{\mathcal{C}}(\lambda), 0)$. Extractable commitments are assumed to be correct commitments that satisfy the **extractability correctness**:

For any committing-extracting key pairs (ck, tk) generated by **ExSetup** on any security parameter λ , for any message m , and any random ρ , we have

$$\text{Extract}(\text{tk}, \text{Com}(\text{ck}, m, \rho)) = m \quad (2.9)$$

at least with overwhelming probability in λ .

Trapdoor Commitments. A commitment scheme defined by the algorithms $(\text{Setup}_c, \text{KeyGen}_{\mathcal{MC}}(\cdot, 1), \text{Com}, \text{Verify}_c, \text{Equiv})$ is called a *trapdoor commitment*. We may also find the notation TrSetup in the literature for $\text{TrSetup}(\lambda) = \text{KeyGen}_{\mathcal{MC}}(\text{Setup}_c(\lambda), 1)$. Trapdoor commitments are assumed to be correct commitments that further satisfy the **equivocability correctness**:

For every committing-trapdoor key pairs (ck, tk) generated by TrSetup on any security parameter λ , for every message m , and every random ρ , we have

$$\text{Verify}_c(\text{ck}, \text{Equiv}(\text{tk}, \text{Com}(\text{ck}, m, \rho), m, m'), m') = 1 \quad (2.10)$$

at least with overwhelming probability in λ .

For the moment we do not introduce an algorithm to simulate the generation of commitments that are equivocable in the above definition. This choice is made to ease the introduction of the numerous schemes that we recall. Furthermore, this simulation would be perfect for the following instantiations so that equivocations can equivalently be made on “real” commitments. Most of the time “fake” commitments are actually defined as commitments on the neutral element message and the perfect hiding property guarantees their indistinguishability.

Flippable Correctness of $\Pi_{\mathcal{FC}}$. A flippable commitment scheme satisfies the *correctness* property if both underlying commitments are correct in their respective sense, in the extractable case when $b = 0$ and in the equivocable case when $b = 1$, and if for any PPT adversary $\mathcal{A}$ it is infeasible to distinguish the setting wherein the public committing key ck is generated:

$$\left| \Pr[1 \leftarrow \mathcal{A}(\text{ck}) \mid \text{ck} \leftarrow \text{ExSetup}(\lambda)] - \Pr[1 \leftarrow \mathcal{A}(\text{ck}) \mid \text{ck} \leftarrow \text{TrSetup}(\lambda)] \right| \quad (2.11)$$

is a negligible function in λ .

2.4.3 Instantiations

Encryption schemes are perfectly binding commitment schemes where the decryption keys make the commitments extractable. So we only recall perfectly hiding commitment schemes. Moreover, they are all homomorphic trapdoor commitments.

Pedersen Commitment

The Pedersen commitment scheme [122] is defined by the following five algorithms:

Setup_c(1^λ): generates a cyclic multiplicative group $\mathbb{G}$ of order q of size λ , together with a generator g and sets $\text{pp} = (q, \mathbb{G}, g)$.

KeyGen_C(pp): chooses a random scalar $x \xleftarrow{\$} \mathbb{Z}_q$ and computes $h = g^x$. It returns $(\text{ck}, \text{tk}) = (h, x)$ where x is the trapdoor key.

Com(ck, m, r): for a message $m \in \mathbb{Z}_q$ and a random scalar $r \xleftarrow{\$} \mathbb{Z}_q$ computes $d = g^m h^r \in \mathbb{G}$ and outputs the commitment-opening pair (d, r) .

Equiv(tk = x, d, a, m, m'): if $\text{Verify}_C(\text{ck}, d, a, m) = 0$ returns $\perp$ otherwise returns (d, r') where $r' = r + (m - m')/x$.

Verify_C(ck, d, a, m): checks whether $d \stackrel{?}{=} g^m h^a$ or not and returns 1 or 0 consistently.

The Pedersen commitment is perfectly hiding since any commitment may be opened on any message (see **Equiv**) and under the DLOG assumption in $\mathbb{G}$ (cf. Assumption 1), the commitment is computationally binding.

Theorem 4 *This commitment is randomizable and homomorphic:*

- **ReRand_C(ck, d; r')** outputs $d' = d \cdot h^{r'}$, for random $r' \xleftarrow{\$} \mathbb{Z}_q$. The randomized commitment is equivalent to a fresh one with randomness $(r + r') \bmod q$.
- **Homom_C(ck, d₁, d₂)** outputs $d = d_1 d_2$. The resulting commitment is a commitment of $m_1 + m_2$ with randomness $(r_1 + r_2) \bmod q$ if d_i is a commitment of m_i with randomness r_i , for $i = 1, 2$ (and so is for the opening values).

An advantage of the perfectly hiding commitment upon the perfectly binding commitment is that the former allows committing to several messages at once. This is impossible for the latter because of the unicity of the committed message.

Multi-Pedersen Commitment

The multi-Pedersen commitment scheme is defined by the following five algorithms:

Setup_C(1^λ): generates a cyclic multiplicative group $\mathbb{G}$ of order q of size λ , together with a generator g and sets $\text{pp} = (q, \mathbb{G}, g)$.

KeyGen_C(pp, n): chooses n random scalars $x_1, \dots, x_n \xleftarrow{\$} \mathbb{Z}_q$ and computes $h_i = g^{x_i}$ for all $i = 1, \dots, n$. It returns $(\text{ck}, \text{tk}) = (h, (x_i)_{i=1}^n)$.

Com(ck, m, r): for a n -vector message $m = (m_1, \dots, m_n) \in \mathbb{Z}_q^n$ and a random scalar $r \xleftarrow{\$} \mathbb{Z}_q$ computes $d = g^r h_1^{m_1} \dots h_n^{m_n} \in \mathbb{G}$ and outputs (d, r) .

Equiv(tk = x, d, a, m, m'): if $\text{Verify}_C(\text{ck}, d, a, m) = 0$ returns $\perp$ otherwise returns (d, r') where $r' = r + x_1(m_1 - m'_1) + \dots + x_n(m_n - m'_n)$.

Verify_C(ck, d , a , m): for $m = (m_1, \dots, m_n)$ checks whether $d \stackrel{?}{=} g^a h_1^{m_1} \dots h_n^{m_n}$ or not and returns 1 or 0 consistently.

The multi-Pedersen commitment is perfectly hiding and computationally binding under the DLOG assumption in $\mathbb{G}$ (cf. Assumption 1).

Multi-AHO Structure-Preserving Commitment

Abe *et al.* proposed an optimal structure-preserving commitment scheme [5] for asymmetric pairing. It allows committing on vectors of group elements and it is defined by the following five algorithms:

Setup_C(1^λ): generates a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h) \leftarrow \Lambda_{\text{asym}}(\lambda)$.

KeyGen_C(pp, n): chooses n random scalars $x_1, \dots, x_n \xleftarrow{\$} \mathbb{Z}_q$ and computes $g_i = g^{x_i}$ for all $i = 1, \dots, n$. It returns $(\text{ck}, \text{tk}) = ((g_i)_{i=1}^n, (x_i)_{i=1}^n)$.

Com(ck, M , ρ): for a n -vector message $M = (M_1, \dots, M_n) \in \mathbb{Z}_q^n$ and a random $(n+1)$ -vector scalar $r = (r_0, r_1, \dots, r_n) \xleftarrow{\$} \mathbb{Z}_q^n$ computes $d_i = M_i h^{r_i} \in \mathbb{G}_2$ for $i = 1$ to n , $d_0 = g^{r_0} g_1^{r_1} \dots g_n^{r_n}$ and $a = h^{r_0}$, and outputs the pair $(d = (d_i)_{i=0}^n, a)$.

Verify_C(ck, d , a , m): outputs 1 if $e(d_0, h) = e(g, a) \prod_{i=1}^n e(g_i, d_i/M_i)$ and 0 otherwise.

Equiv(tk = x , d , a , M , M'): if **Verify_C**(ck, d , a , M) = 0 returns $\perp$ otherwise returns (d, a') where $a' = a \prod_{i=1}^n (M'_i/M_i)^{x_i}$.

This multi-commitment is optimal since the authors showed that the number of elements needed to commit on a vector of group elements with a structure-preserving scheme cannot be smaller than the number of committed elements: *group to group commitments do not shrink*.

The commitment is perfectly hiding and under the DBP assumption in $\mathbb{G}_1$ (cf. Assumption 6), the commitment is computationally binding. The authors also proposed a variant of the scheme for symmetric pairings [5] based on the SDP assumption (cf. Assumption 7).

Linear Flippable Commitment

The linear flippable commitment for group element $M \in \mathbb{G}$ is defined by the following tuple of efficient algorithms

Setup_C(1^λ): generates a cyclic multiplicative group $\mathbb{G}$ of order q of size λ , together with a generator g and sets $\text{pp} = (q, \mathbb{G}, g)$.

KeyGen_{FC}(pp, b): takes in pp and chooses four random scalars $\mu_1, \mu_2, \nu_1, \nu_2 \xleftarrow{\$} \mathbb{Z}_q$, then first computes $f_1 = g^{\mu_1}$, $f_2 = g^{\mu_2}$ and sets $\text{tk}_0 = (\mu_1, \mu_2)$, $\text{tk}_1 = (\nu_1, \nu_2)$ then computes $f_{3,1} = f_1^{\nu_1}$, $f_{3,2} = f_2^{\nu_2}$, $f_{3,3} = g^{b+\nu_1+\nu_2}$ and sets $\text{ek} = (f_1, f_2, f_{3,1}, f_{3,2}, f_{3,3})$.

Com(ck, M ; (α, β, γ)): using the random scalars $a = (\alpha, \beta, \gamma) \xleftarrow{\$} \mathbb{Z}_q^3$ computes $d = (d_1, d_2, d_3)$ with $d_1 = f_1^\alpha f_{3,1}^\gamma$, $d_2 = f_2^\beta f_{3,2}^\gamma$, $d_3 = M \cdot g^{\alpha+\beta} f_{3,3}^\gamma$ and outputs (d, a) .

Verify_C(ck, d, a, M): returns 1 if $d = \text{Com}(\text{ck}, M; a)$ and 0 otherwise.

Extract(tk₀, d): using the trapdoor (extractable) key tk = (μ_1, μ_2) returns $M = d_1^{-1/\mu_1} d_2^{-1/\mu_2} d_3$.

Equiv(tk₁, $d, a, M, M' := Mg^{m'}$): if **Verify_C**(ck, d, a, M) = 0 returns $\perp$, returns (d, a') with $a' = (\alpha', \beta', \gamma')$ such that $\alpha' = \alpha + \nu_1 m'$, $\beta' = \beta + \nu_2 m'$ and $\gamma' = \gamma - m'$.

Note that the trapdoor property allows opening commitments to any group element in $\mathbb{G}$ but the discrete log m' of M'/M in base g must be known. In the **ExSetup** generation (when $b = 0$) the commitment is a BBS encryption of M with the encryption key (f_1, f_2, g) with randomness $\alpha + \nu_1 \gamma$ and $\beta + \nu_2 \gamma$. In the **TrSetup** generation (when $b = 1$) the extra factor g in $f_{3,3}$ makes the scheme perfectly hiding since a commitment now may be seen as an encryption of the message $M \cdot g^\gamma$ where γ is random. In this case the DLOG assumption implies the binding property which is also a consequence of the indistinguishability of both settings. The two key generations are indistinguishable under the DLIN assumption. The commitment is also randomizable and homomorphic.

The linear flippable commitment for scalars $m \in \mathbb{Z}_q$ is defined by the following tuple of efficient algorithms:

Setup_C(1^λ): generates a cyclic multiplicative group $\mathbb{G}$ of order q of size λ , together with a generator g and sets $\text{pp} = (q, \mathbb{G}, g)$.

KeyGen_{FC}(pp, b): takes in pp and chooses four random scalars $\mu_1, \mu_2, \nu_1, \nu_2 \xleftarrow{\$} \mathbb{Z}_q$, then first computes $f_1 = g^{\mu_1}$, $f_2 = g^{\mu_2}$ and sets tk₀ = (μ_1, μ_2) , tk₁ = (ν_1, ν_2) then computes $f_{3,1} = f_1^{\nu_1}$, $f_{3,2} = f_2^{\nu_2}$, $f'_{3,3} = g^{b-1+\nu_1+\nu_2}$ and sets ek = $(f_1, f_2, f_{3,1}, f_{3,2}, f'_{3,3})$.

Com(ck, m ; (α, β)): using the random scalars $a = (\alpha, \beta) \xleftarrow{\$} \mathbb{Z}_q^2$ computes $d = (d_1, d_2, d_3)$ with $d_1 = f_1^\alpha f_{3,1}^m$, $d_2 = f_2^\beta f_{3,2}^m$, $d_3 = g^{\alpha+\beta} (f'_{3,3})^m$ and outputs (d, a) .

Verify_C(ck, d, a, M): returns 1 if $d = \text{Com}(\text{ck}, m; a)$ and 0 otherwise.

Extract(tk₀, d): using the trapdoor (extractable) key tk = (μ_1, μ_2) returns $g^{-m} = d_1^{-1/\mu_1} d_2^{-1/\mu_2} d_3$.

Equiv(tk₁, d, a, m, m'): if **Verify_C**(ck, d, a, m) = 0 returns $\perp$, returns (d, a') with $a' = (\alpha', \beta')$ such that $\alpha' = \alpha + \nu_1(m' - m)$, $\beta' = \beta + \nu_2(m' - m)$.

Note that in the **ExSetup** generation (when $b = 0$) the commitment is a BBS encryption of g^{-m} with the encryption key (f_1, f_2, g) and the randomness

$\alpha + \nu_1 m$ and $\beta + \nu_2 m$. The extractable property only allows recovering g^{-m} and not the scalar m in general since computing the discrete log is hard if the DLIN assumption holds. In the **TrSetup** generation (when $b = 1$) the commitment is a BBS encryption of $1_{\mathbb{G}}$ for any committed message m which makes the scheme perfectly hiding.

Remark. Both commitment schemes for group elements and scalars might be used together. The **KeyGen** algorithms for both schemes are run with the same bit b in order to gather the same setting **ExSetup** or **TrSetup**. Note that $f_1, f_2, f_{3,1}, f_{3,2}$ contained in **ck** may be common and that $f'_{3,3} = g^{-1} f_{3,3}$ whatever b is.

Later, we will also use a vectorial notation with componentwise multiplication $\odot$ to commit to a group element $M \in \mathbb{G}$. This leads to commitments of the form $C_M = \vec{f}_1^\alpha \odot \vec{f}_2^\beta \odot \vec{f}_3^\gamma \odot \iota(M)$ where $\vec{f}_1 = (f_1, 1, g)$, $\vec{f}_2 = (1, f_2, g)$ and $\vec{f}_3 = \vec{f}_1^{\nu_1} \vec{f}_2^{\nu_2} \iota(g)^b$ with the injection $\iota : \mathbb{G} \rightarrow \mathbb{G}^3$ such that $\iota(X) := (1, 1, X)$.

2.4.4 Schemes with Improved Features

Concise Vector Commitment Schemes [111, 44]

We just saw schemes where it is possible to compute commitments on several messages at once. We then obtained short commitments for several committed values. For example, the multi-Pedersen above gives one element d in $\mathbb{G}$ as a commitment for n values m_i in $\mathbb{Z}_q$. When it comes to revealing an opening, the committer simply returns all the m_i 's along with the used randomness. But what happens now if one only wants to learn an opening of the i -th coordinate? It seems difficult to provide an opening of size smaller than n for a componentwise opening. With the generalized Pedersen, receiving m_i and $d_i := g^r \prod_{j \neq i}^n h_j^{m_j}$ should not convince anyone that m_i was really committed. Indeed, checking the equation $d \stackrel{?}{=} d_i \cdot h_i^{m_i}$ would also work with any m'_i with $d'_i := d/h_i^{m'_i}$, which does not lead to a “component” binding commitment in this case. For now, we lose the advantage of committing to a vector at once rather than computing n individual commitments that can be opened separately.

A solution whereby the size of the commitment and the size of the opening are both *concise* is called a concise vector commitment scheme.

LY Concise Vector Commitment

The Libert-Yung concise vector commitment scheme [111] is defined by the following five algorithms (whose formalism was introduced later by [44]):

Setup_C(1^λ): generates **pp** as a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \Lambda_{\text{sym}}(\lambda)$.

KeyGen_C(**pp**, n): chooses $\alpha \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $g_i = g^{\alpha^i}$ for all $i = 1, \dots, n, n+2, \dots, 2n$. It returns $(\text{ck}, \text{tk}) = ((g, g_1, \dots, g_n, g_{n+2}, \dots, g_{2n}), \alpha)$.

- Com**(ck, m , r): for a n -vector message $m = (m_1, \dots, m_n) \in \mathbb{Z}_p^n$ and a random scalar $r \xleftarrow{\$} \mathbb{Z}_q$ computes $d = g^r \prod_{i=1}^n g_{n+1-i}^{m_i} = g^r \cdot g_n^{m_1} \cdots g_1^{m_n} \in \mathbb{G}$ and outputs (d, r) .
- Open**(ck, i , d , a , m): parses m into $(m_1, \dots, m_n)$ and computes the opening $a_i = g_i^a \prod_{j \neq i} g_{n+1+i-j}^{m_j}$, then returns (i, d, a_i, m_i) .
- Equiv**(tk, i , d , a_i , m_i , m'_i): knowing tk = α all the discrete log's are known, then proceeds as the multi-Pedersen commitment and runs **Open** on the additional index i .
- Verify**_C(ck, i , d , a_i , m_i): checks whether $e(g_i, d) = e(g_1, g_n)^{m_i} \cdot e(g, a_i)$ or not and returns 1 or 0 consistently.

The LY commitment scheme is perfectly hiding and computationally componentwise binding under the q -DHE assumption in $\mathbb{G}$ (cf. Assumption 10).

2.5 Cryptographic Proof Systems

Proof systems or arguments allow a prover to convince a verifier that a given string belongs to a predefined language. In cryptography, the prover has an additional private element which enables him/her to *efficiently* perform a valid proof. This element, called a witness, is usually the solution of a hard problem in the verifier's view – otherwise he/she could compute the witness and makes the proof himself/herself. The language can then be interpreted as the existence of a witness in relation with the given string. Furthermore, since the witness is most of the time a prover's sensitive data, as far as possible, performing a proof should not leak information about this witness. We only want to reveal the validity of the proof.

2.5.1 Definition and Security Notions

A family of relations $\mathcal{R}$ is hard if, for a security parameter λ , given a relation $R \leftarrow \mathcal{R}(\lambda)$ it is efficient to compute a pair (x, w) such that $R(x, w) = 1$, also noted $(x, w) \in R$, but, given only x it is hard to recover an w such that $R(x, w) = 1$ in the sense that the probability of computing w efficiently is negligible in λ . Usually, we simply say that R is a hard relation.

As an example, one-way function (cf. Definition 3) may consist of a hard relation and in particular, since the exponentiation in base g in a cyclic group $(p, \mathbb{G}, g)$ is one-way under the DLOG assumption (cf. Assumption 1), we can define a hard relation for discrete logarithms.

A hard relation R defines a language $L_R = \{x \mid \exists w : (x, w) \in R\}$. Proving a statement $x \in L_R$ comes to prove that $R(x, w) = 1$ for some witness w . Finally, we note that the relation R can be defined from public parameters $\text{pp} \leftarrow \text{Setup}(\lambda)$ so that R is generated as $R \leftarrow \mathcal{R}(\text{pp})$ exactly as we did for the KeyGen algorithms of the different schemes introduced before.

Definition 16 (NIZK Proofs [19, 18]) A non-interactive zero-knowledge (NIZK) proof system $\Pi_{\mathcal{P}}$ for a family of hard relations $\mathcal{R}$ is defined by the following algorithms ($\text{Setup}_{\mathcal{P}}$, $\text{KeyGen}_{\mathcal{P}}$, Prove , $\text{Verify}_{\mathcal{P}}$).

Setup $_{\mathcal{P}}$ (1^λ): from the security parameter λ , generates the public parameters pp of the proof system;

KeyGen $_{\mathcal{P}}$ (pp): takes in pp and outputs the common reference string crs that are public elements helping performing a proof for $R \leftarrow \mathcal{R}(\text{pp})$

Prove(crs, x, w): computes a proof π for x using the public crs and the private witness w .

Verify $_{\mathcal{P}}$ (crs, x, π): returns either 1 or 0 if π is a valid proof associated to the language L_R .

The NIZK proof system $\Pi_{\mathcal{P}}$ has the following properties:

Perfect Completeness: for any PPT adversary $\mathcal{A}_1$,

$$\Pr[\text{pp} \leftarrow \text{Setup}_{\mathcal{P}}(\lambda); \text{crs} \leftarrow \text{KeyGen}_{\mathcal{P}}(\text{pp}); (x, w) \leftarrow \mathcal{A}_1(\text{crs}); \\ \pi \leftarrow \text{Prove}(\text{crs}, x, w) : \text{Verify}(\text{crs}, x, \pi) = 1 \text{ if } R(x, w) = 1] = 1,$$

Computational Soundness: for any PPT adversary $\mathcal{A}_2$,

$$\Pr[\text{pp} \leftarrow \text{Setup}_{\mathcal{P}}(\lambda); \text{crs} \leftarrow \text{KeyGen}_{\mathcal{P}}(\text{pp}); (x, \pi) \leftarrow \mathcal{A}_2(\text{crs}) : \\ \text{Verify}_{\mathcal{P}}(\text{crs}, x, \pi) = 1 \wedge \neg(\exists w : R(x, w) = 1)] \in \text{negl}(\lambda),$$

If $\mathcal{A}_2$ is not bounded in time, the property is named statistical soundness. We often talk about arguments rather than proofs for a system that is only computationally sound.

Zero-Knowledge: there exists a PPT simulator (S_1, S_2) such that, for any PPT adversary $\mathcal{A}_3$,

$$\Pr[\text{pp} \leftarrow \text{Setup}_{\mathcal{P}}(\lambda); (\text{crs}, \tau) \leftarrow S_1(\text{pp}) : \mathcal{A}_3^{S_2(\text{crs}, \tau, \cdot)}(\text{crs}) = 1] \\ \approx \Pr[\text{pp} \leftarrow \text{Setup}_{\mathcal{P}}(\lambda); \text{crs} \leftarrow \text{KeyGen}_{\mathcal{P}}(\text{pp}) : \mathcal{A}_3^{\text{P}(\text{crs}, \cdot)}(\text{crs}) = 1],$$

- $\text{P}(\text{crs}, \cdot, \cdot)$ emulates the actual prover. It takes as input (x, w) and outputs a proof π if $(x, w) \in R$. Otherwise, it outputs $\perp$,
- $S_2(\text{crs}, \tau, \cdot, \cdot)$ is an oracle that takes as input (x, w) and outputs a simulated proof $S_2(\text{crs}, \tau, x)$ if $(x, w) \in R$ and $\perp$ if $(x, w) \notin R$.

The above zero-knowledge (ZK) property is computational since $\mathcal{A}_3$ is restricted to be efficient. Without this restriction we talk about statistical ZK or even perfect ZK if the distributions are perfectly indistinguishable.

At a high level, the ZK property says that the information you get from an honest proof is the same as a simulated proof where no witness is used.

In that sense, we can say that a proof for x do not leak information about witnesses. Seeing a proof for $x \in L_R$ does not give any new information than those computable from the relation R . Often, a relaxed notion called *witness-indistinguishability* suffices. It requires that the distributions of proofs for x are indistinguishable regardless of the witness. So a proof could reveal some information about all the solution/witnesses but not the one used by the prover.

Definition 17 (Witness Indistinguishability) *A non-interactive proof system $\Pi_{\mathcal{P}} = (\text{Setup}_{\mathcal{P}}, \text{KeyGen}_{\mathcal{P}}, \text{Prove}, \text{Verify}_{\mathcal{P}})$ for a hard relation R is witness-indistinguishable (NIWI) if, for any PPT adversary $(\mathcal{A}_4, \mathcal{A}_5)$, for any $\text{pp} \leftarrow \text{Setup}_{\mathcal{P}}(\lambda)$ and $\text{crs} \leftarrow \text{KeyGen}_{\mathcal{P}}(\text{pp})$,*

$$\begin{aligned} & \Pr[(x, w_0, w_1, \text{st}) \leftarrow \mathcal{A}_4(\text{crs}); \pi \leftarrow \text{Prove}(\text{crs}, x, w_0) : \mathcal{A}_5(\pi, \text{st}) = 1] \\ & \approx \Pr[(x, w_0, w_1, \text{st}) \leftarrow \mathcal{A}_4(\text{crs}); \pi \leftarrow \text{Prove}(\text{crs}, x, w_1) : \mathcal{A}_5(\pi, \text{st}) = 1], \end{aligned}$$

where $(x, w_0), (x, w_1) \in R$.

The ZK property involves a simulator that generates a trapdoor which allows computing true statements without witness. Sometimes, a trapdoor also allows computing fake proofs, *i.e.* proofs that pass the verification though $x \notin L_R$. At first sight, it may be surprising to consider proof systems with such a property. However, it is a very useful theoretic tool for building CCA-secure cryptosystems, for example based on the Naor-Yung/Sahai [120, 129] paradigm. Seeing a false proof should not break the soundness.

Definition 18 (Simulation Soundness [129]) *A non-interactive proof system $\Pi_{\mathcal{P}} = (\text{Setup}_{\mathcal{P}}, \text{KeyGen}_{\mathcal{P}}, \text{Prove}, \text{Verify}_{\mathcal{P}})$ for a hard relation R is simulation-sound if there exists a PPT simulator (S_1, S_2) such that, for any PPT adversary $\mathcal{A}_6$,*

$$\begin{aligned} & \Pr[\text{pp} \leftarrow \text{Setup}_{\mathcal{P}}(\lambda); (\text{crs}, \tau) \leftarrow S_1(\text{pp}); (x, \pi) \leftarrow \mathcal{A}_6^{S_2(\text{crs}, \tau, \dots)}(\text{crs}) : \\ & \text{Verify}_{\mathcal{P}}(\text{crs}, x, \pi) = 1 \wedge \neg(\exists w : R(x, w) = 1) \wedge (x, \pi) \notin Q] \in \text{negl}(\lambda) \end{aligned}$$

where the adversary is allowed a one-time query to an oracle $S_2(\text{crs}, \tau, \cdot)$ that takes as input a statement x (where x may be outside L_R) and outputs a simulated proof $\pi \leftarrow S_2(\text{crs}, \tau, x)$ before setting $Q = \{(x, \pi)\}$, which is initially empty.

The soundness still holds but only in a computational manner.

2.5.2 Non Interactive Sigma Protocol

Definition 19 (Sigma Protocol [48]) *A sigma protocol for an NP language $\mathcal{L} : \{s \mid \exists w : L(s, w) = 1\}$ is a pair of interactive algorithms (P, V) that work as follows. On input (s, w) for P and s for V , the following three-move interaction takes place:*

1. P outputs a “commitment” $comm$ to the verifier.
2. V selects a challenge $chal$ uniformly at random from a challenge space and sends it to the prover.
3. P sends a “response” $resp$ and halts.

Eventually, V evaluates a predicate Verify on the statement s and the transcript $(a, e, z) := (comm, chal, resp)$ and returns 0 or 1, then halts.

Sigma protocols must satisfy the following security properties:

Completeness An honest run between $P(s, w)$ and $V(s)$ always accepts if $L(s, w) = 1$

Special soundness There exists an extractor that, on the input of two valid transcripts (a, e_0, z_0) and (a, e_1, z_1) w.r.t. s , returns the correct witness w . As a result, it is hard to produce invalid proofs for language based on hard relation.

Special honest verifier ZK It is possible to simulate any proof for x made with an honest verifier and it is possible to do so for any given challenge e .

Perfect ZK The simulated proofs are distributed just as honest ones.

Unique response For any statement, commitment and challenge, there exists at most one response that would lead to an accepting transcript.

Sigma protocols are interactive witness-indistinguishable proofs. Known techniques transform Sigma protocols (SHVZK) into interactive zero-knowledge proofs (in face of dishonest verifiers too). [54, 73]. Sigma protocols can also be turned into NIZK proofs based on the Fiat-Shamir heuristic [63] (assuming the random-oracle model).

Example. For a Pedersen commitment $x = g^r g_1^{m_1} \cdots g_n^{m_n} \in \mathbb{G}$ with witness $w = (r, m_1, \dots, m_n)$, compute the commitment $a = g^s g_1^{y_1} \cdots g_n^{y_n}$. Then, use a hash function (randomly taken in a family) $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ to compute the challenge $e = H(ck, x, a)$ and the response $z = (t, z_1, \dots, z_n) \in \mathbb{Z}_p^{n+1}$ where $t = er + s$ and, for $i = 1$ to n , $z_i = em_i + y_i$. Return (x, e, z) . The verifier checks if $e = H(ck, x, a')$ where $a' = x^{-e} (g^t g_1^{z_1} \cdots g_n^{z_n})$.

The size of the proof is linear in the number of variables.

2.5.3 The Groth-Sahai Methodology

Most of our protocols make use of the Groth-Sahai (GS) methodology in order to provide evidence of the possession of elements verifying a pairing-product equation

$$\prod_{j=1}^n e(\mathcal{A}_j, \mathcal{Y}_j) \cdot \prod_{i=1}^m e(\mathcal{X}_i, \mathcal{B}_i) \cdot \prod_{i=1}^m \prod_{j=1}^n e(\mathcal{X}_i, \mathcal{Y}_j)^{\gamma_{ij}} = t_T, \quad (2.12)$$

for variables $\mathcal{X}_1, \dots, \mathcal{X}_m \in \mathbb{G}_1$, $\mathcal{Y}_1, \dots, \mathcal{Y}_n \in \mathbb{G}_2$ and constants $t_T \in \mathbb{G}_T$, $\mathcal{A}_1, \dots, \mathcal{A}_m \in \mathbb{G}_2$, $\mathcal{B}_1, \dots, \mathcal{B}_n \in \mathbb{G}_1$, $\gamma_{ij} \in \mathbb{Z}_p$, for $(i, j) \in \{1, \dots, m\} \times \{1, \dots, n\}$, or multi-exponentiation equations in $\mathbb{G}_1$ and in $\mathbb{G}_2$

$$\prod_{j=1}^n \mathcal{A}_j^{y_j} \cdot \prod_{i=1}^m \mathcal{X}_i^{b_i} \cdot \prod_{i=1}^m \prod_{j=1}^n \mathcal{X}_i^{y_j \gamma_{ij}} = T, \quad (2.13)$$

for variables $\mathcal{X}_1, \dots, \mathcal{X}_n \in \mathbb{G}_1$, $y_1, \dots, y_m \in \mathbb{Z}_p$ and constants $T, \mathcal{A}_1, \dots, \mathcal{A}_m \in \mathbb{G}_1$, $b_1, \dots, b_n \in \mathbb{Z}_p$ and $\gamma_{ij} \in \mathbb{Z}_p$, for $i \in \{1, \dots, m\}, j \in \{1, \dots, n\}$, and analogously for $\mathbb{G}_2$.

In [84] Groth and Sahai proposed a general way of making such kind of proofs. They interpreted both previous examples as particular cases of a quadratic formula where all the operations are noted additively:

$$\sum_{j=1}^n f(a_j, y_j) + \sum_{i=1}^m f(x_i, b_i) + \sum_{i=1}^m \sum_{j=1}^n \gamma_{ij} f(x_i, y_j) = t_T, \quad (2.14)$$

where $f(x, y)$ is a bilinear map, and with the same convention for constants and variables. Then, for pairing product equation (2.12) f is simply the pairing map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ and for the multi-exponentiation equation (2.13) in $\mathbb{G}_i$ we can set $f_i : \mathbb{G}_i \times \mathbb{Z}_p \rightarrow \mathbb{G}_i$ such that $f_i(a, y) = a^y$, for $i = 1$ or 2 .

Furthermore, by the linearity of the map f in the second component, the double sum of the equation (2.14) can also be written as $\sum_{i=1}^m f(x_i, \sum_{j=1}^n \gamma_{ij} \cdot y_j)$ so that we obtain a compressed vectorial form of the equation

$$\vec{a} \cdot \vec{y} + \vec{x} \cdot \vec{b} + \vec{x} \cdot \Gamma \vec{y} = t_T, \quad (2.15)$$

where the dot product is defined for two equal-dimension vectors such that for $\vec{a} = (a_1, \dots, a_n)^\top$ and $\vec{y} = (y_1, \dots, y_n)^\top$ we have

$$\vec{a} \cdot \vec{y} := \sum_{j=1}^n f(a_j, y_j),$$

and the $m \times n$ matrix Γ has the element γ_{ij} at entry (i, j) .

The GS Proof System

Let $(A, +, 0)$ be an abelian group and $(\mathcal{R}, +, \cdot, 0_{\mathcal{R}}, 1_{\mathcal{R}})$ be a finite commutative ring with an action such that, for all r, s in $\mathcal{R}$ and for all x, y in A we have

$$(r + s)x = rx + sx \quad \wedge \quad r(x + y) = rx + ry \quad \wedge \quad r(sx) = (rs)x \quad \wedge \quad 1_{\mathcal{R}}x = x.$$

That means that A together with $\mathcal{R}$ form what is called an $\mathcal{R}$ -module. A cyclic group of order n naturally defines a $\mathbb{Z}_n$ -module.

To properly define the quadratic equation (2.15), the setup algorithm generates three $\mathcal{R}$ -modules A_1 , A_2 and A_T on the same ring with a bilinear map

$f : A_1 \times A_2 \rightarrow A_T$. An instance of the problem is given by the constants $\text{inst}_{(m,n)} = (\vec{a}, \vec{b}, \Gamma, t_T)$ such that $\vec{a}^\top \in A_1^n$, $\vec{b}^\top \in A_2^m$, $\Gamma \in \mathcal{R}^{m \times n}$ and $t_T \in A_T$. A witness w for the proof (or argument) of satisfiability of the quadratic equation *w.r.t.* the instance $\text{cst} = \text{inst}_{(m,n)}$ is a solution $\text{sol} = (\vec{x}, \vec{y})$ with $\vec{x}^\top \in A_1^m$ and $\vec{y}^\top \in A_2^n$. It induces a relation R_{GS} such that $R_{\text{GS}}(\text{cst}, \text{sol}) = 1$ with the associated language $\mathcal{L}_{\text{GS}} = \{\text{cst} = \text{inst}_{(m,n)} \mid \exists w = \text{sol} : R_{\text{GS}}(\text{cst}, \text{sol}) = 1\}$.

The membership proof works on commitments on each components x_i and y_j . They are of the form

$$\begin{aligned} c_i &= \iota_1(x_i) + \sum_{k=1}^{m'} r_{ik} u_k \in B_1, \\ d_j &= \iota_2(y_j) + \sum_{k=1}^{n'} s_{jk} v_k \in B_2, \end{aligned}$$

where $\iota_1 : A_1 \rightarrow B_1$ and $\iota_2 : A_2 \rightarrow B_2$ are injections into two other $\mathcal{R}$ -modules, while $\vec{u}^\top = (u_1, \dots, u_{m'}) \in B_1^{m'}$ and $\vec{v}^\top = (v_1, \dots, v_{n'}) \in B_2^{n'}$ are committing keys generated by the CRS generator.

Since verification is made on the commitments it must have another bilinear map $F : B_1 \times B_2 \rightarrow B_T$ into an $\mathcal{R}$ -module B_T which follows the properties of f . Then, for all $x \in A_1$ and $y \in A_2$,

$$F(\iota_1(x), \iota_2(y)) = \iota_T(f(x, y))$$

where $\iota_T : A_T \rightarrow B_T$ is a third injection. We also use a similar dot-product notation as for f : for any $\vec{a}^\top = (a_1, \dots, a_n) \in B_1^n$ and $\vec{y}^\top = (y_1, \dots, y_n) \in B_2^n$ we have

$$\vec{a} \bullet \vec{y} := \sum_{j=1}^n F(a_j, y_j)$$

in B_T , and then for the $m \times n$ matrix Γ we also have $\vec{a} \bullet \Gamma \vec{y} = \Gamma^\top \vec{a} \bullet \vec{y}$.

Setup $_{\mathcal{P}}$ (1^λ): from the security parameter λ , generates $\text{pp} := (\mathcal{R}, A_1, A_2, A_T, f)$ as above where the order of each groups is at least 2^λ ;

KeyGen $_{\mathcal{P}}$ (pp, b): takes in pp and outputs the common reference string crs according to the desired setting: perfect soundness¹ if $b = 0$ and witness-indistinguishability if $b = 1$.

Generates the $\mathcal{R}$ -modules B_1, B_2, B_T , the bilinear map F satisfying the above commutative law $F \circ (\iota_1 \times \iota_2) = \iota_T \circ f$ with respect to the linear maps $\iota_1, \iota_2, \iota_T$ and the committing keys $\vec{u}^\top \in B_1^{m'}$ and $\vec{v}^\top \in B_2^{n'}$. Also generates the linear maps p_1, p_2, p_T in the opposite direction of $\iota_1, \iota_2, \iota_T$ so that $p_T \circ F = f \circ (p_1 \times p_2)$

- In the perfect soundness setting: committing keys make the commitment perfectly binding. In particular we must have $p_1(\vec{u}) = \vec{0}$ and $p_2(\vec{v}) = \vec{0}$ (and also $\text{Im } \iota_i \cap \text{Ker } p_i = \{0\}$, for $i = 1, 2, T$).

¹In the original paper, the soundness property is not perfect in the general setting. We only focus on perfect soundness to facilitate the understanding of the system. All our protocols make use of a perfectly sound instantiation.

- In the witness-indistinguishability setting: committing keys make the commitment perfectly hiding. In particular we must have $\text{Im } \iota_1 \subseteq \langle u_1, \dots, u_{m'} \rangle$ and $\text{Im } \iota_2 \subseteq \langle v_1, \dots, v_{n'} \rangle$.

Finally, generates a set of matrices $H_1, \dots, H_\eta \in \mathcal{R}^{m' \times n'}$ that generate all the matrices $H \in \mathcal{R}^{m' \times n'}$ such that $\vec{u} \bullet H \vec{v} = 0$, when the committing keys $\vec{u}, \vec{v}$ are generated in the WI setting. The crs is given by

$$\text{crs} = (\text{pp}, B_1, B_2, B_T, F, \iota_1, p_1, \iota_2, p_2, \iota_T, p_T, \vec{u}, \vec{v}, H_1, \dots, H_\eta).$$

Prove($\text{crs}, \text{cst}, \text{sol}$): generates the vectors $\vec{c} \in B_1^m$, $\vec{d} \in B_2^n$ of commitments on $\vec{x}^\top \in A_1^m$, $\vec{y}^\top \in A_2^n$ from $\text{sol} = (\vec{x}, \vec{y})$ as

$$\vec{c} = \iota_1(\vec{x}) + R\vec{u}, \quad \vec{d} = \iota_2(\vec{y}) + S\vec{v},$$

for random matrices $R \leftarrow \mathcal{R}^{m \times \hat{m}}$ and $S \leftarrow \mathcal{R}^{n \times \hat{n}}$. Picks also $T \leftarrow \mathcal{R}^{\hat{n} \times \hat{m}}$ and $\rho_1, \dots, \rho_\eta \leftarrow \mathcal{R}$ at random. Then, computes

$$\vec{\pi}_1 = S^\top \iota_1(\vec{a}) + S^\top \Gamma^\top \iota_1(\vec{x}) + T\vec{u} \quad (2.16)$$

$$\vec{\pi}_2 = R^\top \iota_2(\vec{b}) + R^\top \Gamma \iota_2(\vec{y}) + R^\top \Gamma S \vec{v} - T^\top \vec{v} + \sum_{i=1}^{\eta} \rho_i H_i \vec{v} \quad (2.17)$$

and returns the commitment $(\vec{c}, \vec{d})$ and the proof $(\vec{\pi}_1, \vec{\pi}_2) \in B_1^n \times B_2^m$.

Verify $_{\mathcal{P}}$ ($\text{crs}, \text{cst}, \pi$): returns 1 if and only if

$$\iota_1(\vec{a}) \bullet \vec{d} + \vec{c} \bullet \iota_2(\vec{b}) + \vec{c} \bullet \Gamma \vec{d} = \iota_T(t_T) + \vec{u} \bullet \vec{\pi}_2 + \vec{\pi}_1 \bullet \vec{v}.$$

It is worth observing that the maps p_1, p_2, p_T are not necessary to perform **Prove** and **Verify**. Their role is to provide the soundness of the scheme. The role of the H -matrices is to show the witness-indistinguishability of the proof system.

Theorem 5 ([84]) *In the witness-indistinguishable setting the GS proof system is a perfect NIWI argument for membership in $\mathcal{L}_{\text{GS}}$ and, in the soundness setting the GS proof system is a non-interactive membership proof for the language $\mathcal{L}_{\text{GS}}$.*

As an important corollary, if the witness-indistinguishable setting and the soundness setting are computationally indistinguishable then, in either setting the GS proof system achieves all these security properties at least computationally. The proof system can be easily extended to *systems* of quadratic equations.

Particular cases. There are two interesting cases:

1. **Linear case.** When $\vec{a} = 0^m$ and $\Gamma = 0^{m \times n}$, the random matrix T can be recomputed from $\vec{\pi}_1$ in 2.16 and then publicly removed from $\vec{\pi}_2$ in 2.17, so that it can simply set to $0^{\hat{m} \times \hat{n}}$ in that case without degrading the security properties. The proof is reduced to

$$\vec{\pi}_{\text{lin}} = R^\top \iota_2(\vec{b}) + \sum_{i=1}^{\eta} \rho_i H_i \vec{v} \quad (2.18)$$

and the verification algorithm checks whether $\vec{c} \bullet \iota_2(\vec{b}) = \iota_T(t_T) + \vec{u} \bullet \vec{\pi}$ holds.

2. **Symmetric case.** When $B := B_1 = B_2$ and the bilinear map F is symmetric the GS proof is also made shorter. Let us suppose that $\hat{m} \leq \hat{n}$ and $\vec{u} = \vec{v}_{|\hat{m}}$. The proof is given by $\vec{\pi} := \vec{\pi}_1 + \vec{\pi}'_2 \in B^{\hat{n}}$ where $\vec{\pi}'_2$ is the vector $\vec{\pi}_2$ padded with $\hat{n} - \hat{m}$ 0's, and the verification checks if $\iota_1(\vec{a}) \bullet \vec{d} + \vec{c} \bullet \iota_2(\vec{b}) + \vec{c} \bullet \Gamma \vec{d} = \iota_T(t_T) + \vec{\pi} \bullet \vec{v}$. This is correct since $\vec{\pi} \bullet \vec{v} = \vec{\pi}_1 \bullet \vec{v} + \vec{\pi}'_2 \bullet \vec{v}$ of which the last term is $\vec{u} \bullet \vec{\pi}_2$ by the symmetric property of F . Another way to perform the proof $\vec{\pi}$ is to replace $\vec{u}$ by $\vec{v}$ in $\vec{\pi}_1$ and extend the matrices R to R' and T to T' with $\hat{n} - \hat{m}$ 0-columns and the matrices $H_1, \dots, H_\eta$ to $H'_1, \dots, H'_\eta$ with $\hat{n} - \hat{m}$ 0-rows. This leads to

$$\begin{aligned} \vec{\pi}_{\text{sym}} &= S^\top \iota_1(\vec{a}) + S^\top \Gamma^\top \iota_1(\vec{x}) + R'^\top \iota_2(\vec{b}) + R'^\top \Gamma \iota_2(\vec{y}) \\ &\quad + R'^\top \Gamma S \vec{v} + \sum_{i=1}^{\eta} \rho_i H'_i \vec{v} \end{aligned} \quad (2.19)$$

There is no need of T matrix in the proof because otherwise the verification equation with the paddings would contain the term $\vec{v} \bullet (T' - (T')^\top) \vec{v}$ which is equal to 0 since the matrix $T' - (T')^\top$ is antisymmetric and F is symmetric. Then $(T' - (T')^\top)$ would lie in $\langle H'_1, \dots, H'_\eta \rangle$.

Linear equations have proofs that only depend on the randomness of the commitments and not on the committed witnesses or on the right-hand side member of the equation in A_T . As observed in [70], this allows a form of malleability.

In both cases the proofs are randomizable. While this is obvious for linear equations, let us see how proofs in the symmetric setting can be publicly rerandomized. For an honestly generated proof $\vec{\pi}_{\text{sym}}$ for a given crs and constants $\text{cst} = (\vec{a}, \vec{b}, \Gamma, t_T)$ with commitments $\vec{c}$ and $\vec{d}$, proceed as follows.

1. Rerandomize the commitments with random $\bar{R} \leftarrow \mathcal{R}^{m \times \hat{m}}, \bar{S} \leftarrow \mathcal{R}^{n \times \hat{n}},$

$$\vec{c}' = \vec{c} + \bar{R} \vec{u}, \quad \vec{d}' = \vec{d} + \bar{S} \vec{v}.$$

Set $\bar{R}'$ as the $\bar{R}$ matrix padded with $\hat{n} - \hat{m}$ 0-columns.

2. Return the commitments $\vec{c}', \vec{d}'$ along with the rerandomized proof

$$\begin{aligned} \vec{\pi}'_{\text{sym}} = & \vec{\pi}_{\text{sym}} + \bar{S}^\top \iota_1(\vec{a}) + \bar{S}^\top \Gamma^\top \iota_1(\vec{c}) \\ & + \bar{R}'^\top \iota_2(\vec{b}) + \bar{R}'^\top \Gamma \iota_2(\vec{d}) + \bar{R}'^\top \Gamma \bar{S} \vec{v} + \sum_{i=1}^{\eta} \bar{\rho}_i H'_i \vec{v} \end{aligned} \quad (2.20)$$

for fresh $\bar{\rho}_1, \dots, \bar{\rho}_\eta \leftarrow \mathcal{R}$.

In the symmetric setting, the resulting proofs $\vec{\pi}'_{\text{sym}}$ and commitments $\vec{c}', \vec{d}'$ have the same distribution as those generated by the **Prove** algorithm. Indeed, $\vec{c}' = \iota_1(\vec{x}) + \hat{R}' \vec{v}$ for a random $\hat{R} = R + \bar{R}$ extended to $\hat{R}'$ with $\hat{n} - \hat{m}$ 0-columns, $\vec{d}' = \iota_2(\vec{y}) + \hat{S} \vec{v}$ for a random $\hat{S} = S + \bar{S}$ and

$$\vec{\pi}'_{\text{sym}} = \hat{S}^\top \iota_1(\vec{a}) + \hat{S}^\top \Gamma^\top \iota_1(\vec{x}) + \hat{R}'^\top \iota_2(\vec{b}) + \hat{R}'^\top \Gamma \iota_2(\vec{y}) + \hat{R}'^\top \Gamma \hat{S} \vec{v} + \hat{H} \vec{v},$$

where $\hat{H} = ((R' \Gamma \bar{S})^\top - R' \Gamma \bar{S}) + \sum_{i=1}^{\eta} (\rho_i + \bar{\rho}_i) H'_i$ is random in $\langle H'_1, \dots, H'_\eta \rangle$ because the first constant term is antisymmetric and F symmetric in the setting.

DLIN-based Instantiation In Symmetric Pairing

In this section we apply the Groth-Sahai methodology in pairing environments with $\mathbb{G} := \mathbb{G}_1 = \mathbb{G}_2$ of prime order p . Then, $\mathbb{G}$ and $\mathbb{G}_T$ are $\mathbb{F}_p$ -vector spaces. We will consider GS-proofs for the pairing-product equations 2.12 and the multi-exponentiation equations 2.13. We detail each kind of proof with their particular cases by specifying the crs and we will also provide some examples.

Due to the symmetric property of the pairing, the pairing-product equation (PPE) may be written

$$\prod_{i=1}^n e(\mathcal{A}_i, \mathcal{Y}_i) \cdot \prod_{i=1}^n \cdot \prod_{j=1}^n e(\mathcal{Y}_i, \mathcal{Y}_j)^{\gamma_{ij}} = t_T, \quad (2.21)$$

for variables $\mathcal{Y}_1, \dots, \mathcal{Y}_n \in \mathbb{G}$ and constants $t_T \in \mathbb{G}_T$, $\mathcal{A}_1, \dots, \mathcal{A}_n \in \mathbb{G}$, $\gamma_{ij} \in \mathbb{Z}_p$, for $i, j \in \{1, \dots, n\}$. Interestingly, the commitments on a group element $\mathcal{X}_i \in \mathbb{G}$ can also be re-used for the multi-exponentiation equation (MEE)

$$\prod_{i=1}^m \mathcal{B}_i^{x_i} \cdot \prod_{j=1}^n \mathcal{Y}_j^{a_j} \cdot \prod_{i=1}^m \cdot \prod_{j=1}^n \mathcal{Y}_j^{x_i \gamma'_{ij}} = T, \quad (2.22)$$

for variables $\mathcal{Y}_1, \dots, \mathcal{Y}_n \in \mathbb{G}$, $x_1, \dots, x_m \in \mathbb{Z}_p$ and constants $T, \mathcal{B}_1, \dots, \mathcal{B}_m \in \mathbb{G}$, $a_1, \dots, a_n \in \mathbb{Z}_p$ and $\gamma'_{ij} \in \mathbb{Z}_p$, for $i \in \{1, \dots, m\}, j \in \{1, \dots, n\}$. Consequently, a global crs can be used to commit to the variables in $\mathbb{G}$ and the variables in $\mathbb{Z}_p$ if a system of equations involved both kinds of equations. To prove that committed variables satisfy a set of relations, the prover computes one commitment per variable and one proof element (made of a constant number of group elements) per relation.

Setup_P(1^λ): from the security parameter λ , generates the parameters of a pairing environment $\mathbf{pp} := (p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \Lambda_{\text{sym}}(\lambda)$. For a PPE equation the parameters for the GS-proof are $\mathbf{pp}_{\text{PPE}} := (\mathbb{Z}_p, \mathbb{G}, \mathbb{G}, \mathbb{G}_T, e)$ and for the MEE equation the parameters are $\mathbf{pp}_{\text{MEE}} := (\mathbb{Z}_p, \mathbb{Z}_p, \mathbb{G}, \mathbb{G}, f)$ where $f(x, Y) = Y^x$.

KeyGen_P($\mathbf{pp}, b$): takes in $\mathbf{pp}$ and runs **KeyGen_{FC}**($\mathbf{pp}, b$) of the linear flippable commitments for group elements and scalars to obtain a tuple of group elements $(f_1, f_2, f_{3,1}, f_{3,2}, f_{3,3}, f'_{3,3}) \in \mathbb{G}^6$.

Defines the vectors $\vec{f}_1 = (f_1, 1, g)$, $\vec{f}_2 = (1, f_2, g)$, $\vec{f}_3 = (f_{3,1}, f_{3,2}, f_{3,3})$ and $\vec{\varphi} = (f_{3,1}, f_{3,2}, f'_{3,3})$ in $\mathbb{G}^3$ so that for any type of equation $B_1 = B_2 = \mathbb{G}^3$ with componentwise multiplication of $\mathbb{G}$. For group element Y sets the linear injection $\iota(X) := (1, 1, X)$ and for scalar x set $\iota'(x) := \vec{\varphi}^x$. Then, $\vec{f}_3 = \iota(g)^b \cdot \vec{f}_1^{\iota_1} \cdot \vec{f}_2^{\iota_2}$ and $\vec{\varphi} = \iota(g)^{-1} \cdot \vec{f}_3$. Sets also $p = p'$ as **Extract**($\text{tk}_0, \cdot$) of the linear commitment for $\text{tk}_0 = (\mu_1, \mu_2)$.

Sets two bilinear maps $F, \tilde{F}$ from $\mathbb{G}^3 \times \mathbb{G}^3$ into $\mathbb{G}_T^{3 \times 3}$. Both are used for PPE and MEE so that $B_T \cong \mathbb{G}_T^9$ with componentwise multiplication of $\mathbb{G}_T$. For $X = (X_1, X_2, X_3)$ and $Y = (Y_1, Y_2, Y_3)$ in $\mathbb{G}^3$ define,

$$\tilde{F}(X, Y) := (e(X_i, Y_j))_{1 \leq i, j \leq 3}, \quad (2.23)$$

$$F(X, Y) := \tilde{F}(X, Y)^{1/2} \cdot \tilde{F}(Y, X)^{1/2}. \quad (2.24)$$

Several situations have to be considered:

PPE For this type of equation, we set $\vec{u} = \vec{v} := (\vec{f}_1, \vec{f}_2, \vec{f}_3)^\top$, $\iota_1 = \iota_2 := \iota$ and $\iota_T(Z) \in \mathbb{G}_T^9$ is defined by $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & Z \end{pmatrix}$ for $Z \in \mathbb{G}_T$. In order to obtain a short proof for the general case, we use the symmetric map F to apply the symmetric GS proof. This implies the use of matrices for which entries are viewed as exponents:

$$H_1 = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad H_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}, \quad H_3 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}.$$

For the linear case ($\gamma_{i,j} = 0$ for all $1 \leq i, j \leq n$), the shorter proof uses $\tilde{F}$ since there is no non-trivial matrix $\tilde{H}$ such that $\vec{u} \bullet \tilde{H} \vec{u} = 0$.

MEE For this type of equation, we set $\vec{u} := (\vec{f}_1, \vec{f}_2)^\top$ and $\vec{v} := (\vec{f}_1, \vec{f}_2, \vec{f}_3)^\top$, and, $\iota_1 := \iota'$ and $\iota_2 := \iota$. We make the same distinction between symmetric F and asymmetric $\tilde{F}$ maps for general and linear equations.

For the general MEE, we set $\iota'_T(Y) := F(\vec{\varphi}, \iota(Y))$ for $Y \in \mathbb{G}$. In the WI setting, $H'_1 = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$ generates all the matrices $H' \in \mathbb{Z}_p^{2 \times 3}$ satisfying $\vec{u} \bullet H' \vec{v} = 0$. In the symmetric GS proof, *i.e.* padded with one 0-row, this gives back H_1 since the equation is equivalent to $\vec{v} \bullet H \vec{v} = 0$.

For the linear MEE, we set $\tilde{\iota}'_T(Y) := \tilde{F}(\vec{\varphi}, \iota(Y))$ for $Y \in \mathbb{G}$. And, there is no need of “H-matrix”.

We intentionally omit to include a description of the linear maps p_T into the **crs** since they are not involved in the next algorithms.

Prove(crs, cst, sol): From $(\vec{f}_1, \vec{f}_2, \vec{f}_3, \vec{\varphi})$ in the **crs**, generates the commitments on all the elements of **sol**, *i.e.* $x \in \mathbb{Z}_p$ and/or $Y \in \mathbb{G}$, as in the linear commitment schemes

$$\vec{C}'_x = \vec{\varphi}^x \cdot \vec{f}_1^{r_1} \cdot \vec{f}_2^{r_2}, \quad \vec{C}_Y = \iota(Y) \cdot \vec{f}_1^{s_1} \cdot \vec{f}_2^{r_2} \cdot \vec{f}_3^{r_3},$$

for random $(r_1, r_2) \leftarrow \mathbb{Z}_p^2$ and/or $(s_1, s_2, s_3) \leftarrow \mathbb{Z}_p^3$. For each equation specified by $(T, \vec{B}, \vec{a}, \Gamma')$ or $(t_T, \vec{A}, \Gamma)$ in **cst**, computes the vectors $\vec{c}$ and/or $\vec{d}$ of the corresponding committed solutions x' and/or Y . In the same order, build the matrices R and/or S of which the lines are composed of the random vectors (r_1, r_2) and/or (s_1, s_2, s_3) used respectively in the commitments $\vec{C}'_x$ and/or $\vec{C}_Y$.

Next, for each tuple $(\vec{c}, \vec{d}, R, S)$ or $(\vec{d}, S)$ associated with its constants, compute the proof relatively to the different cases.

PPE For the commitments $\vec{d}$ with randomness S and constants $(t_T, \vec{A}, \Gamma)$:

- In the general case, picks $\vec{\rho} \leftarrow \mathbb{Z}_p^3$ and computes the proof $\vec{\pi}_{\text{sym}}^{\text{PPE}} \in \mathbb{G}^{3 \times 3}$,

$$\vec{\pi}_{\text{sym}}^{\text{PPE}} = S^\top \iota(\vec{A}) + S^\top (\Gamma^\top + \Gamma) \iota(\vec{Y}) + S^\top \Gamma S \vec{v} + \langle \vec{\rho}, \vec{H} \rangle \vec{v}$$

where $\langle \vec{\rho}, \vec{H} \rangle := \sum_{i=1}^3 \rho_i H_i$ is a random anti-symmetric matrix.

- In the linear case, when $\Gamma = 0^{n \times n}$, computes the proof $\vec{\pi}_{\text{lin}}^{\text{PPE}} \in \mathbb{G}^3$,

$$\vec{\pi}_{\text{lin}}^{\text{PPE}} = S^\top \vec{A}$$

since $\iota(S^\top \vec{A}) = S^\top \iota(\vec{A})$.

MEE For commitments $\vec{c}, \vec{d}$ with randomness R, S and constants $(T, \vec{B}, \vec{a}, \Gamma')$:

- In the general case, picks $\rho' \leftarrow \mathbb{Z}_p$ and computes the proof $\vec{\pi}_{\text{sym}}^{\text{MEE}} \in \mathbb{G}^{3 \times 3}$,

$$\begin{aligned} \vec{\pi}_{\text{sym}}^{\text{MEE}} = & S^\top \iota_1(\vec{a}) + S^\top \Gamma \iota_1(\vec{x}) + R'^\top \iota_2(\vec{b}) \\ & + R'^\top \Gamma \iota_2(\vec{y}) + R'^\top \Gamma S \vec{v} + \rho' H_1 \vec{v} \end{aligned}$$

- In the linear case, when $\Gamma' = 0^{m \times n}$, there are two sub-cases.
 1. If $\vec{B} = 0$, with $\vec{d}, S$ and constants $(T, \vec{a})$, computes $\vec{\pi}_{\text{lin-1}}^{\text{MEE}} \in \mathbb{G}^3$,

$$\vec{\pi}_{\text{lin-1}}^{\text{MEE}} = S^\top \vec{a}$$

since $\iota'(S^\top \vec{a}) = S^\top \iota'(\vec{a})$.

2. If $\vec{a} = 0$, with $\vec{c}, R$ and constants $(T, \vec{B})$, computes $\vec{\pi}_{\text{lin-2}}^{\text{MEE}} \in \mathbb{G}^2$,

$$\vec{\pi}_{\text{lin-2}}^{\text{MEE}} = R^\top \vec{B}$$

since $\iota(R^\top \vec{B}) = R^\top \iota(\vec{B})$.

Where the addition law must be considered as the componentwise multiplication of $\mathbb{G}$ in $\mathbb{G}^{3 \times 3}$.

Verify_P(crs, cst, π): returns 1 if and only if each proof is valid.

PPE For the commitments $\vec{d}$, constants $(t_T, \vec{A}, \Gamma)$ and a proof $\vec{\pi}$:

- For a general pairing-product equation with $\vec{\pi} = \vec{\pi}_{\text{sym}}^{\text{PPE}}$ checks that

$$\iota(\vec{A}) \bullet \vec{d} + \vec{d} \bullet \Gamma \vec{d} = \iota_T(t_T) + \vec{v} \bullet \vec{\pi}_{\text{sym}}^{\text{PPE}}.$$

- For a linear equation $\prod_{i=1}^n e(A_i, Y_i) = t_T$ with $\vec{\pi} = \vec{\pi}_{\text{sym}}^{\text{PPE}}$ checks that

$$\iota(\vec{A}) \tilde{\bullet} \vec{d} = \iota_T(t_T) + \iota(\vec{\pi}_{\text{sym}}^{\text{PPE}}) \tilde{\bullet} \vec{v}.$$

MEE For the commitments $\vec{c}, \vec{d}$, constants $(T, \vec{B}, \vec{a}, \Gamma')$ and a proof $\vec{\pi}$:

- For a general multi-exponentiation equation with $\vec{\pi} = \vec{\pi}_{\text{sym}}^{\text{MEE}}$ checks that

$$\iota'(\vec{a}) \bullet \vec{d} + \vec{c} \bullet \iota_2(\vec{b}) + \vec{c} \bullet \Gamma \vec{d} = \iota'_T(T) + \vec{\pi}_{\text{sym}}^{\text{MEE}} \bullet \vec{v}.$$

- For a linear equation of the form:
 1. $\prod_{i=1}^n Y_i^{a_i} = T$, checks if $\vec{\pi} = \vec{\pi}_{\text{lin-1}}^{\text{MEE}}$ satisfies

$$\iota'(\vec{a}) \tilde{\bullet} \vec{d} = \tilde{\iota}'_T(T) + \iota'(\vec{\pi}_{\text{lin-1}}^{\text{MEE}}) \tilde{\bullet} \vec{v}.$$

2. $\prod_{i=1}^m B_i^{x_i} = T$, checks if $\vec{\pi} = \vec{\pi}_{\text{lin-2}}^{\text{MEE}}$ satisfies

$$\vec{c} \tilde{\bullet} \iota(\vec{B}) = \tilde{\iota}'_T(T) + \vec{u} \tilde{\bullet} \iota(\vec{\pi}_{\text{lin-2}}^{\text{MEE}}).$$

Where the addition law must be considered as the componentwise multiplication of $\mathbb{G}_T$ in $\mathbb{G}_T^{3 \times 3}$.

The randomizability of the GS proofs for linear equation and symmetric setting implies that all the DLIN-based proofs are also publicly randomizable.

Examples

1. Let us see what a proof looks like for the simple pairing-product equation

$$t = e(A, X) \cdot e(Y, Z)^\gamma. \quad (2.25)$$

We have a PPE for a vector of variables $\vec{\mathcal{Y}} = (X, Y, Z)^\top$ with constants

$$t \in \mathbb{G}_T, \quad \vec{\mathcal{A}} = \begin{pmatrix} A \\ 1 \\ 1 \end{pmatrix} \in \mathbb{G}^3, \quad \Gamma = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & \gamma \\ 0 & 0 & 0 \end{pmatrix} \in \mathbb{Z}_p^{3 \times 3}.$$

There are a lot of choices for Γ but² triangular matrices decrease the proof cost. Applying the above description for a vector of commitments $\vec{d} = \iota(\vec{\mathcal{Y}}) + S\vec{v}$,

$$\vec{d} = \begin{pmatrix} \vec{C}_X \\ \vec{C}_Y \\ \vec{C}_Z \end{pmatrix} = \begin{pmatrix} \iota(X) \cdot \vec{f}_1^{x_1} \cdot \vec{f}_2^{x_2} \cdot \vec{f}_3^{x_3} \\ \iota(Y) \cdot \vec{f}_1^{y_1} \cdot \vec{f}_2^{y_2} \cdot \vec{f}_3^{y_3} \\ \iota(Z) \cdot \vec{f}_1^{z_1} \cdot \vec{f}_2^{z_2} \cdot \vec{f}_3^{z_3} \end{pmatrix} \in \mathbb{G}^{3 \times 3},$$

gives the following computations,

$$\begin{aligned} S^\top \iota(\vec{A}) + S^\top (\Gamma + \Gamma^\top) \iota(\vec{\mathcal{Y}}) + S^\top \Gamma S \vec{v} \\ = \begin{pmatrix} \iota(A)^{x_1} \\ \iota(A)^{x_2} \\ \iota(A)^{x_3} \end{pmatrix} \cdot \begin{pmatrix} \iota(Y)^{z_1} \cdot \iota(Z)^{y_1} \\ \iota(Y)^{z_2} \cdot \iota(Z)^{y_2} \\ \iota(Y)^{z_3} \cdot \iota(Z)^{y_3} \end{pmatrix}^\gamma \cdot \begin{pmatrix} \vec{R}_Z^{y_1} \\ \vec{R}_Z^{y_2} \\ \vec{R}_Z^{y_3} \end{pmatrix}^\gamma, \end{aligned}$$

where the vector $\vec{R}_Z$ is the random factor of $\vec{C}_Z = \iota(Z) \cdot \vec{R}_Z$. Multiplying the above result by $\langle \rho, \vec{H} \rangle \vec{v}$, where $\langle \rho, \vec{H} \rangle$ is a random antisymmetric matrix, leads to the final proof $\vec{\pi}$,

$$\vec{\pi} = \begin{pmatrix} \iota(A)^{x_1} \cdot \iota(Y)^{z_1 \gamma} \cdot \vec{C}_Z^{y_1 \gamma} \\ \iota(A)^{x_2} \cdot \iota(Y)^{z_2 \gamma} \cdot \vec{C}_Z^{y_2 \gamma} \\ \iota(A)^{x_3} \cdot \iota(Y)^{z_3 \gamma} \cdot \vec{C}_Z^{y_3 \gamma} \end{pmatrix} \cdot \begin{pmatrix} \vec{f}_2^{\rho_1} \cdot \vec{f}_3^{\rho_2} \\ \vec{f}_1^{-\rho_1} \cdot \vec{f}_3^{\rho_3} \\ \vec{f}_1^{-\rho_2} \cdot \vec{f}_2^{-\rho_3} \end{pmatrix}. \quad (2.26)$$

To verify the proof $\vec{\pi} = (\vec{\pi}_1, \vec{\pi}_2, \vec{\pi}_3)^\top$, we simply check that

$$\iota_T(t) \cdot \prod_{i=1}^3 F(\vec{f}_i, \vec{\pi}_i) = F(\iota(A), \vec{C}_X) \cdot F(\vec{C}_Y, \vec{C}_Z)^\gamma.$$

It is worth observing that any PPE equation may be written as an equality between a $\mathbb{G}_T$ -constant and a product of $e(A_i, X_i) \cdot e(Y_i, Z_i)$ like in the right-hand side of equation 2.25. Interestingly, the GS proofs are so that, computing a proof for a general PPE comes to multiply all the left matrices of the right-hand side member of the equation 2.26 relatively to each factor $e(A_i, X_i) \cdot e(Y_i, Z_i)$ and to multiply their product with only one right matrix $\langle \rho, \vec{H} \rangle \vec{v}$. This gives a general method for proving PPE statements (even if an element lies in several such factors).

2. Let us prove that we possess a valid AHO signature (cf. Section 2.2.2) $\vec{\theta} = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7) \in \mathbb{G}^7$ on a message $\vec{M} = (M_1, \dots, M_n) \in \mathbb{G}^n$. We must show that both pairing-product equations 2.7 and 2.8 of the verification equation are satisfied. But first of all, let us see that the signature can be partially rerandomized. Indeed, both equations contain a factor of the form $e(G, X) \cdot e(Y, Z)$, for some given constant G , and this factor is actually equal to $e(G, X') \cdot e(Y', Z')$ where $X' = X \cdot Z^{-\alpha}$, $Y' = (Y \cdot G^\alpha)^{1/\beta}$, $Z' = Z^\beta$, for any random scalars $\alpha, \beta \leftarrow \mathbb{Z}_p$. This fact was observed by the authors of the signature in [4, 3], and the transformation is denoted by $(X', Y', Z') \leftarrow \text{ReRand}(G, (X, Y, Z))$; the two elements

²All the Γ matrices are equal up to an additive antisymmetric matrix, $\Gamma + \Gamma^\top$ is constant.

Y', Z' are made completely independent. Consequently, for the rerandomized signature with $\theta'_1 = \theta_1$, $(\theta'_2, \theta'_3, \theta'_4) \leftarrow \text{ReRand}(G_r, (\theta_2, \theta_3, \theta_4))$ and $(\theta'_5, \theta'_6, \theta'_7) \leftarrow \text{ReRand}(H_r, (\theta_5, \theta_6, \theta_7))$, the elements $\{\theta'_i\}_{i \in \{3,4,6,7\}}$ can be revealed in clear.

By grouping the $\mathbb{G}_T$ -constants, $t_T^A := A \cdot e(\theta'_3, \theta'_4)^{-1}$ and $t_T^B := B \cdot e(\theta'_6, \theta'_7)^{-1}$, this leads to prove the following *linear* relation.

$$\begin{aligned} t_A &= e(G_z, \theta'_1) \cdot e(G_r, \theta'_2) \cdot \prod_{i=1}^n e(G_i, M_i) \\ t_B &= e(H_z, \theta'_1) \cdot e(H_r, \theta'_5) \cdot \prod_{i=1}^n e(H_i, M_i) \end{aligned} \quad (2.27)$$

The proof $\pi = (\pi_A, \pi_B)$ is made of 3 group elements per equation

$$\vec{\pi}_A = \begin{pmatrix} G_z^{\rho_1} \cdot G_r^{\rho_2} \cdot \prod_{i=1}^n G_i^{\mu_i} \\ G_z^{\sigma_1} \cdot G_r^{\sigma_2} \cdot \prod_{i=1}^n G_i^{\nu_i} \\ G_z^{\tau_1} \cdot G_r^{\tau_2} \cdot \prod_{i=1}^n G_i^{\eta_i} \end{pmatrix}, \quad \vec{\pi}_B = \begin{pmatrix} H_z^{\rho_1} \cdot H_r^{\rho_5} \cdot \prod_{i=1}^n H_i^{\mu_i} \\ H_z^{\sigma_1} \cdot H_r^{\sigma_5} \cdot \prod_{i=1}^n H_i^{\nu_i} \\ H_z^{\tau_1} \cdot H_r^{\tau_5} \cdot \prod_{i=1}^n H_i^{\eta_i} \end{pmatrix}.$$

along with the commitments $\vec{C}_{\theta'_i} = \iota(\theta'_1) \vec{f}_1^{\rho_i} \vec{f}_2^{\sigma_i} \vec{f}_1^{\tau_i}$, for $i = 1, 2, 5$, and $\vec{C}_{M_i} = \iota(M_1) \vec{f}_1^{\mu_i} \vec{f}_2^{\nu_i} \vec{f}_1^{\eta_i}$, for $i = 1$ up to n .

The verifier must recompute t_A and t_B (this is crucial) and check that

$$\begin{aligned} \iota_T(t_A) \cdot \prod_{i=1}^n \tilde{F}(\iota(\pi_i^A), \vec{f}_i) &= \tilde{F}(\iota(G_z), \vec{C}_{\theta'_1}) \cdot \tilde{F}(\iota(G_r), \vec{C}_{\theta'_2}) \cdot \prod_{i=1}^n \tilde{F}(\iota(G_i), \vec{C}_{M_i}) \\ \iota_T(t_B) \cdot \prod_{i=1}^n \tilde{F}(\iota(\pi_i^B), \vec{f}_i) &= \tilde{F}(\iota(H_z), \vec{C}_{\theta'_1}) \cdot \tilde{F}(\iota(H_r), \vec{C}_{\theta'_5}) \cdot \prod_{i=1}^n \tilde{F}(\iota(H_i), \vec{C}_{M_i}) \end{aligned}$$

where $\vec{\pi}_A = (\pi_1^A, \pi_2^A, \pi_3^A)^\top$ and $\vec{\pi}_B = (\pi_1^B, \pi_2^B, \pi_3^B)^\top$.

Note that the first two rows of all the (3×3) -matrices in the verification equations are exclusively composed of $1_T \in \mathbb{G}_T$. For ι_T it is by definition and for the $\tilde{F}$ -factors this is because they all include a term of the form $\iota(G)$ for a group element $G \in \mathbb{G}$. Rather than having to ignore the six trivial equations, sometimes the notation $E(G, \vec{C})$ may be found to express only $(e(G, C_1), e(G, C_3), e(G, C_3))$ the last row of $\tilde{F}(\iota(G), \vec{C})$.

- Let us suppose we have an SDP solution $(Z, R, U) \in \mathbb{G}^3$ verifying the following two linear equations.

$$\begin{aligned} 1_T &= e(G_z, Z) \cdot e(G_r, R) \\ 1_T &= e(H_z, Z) \cdot e(H_u, U) \end{aligned}$$

This also means that $(Z, R, U) \neq (1, 1, 1)$, which is not implied by the above system. Since the pairing map is supposed to be non degenerated, it is equivalent to show that $Z \neq 1$ in $\mathbb{G}$. Then, it suffices to pick $Y \leftarrow \mathbb{G}$ at random, reveal the constant $t := e(Y, Z) \in \mathbb{G}_T \setminus \{1_T\}$ and additionally prove that you also know the element Y .

Generally, if $Z = G^a \neq 1$ for a known $a \in \mathbb{Z}_p$ and a constant G , we rather prove that $e(X, Z) = e(G, G)$ for the committed variable $X = G^{1/a}$ since a must be invertible. This saves the computation of t , but the additional equation remains quadratic.

4. Prove that $\vec{\Upsilon} = (\Upsilon_1, \Upsilon_2, \Upsilon_3)$ is a BBS encryption of a committed variable X . That is, for additional witnesses (z_1, z_2) and (w_1, w_2, w_3) we have $\vec{\Upsilon} = (f_1^{z_1}, f_2^{z_2}, X \cdot g^{z_1+z_2})$ and $\vec{C}_X = \iota(X) \cdot \vec{f}_1^{w_1} \vec{f}_2^{w_2} \vec{f}_3^{w_3}$. (Implicitly, the public encrypting-key ek of the encryption and committing keys of the crs gather the same parameters f_1, f_2 .)

This is equivalent to showing the three (type-2) linear MME equations by considering $(T_1, T_2, T_3) := \vec{C}_X \cdot \vec{\Upsilon}^{-1}$ in $\mathbb{G}^3$:

$$T_1 = f_1^{\chi_1} \cdot f_{3,1}^{\chi_3}, \quad T_2 = f_2^{\chi_2} \cdot f_{3,2}^{\chi_3}, \quad T_3 = g^{\chi_1+\chi_2} \cdot f_{3,3}^{\chi_3}, \quad (2.28)$$

with $\chi_1 = w_1 - z_1$, $\chi_2 = w_2 - z_2$, $\chi_3 = w_3$. Compute a commitment on each *exponent*, $\vec{C}'_{\chi_j} = \vec{\varphi}^{\chi_j} \cdot \vec{f}_1^{\rho_j} \cdot \vec{f}_2^{\sigma_j}$, with $\rho_j, \sigma_j \xleftarrow{\$} \mathbb{Z}_p$ for $j \in \{1, 2, 3\}$. Output them with their proof

$$\vec{\pi}_1 = \begin{pmatrix} f_1^{\rho_1} \cdot f_{3,1}^{\rho_3} \\ f_1^{\sigma_1} \cdot f_{3,1}^{\sigma_3} \end{pmatrix}, \quad \vec{\pi}_2 = \begin{pmatrix} f_2^{\rho_2} \cdot f_{3,2}^{\rho_3} \\ f_2^{\sigma_2} \cdot f_{3,2}^{\sigma_3} \end{pmatrix}, \quad \vec{\pi}_3 = \begin{pmatrix} g^{\rho_1+\rho_2} \cdot f_{3,3}^{\rho_3} \\ g^{\sigma_1+\sigma_2} \cdot f_{3,3}^{\sigma_3} \end{pmatrix}. \quad (2.29)$$

Multi-exponentiation equations admit zero-knowledge (NIZK) proofs at no additional cost. On a simulated CRS (prepared for the WI setting), a trapdoor ($\text{tk}_1 = (\mu_1, \mu_2)$ defined in the linear flippable commitment) makes it possible to simulate proofs without knowing witnesses. Simulated proofs then have the same distribution as real proofs. In contrast, pairing-product equations do not always have NIZK proofs. Fortunately, NIWI proofs will be sufficient most of the time. In the body of the thesis we use the GS proof system in a black box manner.

2.6 Advanced Tools

In this section we recall specific notions of encryption schemes. We start with the notion of tag-based encryption. Next, we move on the notion of hierarchical identity-based encryption and finally, we recall a framework for the notion of broadcast encryption.

2.6.1 Tag-Based Encryption

A tag-based encryption scheme is an encryption scheme where the encryption and the decryption algorithms take in an additional input called a tag. Then, all the ciphertexts are associated to possibly different tags but different ciphertexts can share a same tag. The tag space is defined during the generation of the parameters of the scheme and most of the time a tag is simply a bit-string of appropriate length.

Kiltz's tag-based encryption scheme. In [99], Kiltz described a tag-based encryption scheme that can be transformed into a CCA-secure public-key encryption scheme based on the DLIN assumption.

Setup(λ): chooses a cyclic group $\mathbb{G}$ of prime order $p > 2^\lambda$ with a generator $g \xleftarrow{\$} \mathbb{G}$ and outputs $\mathbf{pp} = (\lambda, p, \mathbb{G}, g)$. A tag τ is simply a λ -bit string and can then be represented as an element of $\mathbb{Z}_p$.

Keygen($\mathbf{pp}$): takes in $\mathbf{pp}$ and picks random exponents $x_1, x_2, y_1, y_2 \xleftarrow{\$} \mathbb{Z}_p$. The private key is $\mathbf{sk} = (x_1, x_2, y_1, y_2) \in \mathbb{Z}_p^4$ and the public key is $\mathbf{pk} = (g, f_1, f_2, U, V) = (g, g^{x_1}, g^{x_2}, g^{y_1}, g^{y_2})$.

Encrypt($\mathbf{pk}, M, \tau$): to encrypt $M \in \mathbb{G}$ with the tag $\tau \in \mathbb{Z}_p$, picks $z_1, z_2 \xleftarrow{\$} \mathbb{Z}_p$, computes and returns $C = (c_0, c_1, c_2, c_3, c_4) \in \mathbb{G}^5$ such that $c_0 = M \cdot g^{z_1+z_2}$, $c_1 = f_1^{z_1}$, $c_2 = f_2^{z_2}$, $c_3 = (g^\tau \cdot U)^{z_1}$ and $c_4 = (g^\tau \cdot V)^{z_2}$.

Decrypt($\mathbf{sk}, C, \tau$): checks that $c_3 = c_1^{(\tau+y_1)/x_1}$ and $c_4 = c_2^{(\tau+y_2)/x_2}$. If so, outputs the plaintext $M = c_0 / (c_1^{1/x_1} c_2^{1/x_2})$ and $\perp$ otherwise.

When the scheme is instantiated in a symmetric bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g)$, validity of ciphertexts is publicly verifiable. Indeed, it suffices to check whether $e(f_1, c_3) = e(c_1, g^\tau U)$ and $e(f_2, c_4) = e(c_2, g^\tau V)$.

The CCA transformation [99]. The scheme can be turned into a CCA-secure encryption scheme using a strongly unforgeable one-time signature scheme Σ .

Using the public parameters $\mathbf{pp}$ of the scheme, the **Setup** algorithm also selects $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ and an injective encoding function $\mathcal{H}$ so that $\mathcal{H}(\mathbf{VK}) \in \mathbb{Z}_p$ for any one-time verification key $\mathbf{VK}$ generated by $(\mathbf{SK}, \mathbf{VK}) \leftarrow \mathcal{G}(\mathbf{pp})$. To encrypt M , **Encrypt'** generates a one-time key-pair $(\mathbf{SK}, \mathbf{VK}) \leftarrow \mathcal{G}(\mathbf{pp})$, runs **Encrypt**($\mathbf{pk}, M, \tau$) with $\tau = \mathcal{H}(\mathbf{VK})$ to get C and returns

$$C' = (\mathbf{VK}, C, \sigma),$$

where $\sigma = \mathcal{S}(\mathbf{SK}, C)$. To decrypt C' , **Decrypt'** first verifies the one-time signature σ using $\mathcal{V}(\mathbf{VK}, \sigma, C)$ and then runs **Decrypt**($\mathbf{sk}, C, \tau$) with $\tau = \mathcal{H}(\mathbf{VK})$.

2.6.2 Hierarchical Identity-Based Encryption

Consider a hierarchy of entities, each of which has a unique address $\text{ID} = (I_1, \dots, I_\ell)$, with $I_i \in \{0, 1\}^*$ for $1 \leq i \leq \ell$, at level ℓ . For any $i \leq \ell$, $\text{ID}|_i$ denotes the prefix $(I_1, \dots, I_i)$ of ID . The address of a node at level i is obtained by appending its local identifier I_i to its father's address $\text{ID}|_{i-1}$.

A HIBE scheme [89, 78] is a tuple (**Setup**, **Keygen**, **Derive**, **Encrypt**, **Decrypt**) of algorithms³ working as follows. **Setup** is run by a trusted private key generator (PKG) to generate a master public key mpk and a master secret key msk . The latter is used by the PKG, at the root of the hierarchy, to derive private keys from users' identities at level 1. The key generation algorithm **Keygen** takes as input the master secret key msk and a hierarchical identity

³We use the syntax of [32] which involves an explicit delegation algorithm **Derive**. Although this algorithm is not explicitly written in [22], it exists as noted below.

$ID = (I_1, \dots, I_k)$ and returns a private key d_{ID} for that identity. Algorithm **Derive** is used by a ℓ -th level entity with address $ID = (I_1, \dots, I_\ell)$ to compute private keys for its children labelled as $(I_1, \dots, I_\ell, *)$ at depth $\ell + 1$. It takes in a ℓ -th level private key d_{ID} and a vector $ID' = (I_1, \dots, I_\ell, I_{\ell+1})$, where $ID'|_\ell = ID$, to generate a $(\ell + 1)$ -th level secret key $d_{ID'}$. Algorithm **Encrypt** takes in a plaintext $m \in \mathcal{M}$, where $\mathcal{M}$ denotes the plaintext space, the master public key mpk and the receiver's address $ID = (I_1, \dots, I_\ell)$ to produce a ciphertext C that can be undone by the receiver having obtained d_{ID} from its father.

In many HIBE constructions (such as [22, 32]) delegated private keys (produced by **Derive**) have the same distribution as original keys (generated by **Keygen**). In the upcoming security definitions, we assume that this property is satisfied by the considered HIBE system.

In the following, we say that a HIBE scheme is *key-partitioned* if private keys $d_{ID} = (D_{ID}, K_{ID})$ consist of two distinct parts: the first one D_{ID} is called *decryption component* and it is only used to decrypt messages; the second part K_{ID} is called *delegation component* and its sole purpose is to derive private keys for children nodes. Many HIBE systems in the literature (e.g., [22, 32]) are key-partitioned.

Security Definitions for HIBE The standard security notion [78] captures that any coalition of hierarchy entities that are not ancestors of some user should be unable to gain information on messages encrypted for that user.

In [42], Canetti, Halevi and Katz suggested a weaker security notion, called *selective security*, where the adversary has to choose its target identity upfront, i.e. the identity for which he/she will try to learn information on encrypted messages.

Definition 20 ([42]) *A HIBE scheme is selectively secure (or IND-sID-CPA secure) for ℓ levels if no PPT adversary $\mathcal{A}$ has non-negligible advantage in the following game:*

1. The adversary $\mathcal{A}$ chooses a target identity $ID^* = (I_1^*, \dots, I_{\ell^*}^*)$ at depth $\ell^* < \ell$, for some ℓ^* of its choice. The challenger runs $(mpk, msk) \leftarrow \text{Setup}(\lambda)$ and hands mpk to $\mathcal{A}$.
2. $\mathcal{A}$ issues a number of key extraction queries under the rule that no prefix of ID^* can be the input of a key extraction query. On input of an identity $ID = (I_1, \dots, I_k)$, with $k \leq \ell$, the challenger responds with $d_{ID} \leftarrow \text{Keygen}(msk, ID)$.
3. When $\mathcal{A}$ decides that the first phase is over, it chooses messages m_0, m_1 . The challenger flips a coin $d \xleftarrow{\$} \{0, 1\}$ and responds with a challenge $C^* = \text{Encrypt}(mpk, ID^*, m_d)$.
4. $\mathcal{A}$ issues new queries but cannot ask for the private key of a prefix of ID^* .
5. $\mathcal{A}$ finally outputs a bit $d' \in \{0, 1\}$ and wins if $d' = d$. As usual, $\mathcal{A}$'s advantage is quantified as the distance $\text{Adv}^{\text{hibe}}(\mathcal{A}) := |\Pr[d' = d] - 1/2|$.

For our purposes, we need a different and non-standard form of selective security, which mandates that the adversary be unable to maul a private key that it obtained for some target identity $ID^\dagger$ *even* knowing the master secret key. By “mauling”, we mean computing a private key for a different identity $ID' \neq ID^\dagger$ but under the same randomness as the received key $d_{ID^\dagger}$.

Definition 21 *A key-partitioned HIBE system with ℓ levels is said selectively key-robust if no PPT adversary $\mathcal{A}$ has non-negligible advantage in the following game:*

1. *The adversary $\mathcal{A}$ chooses an identity $ID^\dagger = (I_1^\dagger, \dots, I_{\ell^\dagger}^\dagger)$ that it wishes to be challenged upon at the depth $\ell^\dagger < \ell$ of its choice. The challenger runs $(mpk, msk) \leftarrow \text{Setup}(\lambda)$ and hands (msk, mpk) to $\mathcal{A}$ along with a challenge consisting of a private key $d_{ID^\dagger}$ for the identity $ID^\dagger$.*
2. *$\mathcal{A}$ outputs an identity ID' such that $ID^\dagger$ is not a prefix⁴ of ID' and a decryption component $D_{ID'}$ (i.e., a private key without delegation component). The adversary wins if: (i) $D_{ID'}$ is a valid decryption component for ID' ; (ii) $D_{ID'}$ and $d_{ID^\dagger}$ corresponds to the same randomness of the key generation algorithm.*

In Definition 21, we insist that $\mathcal{A}$ is given a full private key for the target identity $ID^\dagger$ but it only has to output a valid decryption component $D_{ID'}$ for ID' .

The Boneh-Boyen-Goh HIBE In [22], Boneh, Boyen and Goh (BBG) described the first HIBE scheme where the size of ciphertext does not depend on the depth of the receiver in the hierarchy. The construction bears resemblance with the first selectively secure IBE scheme of Boneh and Boyen [20], which can be seen as a single-level variant of the BBG HIBE. The latter works as follows.

Setup (λ, ℓ) : given a security parameter $\lambda \in \mathbb{N}$ and the number of levels $\ell \in \mathbb{N}$ in the hierarchy, chooses bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order p , where $p > 2^\lambda$. Chooses $\alpha \xleftarrow{\$} \mathbb{Z}_p$, $g, g_2, h_0, h_1, \dots, h_\ell \xleftarrow{\$} \mathbb{G}$ and computes $g_1 = g^\alpha$. The master public key is defined to be

$$mpk_{\text{BBG}} := \left((\mathbb{G}, \mathbb{G}_T), g, g_1, g_2, \{h_i\}_{i=0}^\ell \right)$$

while the master secret key consists of $msk_{\text{BBG}} := g_2^\alpha$. The space of hierarchical identities is $\mathcal{I} = (\mathbb{Z}_p^*)^{\leq \ell}$.

Keygen $(msk_{\text{BBG}}, ID = (I_1, \dots, I_k))$: in order to generate a private key, at level k , for $ID = (I_1, \dots, I_k) \in (\mathbb{Z}_p^*)^k$ using $msk_{\text{BBG}} = g_2^\alpha$, chooses $r \xleftarrow{\$} \mathbb{Z}_p$.

⁴We assume that hierarchical identities are prefixes of themselves for simplicity.

Then, computes and returns $d_{\text{ID}} = (D_1, D_2, K_{k+1}, \dots, K_\ell)$ such that

$$d_{\text{ID}} = \left(g_2^\alpha \cdot \left(h_0 \cdot \prod_{i=1}^k h_i^{I_i} \right)^r, g^r, h_{k+1}^r, \dots, h_\ell^r \right) \in \mathbb{G}^{\ell-k+2}. \quad (2.30)$$

Derive($d_{\text{ID}}, \text{ID}' = (I_1, \dots, I_k, I_{k+1})$): given a private key d_{ID} of the form (2.30) for the hierarchical identity $\text{ID} = (I_1, \dots, I_k)$, it is easy to derive a key for the identity $\text{ID}' = (I_1, \dots, I_k, I_{k+1}) \in (\mathbb{Z}_p^*)^{k+1}$ by choosing $r' \xleftarrow{\$} \mathbb{Z}_p$ and computing $d_{\text{ID}'} = (D'_1, D'_2, K'_{k+2}, \dots, K'_\ell)$ such that

$$\begin{aligned} d_{\text{ID}'} &= \left(D_1 \cdot K_{d+1}^{I_{k+1}} \cdot \left(h_0 \cdot \prod_{i=1}^{k+1} h_i^{I_i} \right)^{r'}, D_2 \cdot g^{r'}, K_{k+2} \cdot h_{k+2}^{r'}, \dots, K_\ell \cdot h_\ell^{r'} \right) \\ &= \left(g_2^\alpha \cdot \left(h_0 \cdot \prod_{i=1}^{k+1} h_i^{I_i} \right)^{r''}, g^{r''}, h_{k+2}^{r''}, \dots, h_\ell^{r''} \right) \in \mathbb{G}^{\ell-k+1}, \end{aligned} \quad (2.31)$$

where $r'' = r + r'$.

Encrypt($\text{mpk}_{\text{BBG}}, \text{ID} = (I_1, \dots, I_d), M$): in order to encrypt $M \in \mathbb{G}$ under $\text{ID} = (I_1, \dots, I_d) \in (\mathbb{Z}_p^*)^d$, chooses $s \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$C_0 = M \cdot e(g_1, g_2)^s, \quad C_1 = g^s, \quad C_2 = (h_0 \cdot h_1^{I_1} \dots h_d^{I_d})^s$$

The ciphertext is $C = (C_0, C_1, C_2)$.

Decrypt($\text{mpk}_{\text{BBG}}, d_{\text{ID}}, C$): parses d_{ID} as $(D_1, D_2, K_{d+1}, \dots, K_\ell) \in \mathbb{G}^{\ell-d+2}$ and the ciphertext C as (C_0, C_1, C_2) . Then, computes and outputs

$$M = C_0 \cdot e(C_1, D_1)^{-1} \cdot e(C_2, D_2).$$

It is easy to see that this construction is key-partitioned since the private key can be divided into $(D_{\text{ID}}, K_{\text{ID}})$, where $D_{\text{ID}} = (D_1, D_2) \in \mathbb{G}^2$ is only used to decrypt and $K_{\text{ID}} = (K_{k+1}, \dots, K_\ell) \in \mathbb{G}^{\ell-k+2}$ is only useful for delegations.

When the private key for ID' is derived from the private key for ID , the randomizer $r' \in \mathbb{Z}_p$ in (2.31) allows making sure that derived private keys are indistinguishable from original keys that are generated directly at level $k + 1$.

2.6.3 The>NNL Framework for Broadcast Encryption

The Subset Cover framework of Naor *et al.* [119] considers secret-key broadcast encryption schemes with $N = 2^\ell$ registered receivers. Each one of them is associated with a leaf of a complete binary tree $\mathbf{T}$ of height ℓ and each tree node is assigned a secret key. If $\mathcal{N}$ denotes the universe of users and $\mathcal{R} \subset \mathcal{N}$ is the set of revoked receivers, the idea of the framework is to partition the set of non-revoked users into m disjoint subsets $S_1, \dots, S_m$ such that $\mathcal{N} \setminus \mathcal{R} = S_1 \cup \dots \cup S_m$. Depending on the way to partition $\mathcal{N} \setminus \mathcal{R}$ and the distribution of keys to users, different instantiations and tradeoffs are possible.

THE COMPLETE SUBTREE METHOD. In this technique, each subset S_i consists of the leaves of a complete subtree rooted at some node x_i of T . Upon registration, each user obtains secret keys for all nodes on the path connecting his/her leaf to the root of T (and thus $O(\ell)$ keys overall). By doing so, users in $\mathcal{N} \setminus \mathcal{R}$ can decrypt the content if the latter is enciphered using symmetric keys $K_1, \dots, K_m$ corresponding to the roots of subtrees $S_1, \dots, S_m$. As showed in [119], the CS partitioning method entails at most $m \leq r \cdot \log(N/r)$ subsets, where $r = |\mathcal{R}|$. Each transmission requires to send $O(r \cdot \log N)$ symmetric encryptions while, at each user, the storage complexity is $O(\log N)$.

As noted in [119, 59], a single-level identity-based encryption scheme allows implementing a public-key variant of the CS method. The master public key of the IBE scheme forms the public key of the broadcast encryption system, which allows for public keys of size $O(1)$ (instead of $O(N)$ in instantiations using ordinary public-key encryption). When users join the system, they obtain $O(\ell)$ IBE private keys (in place of symmetric keys) associated with the “identities” of nodes on the path between their leaf and the root.

THE SUBSET DIFFERENCE METHOD. The SD method reduces the transmission cost to $O(r)$ at the expense of increased storage requirements. For each node $x_j \in T$, we call T_{x_j} the subtree rooted at x_j . The set $\mathcal{N} \setminus \mathcal{R}$ is now divided into disjoint subsets $S_{k_1, u_1}, \dots, S_{k_m, u_m}$. For each $i \in \{1, \dots, m\}$, the subset S_{k_i, u_i} is determined by a node x_{k_i} and one of its descendants x_{u_i} – which are called *primary* and *secondary* roots of S_{k_i, u_i} , respectively – and it consists of the leaves of $T_{x_{k_i}}$ that are not in $T_{x_{u_i}}$. Each user thus belongs to many more generic subsets than in the CS method and this allows reducing the maximal number of subsets to $m = 2r - 1$ (see [119] for a proof of this bound).

A more complex key distribution is necessary to avoid a prohibitive storage overhead. Each subset S_{k_i, u_i} is assigned a “proto-key” $P_{x_{k_i}, x_{u_i}}$ that allows deriving the actual symmetric encryption key K_{k_i, u_i} for S_{k_i, u_i} and as well as proto-keys $P_{x_{k_i}, x_{u_l}}$ for any descendant x_{u_l} of x_{u_i} . At the same time, $P_{x_{k_i}, x_{u_l}}$ should be hard to compute without a proto-key $P_{x_{k_i}, x_{u_i}}$ for an ancestor x_{u_i} of x_{u_l} . The key distribution phase then proceeds as follows. Let user i be assigned a leaf v_i and let $\epsilon = x_0, x_1, \dots, x_\ell = v_i$ denote the path from the root ϵ to v_i . For each subtree T_{x_j} (with $j \in \{1, \dots, \ell\}$), if copath_{x_j} denotes the set of all siblings of nodes on the path from x_j to v_i , user i must obtain proto-keys $P_{x_j, w}$ for each node $w \in \text{copath}_{x_j}$ because he/she belongs to the generic subset whose primary root is x_j and whose secondary root is w . By storing $O(\ell^2)$ proto-keys (*i.e.*, $O(\ell)$ for each subtree T_{x_j}), users will be able to derive keys for all generic subsets they belong to.

In [59], Dodis and Fazio extended the SD method to the public-key setting using hierarchical identity-based encryption. In the tree, each node w at depth $\leq \ell$ has a label $\langle w \rangle$ which is defined by assigning the label ϵ to the root (at depth 0). The left and right children of w are then labelled with $\langle w \rangle || 0$ and $\langle w \rangle || 1$, respectively. For each subset S_{k_i, u_i} of $\mathcal{N} \setminus \mathcal{R}$, the sender considers the primary and secondary roots x_{k_i}, x_{u_i} and parses the label $\langle x_{u_i} \rangle$ as $\langle x_{k_i} \rangle || u_{i, \ell_{i,1}} \dots u_{i, \ell_{i,2}}$, with $u_{i,j} \in \{0, 1\}$ for each $j \in \{\ell_{i,1}, \dots, \ell_{i,2}\}$. Then, he/she computes a HIBE ciphertext for the hierarchical identity $(\langle x_{k_i} \rangle, u_{i, \ell_{i,1}}, \dots, u_{i, \ell_{i,2}})$ at level $\ell_{i,2}$ –

$\ell_{i,1} + 2$. Upon registration, if $\epsilon = x_0, \dots, x_\ell = v_i$ denotes the path from the root to his/her leaf v_i , for each subtree T_{x_j} , user i receives exactly one HIBE private key for each $w \in \text{copath}_{x_j}$: namely, for each $w \in \text{copath}_{x_j}$, there exist $\ell_1, \ell_2 \in \{1, \dots, \ell\}$ such that $\langle w \rangle = \langle x_j \rangle || w_{\ell_1} \dots w_{\ell_2}$ with $w_j \in \{0, 1\}$ for all $j \in \{\ell_1, \dots, \ell_2\}$ and user i obtains a HIBE private key for the hierarchical identity $(\langle x_j \rangle, w_{\ell_1}, \dots, w_{\ell_2})$. By construction, this key will allow user i to decrypt any HIBE ciphertext encrypted for a subset whose primary root is x_j and whose secondary root is a descendant of w . Overall, each user thus has to store $O(\log^2 N)$ HIBE private keys.

Part II

Signature-Based Primitives

Chapter 3

Revocable Group Signatures

Contents

3.1	Definition and Security Notions	66
3.1.1	Definition of Group Signatures with Revocation . .	66
3.1.2	Security Model for Revocable Group Signatures . .	68
3.2	The First Scalable Group Signature with Revocation	72
3.2.1	Intuition and Technical Lemma	72
3.2.2	Scalable Construction	75
3.2.3	Security Analysis	80
3.3	Group Signature with Almost-for-free Revocation	87
3.3.1	Intuition behind our Scheme	88
3.3.2	Construction with Compact Keys	89
3.3.3	Security Analysis	95
3.3.4	Efficiency of this second Scheme	100
3.4	Efficiency Comparison and Comments	100

Group signatures are a central cryptographic primitive where users can anonymously and accountably sign messages in the name of the group they belong to. Several efficient constructions with security proofs in the standard model (*i.e.*, without the random oracle idealization) have appeared in recent years. However, like standard PKIs, group signatures need an efficient revocation system to be practical. Despite years of research, membership revocation remains a non-trivial problem: many existing solutions do not scale well due to either high overhead (e.g., large group public key size) or constraining operational requirements (like the need for all users to update their keys after each revocation which forces them to follow the entire history of the system). In the standard model, the situation is even worse as many existing solutions

are not readily adaptable. To fill this gap and tackle this challenge, we describe a new revocation approach based, perhaps somewhat unexpectedly, on the Naor-Naor-Lotspeich framework which was introduced for a different problem (namely, that of broadcast encryption). Our mechanism yields efficient and scalable revocable group signatures in the standard model. Moreover, the schemes are history-independent: unrevoked group members do not have to update their keys when a revocation occurs.

In this chapter we propose two constructions based on the NLL subset difference method after formalizing the revocable group signature (R-GS) primitive. In particular, our first construction is the first scheme to enjoy a size of signatures and a verification cost that are independent of the number of revocations and the maximal cardinality N of the group while the other complexities are at most polylogarithmic in N . In order to decrease the storage requirements at group members, namely, membership certificates, which used to have constant size in existing standard model constructions, our second construction shows how to obtain private keys of *constant* size at the expense of strengthening the underlying security assumptions.

3.1 Definition and Security Notions

We consider group signature schemes that have their lifetime divided into revocation periods at the beginning of which group managers update their revocation lists. The syntax and the security model are built on those defined by Kiayias and Yung [97]. Like the Bellare-Shi-Zhang model [16], the Kiayias-Yung (KY) model assumes an interactive *join* protocol whereby a prospective user becomes a group member by interacting with the group manager. This protocol provides the user with a membership certificate and a membership secret.

3.1.1 Definition of Group Signatures with Revocation

Syntax. We denote by $N \in \text{poly}(\lambda)$ the maximal number of group members. At the beginning of each revocation period t , the group manager publicizes an up-to-date revocation list RL_t and we denote by $\mathcal{R}_t \subset \{1, \dots, N\}$ the corresponding set of revoked users (we assume that $\mathcal{R}_t$ is part of RL_t). A revocable group signature (R-GS) scheme consists of the following algorithms or protocols.

Setup(λ, N): given a security parameter $\lambda \in \mathbb{N}$ and a maximal number of group members $N \in \mathbb{N}$, this algorithm (which is run by a trusted party) generates a group public key $\mathcal{V}$, the group manager's private key $\mathcal{S}_{\text{GM}}$ and the opening authority's private key $\mathcal{S}_{\text{OA}}$. Keys $\mathcal{S}_{\text{GM}}$ and $\mathcal{S}_{\text{OA}}$ are given to the appropriate authority while $\mathcal{V}$ is publicized. The algorithm also

initializes a public state St comprising a set data structure $St_{\text{users}} = \emptyset$ and a string data structure $St_{\text{trans}} = \epsilon$.

Join: is an interactive protocol between the group manager GM and a user $\mathcal{U}_i$ where the latter becomes a group member. The protocol involves two interactive Turing machines J_{user} and J_{GM} that both take $\mathcal{Y}$ as input. The execution, denoted as $[J_{\text{user}}(\lambda, \mathcal{Y}), J_{\text{GM}}(\lambda, St, \mathcal{Y}, \mathcal{S}_{\text{GM}})]$, ends with user $\mathcal{U}_i$ obtaining a membership secret sec_i , that no one else knows, and a membership certificate cert_i . If the protocol is successful, the group manager updates the public state St by setting $St_{\text{users}} := St_{\text{users}} \cup \{i\}$ as well as $St_{\text{trans}} := St_{\text{trans}} || \langle i, \text{transcript}_i \rangle$.

Revoke: is a (possibly randomized) algorithm allowing the GM to generate an updated revocation list RL_t for the new revocation period t . It takes as input a public key $\mathcal{Y}$ and a set $\mathcal{R}_t \subset St_{\text{users}}$ that identifies the users to be revoked. It outputs an updated revocation list RL_t for period t .

Sign: given a revocation period t with its revocation list RL_t , a membership certificate cert_i , a membership secret sec_i and a message M , this algorithm outputs $\perp$ if $i \in \mathcal{R}_t$ and a signature σ otherwise.

Verify: given a signature σ , a revocation period t , the corresponding revocation list RL_t , a message M and a group public key $\mathcal{Y}$, this deterministic algorithm returns either 0 or 1.

Open: takes as input a message M , a valid signature σ w.r.t. $\mathcal{Y}$ for the indicated revocation period t , the opening authority's private key $\mathcal{S}_{\text{OA}}$ and the public state St . It outputs $i \in St_{\text{users}} \cup \{\perp\}$, which is the identity of a group member or a symbol indicating an opening failure.

Each membership certificate contains a unique tag that identifies the user.

Correctness for Revocable Group Signatures. First of all, following the KY terminology, we say that a public state St is *valid* if it is reachable from $St = (\emptyset, \epsilon)$ by a Turing machine having oracle access to J_{GM} . Also, a state St' is said to *extend* another state St if it is within reach from St .

Moreover, still as in [97, 96], when we write $\text{cert}_i \Leftarrow_{\mathcal{Y}} \text{sec}_i$, it means that there exists coin tosses ϖ for J_{GM} and J_{user} such that, for some valid public state St' , the execution of $[J_{\text{user}}(\lambda, \mathcal{Y}), J_{\text{GM}}(\lambda, St', \mathcal{Y}, \mathcal{S}_{\text{GM}})](\varpi)$ provides J_{user} with $\langle i, \text{sec}_i, \text{cert}_i \rangle$.

Definition 22 (Correctness of R-GS.) *A R-GS scheme is correct if the following conditions are all satisfied:*

1. *In a valid state St , $|St_{\text{users}}| = |St_{\text{trans}}|$ always holds and two distinct entries of St_{trans} always contain certificates with distinct tag.*

2. If $[J_{\text{user}}(\lambda, \mathcal{Y}), J_{\text{GM}}(\lambda, St, \mathcal{Y}, \mathcal{S}_{\text{GM}})]$ is run by two honest parties following the protocol and $\langle i, \text{cert}_i, \text{sec}_i \rangle$ is obtained by J_{user} , then it holds that $\text{cert}_i \Leftarrow_{\mathcal{Y}} \text{sec}_i$.
3. For each revocation period t and any $\langle i, \text{cert}_i, \text{sec}_i \rangle$ such that $\text{cert}_i \Leftarrow_{\mathcal{Y}} \text{sec}_i$, satisfying condition 2, $\text{Verify}(\text{Sign}(\mathcal{Y}, t, RL_t, \text{cert}_i, \text{sec}_i, M), M, t, RL_t, \mathcal{Y}) = 1$ always holds if $i \notin \mathcal{R}_t$.
4. For any outcome $\langle i, \text{cert}_i, \text{sec}_i \rangle$ of the interaction $[J_{\text{user}}(\cdot, \cdot), J_{\text{GM}}(\cdot, St, \cdot, \cdot)]$ for some valid period St , any revocation period t such that $i \notin \mathcal{R}_t$, if $\sigma = \text{Sign}(\mathcal{Y}, t, RL_t, \text{cert}_i, \text{sec}_i, M)$, then

$$\text{Open}(M, t, RL_t, \sigma, \mathcal{S}_{\text{OA}}, \mathcal{Y}, St') = i.$$

Correctness of (revocable) group signature schemes is implicit in the following. At a high level, it means that if all parties *honestly* run the protocols, everything goes well. Until now, nothing is specified when some users misbehave and deviate from the procedures. Protecting honest parties from other ones' misbehaviours is captured by the following security model.

3.1.2 Security Model for Revocable Group Signatures

A R-GS scheme must satisfy three security notions defined hereafter. The first one is called *security against misidentification attacks*. It requires that, even if the adversary can introduce and revoke users at will, it cannot produce a signature that traces outside the set of unrevoked adversarially-controlled users. As in ordinary (*i.e.*, non-revocable) group signatures, the notion of *security against framing attacks* captures that under no circumstances should an honest user be held accountable for messages that he did not sign, even if the whole system conspired against him. Finally, the notion of *anonymity* is also defined by granting the adversary access to a signature opening oracle as in the models of [16, 97].

We formalize security properties via experiments where the adversary interacts with a stateful interface $\mathcal{I}$ that maintains the following variables:

- $\text{state}_{\mathcal{I}}$: is a data structure representing the state of the interface as the adversary invokes the various oracles. It is initialized as $\text{state}_{\mathcal{I}} = (St, \mathcal{Y}, \mathcal{S}_{\text{GM}}, \mathcal{S}_{\text{OA}}) \leftarrow \text{Setup}(\lambda, N)$. It includes the (initially empty) set St_{users} of group members and a dynamically growing database St_{trans} storing the transcripts of previously executed join protocols. Finally, $\text{state}_{\mathcal{I}}$ includes a counter t (which is initialized to 0) indicating the number of user revocation queries so far.
- $n = |St_{\text{users}}| < N$ denotes the current cardinality of the group.
- **Sigs**: is a database of signatures created by the signing oracle. Each entry consists of a triple (i, t, M, σ) indicating that message M was signed by user i at period t .

- U^a : is the set of users that were introduced by the adversary in the system in an execution of the join protocol.
- U^b : is the set of honest users that the adversary, acting as a dishonest group manager, introduced in the system. For these users, the adversary obtains the transcript of the join protocol but not the user's membership secret.

When mounting attacks, adversaries will be granted access to the following oracles.

- Q_{pub} , Q_{keyGM} and Q_{keyOA} : when these oracles are invoked, the interface looks up $\text{state}_{\mathcal{I}}$ and returns the group public key $\mathcal{Y}$, the GM's private key S_{GM} and the opening authority's private key S_{OA} respectively.
- $Q_{\text{a-join}}$: allows the adversary to introduce users under his control in the group. On behalf of the GM, the interface runs J_{GM} in interaction with the J_{user} -executing adversary who plays the role of the prospective user in the join protocol. If this protocol successfully ends, the interface increments N , updates St by inserting the new user n in both sets St_{users} and U^a . It also sets $St_{\text{trans}} := St_{\text{trans}} || \langle n, \text{transcript}_n \rangle$.
- $Q_{\text{b-join}}$: allows the adversary, acting as a corrupted group manager, to introduce new honest group members of his/her choice. The interface triggers an execution of $[J_{\text{user}}, J_{\text{GM}}]$ and runs J_{user} in interaction with the adversary who runs J_{GM} . If the protocol successfully completes, the interface increments n , adds user n to St_{users} and U^b and sets $St_{\text{trans}} := St_{\text{trans}} || \langle n, \text{transcript}_n \rangle$. It stores the membership certificate cert_n and the membership secret sec_n in a *private* part of $\text{state}_{\mathcal{I}}$.
- Q_{sig} : given a message M , an index i , the interface checks whether the private area of $\text{state}_{\mathcal{I}}$ contains a certificate cert_i and a membership secret sec_i such that $i \notin \mathcal{R}_t$, where t is the current revocation period. If no such elements $(\text{cert}_i, \text{sec}_i)$ exist or if $i \notin U^b$, the interface returns $\perp$. Otherwise, it outputs a signature σ on behalf of user i for period t and also sets $\text{Sigs} \leftarrow \text{Sigs} || (i, t, M, \sigma)$.
- Q_{open} : when this oracle is invoked on input of a valid pair (M, σ) for some revocation period t , the interface runs algorithm Open using the current state St . When S is a set of triples of the form (M, σ, t) , $Q_{\text{open}}^{\neg S}$ denotes a restricted oracle that only applies the opening algorithm to triples (M, σ, t) which are not in S .
- Q_{read} and Q_{write} : are used by the adversary to read and write the content of $\text{state}_{\mathcal{I}}$. Namely, at each invocation, Q_{read} outputs the whole $\text{state}_{\mathcal{I}}$ but the public/private keys and the private part of $\text{state}_{\mathcal{I}}$ where membership secrets are stored after $Q_{\text{b-join}}$ -queries. By using Q_{write} , the adversary can modify $\text{state}_{\mathcal{I}}$ at will as long as it does not remove or alter elements of St_{users} , St_{trans} or invalidate the public state St : for example, the

adversary is allowed to create dummy users as long as he/she does not re-use already existing certificate tags.

- Q_{revoke} : is a revocation oracle. Given an index i such that $i \in St_{\text{users}}$, the interface checks if i appears in the appropriate user set (namely, U^a or U^b depending on the considered security notion) and if the database St_{trans} contains a record $\langle i, \text{transcript}_i \rangle$ such that $i \notin \mathcal{R}_t$, where t is the current revocation period. If not, it returns $\perp$. Otherwise, it increments t , adds i to $\mathcal{R}_t$ and generates an updated revocation list RL_t which is made available to the adversary. For simplicity purposes, we assume that the adversary only revokes one user per query to Q_{revoke} but the model easily extends to allow multiple revocations at once.

Using this formalism, we can now properly define the three announced security properties.

Security Against Misidentification In a misidentification attack, the adversary can corrupt the opening authority using the Q_{keyOA} oracle. Moreover, he/she can also introduce malicious users in the group via $Q_{\text{a-join}}$ -queries and revoke users at any time using Q_{revoke} . His/her purpose is to come up with a signature σ^* that verifies w.r.t. RL_{t^*} , where t^* denotes the current revocation period (*i.e.*, the number of Q_{revoke} -queries). He/she is deemed successful if the produced signature σ^* does not open to any unrevoked adversarially-controlled.

Definition 23 (Security Against Misidentification) *A R-GS scheme is secure against misidentification attacks if, for any PPT adversary $\mathcal{A}$ involved in the experiment hereafter, $\text{Adv}_{\mathcal{A}}^{\text{mis-id}}(\lambda) = \Pr[\text{Exp}_{\mathcal{A}}^{\text{mis-id}}(\lambda) = 1] \in \text{negl}(\lambda)$.*

Experiment $\text{Exp}_{\mathcal{A}}^{\text{mis-id}}(\lambda)$

1. $\text{state}_{\mathcal{I}} = (St, \mathcal{Y}, \mathcal{S}_{\text{GM}}, \mathcal{S}_{\text{OA}}) \leftarrow \text{Setup}(\lambda, N)$;
2. $(M^*, \sigma^*) \leftarrow \mathcal{A}(Q_{\text{pub}}, Q_{\text{a-join}}, Q_{\text{revoke}}, Q_{\text{read}}, Q_{\text{keyOA}})$;
3. If $\text{Verify}(\sigma^*, M, t^*, RL_{t^*}, \mathcal{Y}) = 0$ returns 0;
4. $i = \text{Open}(M^*, t^*, RL_{t^*}, \sigma^*, \mathcal{S}_{\text{OA}}, \mathcal{Y}, St')$;
5. If $(i \notin U^a \setminus \mathcal{R}_{t^*})$ returns 1;
6. Returns 0;

This definition extends the usual definition [97] in that $\mathcal{A}$ also wins if his/her forgery σ^* verifies w.r.t. RL_{t^*} but opens to an adversarially-controlled user that was revoked during the revocation period t^* .

Non-Frameability Framing attacks consider the situation where the entire system, including the group manager and the opening authority, is colluding against some honest user. The adversary can corrupt the group manager as well as the opening authority (via oracles Q_{keyGM} and Q_{keyOA} , respectively). He/she is also allowed to introduce honest group members (via $Q_{\text{b-join}}$ -queries), observe the system while these users sign messages and create dummy users

using Q_{write} . In addition, before the possible corruption of the group manager, the adversary can revoke group members at any time by invoking the Q_{revoke} oracle. As a potentially corrupted group manager, $\mathcal{A}$ is allowed to come up with his/her own revocation list RL_{t^*} at the end of the game. We assume that anyone can publicly verify that RL_{t^*} is correctly formed (*i.e.*, that it could be a legitimate output of **Revoke**) so that the adversary does not come up with an ill-formed revocation list. For consistency, if $\mathcal{A}$ chooses not to corrupt the GM, the produced revocation list RL_{t^*} must be the one determined by the history of Q_{revoke} -queries. The adversary eventually aims at framing an honest group member.

Definition 24 (Non-Frameability) *A R-GS scheme is secure against framing attacks if, for any PPT adversary $\mathcal{A}$, $\text{Adv}_{\mathcal{A}}^{\text{fra}}(\lambda) = \Pr[\text{Exp}_{\mathcal{A}}^{\text{fra}}(\lambda) = 1] \in \text{negl}(\lambda)$ holds.*

Experiment $\text{Exp}_{\mathcal{A}}^{\text{fra}}(\lambda)$
 $\text{state}_{\mathcal{I}} = (St, \mathcal{Y}, S_{\text{GM}}, S_{\text{OA}}) \leftarrow \text{Setup}(\lambda, N);$
 $(M^*, \sigma^*, t^*, RL_{t^*})$
 $\leftarrow \mathcal{A}(Q_{\text{pub}}, Q_{\text{keyGM}}, Q_{\text{keyOA}}, Q_{\text{b-join}}, Q_{\text{revoke}}, Q_{\text{sig}}, Q_{\text{read}}, Q_{\text{write}});$
If $\text{Verify}(\sigma^*, M^*, t^*, RL_{t^*}, \mathcal{Y}) = 0$ *then returns* 0;
 $i = \text{Open}(M^*, t^*, RL_{t^*}, \sigma^*, S_{\text{OA}}, \mathcal{Y}, St);$
If $i \notin U^b$ *returns* 0;
If $(\bigwedge_{j \in U^b \text{ s.t. } j=i} (j, t^*, M^*, *) \notin \text{Sigs})$ *then returns* 1;
Returns 0;

Full Anonymity The notion of anonymity is formalized by means of a game involving a two-stage adversary. The first stage is called **play** stage and allows the adversary $\mathcal{A}$ to modify $\text{state}_{\mathcal{I}}$ via Q_{write} -queries and open arbitrary signatures by probing Q_{open} . When the **play** stage ends, $\mathcal{A}$ chooses a message-period pair (M^*, t^*) as well as two pairs $(\text{sec}_0^*, \text{cert}_0^*), (\text{sec}_1^*, \text{cert}_1^*)$, consisting of a valid membership certificate and a corresponding membership secret. Then, the challenger flips a coin $d \leftarrow \{0, 1\}$ and computes a challenge signature σ^* using $(\text{sec}_d^*, \text{cert}_d^*)$. The adversary is given σ^* with the task of eventually guessing the bit $d \in \{0, 1\}$. Before doing so, he/she is allowed further oracle queries throughout the second stage, called **guess** stage, but is restricted not to query Q_{open} for (M^*, σ^*, t^*) .

Definition 25 (Full Anonymity) *A R-GS scheme is fully anonymous if, for any PPT adversary $\mathcal{A}$ involved in the following experiment, $\text{Adv}_{\mathcal{A}}^{\text{anon}}(\lambda) := |\Pr[\text{Exp}_{\mathcal{A}}^{\text{anon}}(\lambda) = 1] - 1/2|$ is negligible.*

Experiment $\mathbf{Exp}_A^{\text{anon}}(\lambda)$
 $\text{state}_{\mathcal{I}} = (St, \mathcal{Y}, \mathcal{S}_{\text{GM}}, \mathcal{S}_{\text{OA}}) \leftarrow \text{Setup}(\lambda);$
 $(aux, M^*, t^*, RL_{t^*}, (\text{sec}_0^*, \text{cert}_0^*), (\text{sec}_1^*, \text{cert}_1^*))$
 $\leftarrow \mathcal{A}(\text{play} : Q_{\text{pub}}, Q_{\text{keyGM}}, Q_{\text{revoke}}, Q_{\text{open}}, Q_{\text{read}}, Q_{\text{write}});$
If $\neg(\text{cert}_b \Rightarrow_{\mathcal{Y}} \text{sec}_b)$ for $b \in \{0, 1\}$ or if $\text{cert}_0 = \text{cert}_1$ returns 0;
Picks random bit $d \leftarrow \{0, 1\}$; $\sigma^* \leftarrow \text{Sign}(\mathcal{Y}, t^*, \text{cert}_d^*, \text{sec}_d^*, M^*);$
 $d' \leftarrow \mathcal{A}(\text{guess} : \sigma^*, aux, Q_{\text{pub}}, Q_{\text{keyGM}}, Q_{\text{open}}^{-\{(M^*, \sigma^*, t^*)\}}, Q_{\text{read}}, Q_{\text{write}});$
If $d' = d$ then returns 1;
Returns 0;

3.2 The First Scalable Group Signature with Revocation

For an arbitrary but polynomially bounded number N of users, in the security parameter, we build the first group signature scheme supporting revocation whose complexity dependency in N is at most poly-logarithmic.

3.2.1 Intuition and Technical Lemma

As already mentioned, the idea is to turn the NNL global ciphertext into a revocation list in the group signature. Each group member is associated with a leaf of a binary tree of height ℓ and the outcome of the join protocol is the user obtaining a membership certificate that contains the same key material as in the public-key variant of the SD method – *i.e.*, $O(\ell^2)$ HIBE private keys, as recalled in section 2.6.3. To ensure traceability and non-frameability, these NNL private keys are linked to a group element X , of which only the user knows the discrete logarithm, by means of structure-preserving signatures.

At each revocation period t , the group manager generates an up-to-date revocation list RL_t consisting of $O(r)$ HIBE ciphertexts, each of which is signed using a structure-preserving signature. When it comes to sign a message, the user $\mathcal{U}_i$ proves that he/she is not revoked by providing evidence that he/she is capable of decrypting one of the HIBE ciphertexts in RL_t . To this end, $\mathcal{U}_i$ commits to that HIBE ciphertext C_l and proves that he/she holds a key that decrypts C_l . To convince the verifier that C_l belongs to RL_t , he/she proves knowledge of a signature on the committed HIBE ciphertext C_l (this technique is borrowed from the set membership and range proofs of [134, 34]). Of course, to preserve the anonymity of signers, we need a HIBE scheme with constant-size ciphertexts (otherwise, the length of the committed ciphertext could betray the signer's location in the tree), which is why the Boneh-Boyen-Goh construction [22] is the ideal candidate.

More precisely, we require that the key be derived without any randomization. Namely, given a private key $d_{\text{ID}} = (D_1, D_2, K_{k+1}, \dots, K_{\ell})$ for $\text{ID} = (I_1, \dots, I_k)$ at level k , a private key for $\text{ID}' = (I_1, \dots, I_k, I_{k+1})$ is always de-

rived as per

$$\begin{aligned} d_{\text{ID}'} &= (D'_1, D'_2, K'_{k+2}, \dots, K'_\ell) = (D_1 \cdot K_{d+1}^{I_{k+1}}, D_2, K_{k+2}, \dots, K_\ell) \\ &= \left(g_2^\alpha \cdot (h_0 \cdot \prod_{i=1}^{k+1} h_i^{I_i})^r, g^r, h_{k+2}^r, \dots, h_\ell^r \right) \in \mathbb{G}^{\ell-k+1}. \end{aligned}$$

For this reason, a private key and its descendants will always share the same component D_2 . However, it does not affect the security of our construction since, in the join protocol, users are always given freshly generated HIBE private keys.

The scheme is made anonymous and non-frameable using the same techniques as Groth [81] in steps 4-6 of the signing algorithm. As for the security against misidentification attacks, we cannot prove it by relying on the standard collusion-resistance (captured by Definition 20 in Section 2.6.2) of the HIBE scheme. In the proof of Theorem 6, the problem appears in the treatment of forgeries that open to a revoked user: while this user cannot have obtained a private key that decrypts the committed HIBE ciphertext of the forgery (because he/she is revoked), unrevoked adversarially-controlled users can. To solve this problem, we need to rest on a non-standard security property (formalized by Definition 21) called “key-robustness”. This notion asks that, given a private key generated for some hierarchical identity using specific random coins, it be infeasible to compute the private key of a different identity for the *same random coins* and even *knowing* the *master secret key* of the HIBE scheme. It is also worth observing that a selective flavour of key-robustness suffices. Indeed, since the number of group members is always polynomial, the target identity $\text{ID}^\dagger$ can be guessed upfront with non-negligible probability in the proof of Lemma 2 on which relies the proof of Theorem 6. While unusual, this property can be proved (by Lemma 1 below) under the standard Diffie-Hellman assumption for the BBG construction.

Perhaps surprisingly, even though we rely on the BBG HIBE, we do not need its underlying q -type assumption [22]. The reason is that the master secret key of the scheme is unnecessary here as its role is taken over by the private key of a structure-preserving signature. In the ordinary BBG system (recalled in Section 2.6.2), private keys contain components of the form $(g_2^\alpha \cdot F(\text{ID})^r, g^r)$, for some $r \in \mathbb{Z}_p$, where g_2^α is the master secret key and $F(\text{ID})$ is a function of the hierarchical identity. In the join protocol, the master key g_2^α disappears: the user obtains a private key of the form $(F(\text{ID})^r, g^r)$ and an AHO signature is used to bind the user’s membership public key X to $D_2 = g^r$. The latter can be thought of as a public key for a one-time variant (the one-time nature is what allows for a proof of selective-message security in the standard model) of the Boneh-Lynn-Shacham signature [27]. The underlying one-time private key $r \in \mathbb{Z}_p$ is used to compute $F(\text{ID})^r$ as well as a number of delegation components allowing the derivation of signatures for messages of which ID is a prefix.

Technical Lemma The following lemma demonstrates that the BBG HIBE is selectively key-robust under the Diffie-Hellman assumption. The proof im-

licitly relies on the fact (implicitly noted in [88, 87]) that BLS-type signatures [27] can be proved secure in the standard model when the number of signing queries is bounded by a small constant (such as one here since the one-time public key g^r is used as a one-time public key).

Lemma 1 *The BBG HIBE scheme is selectively key-robust assuming that the CDH assumption holds in $\mathbb{G}$. More precisely, a selective key-robustness adversary $\mathcal{A}$ with advantage ε implies an algorithm $\mathcal{B}$ solving the CDH problem with advantage $\varepsilon \cdot (1 - 1/p)$.*

Proof. Towards a contradiction, let us assume that a selective adversary $\mathcal{A}$ has non-negligible advantage in the game of Definition 21. We show that $\mathcal{A}$ allows breaking the CDH assumption.

Algorithm $\mathcal{B}$ receives as input a CDH instance $(g, g^a, g^b) \in \mathbb{G}^3$ and undertakes to compute g^{ab} . At the beginning of its interaction with $\mathcal{A}$, the latter chooses a target identity $\text{ID}^\dagger = (I_1^\dagger, \dots, I_{\ell^\dagger}^\dagger) \in (\mathbb{Z}_p^*)^{\ell^\dagger}$ at the depth $\ell^\dagger \leq \ell$ of his/her choice. Then, $\mathcal{B}$ generates the master key pair $(\text{msk}_{\text{BBG}}, \text{mpk}_{\text{BBG}})$ by choosing $\alpha \xleftarrow{\$} \mathbb{Z}_p$, $g_2 \xleftarrow{\$} \mathbb{G}$ and setting $g_1 = g^\alpha$ as in the normal setup algorithm. Then, he/she picks $\gamma_1, \dots, \gamma_{\ell^\dagger} \xleftarrow{\$} \mathbb{Z}_p$, $\delta_0, \dots, \delta_\ell \xleftarrow{\$} \mathbb{Z}_p$ and defines

$$\begin{aligned} h_0 &= g^{\delta_0} \cdot (g^b)^{-\sum_{i=1}^{\ell^\dagger} \gamma_i I_i^\dagger} \\ h_i &= g^{\delta_i} \cdot (g^b)^{\gamma_i} & \text{for } i = 1, \dots, \ell^\dagger \\ h_i &= g^{\delta_i} & \text{for } i = \ell^\dagger + 1, \dots, \ell. \end{aligned}$$

To generate a private key $d_{\text{ID}^\dagger} = (D_1, D_2, K_{\ell^\dagger+1}, \dots, K_\ell)$ for the target identity $\text{ID}^\dagger$, $\mathcal{B}$ sets

$$\begin{aligned} D_1 &= g_2^\alpha \cdot (g^a)^{\delta_0 + \sum_{i=1}^{\ell^\dagger} \delta_i I_i^\dagger} \\ D_2 &= g^a \\ K_i &= (g^a)^{\delta_i} & \text{for } i = \ell^\dagger + 1, \dots, \ell, \end{aligned}$$

which form a valid private key for the random exponent $r = a$.

The adversary $\mathcal{A}$ is given $(\text{mpk}_{\text{BBG}}, \text{msk}_{\text{BBG}} = g_2^\alpha)$ and the private key $d_{\text{ID}^\dagger}$. His/her goal will be to produce a valid decryption component $D_{\text{ID}'} = (D'_1, D'_2)$ corresponding to an identity $\text{ID}' = (I'_1, \dots, I'_k) \in (\mathbb{Z}_p^*)^k$, for some $k \in \{1, \dots, \ell\}$, of which $\text{ID}^\dagger$ is not a prefix. In addition, $D_{\text{ID}'}$ should correspond to the same random exponent $r = a$ as $d_{\text{ID}^\dagger}$ (in other words, $D'_1 = D_2 = g^a$).

When $\mathcal{A}$ outputs his/her result $(\text{ID}', D_{\text{ID}'})$, we distinguish the following situations.

- If $k \leq \ell^\dagger$, we have

$$h_0 \cdot \prod_{i=1}^k h_i^{I'_i} = g^{\delta_0 + \sum_{i=1}^k \delta_i I'_i} \cdot (g^b)^{\sum_{i=1}^k \gamma_i \cdot (I'_i - I_i^\dagger) - \sum_{i=k+1}^{\ell^\dagger} \gamma_i I_i^\dagger} \quad (3.1)$$

and, with overwhelming probability $1 - 1/p$, it holds that

$$\sum_{i=1}^k \gamma_i \cdot (I'_i - I_i^\dagger) - \sum_{i=k+1}^{\ell^\dagger} \gamma_i I_i^\dagger \neq 0. \quad (3.2)$$

Indeed, the vector $\vec{\gamma} = (\gamma_1, \dots, \gamma_{\ell^\dagger})$ is chosen uniformly in $\mathbb{Z}_p^{\ell^\dagger}$ and it is independent of $\mathcal{A}$'s view.

- If $k = \ell^\dagger$, we must have $I'_i \neq I_i^\dagger$ for at least one $i \in \{1, \dots, \ell^\dagger\}$. Since the coordinates of $\vec{\gamma}$ are independent and uniformly distributed, the probability to have $\sum_{i=1}^{\ell^\dagger} \gamma_i \cdot (I'_i - I_i^\dagger) = 0$ is at most $1/p$ since we are bounding the probability of a random vector $\vec{\gamma}$ to be orthogonal to a given non-zero vector of $\mathbb{Z}_p^{\ell^\dagger}$.
- If $k < \ell^\dagger$, we may have $I'_i = I_i^\dagger$ for each $i \in \{1, \dots, k\}$ (i.e., ID' may be a prefix of $\text{ID}^\dagger$). In this situation, the probability to have $\sum_{i=k+1}^{\ell^\dagger} \gamma_i I_i^\dagger = 0$ is also $1/p$ since $(\gamma_{k+1}, \dots, \gamma_{\ell^\dagger})$ is independent of $\mathcal{A}$'s view and identities $I_{k+1}^\dagger, \dots, I_{\ell^\dagger}^\dagger$ are always non-zero. Finally, if ID' is not a prefix of $\text{ID}^\dagger$, there exists $i \in \{1, \dots, k\}$ such that $I'_i \neq I_i^\dagger$. Then, the same argument as in previous cases applies.

Since $\mathcal{A}$ presumably outputs a decryption component $D_{\text{ID}'} = (D_1, D_2) = (g_2^\alpha \cdot (h_0 \cdot \prod_{i=1}^k h_i^{I'_i})^a, g^a)$ with non-negligible probability ε , $\mathcal{B}$ can compute

$$g^{ab} = \left(\frac{D_1}{g_2^\alpha \cdot (g^a)^{\delta_0 + \sum_{i=1}^k \delta_i I'_i}} \right)^{1/(\sum_{i=1}^k \gamma_i \cdot (I'_i - I_i^\dagger) - \sum_{i=k+1}^{\ell^\dagger} \gamma_i I_i^\dagger)}$$

with probability $\varepsilon \cdot (1 - 1/p)$.

- If $k > \ell^\dagger$, there exists $i \in \{1, \dots, \ell^\dagger\}$ such that $I'_i \neq I_i^\dagger$ since $\text{ID}^\dagger$ cannot be a prefix of ID' . In this case, we can write

$$h_0 \cdot \prod_{i=1}^k h_i^{I'_i} = g^{\delta_0 + \sum_{i=1}^k \delta_i I'_i} \cdot (g^b)^{\sum_{i=1}^{\ell^\dagger} \gamma_i \cdot (I'_i - I_i^\dagger)}$$

where $\sum_{i=1}^{\ell^\dagger} \gamma_i \cdot (I'_i - I_i^\dagger) \neq 0$ with probability at least $1 - 1/p$. Then, the CDH solution g^{ab} can be found in the same way as in the case $k \leq \ell^\dagger$.

In either case $\mathcal{B}$ solves the CDH instance with the announced probability. $\square$

3.2.2 Scalable Construction

As in Section 2.6.3, $\langle x \rangle$ denotes the label of node $x \in \mathbb{T}$ and, for any sub-tree $\mathbb{T}_{x_j}$ rooted at x_j and any leaf v_i of $\mathbb{T}_{x_j}$, copath_{x_j} denotes the set of all siblings of nodes on the path from x_j to v_i , not counting x_j itself.

As is standard in group signatures, the description below assumes that, before joining the group, user $\mathcal{U}_i$ chooses a long term key pair $(\text{usk}[i], \text{upk}[i])$ and registers it in some PKI.

Setup (λ, N) : given a security parameter $\lambda \in \mathbb{N}$ and the permitted number of users $N = 2^\ell$,

1. Chooses bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$, with a generator $g \xleftarrow{\$} \mathbb{G}$.
2. Generates two key pairs $(sk_{\text{AHO}}^{(0)}, pk_{\text{AHO}}^{(0)})$ and $(sk_{\text{AHO}}^{(1)}, pk_{\text{AHO}}^{(1)})$ for the AHO signature of Section 2.2 in order to sign messages of two group elements. These key pairs consist of

$$pk_{\text{AHO}}^{(d)} = \left(G_r^{(d)}, H_r^{(d)}, G_z^{(d)} = G_r^{\gamma_z^{(d)}}, H_z^{(d)} = H_r^{\delta_z^{(d)}}, \right. \\ \left. \{G_i^{(d)} = G_r^{\gamma_i^{(d)}}, H_i^{(d)} = H_r^{\delta_i^{(d)}}\}_{i=1}^2, A^{(d)}, B^{(d)} \right)$$

and $sk_{\text{AHO}}^{(d)} = (\alpha_a^{(d)}, \alpha_b^{(d)}, \gamma_z^{(d)}, \delta_z^{(d)}, \{\gamma_i^{(d)}, \delta_i^{(d)}\}_{i=1}^2)$, where $d \in \{0, 1\}$.

3. As a CRS for the NIWI proof system, selects vectors $\mathbf{f} = (\vec{f}_1, \vec{f}_2, \vec{f}_3)$ s.t. $\vec{f}_1 = (f_1, 1, g) \in \mathbb{G}^3$, $\vec{f}_2 = (1, f_2, g) \in \mathbb{G}^3$, and $\vec{f}_3 = \vec{f}_1^{\xi_1} \cdot \vec{f}_2^{\xi_2}$, with $f_1 = g^{\beta_1}$, $f_2 = g^{\beta_2} \xleftarrow{\$} \mathbb{G}$ and $\beta_1, \beta_2, \xi_1, \xi_2 \xleftarrow{\$} \mathbb{Z}_p^*$.
4. Chooses $(U, V) \xleftarrow{\$} \mathbb{G}^2$ that, together with f_1, f_2, g , will form a public encryption key for the tag-based cryptosystem of Section 2.6.1.
5. Generates a master public key mpk_{BBG} for the Boneh-Boyen-Goh HIBE. Such a public key consists¹ of $mpk_{\text{BBG}} = (\{h_i\}_{i=0}^\ell)$, where $\ell = \log_2(N)$, and no master secret key is needed.
6. Selects an injective encoding² function $\mathcal{H} : \{0, 1\}^{\leq \ell} \rightarrow \mathbb{Z}_p^*$ and a strongly unforgeable one-time signature $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$.
7. Sets $\mathcal{S}_{\text{GM}} := (sk_{\text{AHO}}^{(0)}, sk_{\text{AHO}}^{(1)})$, $\mathcal{S}_{\text{OA}} := (\beta_1, \beta_2)$ as authorities' private keys and the group public key is

$$\mathcal{Y} := \left(g, pk_{\text{AHO}}^{(0)}, pk_{\text{AHO}}^{(1)}, mpk_{\text{BBG}}, \mathbf{f}, (U, V), \mathcal{H}, \Sigma \right).$$

Join $(\text{GM}, \mathcal{U}_i)$: the GM and the prospective user $\mathcal{U}_i$ run the following protocol $[\text{J}_{\text{user}}(\lambda, \mathcal{Y}), \text{J}_{\text{GM}}(\lambda, St, \mathcal{Y}, \mathcal{S}_{\text{GM}})]$:

1. $\text{J}_{\text{user}}(\lambda, \mathcal{Y})$ picks $x \xleftarrow{\$} \mathbb{Z}_p$ and computes $X = g^x$ which is sent to $\text{J}_{\text{GM}}(\lambda, St, \mathcal{Y}, \mathcal{S}_{\text{GM}})$. If the value X already appears in some entry transcript_j of the database St_{trans} , J_{GM} aborts and returns $\perp$ to J_{user} .

¹As mentioned earlier, in comparison with the original HIBE scheme (recalled in appendix 2.6.2) where mpk_{BBG} includes $(g_1 = g^\alpha, g_2)$ and $msk_{\text{BBG}} = g_2^\alpha$, the public elements g_1 and g_2 have disappeared.

²This encoding allows making sure that “identities” will be non-zero at each level. Since the set $\{0, 1\}^{\leq \ell}$ is of cardinality $\sum_{i=0}^\ell 2^i = 2^{\ell+1} - 1 < p - 1$, such a function can be efficiently constructed without any intractability assumption.

2. J_{GM} assigns to $\mathcal{U}_i$ an available leaf v_i of label $\langle v_i \rangle = v_{i,1} \dots v_{i,\ell} \in \{0,1\}^\ell$ in the tree T . Let $x_0 = \epsilon$, $x_1, \dots, x_{\ell-1}$, $x_\ell = v_i$ be the path from v_i to the root ϵ of T . For $j = 0$ to ℓ , J_{GM} does the following.
 - a. Considers the sub-tree T_{x_j} rooted at node x_j . Lets copath_{x_j} be the co-path from x_j to v_i .
 - b. For each node $w \in \text{copath}_{x_j}$, since x_j is an ancestor of w , $\langle x_j \rangle$ is a prefix of $\langle w \rangle$ and we denote by $w_{\ell_1} \dots w_{\ell_2} \in \{0,1\}^{\ell_2 - \ell_1 + 1}$, for some $\ell_1 \leq \ell_2 \leq \ell$, the suffix of $\langle w \rangle$ coming right after $\langle x_j \rangle$.
 - b.1 Chooses a random $r \xleftarrow{\$} \mathbb{Z}_p$ and computes a HIBE private key $d_w = (D_{w,1}, D_{w,2}, K_{w,\ell_2 - \ell_1 + 3}, \dots, K_{w,\ell})$ equals to

$$\left((h_0 \cdot h_1^{\mathcal{H}(\langle x_j \rangle)} \cdot h_2^{\mathcal{H}(w_{\ell_1})} \dots h_{\ell_2 - \ell_1 + 2}^{\mathcal{H}(w_{\ell_2})})^r, g^r, h_{\ell_2 - \ell_1 + 3}^r, \dots, h_\ell^r \right)$$
 for the identity $(\mathcal{H}(\langle x_j \rangle), \mathcal{H}(w_{\ell_1}), \dots, \mathcal{H}(w_{\ell_2})) \in (\mathbb{Z}_p^*)^{\ell_2 - \ell_1 + 2}$.
 - b.2 Generates an AHO signature $\sigma_w = (\theta_{w,1}, \dots, \theta_{w,7})$ on the vector $(X, D_{w,2}) \in \mathbb{G}^2$ using $sk_{\text{AHO}}^{(0)}$ so as to bind the HIBE private key d_w to the value X that identifies $\mathcal{U}_i$.
3. J_{GM} sends $\langle v_i \rangle \in \{0,1\}^\ell$, and the HIBE private keys $\{d_w\}_{w \in \text{copath}_{x_j}}$ for $j = 0, \dots, \ell$ to J_{user} that verifies their validity. If these keys are all well-formed, J_{user} acknowledges them by generating a digital signature $\text{sig}_i = \text{Sign}_{\text{usk}[i]}(X || \{\{d_w\}_{w \in \text{copath}_{x_j}}\}_{j=0}^\ell)$ and sends it back to J_{GM} .
4. J_{GM} checks if $\text{Verify}_{\text{upk}[i]}(X || \{\{d_w\}_{w \in \text{copath}_{x_j}}\}_{j=0}^\ell, \text{sig}_i) = 1$ holds or not. If not J_{GM} aborts. Otherwise, J_{GM} returns the AHO signatures $\{\{\sigma_w\}_{w \in \text{copath}_{x_j}}\}_{j=0}^\ell$ to J_{user} and stores the conversation transcript $\text{transcript}_i = (X, \{\{d_w, \sigma_w\}_{w \in \text{copath}_{x_j}}\}_{j=0}^\ell, \text{sig}_i)$ in St_{trans} .
5. J_{user} defines $\text{cert}_i = (\langle v_i \rangle, \{\{d_w, \sigma_w\}_{w \in \text{copath}_{x_j}}\}_{j=0}^\ell, X)$ as the membership certificate, where X will serve as the tag that identifies $\mathcal{U}_i$. The membership secret sec_i is defined to be $\text{sec}_i = x$.

Revoke($\mathcal{Y}, \mathcal{S}_{\text{GM}}, t, \mathcal{R}_t$):

1. Parses $\mathcal{S}_{\text{GM}}$ as $\mathcal{S}_{\text{GM}} := (sk_{\text{AHO}}^{(0)}, sk_{\text{AHO}}^{(1)})$.
2. Using the SD covering algorithm, finds a cover of the unrevoked user set $\{1, \dots, N\} \setminus \mathcal{R}_t$ as the union of disjoint subsets of the form $S_{k_1, u_1}, \dots, S_{k_m, u_m}$, with $m \leq 2 \cdot |\mathcal{R}_t| - 1$.
3. For $i = 1$ to m , does the following.
 - a. Considers S_{k_i, u_i} as the difference between sub-trees rooted at an internal node x_{k_i} and one of its descendants x_{u_i} . The label of x_{u_i} can be written $\langle x_{u_i} \rangle = \langle x_{k_i} \rangle || u_{i,\ell_{i,1}} \dots u_{i,\ell_{i,2}}$ for some $\ell_{i,1} < \ell_{i,2} \leq \ell$ and where $u_{i,\kappa} \in \{0,1\}$ for each $\kappa \in \{\ell_{i,1}, \dots, \ell_{i,2}\}$. Then, computes an encoding of S_{k_i, u_i} as a group element

$$C_i = h_0 \cdot h_1^{\mathcal{H}(\langle x_{k_i} \rangle)} \cdot h_2^{\mathcal{H}(u_{i,\ell_{i,1}})} \dots h_{\ell_{i,2} - \ell_{i,1} + 2}^{\mathcal{H}(u_{i,\ell_{i,2}})}.$$

Note that C_i can be thought of as a de-randomized HIBE ciphertext for $(\mathcal{H}(\langle x_{k_i} \rangle), \mathcal{H}(u_{i,\ell_{i,1}}), \dots, \mathcal{H}(u_{i,\ell_{i,2}})) \in (\mathbb{Z}_p^*)^{\ell_{i,2}-\ell_{i,1}+2}$ as the hierarchical identity.

- b. To authenticate the HIBE ciphertext C_i and bind it to the revocation period t , uses $sk_{\text{AHO}}^{(1)}$ to generate an AHO signature $\Theta_i = (\Theta_{i,1}, \dots, \Theta_{i,7}) \in \mathbb{G}^7$ on the pair $(C_i, g^t) \in \mathbb{G}^2$, where the period number t is interpreted as an element of $\mathbb{Z}_p$.

Returns the revocation data RL_t which is defined to be

$$RL_t = \left(t, \mathcal{R}_t, \{ \langle x_{k_i} \rangle, \langle x_{u_i} \rangle, (C_i, \Theta_i = (\Theta_{i,1}, \dots, \Theta_{i,7})) \}_{i=1}^m \right) \quad (3.3)$$

Sign($\mathcal{Y}, t, RL_t, \text{cert}_i, \text{sec}_i, M$): returns $\perp$ if $i \in \mathcal{R}_t$. Otherwise, to sign $M \in \{0, 1\}^*$, generates a one-time signature key pair $(\text{SK}, \text{VK}) \leftarrow \mathcal{G}(\lambda)$. Parses cert_i as $(\langle v_i \rangle, \{ \{(d_w, \sigma_w)\}_{w \in \text{copath}_{x_j}}\}_{j=0}^\ell, X)$ and sec_i as $x \in \mathbb{Z}_p$. Then, $\mathcal{U}_i$ conducts the following steps.

1. Using RL_t , determines the set S_{k_l, u_l} , with $l \in \{1, \dots, m\}$, that contains the leaf v_i (this subset must exist since $i \notin \mathcal{R}_t$) and lets x_{k_l} and x_{u_l} denote the primary and secondary roots of S_{k_l, u_l} . Since x_{k_l} is an ancestor of x_{u_l} , we can write $\langle x_{u_l} \rangle = \langle x_{k_l} \rangle || u_{l,\ell_1} \dots u_{l,\ell_2}$, for some $\ell_1 < \ell_2 \leq \ell$ and with $u_{l,\kappa} \in \{0, 1\}$ for each $\kappa \in \{\ell_1, \dots, \ell_2\}$. The signer $\mathcal{U}_i$ computes a HIBE decryption key of the form

$$(D_{l,1}, D_{l,2}) = \left((h_0 \cdot h_1^{\mathcal{H}(\langle x_{k_l} \rangle)} \cdot h_2^{\mathcal{H}(u_{l,\ell_1})} \dots h_{\ell_2-\ell_1+2}^{\mathcal{H}(u_{l,\ell_2})})^r, g^r \right). \quad (3.4)$$

This is possible since, if we denote by $\langle x_{k_l} \rangle || u_{l,\ell_1} \dots u_{l,\ell'_1}$ the shortest prefix of $\langle x_{u_l} \rangle$ that is not a prefix of $\langle v_i \rangle$, the key material $\{d_w\}_{w \in \text{copath}_{x_{k_l}}}$ corresponding to the sub-tree rooted at x_{k_l} contains a HIBE private key $d_w = (D_{w,1}, D_{w,2}, K_{w,\ell'_1-\ell_1+3}, \dots, K_{w,\ell})$ such that

$$d_w = \left((h_0 \cdot h_1^{\mathcal{H}(\langle x_{k_l} \rangle)} \cdot h_2^{\mathcal{H}(u_{l,\ell_1})} \dots h_{\ell'_1-\ell_1+2}^{\mathcal{H}(u_{l,\ell'_1})})^r, g^r, h_{\ell'_1-\ell_1+3}^r, \dots, h_\ell^r \right),$$

which allows deriving a key of the form (3.4) for the same $r \in \mathbb{Z}_p$ (i.e., $D_{l,2} = D_{w,2}$).

2. To prove his/her ability to “decrypt” C_l , $\mathcal{U}_i$ first re-randomizes Θ_l as $\{\Theta'_{l,i}\}_{i=1}^7 \leftarrow \text{ReRand}(pk_{\text{AHO}}^{(1)}, \Theta_l)$. Then, he/she computes a Groth-Sahai commitment com_{C_l} to C_l as well as commitments $\{\text{com}_{\Theta'_{l,i}}\}_{i \in \{1,2,5\}}$ to $\{\Theta'_{l,i}\}_{i \in \{1,2,5\}}$. He/she generates a proof π_{C_l} that C_l is a certified HIBE ciphertext for period t : i.e., π_{C_l} provides evidence that, for $T_A^{(1)} = A^{(1)} \cdot e(\Theta'_{l,3}, \Theta'_{l,4})^{-1} \cdot e(G_2^{(1)}, g^t)^{-1}$ and $T_B^{(1)} = B^{(1)} \cdot e(\Theta'_{l,6}, \Theta'_{l,7})^{-1} \cdot e(H_2^{(1)}, g^t)^{-1}$ we have

$$\begin{aligned} T_A^{(1)} &= e(G_z^{(1)}, \Theta'_{l,1}) \cdot e(G_r^{(1)}, \Theta'_{l,2}) \cdot e(G_1^{(1)}, C_l), \\ T_B^{(1)} &= e(H_z^{(1)}, \Theta'_{l,1}) \cdot e(H_r^{(1)}, \Theta'_{l,5}) \cdot e(H_1^{(1)}, C_l). \end{aligned} \quad (3.5)$$

Then, $\mathcal{U}_i$ generates commitments $\{com_{D_{l,i}}\}_{i=1}^2$ to the HIBE key components $\{D_{l,i}\}_{i=1}^2$ derived at step 1 and computes a proof π_{D_l} that $e(D_{l,1}, g) = e(C_l, D_{l,2})$. The latter is quadratic and requires 9 group elements. Since $\{\Theta'_{l,i}\}_{i \in \{3,4,6,7\}}$ are constants, equations (3.5) are linear and require 3 elements each (see details in 2.27). Hence, π_{C_l} and π_{D_l} take 15 elements altogether.

3. Lets $\sigma_l = (\theta_{l,1}, \dots, \theta_{l,7})$ be the AHO signature on $(X, D_{l,2})$. Computes $\{\theta'_{l,i}\}_{i=1}^7 \leftarrow \text{ReRand}(pk_{\text{AHO}}^{(0)}, \sigma_l)$ and generates commitments $\{com_{\theta'_{l,i}}\}_{i \in \{1,2,5\}}$ to $\{\theta'_{l,i}\}_{i \in \{1,2,5\}}$ as well as a commitment com_X to X . Then, generates a proof π_{σ_l} that committed variables satisfy the verification equations with $T_A^{(0)} = A^{(0)} \cdot e(\theta'_{l,3}, \theta'_{l,4})^{-1}$ and $T_B^{(0)} = B^{(0)} \cdot e(\theta'_{l,6}, \theta'_{l,7})^{-1}$

$$\begin{aligned} T_A^{(0)} &= e(G_z^{(0)}, \theta'_{l,1}) \cdot e(G_r^{(0)}, \theta'_{l,2}) \cdot e(G_1^{(0)}, X) \cdot e(G_2^{(0)}, D_{l,2}), \\ T_B^{(0)} &= e(H_z^{(0)}, \theta_{l,1}) \cdot e(H_r^{(0)}, \theta'_{l,5}) \cdot e(H_1^{(0)}, X) \cdot e(H_2^{(0)}, D_{l,2}). \end{aligned}$$

Since these equations are linear, π_{σ_l} requires 6 group elements.

4. Using VK as a tag (we assume that it is first hashed onto $\mathbb{Z}_p$ in such a way that it can be interpreted as a $\mathbb{Z}_p$ element), computes a Kiltz's tag-based encryption (of Section 2.6.1) of X by drawing $z_1, z_2 \xleftarrow{\$} \mathbb{Z}_p$ and setting

$$(\Psi_1, \Psi_2, \Psi_3, \Psi_4, \Psi_5) = (f_1^{z_1}, f_2^{z_2}, X \cdot g^{z_1+z_2}, (g^{\text{VK}} \cdot U)^{z_1}, (g^{\text{VK}} \cdot V)^{z_2}).$$

5. Generates a NIZK proof that $com_X = (1, 1, X) \cdot \vec{f}_1^{\phi_{X,1}} \cdot \vec{f}_2^{\phi_{X,2}} \cdot \vec{f}_3^{\phi_{X,3}}$ and (Ψ_1, Ψ_2, Ψ_3) are BBS encryptions of the same value X . If we write $\vec{f}_3 = (f_{3,1}, f_{3,2}, f_{3,3})$, the commitment com_X can be written as $(f_1^{\phi_{X,1}} \cdot f_{3,1}^{\phi_{X,3}} \cdot f_2^{\phi_{X,2}} \cdot f_{3,2}^{\phi_{X,3}}, X \cdot g^{\phi_{X,1}+\phi_{X,2}} \cdot f_{3,3}^{\phi_{X,3}})$, so that we have

$$com_X \odot (\Psi_1, \Psi_2, \Psi_3)^{-1} = (f_1^{\tau_1} \cdot f_{3,1}^{\tau_3}, f_2^{\tau_2} \cdot f_{3,2}^{\tau_3}, g^{\tau_1+\tau_2} \cdot f_{3,3}^{\tau_3}) \quad (3.6)$$

with $\tau_1 = \phi_{X,1} - z_1$, $\tau_2 = \phi_{X,2} - z_2$, $\tau_3 = \phi_{X,3}$. The signer $\mathcal{U}_i$ commits to $\tau_1, \tau_2, \tau_3 \in \mathbb{Z}_p$ (by computing $com_{\tau_j} = \vec{\varphi}^{\tau_j} \cdot \vec{f}_1^{\phi_{\tau_j,1}} \cdot \vec{f}_2^{\phi_{\tau_j,2}}$, for $j \in \{1, 2, 3\}$, using the vector $\vec{\varphi} = \vec{f}_3 \cdot (1, 1, g)$ and random $\{\phi_{\tau_j,1}, \phi_{\tau_j,2}\}_{j=1}^3$), and generates proofs $\{\pi_{eq-com,j}\}_{j=1}^3$ that τ_1, τ_2, τ_3 satisfy the three relations (3.6). Since these are linear equations, proofs $\{\pi_{eq-com,j}\}_{j=1}^3$ cost 2 elements each (see details in 2.28).

6. Computes $\sigma_{\text{VK}} = g^{1/(x+\text{VK})}$ and generates a commitment $com_{\sigma_{\text{VK}}}$ to σ_{VK} . Then, generates a NIWI proof that committed variables σ_{VK} and X satisfy $e(\sigma_{\text{VK}}, X \cdot g^{\text{VK}}) = e(g, g)$. This relation is quadratic and requires a proof consisting of 9 group elements. We denote this proof by $\pi_{\sigma_{\text{VK}}} = (\vec{\pi}_{\sigma_{\text{VK}},1}, \vec{\pi}_{\sigma_{\text{VK}},2}, \vec{\pi}_{\sigma_{\text{VK}},3})$.

7. Computes $\sigma_{ots} = \mathcal{S}(\text{SK}, (M, RL_t, \Psi_1, \Psi_2, \Psi_3, \Psi_4, \Psi_5, \Omega, \mathbf{com}, \mathbf{\Pi}))$ where $\Omega = \{\Theta'_{l,i}, \theta'_{l,i}\}_{i \in \{3,4,6,7\}}$ and

$$\begin{aligned} \mathbf{com} &= (com_{C_l}, \{com_{D_{l,i}}\}_{i=1}^2, com_X, \{com_{\Theta'_{l,i}}\}_{i \in \{1,2,5\}}, \\ &\quad \{com_{\theta'_{l,i}}\}_{i \in \{1,2,5\}}, \{com_{\tau_i}\}_{i=1}^3, com_{\sigma_{VK}}) \\ \mathbf{\Pi} &= (\pi_{C_l}, \pi_{D_l}, \pi_{\sigma_l}, \pi_{eq-com,1}, \pi_{eq-com,2}, \pi_{eq-com,3}, \pi_{\sigma_{VK}}) \end{aligned}$$

Returns the signature $\sigma = (\text{VK}, \Psi_1, \Psi_2, \Psi_3, \Psi_4, \Psi_5, \Omega, \mathbf{com}, \mathbf{\Pi}, \sigma_{ots})$.

Verify($\sigma, M, t, RL_t, \mathcal{Y}$): parses σ as above and does the following.

1. If $\mathcal{V}(\text{VK}, (\Psi_1, \Psi_2, \Psi_3, \Psi_4, \Psi_5, \Omega, \mathbf{com}, \mathbf{\Pi}), \sigma_{ots}) = 0$, returns 0.
2. Returns 0 if $e(\Psi_1, g^{\text{VK}} \cdot U) \neq e(f_1, \Psi_4)$ or $e(\Psi_2, g^{\text{VK}} \cdot V) \neq e(f_2, \Psi_5)$.
3. Returns 1 if all proofs properly verify. Otherwise, return 0.

Open($M, t, RL_t, \sigma, \mathcal{S}_{\text{OA}}, \mathcal{Y}, St$): given $\mathcal{S}_{\text{OA}} = (\beta_1, \beta_2)$, parses the signature σ as above and returns $\perp$ if $\text{Verify}(\sigma, M, t, RL_t, \mathcal{Y}) = 0$. Otherwise, computes $\tilde{X} = \Psi_3 \cdot \Psi_1^{-1/\beta_1} \cdot \Psi_2^{-1/\beta_2}$. In the database St_{trans} , finds a record $\langle i, \text{transcript}_i = (X, \{\{d_w, \sigma_w\}_{w \in \text{copath}_{x_j}}\}_{j=0}^\ell, sig_i) \rangle$ such that $X = \tilde{X}$. If no such record exists in St_{trans} , returns $\perp$. Otherwise, returns i .

From an efficiency point of view, for each $i \in \{1 \dots, m\}$, RL_t comprises 8 group elements plus the labels of nodes that identify S_{k_i, u_i} . If $\lambda_{\mathbb{G}}$ denotes the bitlength of a group element, the number of bits of RL_t is thus bounded by $2 \cdot |\mathcal{R}_t| \cdot (8 \cdot \lambda_{\mathbb{G}} + 2 \log N) < 2 \cdot |\mathcal{R}_t| \cdot (9\lambda_{\mathbb{G}})$ bits (as $\log N < \lambda_{\mathbb{G}}/2$ since $\lambda \leq \lambda_{\mathbb{G}}$ and N is polynomial). The size of revocation lists thus amounts to that of at most $18 \cdot |\mathcal{R}_t|$ group elements.

Group members need $O(\log^3 N)$ group elements to store their membership certificate. As far as the size of signatures goes, $\mathbf{com}$ and $\mathbf{\Pi}$ require 42 and 36 group elements, respectively. If the one-time signature of Section 2.2.2 is used, σ consists of 96 group elements, which is less than twice the size of Groth's signatures [81]. At the 128-bit security level, if each element has a representation of 512 bits, a signature takes 6 kB.

Verifying signatures takes constant time. The cost of each signature generation is dominated by at most $\ell = \log N$ exponentiations to derive a HIBE private key at step 1. However, this step only has to be executed once per revocation period, at the first signature of that period.

3.2.3 Security Analysis

Security Against Misidentification

Theorem 6 (Misidentification) *The scheme is secure against misidentification attacks assuming that the q -SFP problem is hard for $q = \max(\ell^2 \cdot q_a, q_r^2)$, where q_a and q_r denote the maximal numbers of $Q_{\text{a-join}}$ queries and Q_{revoke} queries, respectively, and $\ell = \log N$.*

Proof. To mount a successful misidentification attack, the adversary $\mathcal{A}$ must output a non-trivial signature for which the opening algorithm fails to point to an unrevoked adversarially-controlled group member.

Let $\sigma^* = (\text{VK}^*, \Psi_1^*, \Psi_2^*, \Psi_3^*, \Psi_4^*, \Psi_5^*, \Omega^*, \mathbf{com}^*, \Pi^*, \sigma_{ots}^*)$ denote $\mathcal{A}$'s forgery and parse $\mathbf{com}^*$ as

$$\mathbf{com}^* = \left(com_C^*, \{com_{D_i}^*\}_{i=1}^2, com_X^*, \{com_{\Theta_i}^*\}_{i \in \{1,2,5\}}, \right. \\ \left. \{com_{\theta_i'}^*\}_{i \in \{1,2,5\}}, \{com_{\tau_i}^*\}_{i=1}^3, com_{\sigma_{VK}^*}^* \right).$$

By hypothesis, it must be the case that $\text{Open}(M^*, t^*, RL_{t^*}, \sigma^*, \mathcal{S}_{OA}, \mathcal{V}, St) \notin U^a \setminus \mathcal{R}_{t^*}$, where U^a denotes the set of adversarially-controlled users. Depending on extractable commitments com_X^* , com_C^* , $\{com_{D_i}^*\}_{i=1}^2$, $\{com_{\Theta_i}^*\}_{i \in \{1,2,5\}}$, $\{com_{\theta_i'}^*\}_{i \in \{1,2,5\}}$ and their contents, we distinguish the following cases:

- **Type I forgeries** are those for which $com_{C_l}^*$ contains a group element C^* such that (C^*, g^{t^*}) was never signed when the latest revocation list RL_{t^*} was generated.
- **Type II forgeries** are such that com_C^* contains a properly certified HIBE ciphertext for period t^* (say $C^* = C_l^*$, for some $l \in \{1, \dots, m\}$, where $C_1^*, \dots, C_m^*$ are the HIBE ciphertexts of RL_{t^*}). However, the execution of **Open** reveals a previously unseen X^* or points some revoked user $i \in U^a \cap \mathcal{R}_{t^*}$ although σ^* provides convincing evidence that the committed private key (D_1^*, D_2^*) allows decrypting C_l^* and that committed elements $\{\theta_i^*\}_{i=1}^7$ form a valid signature on (X^*, D_2^*) . In this case, we have two situations:
 - a. The pair (X^*, D_2^*) was *not* signed by J_{GM} in any execution of **Join**. It means that either: (1) **Open** uncovers a value X^* that does not appear anywhere in St_{trans} . (2) The traced user $i \in U^a \cap \mathcal{R}_{t^*}$ colluded with some unrevoked user $j \in U^a$ whose leaf is in S_{k_l, u_l} – of which C_l^* is an encoding – and managed to forge an AHO signature so as to link his X^* to an authorized key (D_1^*, D_2^*) for S_{k_l, u_l} .
 - b. The pair (X^*, D_2^*) was signed by J_{GM} at some execution of **Join**. At first, we would like to use the Type II.b adversary to break the standard selective semantic security of the HIBE system (cf. Definition 20 in Section 2.6.2). As it turns out, even if we were using the original BBG HIBE (with its master secret key), such a reduction would be unlikely to work because the set S_{k_l, u_l} may contain unrevoked users in U^a , so that $\mathcal{A}$ obtained private keys that do decrypt C_l^* . Instead, we rely on the security property that we call “selective key-robustness” (cf. Definition 21) and which relies on a weaker assumption than the standard security of the BBG HIBE. Observe that, when user i joined the group, he/she cannot have been issued the private key (D_1^*, D_2^*) (which is an authorized key for S_{k_l, u_l}) since he/she is revoked at period t^* . However, since (X^*, D_2^*) was signed

by J_{GM} , user i must have obtained from J_{GM} a HIBE private key of the form (D_1, D_2^*) , where $D_1 \neq D_1^*$, for an identity other than the one encoded by S_{k_l, u_l} . In this case, as will be showed in Lemma 2, the key-robustness property is necessarily broken in the HIBE scheme.

It is easy to see that Type I and Type II.a forgeries imply a forger against the AHO signature scheme (the proof is straightforward and omitted).

Lemma 2 below demonstrates that a Type II.b attack necessarily contradicts the key-robustness property of the Boneh-Boyen-Goh HIBE scheme and thus the Diffie-Hellman assumption, as established by Lemma 1.

Finally, one can readily check that an adversary cannot produce a signature σ^* to win the misidentification game without resorting to one of the above kinds of forgeries. The result of the theorem follows from the fact that the SFP assumption implies the CDH assumption. $\square$

Lemma 2 *The advantage of any Type II.b forger $\mathcal{A}$ is at most*

$$\text{Adv}_{\mathcal{A}}^{\text{mis-id-II.b}}(\lambda) \leq 4 \cdot N^2 \cdot \left(1 - \frac{1}{p}\right) \cdot \text{Adv}^{\text{CDH}}(\lambda)$$

where N denotes the maximal number of users.

Proof. At the beginning of its interaction with its challenger, our selective key-robustness adversary $\mathcal{B}$ chooses a random node $x_j \in \mathbb{T}$ and a random descendant x'_j of x_j (alternatively, $\mathcal{B}$ can more simply choose two distinct random nodes in the tree and, with some probability, x'_j will be in the subtree rooted at x_j). Since x'_j is a descendant of x_j , its label $\langle x'_j \rangle$ can be written $\langle x'_j \rangle = \langle x_j \rangle || w_{\ell_1} \dots w_{\ell_2}$, for some integers $\ell_1, \ell_2 \in \{1, \dots, \ell\}$ and where $w_i \in \{0, 1\}$ for each $i \in \{\ell_1, \dots, \ell_2\}$. Then, $\mathcal{B}$ declares $\text{ID}^\dagger = (\mathcal{H}(\langle x_j \rangle), \mathcal{H}(w_{\ell_1}), \dots, \mathcal{H}(w_{\ell_2}))$ as its target identity at level $\ell_2 - \ell_1 + 2$. The key-robustness challenger replies by returning a master key pair $(msk_{\text{BBG}}, mpk_{\text{BBG}})$ consisting of

$$mpk_{\text{BBG}} = \left((\mathbb{G}, \mathbb{G}_T), g, g_1 = g^\alpha, g_2, \{h_i\}_{i=0}^\ell \right), \quad msk_{\text{BBG}} = g_2^\alpha$$

together with a private key $d_{\text{ID}^\dagger} = (D_1^\dagger, D_2^\dagger, K_{\ell_2 - \ell_1 + 3}^\dagger, \dots, K_\ell^\dagger)$ for the identity $\text{ID}^\dagger$.

Then, $\mathcal{B}$ uses mpk_{BBG} to construct the group public key $\mathcal{Y}$ and generates all other public key elements (including $pk_{\text{AHO}}^{(0)}$ and $pk_{\text{AHO}}^{(1)}$) according to the specification of the setup algorithm. In particular, $\mathcal{B}$ retains the group manager's secret key $\mathcal{S}_{GM} = (sk_{\text{AHO}}^{(0)}, sk_{\text{AHO}}^{(1)})$ and uses it to answer $Q_{\text{a-join}}$ -queries.

At each $Q_{\text{a-join}}$ -query, $\mathcal{B}$ executes the join protocol on behalf of J_{GM} and proceeds exactly as the real J_{GM} does (remember that it knows $\mathcal{S}_{GM}$ and can thus perfectly simulate J_{GM}) with one exception. Namely, when executing step b.1 of Join, if the private key D_w has to be computed for the target identity $\text{ID}^\dagger$, $\mathcal{B}$ uses the private key $d_{\text{ID}^\dagger}$ that it received from its challenger to compute

$$(D_{w,1}, D_{w,2}, K_{w, \ell_2 - \ell_1 + 3}, \dots, K_{w, \ell}) = (D_1^\dagger / g_2^\alpha, D_2^\dagger, K_{\ell_2 - \ell_1 + 3}^\dagger, \dots, K_\ell^\dagger).$$

As for Q_{pub} , Q_{revoke} , Q_{read} and Q_{OA} -queries, $\mathcal{B}$ simply answers them as the real oracles would.

If the game terminates and $\mathcal{B}$ did not have to compute a private key for $\text{ID}^\dagger$ at some $Q_{\text{a-join}}$ -query, then $\mathcal{B}$ halts and reports failure since it must have guessed the wrong tree nodes x_j and x'_j . Otherwise (i.e., if $d_{\text{ID}^\dagger}$ was used to compute part of a membership certificate), we know that $\mathcal{A}$'s forgery σ^* will contain a committed HIBE private key (D_1^*, D_2^*) and a committed value X^* such that (X^*, D_2^*) is one of the pairs that were signed by $\mathcal{B}$ during some $Q_{\text{a-join}}$ -query. Moreover, the pair (X^*, D_2^*) was associated with some HIBE private key $d_{\text{ID}^\diamond} = (D_1, D_2^*, K_{\ell_2-\ell_1+3}, \dots, K_\ell)$ for a certain hierarchical identity $\text{ID}^\diamond$ at step b.1 of Join. Since that identity is entirely defined by two nodes at the extremities of a path in the tree T , these two nodes happen to be x_j and x'_j – so that $\text{ID}^\diamond = \text{ID}^\dagger$ – with non-negligible probability $1/(2N-1)^2 > 1/4N^2$.

Therefore, with probability at least $1/4N^2$, the value D_2^* is precisely the element $D_2^\dagger$ of the challenge private key $d_{\text{ID}^\dagger}$ sent by the key-robustness challenger. Also, we note that $d_{\text{ID}^\diamond}$ and (D_1^*, D_2^*) necessarily correspond to distinct identities as the signature σ^* would not trace to a revoked user otherwise. If the desirable event $\text{ID}^\diamond = \text{ID}^\dagger$ comes about, this implies that either:

- $(D_1^*, D_2^*) = (D_1^\dagger, D_2^\dagger)$, which means that $d_{\text{ID}^\dagger}$ and (D_1^*, D_2^*) correspond to distinct hierarchical identities $\text{ID}^\dagger = (I_1, \dots, I_{\ell_2-\ell_1+2})$ and $(I'_1, \dots, I_k)$, with $k \in \{1, \dots, \ell\}$, such that

$$h_0 \cdot \prod_{i=1}^{\ell_2-\ell_1+2} h_i^{I_i} = h_0 \cdot \prod_{i=1}^{k'} h_i^{I'_i}.$$

Such a collision is known (as shown in [88][Section 1.2], for example) to occur with negligible probability under the discrete logarithm assumption.

- $D_2^* = D_2^\dagger$ and $D_1^* \neq D_1^\dagger$, in which case $\mathcal{B}$ wins the selective key-robustness game. It does so by outputting the decryption component $(g_2^\alpha \cdot D_1^*, D_2^*)$ – after having extracted (D_1^*, D_2^*) from $\{\text{com}_{D_i}^*\}_{i=1}^2$ using (β_1, β_2) – and the identity ID' corresponding to the HIBE ciphertext C_l^* of the revocation list. Indeed, if $\mathcal{B}$ correctly guessed x_j and x'_j , ID' cannot be a descendant of $\text{ID}^\dagger$ as long as σ^* opens to a revoked user in $U^a \cap \mathcal{R}_{t^*}$.

Since the probability to have $\text{ID}^\diamond = \text{ID}^\dagger$ is at least $1/4N^2$ and due to the multiplicative factor $(1 - 1/p)$ in the statement of Lemma 1, the announced result follows. $\square$

Security against Framing Attacks

Theorem 7 (Non-frameability) *The scheme is secure against framing attacks assuming that: (i) the q_b -SDH assumption holds in $\mathbb{G}$, where q_b is the maximal number of $Q_{\text{b-join}}$ -queries; (ii) Σ is a strongly unforgeable one-time signature.*

Proof. As in [81], we consider two kinds of framing attacks that can be possibly mounted by a non-frameability adversary $\mathcal{A}$.

- **Type I attacks:** the adversary $\mathcal{A}$ generates a forgery $\sigma^* = (\text{VK}^*, \Psi_1^*, \Psi_2^*, \Psi_3^*, \Psi_4^*, \Psi_5^*, \Omega^*, \text{com}^*, \Pi^*, \sigma_{ots}^*)$ for which the one-time verification key VK^* was used by some honest group member $i \in U^b$ when answering a Q_{sig} -query.
- **Type II attacks:** $\mathcal{A}$ outputs a forgery $\sigma^* = (\text{VK}^*, \Psi_1^*, \Psi_2^*, \Psi_3^*, \Psi_4^*, \Psi_5^*, \Omega^*, \text{com}^*, \Pi^*, \sigma_{ots}^*)$ for which the one-time verification key VK^* was never used by Q_{sig} to answer a signing query on behalf of an honest user $i \in U^b$.

It is immediate that Type I attacks imply a breach in the unforgeability of the one-time signature. Lemma 3 shows that no PPT adversary can produce a Type II forgery as long as the Strong Diffie-Hellman assumption holds. $\square$

Lemma 3 *The scheme is secure against framing attacks of Type II if the q_s -SDH problem is hard. More precisely, the advantage of any adversary after q_s Q_{sig} -queries and q_b $Q_{\text{b-join}}$ -queries is at most $\text{Adv}^{\text{fra-II}}(\lambda) \leq q_b \cdot \text{Adv}^{q_s\text{-SDH}}(\lambda)$.*

Proof. By hypothesis, the adversary $\mathcal{A}$ comes up with a forgery (M^*, σ^*) that opens to some honest user $i \in U^b$ and that did not issue a signature containing the verification key VK^* . The same proof as in [81] shows that the Strong Diffie-Hellman assumption can be broken.

Given a problem instance $(\tilde{g}, \tilde{g}^a, \dots, \tilde{g}^{(a^{q_s})}) \in \mathbb{G}^{q_s+1}$, the simulator $\mathcal{B}$ generates q_s one-time signature keys pairs $(\text{SK}_i, \text{VK}_i) \leftarrow \mathcal{G}(\lambda)$ for $i = 1$ to q_s . Then, using standard techniques (see [21][Lemma 3.2]) it builds a generator g and a randomly distributed public value $X^\dagger = g^a$ – which implicitly defines $x^\dagger = \log_g(X^\dagger) = a$ – such that it knows $\{(g^{1/(a+\text{VK}_i)}, \text{VK}_i)\}_{i=1}^{q_s}$.

Next, using the newly generated g , $\mathcal{B}$ generates key pairs $\{(sk_{\text{AHO}}^{(b)}, pk_{\text{AHO}}^{(b)})\}_{b=0,1}$ for the AHO signature (note that group elements of $\{pk_{\text{AHO}}^{(b)}\}_{b=0,1}$ are computed as powers of g) and uses $pk_{\text{AHO}}^{(0)}, pk_{\text{AHO}}^{(1)}$ to form the group public key

$$\mathcal{V} = (g, pk_{\text{AHO}}^{(0)}, pk_{\text{AHO}}^{(1)}, mpk_{\text{BEG}}, \mathbf{f}, U, V, \mathcal{H}, \Sigma).$$

In the latter, the Groth-Sahai CRS $\mathbf{f} = (\vec{f}_1, \vec{f}_2, \vec{f}_3)$ is prepared for the perfect soundness setting, *i.e.*, with $\vec{f}_1 = (f_1 = g^{\beta_1}, 1, g)$, $\vec{f}_2 = (1, f_2 = g^{\beta_2}, g)$ and $\vec{f}_3 = \vec{f}_1^{\xi_1} \odot \vec{f}_2^{\xi_2}$, where $\xi_1, \xi_2 \xleftarrow{\$} \mathbb{Z}_p^*$.

Should the adversary $\mathcal{A}$ decide to corrupt the group manager or the opening authority during the game, $\mathcal{B}$ has $\mathcal{S}_{\text{GM}} = (sk_{\text{AHO}}^{(0)}, sk_{\text{AHO}}^{(1)})$ and $\mathcal{S}_{\text{OA}} = (\beta_1, \beta_2) = (\log_g(f_1), \log_g(f_2))$ at its disposal. At the beginning of the game, $\mathcal{B}$ picks a random index $j^* \xleftarrow{\$} \{1, \dots, q_b\}$ and interacts with $\mathcal{A}$ as follows.

- Q_{keyGM} -queries: if $\mathcal{A}$ decides to corrupt the group manager, $\mathcal{B}$ surrenders $\mathcal{S}_{\text{GM}} = (sk_{\text{AHO}}^{(0)}, sk_{\text{AHO}}^{(1)})$.

- $Q_{\text{b-join}}$ -queries: when $\mathcal{A}$, acting as a rogue group manager, requests the introduction of a new honest user i in the group, $\mathcal{B}$ starts interacting with $\mathcal{A}$ in an execution of Join and runs J_{user} on behalf of the prospective user. Namely, $\mathcal{B}$'s behaviour depends on the index $j \in \{1, \dots, q_b\}$ of the $Q_{\text{b-join}}$ -query.
 - If $j \neq j^*$, $\mathcal{B}$ follows exactly the specification of J_{user} .
 - If $j = j^*$, $\mathcal{B}$ sends the value $X^\dagger$ to J_{GM} at step 1 of Join . This implicitly defines user j^* 's membership secret to be the unknown exponent $\text{sec}_{j^*} = a$ of the SDH instance. In subsequent steps of the join protocol, $\mathcal{B}$ proceeds as the real J_{user} would. When Join terminates, the challenger $\mathcal{B}$ obtains a membership certificate $\text{cert}_{j^*} = (\langle v_j \rangle, \{\{d_w, \sigma_w\}_{w \in \text{copath}_{x_j}}\}_{j=0}^\ell, X^\dagger)$.
- Q_{pub} -queries: can be treated as in the real game, by having the simulator return $\mathcal{Y}$.
- Q_{sig} -queries: when $\mathcal{A}$ asks user $i \in U^b$ to sign a message M , the simulator $\mathcal{B}$ can answer the query by running the real signature generation algorithm if $i \neq j^*$. Otherwise (namely, if $i = j^*$), $\mathcal{B}$ uses the next available pair $\{(g^{1/(a+\text{VK}_i)}, \text{VK}_i)\}_{i=1}^{q_s}$ to define σ_{VK_i} . $\mathcal{B}$ also recalls the membership certificate $\text{cert}_{j^*} = (\langle v_j \rangle, \{\{d_w, \sigma_w\}_{w \in \text{copath}_{x_j}}\}_{j=0}^\ell, X^\dagger)$ obtained from the J_{GM} -executing adversary at the j^* -th $Q_{\text{b-join}}$ -query. It is easy to see that, using σ_{VK_i} and cert_{j^*} , $\mathcal{B}$ can easily generate all signature components and sign them all using SK_i .

Finally, $\mathcal{A}$ outputs a signature $\sigma^* = (\text{VK}^*, \Psi_1^*, \Psi_2^*, \Psi_3^*, \Psi_4^*, \Psi_5^*, \Omega^*, \text{com}^*, \Pi^*, \sigma_{\text{ots}}^*)$, for some message M^* , that opens to some user $i^* \in U^b$ who did not sign M^* . At this point, $\mathcal{B}$ halts and declares failure if it turns out that $X^\dagger \neq \Psi_3^* \cdot \Psi_1^{*-1/\beta_1} \cdot \Psi_2^{*-1/\beta_2}$ since, in this case, it was unfortunate when drawing the random index j^* . Still, with probability $1/q_b$, the signature σ^* opens to the user introduced at the j^* -th $Q_{\text{b-join}}$ -query and $(\Psi_1^*, \Psi_2^*, \Psi_3^*)$ does decrypt to X^* . In this situation, the perfect soundness of the proof system ensures that $\text{com}_{\sigma_{\text{VK}^*}}^*$ is a commitment to a group element $\sigma_{\text{VK}^*}^*$ such that $e(\sigma_{\text{VK}^*}^*, X^\dagger \cdot g^{\text{VK}^*}) = e(g, g)$. Since σ^* is a Type II forgery, $\mathcal{B}$ can use β_1, β_2 to compute a BBS decryption of $\text{com}_{\sigma_{\text{VK}^*}}^*$ and obtain a solution $(\sigma_{\text{VK}^*}, \text{VK}^*)$ to the q_b -SDH instance. $\square$

Anonymity

As for the anonymity property, it naturally relies on the DLIN assumption. The proof is essentially identical to that of Lemma 5 in [81] but we give it for completeness.

Theorem 8 (Anonymity) *The advantage of any anonymity adversary is at most*

$$\text{Adv}^{\text{anon}}(\mathcal{A}) \leq \text{Adv}^{\text{ots}}(\lambda) + 3 \cdot \text{Adv}^{\text{DLIN}}(\lambda),$$

where the first term is $\mathcal{A}$'s probability of breaking the strong unforgeability of the one-time signature.

Proof. We consider a sequence of games at the end of which even an unbounded adversary has no advantage. In Game i , we call S_i the event that $\mathcal{A}$ wins and define $\text{Adv}_i = |\Pr[S_i] - 1/2|$.

Game 1: is the experiment of definition 25. In the play stage, the adversary $\mathcal{A}$ can obtain the group public key $\mathcal{Y}$, the group manager's private key $\mathcal{S}_{\text{GM}} = (sk_{\text{AHO}}^{(0)}, sk_{\text{AHO}}^{(1)})$. It can also ask for the opening of any group signature and read/write the content of $\text{state}_{\mathcal{L}}$. When it decides to enter the challenge phase, it outputs a message M^* , a period index t^* and two membership certificate/secret $(\text{cert}_0^*, \text{sec}_0^*)$ and $(\text{cert}_1^*, \text{sec}_1^*)$ such that $\text{cert}_b^* \Leftarrow_{\mathcal{Y}} \text{sec}_b^*$ for $b = 0, 1$. The simulator $\mathcal{B}$ flips a fair coin $d \xleftarrow{\$} \{0, 1\}$ and computes $\sigma^* \leftarrow \text{Sign}(\mathcal{Y}, t^*, RL_{t^*}, \text{cert}_d^*, \text{sec}_d^*, M^*)$, where t^* is determined by the history of Q_{revoke} -queries. The signature σ^* is given as a challenge to $\mathcal{A}$ who has to guess $d \in \{0, 1\}$ after another series of queries (under the natural restriction of not querying the opening of σ^*). We have $\text{Adv}_1 = \mathbf{Adv}^{\text{anon}}(\mathcal{A})$.

Game 2: is as **Game 1** but $\mathcal{B}$ halts if $\mathcal{A}$ queries the opening of a signature σ containing the same one-time verification key VK^* as in the challenge phase (we assume w.l.o.g. that $(\text{SK}^*, \text{VK}^*)$ is generated at the outset of the game). If such a query is made before the challenge phase, it means that $\mathcal{A}$ was able to forge a one-time signature even without having seen a signature. If the query occurs after the challenge phase, then the strong unforgeability of Σ is broken. We can thus write $|\Pr[S_2] - \Pr[S_1]| \leq \mathbf{Adv}^{\text{ots}}(\lambda)$.

Game 3: we change the generation of $\mathcal{Y}$ so as to answer Q_{open} -queries without using the secret exponents $\beta_1, \beta_2 \in \mathbb{Z}_p$ that define $\mathcal{S}_{\text{OA}}$. To this end, $\mathcal{B}$ chooses $\alpha_u, \alpha_v \xleftarrow{\$} \mathbb{Z}_p^*$, and defines $U = g^{-\text{VK}^*} \cdot f_1^{\alpha_u}$, and $V = g^{-\text{VK}^*} \cdot f_2^{\alpha_v}$. It is not hard to see (see [99] for details) that, for any Q_{open} -query containing a BBS encryption $(\Psi_1, \Psi_2, \Psi_3) = (f_1^{z_1}, f_2^{z_2}, X \cdot g^{z_1+z_2})$, the values (Ψ_4, Ψ_5) reveal g^{z_1} and g^{z_2} (and thus the encrypted X) since $\text{VK} \neq \text{VK}^*$ unless the event introduced in Game 2 occurs. To generate the challenge signature σ^* at period t^* , $\mathcal{B}$ first computes $(\Psi_1^*, \Psi_2^*, \Psi_3^*)$ and then $(\Psi_4^*, \Psi_5^*) = (\Psi_1^{\alpha_u}, \Psi_2^{\alpha_v})$. It sets the challenge signature to be $\sigma^* = (\text{VK}^*, \Psi_1^*, \Psi_2^*, \Psi_3^*, \Psi_4^*, \Psi_5^*, \Omega^*, \text{com}^*, \Pi^*, \sigma_{\text{ots}}^*)$. It can be checked that the distributions of $\mathcal{Y}$ and σ^* are unchanged and we have $\Pr[S_3] = \Pr[S_2]$.

Game 4: in the setup phase, we generate the CRS $\mathbf{f} = (\vec{f}_1, \vec{f}_2, \vec{f}_3)$ of the proof system for the perfect WI setting. We choose $\vec{f}_3 = \vec{f}_1^{\xi_1} \odot \vec{f}_2^{\xi_2} \odot (1, 1, g)^{-1}$ instead of $\vec{f}_3 = \vec{f}_1^{\xi_1} \odot \vec{f}_2^{\xi_2}$ so that $\vec{f}_1, \vec{f}_2$ and $\vec{f}_3$ are linearly independent. Any significant change in $\mathcal{A}$'s behaviour yields a distinguisher for the DLIN problem and we can write $|\Pr[S_4] - \Pr[S_3]| = 2 \cdot \mathbf{Adv}^{\text{DLIN}}(\mathcal{B})$. As noted in [84], proofs in the WI setting reveal no information on which witnesses they were generated from.

Game 5: we modify the generation of the challenge σ^* and use the trapdoor of the CRS (i.e., ξ_1, ξ_2 s.t. $\vec{\varphi} = \vec{f}_1^{\xi_1} \odot \vec{f}_2^{\xi_2}$) to simulate proofs $\{\pi_{\text{eq-com}, j}\}_{j=1}^3$

that $(\Psi_1^*, \Psi_2^*, \Psi_3^*)$ and com_X encrypt of the same value. It is known [84] that linear multi-exponentiation equations always have perfectly NIZK proofs on a simulated CRS. For any satisfiable relation, (ξ_1, ξ_2) allows generating proofs without using the witnesses τ_1, τ_2, τ_3 for which (3.6) holds and simulated proofs are perfectly indistinguishable from real ones. Hence, $\Pr[S_5] = \Pr[S_4]$.

Game 6: in the computation of Ψ_3^* , we now replace $g^{z_1+z_2}$ by a random group element in the challenge σ^* . Since $\mathcal{B}$ does not explicitly use $z_1 = \log_{f_1}(\Psi_1^*)$, $z_2 = \log_{f_2}(\Psi_2^*)$, any change in $\mathcal{A}$'s behaviour yields a distinguisher for the DLIN problem and $|\Pr[S_6] - \Pr[S_5]| \leq \mathbf{Adv}^{\text{DLIN}}(\mathcal{B})$. In Game 6, we have $\Pr[S_6] = 1/2$. Indeed, when we consider the challenge σ^* , Groth-Sahai commitments are all perfectly hiding in the WI setting and proofs Π reveal nothing about the underlying witnesses (in particular, NIZK proofs $\{\pi_{eq-com,j}\}_{j=1}^3$ are generated without using them) and $(\Psi_1^*, \Psi_2^*, \Psi_3^*)$ perfectly hides X^* . Finally, randomized signature components $\Omega^* = \{\Theta'_{l,i}^*, \theta'_{l,i}^*\}_{i \in \{3,4,6,7\}}$ are information-theoretically independent of the corresponding messages and the remaining components of AHO signatures Θ_l^* and θ_l^* .

When combining the above, $\mathcal{A}$'s advantage can be bounded by $\mathbf{Adv}^{\text{anon}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{ots}}(\lambda) + 3 \cdot \mathbf{Adv}^{\text{DLIN}}(\lambda)$ as stated by the theorem. $\square$

3.3 Group Signature with Almost-for-free Revocation

In the scalable group signature with revocation of the previous section, group members have to store a polylogarithmic number of group elements. In many applications, however, this can rapidly become unwieldy even for moderately large groups: for example, with $N = 1000 \approx 2^{10}$, users may have to privately store thousands of group elements. In order to be competitive with other group signatures in the standard model such as [81] and still be able to revoke members while keeping them “stateless”, it is highly desirable to reduce this complexity.

In this section, we start with the same approach so as to instantiate the Subset Difference method, but obtain private keys of *constant* size without degrading other performance criteria. This may sound somewhat surprising since, in the SD method, (poly)logarithmic complexities inherently seem inevitable in several metrics. Here, the only dependency on N is a $O(\log N)$ -size group public key. Our system uses revocation lists of size $O(r)$, which is on par with Certificate Revocation Lists (CRLs) of standard PKIs. It is worth noting that RLs are *not* part of the group public key: verifiers only need to know the number of the latest revocation period and should not bother to read RLs entirely.

To this end, we introduce a new technique to leverage the NNL subset cover framework in the context of group signatures. Namely, we provide a way for users to efficiently prove their membership to one of the generic subsets in the NNL subset cover framework. This technique makes our revocable group sig-

natures competitive with ordinary group signatures (*i.e.*, without revocation) in the standard model. Moreover, unrevoked members (as in PKIs) still do not need to update their keys at each revocation.

3.3.1 Intuition behind our Scheme

The key idea is as follows. As in the NNL framework, group members are assigned to a leaf of a binary tree and each unrevoked member should belong to exactly one subset in the cover of authorized leafs determined by the group manager. Instead of relying on hierarchical identity-based encryption as in the public-key variant of NNL, we use a novel way for users to non-interactively prove their membership to some generic subset of the SD method using a proof of constant size.

To construct these “compact anonymous membership proofs”, we employ *concise* vector commitment schemes (cf. Section 2.4.4), where each commitment can be opened w.r.t. individual coordinates in a space-efficient manner (namely, the size of a coordinate-wise opening does not depend on the length of the vector). These vector commitments interact nicely with the specific shape of subsets – as differences between two subtrees – in the SD method. Using them, we compactly encode as a vector the path from the user’s leaf to the root. To provide evidence of their inclusion in one of the SD subsets, group members successively prove the equality and inequality between two coordinates of their vector (*i.e.*, two nodes of the path from their leaf to the root) and specific node labels indicated by an appropriate entry of the revocation list. This is where the position-wise openability of concise commitments is very handy. Of course, for anonymity purposes, the relevant entry of the revocation list only appears in committed form in the group signature. In order to prove that he/she is using a legal entry of the revocation list, the user generates a set membership proof.

Technical Intuition. The number of users is assumed to be $N = 2^{\ell-1} \in \text{poly}(\lambda)$, for some integer ℓ , so that each group member is assigned to a leaf of the tree. Each node is assigned a unique identifier. For simplicity, the root is identified by $\text{ID}(\epsilon) = 1$ and, for each other node x , we define the identifier $\text{ID}(x) \in \{1, \dots, 2N - 1\}$ to be $\text{ID}(x) = 2 \cdot \text{ID}(\text{parent}(x)) + b$, where $\text{parent}(x)$ denotes x ’s father in the tree and $b = 0$ (resp. $b = 1$) if x is the left (resp. right) child of its father. The root of the tree is assigned the identifier $\text{ID}_\epsilon = 1$.

At the beginning of each revocation period t , the GM generates an up-to-date revocation list RL_t containing one entry for each generic subset $S_{k_1, u_1}, \dots, S_{k_m, u_m}$ produced by the Subset Difference method. These subsets are encoded in such a way that unrevoked users can anonymously prove their membership of one of them. Our technique allows doing this using a proof of *constant* size.

The intuition is as follows. In the generation of RL_t , for each $i \in \{1, \dots, m\}$, if x_{k_i} (resp. x_{u_i}) denotes the primary (resp. secondary) root of S_{k_i, u_i} , the GM encodes S_{k_i, u_i} as a vector of group elements R_i that determines the levels of

nodes x_{k_i} and x_{u_i} in the tree (which are called ϕ_i and ψ_i hereafter) and the identifiers $\text{ID}(x_{k_i})$ and $\text{ID}(x_{u_i})$. Then, the resulting vector R_i is authenticated by means of a structure-preserving signature Θ_i , which is included in RL_t and will be used in a set membership proof.

During the join protocol, users obtain from the GM a structure-preserving signature on a compact encoding C_v – which is computed as a concise commitment to a vector of node identifiers $(I_1, \dots, I_\ell)$ – of the path $(I_1, \dots, I_\ell)$ between their leaf v and the root ϵ . This path is encoded as a single group element.

The group manager uses two key pairs for the AHO structure-preserving signature of Section 2.2.2. The first one is used during the join protocol to bind a group element X chosen by the user, who knows $x = \log_g(X)$, to the path from the user's leaf v to the root ϵ .

In order to anonymously prove his/her non-revocation, a group member $\mathcal{U}_i$ uses RL_t to determine the generic subset S_{k_l, u_l} , with $l \in \{1, \dots, m\}$, where his/her leaf v_i lies. He/she commits to the corresponding vector of group elements R_l that encodes the node identifiers $\text{ID}(x_{k_l})$ and $\text{ID}(x_{u_l})$ of the primary and secondary roots of S_{k_l, u_l} at levels ϕ_l and ψ_l , respectively. If $(I_1, \dots, I_\ell)$ identifies the path from his/her leaf v_i to ϵ , the unrevoked member $\mathcal{U}_i$ generates a membership proof for the subset S_{k_l, u_l} by proving that $\text{ID}(x_{k_l}) = I_{\phi_l}$ and $\text{ID}(x_{u_l}) \neq I_{\psi_l}$ (in other words, that x_{k_l} is an ancestor of v_i and x_{u_l} is not). To succinctly prove these statements, $\mathcal{U}_i$ uses the properties of the LY concise vector commitment scheme³ recalled in Section 2.4.4. Finally, in order to convince the verifier that he used a legal element of RL_t , $\mathcal{U}_i$ follows the technique of [34] and proves knowledge of a signature Θ_l on the committed vector of group elements R_l . By doing so, $\mathcal{U}_i$ thus provides evidence that his/her leaf v_i is a member of some authorized subset S_{k_l, u_l} without revealing $l \in \{1, \dots, m\}$.

In order to obtain the strongest flavour of anonymity (*i.e.*, where the adversary has access to a signature opening oracle), the scheme uses Kiltz's tag-based encryption scheme (cf. Section 2.6.1) as in Groth's construction [81] exactly as we did in the previous construction. In non-frameability concerns, the group member $\mathcal{U}_i$ also generates a weak Boneh-Boyen signature of Section 2.2.2 (which yields a fully secure signature when combined with a one-time signature) using $x = \log_g(X)$, where $X \in \mathbb{G}$ is a group element certified by the GM and bound to the path $(I_1, \dots, I_\ell)$ during the join protocol.

3.3.2 Construction with Compact Keys

As in standard security models for group signatures, we assume that, before joining the group, user $\mathcal{U}_i$ chooses a long term key pair $(\text{usk}[i], \text{upk}[i])$ and registers it in some PKI.

Setup(λ, N): given a security parameter $\lambda \in \mathbb{N}$ and the permitted number of users $N = 2^{\ell-1}$,

³No randomness is required here. We only need the componentwise binding property.

1. Chooses bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$, with a random generator $g \leftarrow \mathbb{G}$.
2. Defines $n_0 = 2$ and $n_1 = 5$. Generates two key pairs $(sk_{\text{AHO}}^{(0)}, pk_{\text{AHO}}^{(0)})$ and $(sk_{\text{AHO}}^{(1)}, pk_{\text{AHO}}^{(1)})$ for the AHO signature in order to sign messages of n_0 and n_1 group elements, respectively. These key pairs are

$$pk_{\text{AHO}}^{(d)} = \left(G_r^{(d)}, H_r^{(d)}, G_z^{(d)} = G_r^{\gamma_z^{(d)}}, H_z^{(d)} = H_r^{\delta_z^{(d)}}, \right. \\ \left. \{G_i^{(d)} = G_r^{\gamma_i^{(d)}}, H_i^{(d)} = H_r^{\delta_i^{(d)}}\}_{i=1}^{n_d}, A^{(d)}, B^{(d)} \right)$$

and $sk_{\text{AHO}}^{(d)} = (\alpha_a^{(d)}, \alpha_b^{(d)}, \gamma_z^{(d)}, \delta_z^{(d)}, \{\gamma_i^{(d)}, \delta_i^{(d)}\}_{i=1}^{n_d})$, where $d \in \{0, 1\}$. These two schemes will be used to sign messages consisting of 2 and 5 group elements, respectively.

3. Generates a public key $ck = (g_1, \dots, g_\ell, g_{\ell+2}, \dots, g_{2\ell}) \in \mathbb{G}^{2\ell-1}$ for ℓ -dimension vectors of the concise vector commitment scheme recalled in section 2.4.4. The trapdoor $g_{\ell+1}$ is not needed and can be discarded.
4. As a CRS for the NIWI proof system, selects vectors $\mathbf{f} = (\vec{f}_1, \vec{f}_2, \vec{f}_3)$ s.t. $\vec{f}_1 = (f_1, 1, g) \in \mathbb{G}^3$, $\vec{f}_2 = (1, f_2, g) \in \mathbb{G}^3$, and $\vec{f}_3 = \vec{f}_1^{\xi_1} \cdot \vec{f}_2^{\xi_2}$, with $f_1 = g^{\beta_1}$, $f_2 = g^{\beta_2}$ in $\mathbb{G}$ and random $\beta_1, \beta_2, \xi_1, \xi_2 \leftarrow \mathbb{Z}_p^*$. We also define the vector $\vec{\varphi} = \vec{f}_3 \cdot (1, 1, g)$.
5. Chooses random $(U, V) \leftarrow \mathbb{G}^2$ that, together with generators $f_1, f_2, g \in \mathbb{G}$, will form a public encryption key.
6. Selects a strongly unforgeable one-time signature $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$.
7. Sets $\mathcal{S}_{\text{GM}} := (sk_{\text{AHO}}^{(0)}, sk_{\text{AHO}}^{(1)})$, $\mathcal{S}_{\text{OA}} := (\beta_1, \beta_2)$ as authorities' private keys and the group public key is

$$\mathcal{V} := \left(g, pk_{\text{AHO}}^{(0)}, pk_{\text{AHO}}^{(1)}, ck, \mathbf{f}, \vec{\varphi}, (U, V), \Sigma \right).$$

Join^(GM, $\mathcal{U}_i$): the GM and the prospective user $\mathcal{U}_i$ run the following protocol $[\text{J}_{\text{user}}(\lambda, \mathcal{V}), \text{J}_{\text{GM}}(\lambda, St, \mathcal{V}, \mathcal{S}_{\text{GM}})]$:

1. $\text{J}_{\text{user}}(\lambda, \mathcal{V})$ draws $x \leftarrow \mathbb{Z}_p$ at random and computes $X = g^x$ which is sent to $\text{J}_{\text{GM}}(\lambda, St, \mathcal{V}, \mathcal{S}_{\text{GM}})$. If $X \in \mathbb{G}$ already appears in some entry transcript_j of the database St_{trans} , J_{GM} halts and returns $\perp$ to J_{user} .
2. J_{GM} assigns to $\mathcal{U}_i$ an available leaf v of identifier $\text{ID}(v)$ in the tree $\mathbb{T}$. Let $x_1, \dots, x_\ell$ be the path from $x_\ell = v$ to the root $x_1 = \epsilon$ of $\mathbb{T}$. Let also $(I_1, \dots, I_\ell) = (\text{ID}(x_1), \dots, \text{ID}(x_\ell))$ be the corresponding vector of identifiers (with $I_1 = 1$ and $I_\ell = \text{ID}(v) \in \{N, \dots, 2N-1\}$). Then, J_{GM} does the following.
 - (a) Computes a compact encoding $C_v = \prod_{\kappa=1}^{\ell} g_{\ell+1-\kappa}^{I_\kappa} = g_\ell^{I_1} \cdots g_1^{I_\ell}$ of $(I_1, \dots, I_\ell)$.

- (b) Using $sk_{\text{AHO}}^{(0)}$, generates an AHO signature $\sigma_v = (\theta_{v,1}, \dots, \theta_{v,7})$ on the pair $(X, C_v) \in \mathbb{G}^2$ so as to bind the encoded path C_v to the value X that identifies $\mathcal{U}_i$.
3. J_{GM} sends $\text{ID}(v) \in \{N, \dots, 2N - 1\}$ and C_v to J_{user} that halts if $\text{ID}(v) \notin \{N, \dots, 2N - 1\}$ or if C_v is found incorrect. Otherwise, J_{user} sends a signature $\text{sig}_i = \text{Sign}_{\text{usk}[i]}(X || (I_1, \dots, I_\ell))$ to J_{GM} .
4. J_{GM} checks that $\text{Verify}_{\text{upk}[i]}((X || (I_1, \dots, I_\ell)), \text{sig}_i) = 1$. If not J_{GM} aborts. Otherwise, J_{GM} returns the AHO signature σ_v to J_{user} and stores $\text{transcript}_i = (X, \text{ID}(v), C_v, \sigma_v, \text{sig}_i)$ in the database St_{trans} .
5. J_{user} defines the membership certificate cert_i as $\text{cert}_i = (\text{ID}(v), X, C_v, \sigma_v) \in \{N, \dots, 2N - 1\} \times \mathbb{G}^9$, where X will serve as the tag identifying $\mathcal{U}_i$. The membership secret sec_i is defined as $\text{sec}_i = x \in \mathbb{Z}_p$.

Revoke($\mathcal{Y}, \mathcal{S}_{\text{GM}}, t, \mathcal{R}_t$): Parses $\mathcal{S}_{\text{GM}}$ as $\mathcal{S}_{\text{GM}} := (sk_{\text{AHO}}^{(0)}, sk_{\text{AHO}}^{(1)})$ and does the following.

1. Using the covering algorithm of the SD method, finds a cover of the unrevoked user set $\{1, \dots, N\} \setminus \mathcal{R}_t$ as the union of disjoint subsets of the form $S_{k_1, u_1}, \dots, S_{k_m, u_m}$, with $m \leq 2 \cdot |\mathcal{R}_t| - 1$.
2. For $i = 1$ to m , does the following.
 - (a) Considers S_{k_i, u_i} as the difference between sub-trees rooted at an internal node x_{k_i} and one of its descendants x_{u_i} . Lets $\phi_i, \psi_i \in \{1, \dots, \ell\}$ be the depths of x_{k_i} and x_{u_i} , respectively, in $\mathbb{T}$ assuming that the root ϵ is at depth 1. Encodes S_{k_i, u_i} as a vector $(g_{\phi_i}, g_1^{\text{ID}(x_{k_i})}, g_{\psi_i}, g^{\text{ID}(x_{u_i})})$.
 - (b) To authenticate S_{k_i, u_i} and bind it to the revocation period t , uses $sk_{\text{AHO}}^{(1)}$ to generate an AHO signature $\Theta_i = (\Theta_{i,1}, \dots, \Theta_{i,7}) \in \mathbb{G}^7$ on the message $R_i = (g^t, g_{\phi_i}, g_1^{\text{ID}(x_{k_i})}, g_{\psi_i}, g^{\text{ID}(x_{u_i})}) \in \mathbb{G}^5$, where the period number t is interpreted as an element of $\mathbb{Z}_p$.

Returns the revocation data

$$RL_t = \left(t, \mathcal{R}_t, \{ \phi_i, \psi_i, \text{ID}(x_{k_i}), \text{ID}(x_{u_i}), \Theta_i = (\Theta_{i,1}, \dots, \Theta_{i,7}) \}_{i=1}^m \right). \quad (3.7)$$

Sign($\mathcal{Y}, t, RL_t, \text{cert}_i, \text{sec}_i, M$): returns $\perp$ if $i \in \mathcal{R}_t$. Otherwise, to sign $M \in \{0, 1\}^*$, generates a one-time signature key pair $(\text{SK}, \text{VK}) \leftarrow \mathcal{G}(\lambda)$. Parses cert_i as $\text{cert}_i = (\text{ID}(v_i), X, C_{v_i}, \sigma_{v_i}) \in \{N, \dots, 2N - 1\} \times \mathbb{G}^9$ and sec_i as $x \in \mathbb{Z}_p$. Lets $\epsilon = x_1, \dots, x_\ell = v_i$ be the path connecting v_i to the root ϵ of $\mathbb{T}$ and lets $(I_1, \dots, I_\ell) = (\text{ID}(x_1), \dots, \text{ID}(x_\ell))$ be the vector of node identifiers. First, $\mathcal{U}_i$ generates a commitment $\text{com}_{C_{v_i}}$ to the encoding C_{v_i} of the path $(I_1, \dots, I_\ell)$ from v_i to the root. Then, he does the following.

1. Using RL_t , finds the set S_{k_l, u_l} , with $l \in \{1, \dots, m\}$, that contains the leaf v_i identified by $\text{ID}(v_i)$. Lets x_{k_l} and x_{u_l} denote the primary and

secondary roots of S_{k_l, u_l} at depths ϕ_l and ψ_l , respectively. Since x_{k_l} is an ancestor of v_i but x_{u_l} is not, it must be the case that $I_{\phi_l} = \text{ID}(x_{k_l})$ and $I_{\psi_l} \neq \text{ID}(x_{u_l})$.

2. In order to prove that v_i belongs to S_{k_l, u_l} without leaking l , $\mathcal{U}_i$ first re-randomizes the l -th AHO signature Θ_l contained in RL_t as $\{\Theta'_{l,i}\}_{i=1}^7 \leftarrow \text{ReRand}(pk_{\text{AHO}}^{(1)}, \Theta_l)$. Then, he/she commits to the l -th revocation message

$$R_l = (R_{l,1}, \dots, R_{l,5}) = (g^t, g_{\phi_l}, g_1^{\text{ID}(x_{k_l})}, g_{\psi_l}, g^{\text{ID}(x_{u_l})}) \quad (3.8)$$

and its signature $\Theta'_l = (\Theta'_{l,1}, \dots, \Theta'_{l,7})$ by computing Groth-Sahai commitments $\{com_{R_{l,\tau}}\}_{\tau=2}^5$ and $\{com_{\Theta'_{l,j}}\}_{j \in \{1,2,5\}}$ to $\{R_{l,\tau}\}_{\tau=2}^5$ and $\{\Theta'_{l,j}\}_{j \in \{1,2,5\}}$ respectively.

- (a) To prove that identifier $I_{\phi_l} = \text{ID}(x_{k_l})$, $\mathcal{U}_i$ first computes $W_{\phi_l} = \prod_{\kappa=1, \kappa \neq \phi_l}^{\ell} g_{\ell+1-\kappa+\phi_l}^{I_{\kappa}}$ that satisfies the equality $e(g_{\phi_l}, C_{v_i}) = e(g_1, g_{\ell})^{I_{\phi_l}} \cdot e(g, W_{\phi_l})$. Then, $\mathcal{U}_i$ generates a Groth-Sahai commitment $com_{W_{\phi_l}}$ to W_{ϕ_l} . He/she computes a NIWI proof that committed variables $(R_{l,2}, R_{l,3}, C_{v_i}, W_{\phi_l})$ satisfy

$$e(R_{l,2}, C_{v_i}) = e(R_{l,3}, g_{\ell}) \cdot e(g, W_{\phi_l}). \quad (3.9)$$

We denote by π_{eq} the proof for the quadratic equation (3.9), which requires 9 group elements.

- (b) To prove that identifier $I_{\psi_l} \neq \text{ID}(x_{u_l})$, $\mathcal{U}_i$ computes $W_{\psi_l} = \prod_{\kappa=1, \kappa \neq \psi_l}^{\ell} g_{\ell+1-\kappa+\psi_l}^{I_{\kappa}}$ that satisfies the equality $e(g_{\psi_l}, C_{v_i}) = e(g_1, g_{\ell})^{I_{\psi_l}} \cdot e(g, W_{\psi_l})$. Then, he/she computes a Groth-Sahai commitment $com_{W_{\psi_l}}$ to W_{ψ_l} as well as commitments com_{Γ_l} and $\{com_{\Psi_{l,\tau}}\}_{\tau \in \{0,1,2\ell\}}$ to the group elements

$$(\Gamma_l, \Psi_{l,0}, \Psi_{l,1}, \Psi_{l,2\ell}) = (g^{1/(I_{\psi_l} - \text{ID}(x_{u_l}))}, g^{I_{\psi_l}}, g_1^{I_{\psi_l}}, g_{2\ell}^{I_{\psi_l}}).$$

Then, $\mathcal{U}_i$ provides evidence that committed variables $(R_{l,4}, R_{l,5}, C_{v_i}, \Gamma_l, \Psi_{l,0}, \Psi_{l,1}, \Psi_{l,2\ell})$ satisfy

$$e(R_{l,4}, C_{v_i}) = e(\Psi_{l,1}, g_{\ell}) \cdot e(g, W_{\psi_l}), \quad (3.10)$$

$$e(\Psi_{l,0}/R_{l,5}, \Gamma_l) = e(g, g), \quad (3.11)$$

$$e(\Psi_{l,1}, g) = e(g_1, \Psi_{l,0}), \quad (3.12)$$

$$e(\Psi_{l,2\ell}, g) = e(g_{2\ell}, \Psi_{l,0}). \quad (3.13)$$

We denote this NIWI proof by $\pi_{neq} = (\pi_{neq,1}, \pi_{neq,2}, \pi_{neq,3}, \pi_{neq,4})$. Since the first two equations (3.10) and (3.11) are quadratic, $\pi_{neq,1}$ and $\pi_{neq,2}$ consist of 9 elements each. The last two equations (3.12) and (3.13) are linear and both cost 3 elements to prove.

3. $\mathcal{U}_i$ provides evidence that the tuple R_l of (3.8) is a certified revocation message for period t : namely, he/she computes a NIWI proof

π_{R_l} that committed message elements $\{R_{l,\tau}\}_{\tau=2}^5$ and signature components $\{\Theta'_{l,j}\}_{j \in \{1,2,5\}}$ satisfy the equations

$$\begin{aligned}
& A^{(1)} \cdot e(\Theta'_{l,3}, \Theta'_{l,4})^{-1} \cdot e(G_1^{(1)}, g^t)^{-1} \\
&= e(G_z^{(1)}, \Theta'_{l,1}) \cdot e(G_r^{(1)}, \Theta'_{l,2}) \cdot \prod_{\tau=2}^5 e(G_\tau^{(1)}, R_{l,\tau}), \\
& B^{(1)} \cdot e(\Theta'_{l,6}, \Theta'_{l,7})^{-1} \cdot e(H_1^{(1)}, g^t)^{-1} \\
&= e(H_z^{(1)}, \Theta'_{l,1}) \cdot e(H_r^{(1)}, \Theta'_{l,5}) \cdot \prod_{\tau=2}^5 e(H_\tau^{(1)}, R_{l,\tau}).
\end{aligned} \tag{3.14}$$

Since $\{\Theta'_{l,j}\}_{j \in \{3,4,6,7\}}$ are constants, equations (3.14) are both linear and thus require 3 elements each. Hence, π_{R_l} takes 6 elements altogether.

4. Lets $\sigma_{v_i} = (\theta_{v_i,1}, \dots, \theta_{v_i,7})$ be the AHO signature on (X, C_{v_i}) . Re-randomizes σ_{v_i} as $\{\theta'_{v_i,j}\}_{j=1}^7 \leftarrow \text{ReRand}(pk_{\text{AHO}}^{(0)}, \sigma_{v_i})$ and generates commitments $\{com_{\theta'_{v_i,j}}\}_{j \in \{1,2,5\}}$ to $\{\theta'_{v_i,j}\}_{j \in \{1,2,5\}}$ as well as a commitment com_X to X . Then, generates a NIWI proof $\pi_{\sigma_{v_i}}$ that committed variables satisfy the verification equations

$$\begin{aligned}
A^{(0)} \cdot e(\theta'_{l,3}, \theta'_{l,4})^{-1} &= e(G_z^{(0)}, \theta'_{l,1}) \cdot e(G_r^{(0)}, \theta'_{l,2}) \cdot e(G_1^{(0)}, X) \cdot e(G_2^{(0)}, C_{v_i}), \\
B^{(0)} \cdot e(\theta'_{l,6}, \theta'_{l,7})^{-1} &= e(H_z^{(0)}, \theta_{l,1}) \cdot e(H_r^{(0)}, \theta'_{l,5}) \cdot e(H_1^{(0)}, X) \cdot e(H_2^{(0)}, C_{v_i}).
\end{aligned}$$

Since these equations are linear, $\pi_{\sigma_{v_i}}$ requires 6 group elements.

5. Using VK as a tag (by first hashing it onto $\mathbb{Z}_p$ in such a way that it can be interpreted as a $\mathbb{Z}_p$ element), computes a tag-based encryption [99] of X by drawing $z_1, z_2 \leftarrow \mathbb{Z}_p$ at random and setting $(\Upsilon_1, \Upsilon_2, \Upsilon_3, \Upsilon_4, \Upsilon_5) = (f_1^{z_1}, f_2^{z_2}, X \cdot g^{z_1+z_2}, (g^{\text{VK}} \cdot U)^{z_1}, (g^{\text{VK}} \cdot V)^{z_2})$.
6. Generates a NIZK proof that $com_X = (1, 1, X) \cdot \vec{f}_1^{w_{X,1}} \cdot \vec{f}_2^{w_{X,2}} \cdot \vec{f}_3^{w_{X,3}}$ and $(\Upsilon_1, \Upsilon_2, \Upsilon_3)$ are BBS encryptions of the same value X . If we write $\vec{f}_3 = (f_{3,1}, f_{3,2}, f_{3,3})$, the Groth-Sahai commitment com_X can be written as $(f_1^{w_{X,1}} \cdot f_{3,1}^{w_{X,3}}, f_2^{w_{X,2}} \cdot f_{3,2}^{w_{X,3}}, X \cdot g^{w_{X,1}+w_{X,2}} \cdot f_{3,3}^{w_{X,3}})$, so that we have

$$com_X \cdot (\Upsilon_1, \Upsilon_2, \Upsilon_3)^{-1} = (f_1^{\chi_1} \cdot f_{3,1}^{\chi_3}, f_2^{\chi_2} \cdot f_{3,2}^{\chi_3}, g^{\chi_1+\chi_2} \cdot f_{3,3}^{\chi_3}) \tag{3.15}$$

with $\chi_1 = w_{X,1} - z_1$, $\chi_2 = w_{X,2} - z_2$, $\chi_3 = w_{X,3}$. Computes $com_{\chi_j} = \vec{\varphi}^{\chi_j} \cdot \vec{f}_1^{w_{\chi_j,1}} \cdot \vec{f}_2^{w_{\chi_j,2}}$, with $w_{\chi_j,1}, w_{\chi_j,2} \xleftarrow{\$} \mathbb{Z}_p$ for $j \in \{1, 2, 3\}$, as commitments to $\{\chi_j\}_{j=1}^3$ and generates proofs $\{\pi_{eq-com,j}\}_{j=1}^3$ that χ_1, χ_2, χ_3 satisfy the three linear relations (3.15). These latter proofs $\{\pi_{eq-com,j}\}_{j=1}^3$ cost 2 elements each. See (2.29) for details.

7. Computes a weak Boneh-Boyen signature $\sigma_{\text{VK}} = g^{1/(x+\text{VK})}$ on VK and a commitment $com_{\sigma_{\text{VK}}}$ to σ_{VK} . Then, generate a NIWI proof $\pi_{\sigma_{\text{VK}}} = (\vec{\pi}_{\sigma_{\text{VK},1}}, \vec{\pi}_{\sigma_{\text{VK},2}}, \vec{\pi}_{\sigma_{\text{VK},3}}) \in \mathbb{G}^9$ that the committed variables $(\sigma_{\text{VK}}, X) \in \mathbb{G}^2$ satisfy the quadratic equation $e(\sigma_{\text{VK}}, X \cdot g^{\text{VK}}) = e(g, g)$.

8. Computes $\sigma_{ots} = \mathcal{S}(\text{SK}, (M, RL_t, \Upsilon_1, \Upsilon_2, \Upsilon_3, \Upsilon_4, \Upsilon_5, \Omega, \mathbf{com}, \mathbf{\Pi}))$ where $\Omega = \{\Theta'_{l,i}, \theta'_{l,i}\}_{i \in \{3,4,6,7\}}$ and

$$\begin{aligned} \mathbf{com} = & (com_{C_{v_i}}, com_X, \{com_{R_{l,\tau}}\}_{\tau=2}^5, com_{W_{\phi_l}}, com_{W_{\psi_l}}, \\ & com_{\Gamma_l}, \{com_{\Psi_{l,\tau}}\}_{\tau \in \{0,1,2\ell\}}, \{com_{\Theta'_{l,j}}\}_{j \in \{1,2,5\}}, \\ & \{com_{\theta'_{l,j}}\}_{j \in \{1,2,5\}}, \{com_{\chi_j}\}_{j=1}^3, com_{\sigma_{\text{VK}}}), \\ \mathbf{\Pi} = & (\pi_{eq}, \pi_{neq}, \pi_{R_l}, \pi_{\sigma_{v_i}}, \{\pi_{eq-com,j}\}_{j=1}^3, \pi_{\sigma_{\text{VK}}}). \end{aligned}$$

Returns the signature $\sigma = (\text{VK}, \Upsilon_1, \Upsilon_2, \Upsilon_3, \Upsilon_4, \Upsilon_5, \Omega, \mathbf{com}, \mathbf{\Pi}, \sigma_{ots})$.

Verify($\sigma, M, t, RL_t, \mathcal{Y}$): parses σ as above. If $\mathcal{V}(\text{VK}, (M, RL_t, \Upsilon_1, \Upsilon_2, \Upsilon_3, \Upsilon_4, \Upsilon_5, \Omega, \mathbf{com}, \mathbf{\Pi}), \sigma_{ots}) = 0$ or if $(\Upsilon_1, \Upsilon_2, \Upsilon_3, \Upsilon_4, \Upsilon_5)$ is not a well-formed tag-based encryption (that is, if $e(\Upsilon_1, g^{\text{VK}} \cdot U) \neq e(f_1, \Upsilon_4)$ or $e(\Upsilon_2, g^{\text{VK}} \cdot V) \neq e(f_2, \Upsilon_5)$), returns 0. Then, returns 1 if all proofs properly verify. Otherwise, returns 0.

Open($M, t, RL_t, \sigma, \mathcal{S}_{\text{OA}}, \mathcal{Y}, St$): parses σ as above and returns $\perp$ if **Verify**($\sigma, M, t, RL_t, \mathcal{Y}$) = 0. Otherwise, given $\mathcal{S}_{\text{OA}} = (\beta_1, \beta_2)$, computes $\tilde{X} = \Upsilon_3 \cdot \Upsilon_1^{-1/\beta_1} \cdot \Upsilon_2^{-1/\beta_2}$. In the database St_{trans} , finds a record $\langle i, \text{transcript}_i = (X_i, \text{ID}(v_i), C_{v_i}, \sigma_{v_i}, \text{sig}_i) \rangle$ such that $X_i = \tilde{X}$. If no such record exists in St_{trans} , returns $\perp$. Otherwise, returns i .

At first glance, the variable $\Psi_{l,2\ell}$ and the proof of the second equality (3.12) may seem unnecessary in step 2.b of the signing algorithm. However, this element plays a crucial role when it comes to proving the security under the ℓ -FlexDHE assumption. Indeed, the proof of security against misidentification attacks (more precisely, the proof of Lemma 4 in Section 3.3.3) ceases to go through if we remove $\Psi_{l,2\ell}$ and its corresponding proof.

As far as efficiency goes, each entry of RL_t contains 7 group elements and two node identifiers of $O(\log N)$ bits each. If $\lambda_{\mathbb{G}}$ is the bitlength of a group element, we have $\log N \ll \lambda_{\mathbb{G}}/2$ (since $\lambda \leq \lambda_{\mathbb{G}}$ and N is polynomial), so that the number of bits of RL_t is bounded by $2 \cdot |\mathcal{R}_t| \cdot (7 \cdot \lambda_{\mathbb{G}} + 2 \log N + 2 \log \log N) < 2 \cdot |\mathcal{R}_t| \cdot (9\lambda_{\mathbb{G}})$ bits. The size of RL_t is thus bounded by that of $18 \cdot |\mathcal{R}_t|$ group elements.

Unlike our first scalable construction [109] in the previous section, group members only need to store 9 group elements in their membership certificate. As far as the size of signature goes, $\mathbf{com}$ and $\mathbf{\Pi}$ require 66 and 60 group elements, respectively. If the one-time signature of [80] is used, VK and σ_{ots} consist of 3 elements of $\mathbb{G}$ and 2 elements of $\mathbb{Z}_p$, respectively. The global size σ amounts to that of 144 group elements, which is about 50% longer than [109]. In comparison with [81] (which does not natively support revocation), signatures are only longer by a factor of 3. At the 128-bit security level, each group element should have a 512-bit representation and a signature takes 9 kB.

Verifying signatures takes constant time. The signer has to compute at most $2\ell = O(\log N)$ exponentiations to obtain W_{ϕ_l} and W_{ψ_l} at the beginning of each

revocation period. Note that these exponentiations involve short exponents of $O(\log N)$ bits each. Hence, computing W_{ϕ_l} and W_{ψ_l} requires $O(\log^2 N)$ multiplications in $\mathbb{G}$. For this reason, since we always have $\log^2 N \ll \lambda$ (as long as $N \ll 2^{\lambda^{1/2}}$), this cost is dominated by that of a single exponentiation in $\mathbb{G}$.

3.3.3 Security Analysis

Security Against Misidentification

Theorem 9 (Misidentification) *The scheme is secure against misidentification attacks assuming that the q -SFP and the ℓ -FlexDHE problems are both hard for $q = \max(q_a, q_r^2)$ and $\ell = \log N$, where q_a and q_r denote the maximal numbers of $Q_{a\text{-join}}$ queries and Q_{revoke} queries, respectively, and N is the maximal number of group members.*

Proof. Towards a contradiction, let us assume that the adversary $\mathcal{A}$ outputs a non-trivial signature that does not open to an unrevoked adversarially-controlled group member.

Let $\sigma^* = (\text{VK}^*, \Upsilon_1^*, \Upsilon_2^*, \Upsilon_3^*, \Upsilon_4^*, \Upsilon_5^*, \Omega^*, \mathbf{com}^*, \Pi^*, \sigma_{ots}^*)$ denote $\mathcal{A}$'s forgery and parse $\mathbf{com}^*$ as

$$\begin{aligned} \mathbf{com}^* = & (com_{C_{v_i}}^*, com_X^*, \{com_{R_{l,\tau}}^*\}_{\tau=2}^5, com_{W_{\phi_l}}^*, \\ & com_{W_{\psi_l}}^*, com_{\Gamma_l}^*, \{com_{\Psi_{l,\tau}}^*\}_{\tau \in \{0,1,2\ell\}}, \{com_{\Theta'_{l,j}}^*\}_{j \in \{1,2,5\}}, \\ & \{com_{\theta'_{l,j}}^*\}_{j \in \{1,2,5\}}, \{com_{\chi_j}^*\}_{j=1}^3, com_{\sigma_{VK}}^*) \end{aligned}$$

We thus have $\text{Open}(M^*, t^*, RL_{t^*}, \sigma^*, \mathcal{S}_{OA}, \mathcal{Y}, St) \notin U^a \setminus \mathcal{R}_{t^*}$, where U^a denotes the set of adversarially-controlled users. Depending on extractable commitments com_X^* , $com_{C_{v_i}}^*$, $\{com_{R_{l,\tau}}^*\}_{\tau=2}^5$, $\{com_{\Psi_{l,\tau}}^*\}_{i \in \{0,1,2\ell\}}$, $com_{W_{\phi_l}}^*$, $com_{W_{\psi_l}}^*$, $com_{\Gamma_l}^*$, and their contents, we distinguish the following cases:

- **Type I forgeries** are those for which $\{com_{R_{l,\tau}}^*\}_{\tau=2}^5$ contains group elements $(R_{l,2}^*, \dots, R_{l,5}^*)$ such that $(g^{t^*}, R_{l,2}^*, \dots, R_{l,5}^*)$ was never signed when the latest revocation list RL_{t^*} was generated.
- **Type II forgeries** are such that $\{com_{R_{l,\tau}}^*\}_{\tau=2}^5$ contains group elements $(R_{l,2}^*, \dots, R_{l,5}^*)$ for which the message $(g^{t^*}, R_{l,2}^*, \dots, R_{l,5}^*)$ was signed when the latest revocation list RL_{t^*} was publicized at period t^* . At the same time, Open uncovers a user's tag X^* for which one of the following two situations occurs:
 - a. The pair $(X^*, C_{v_i}^*)$ was not signed using $sk_{\text{AHO}}^{(0)}$.
 - b. $(X^*, C_{v_i}^*)$ was signed when answering some $Q_{a\text{-join}}$ -query. However, $C_{v_i}^*$ encodes the path $(I_1^*, \dots, I_\ell^*)$ of a leaf v_i^* assigned to a revoked

user i^* even though the forgery σ^* provides convincing evidence that the committed values $C_{v_i}^*$, $(R_{l,2}^*, R_{l,3}^*, R_{l,4}^*, R_{l,5}^*)$, $(\Psi_{l,0}^*, \Psi_{l,1}^*, \Psi_{l,2\ell}^*)$ and $(\Gamma_l^*, W_{\phi_l}^*, W_{\psi_l}^*)$ satisfy the relations

$$e(R_{l,2}^*, C_{v_i}^*) = e(R_{l,3}^*, g_\ell) \cdot e(g, W_{\phi_l}^*), \quad (3.16)$$

and

$$e(R_{l,4}^*, C_{v_i}^*) = e(\Psi_{l,1}^*, g_\ell) \cdot e(g, W_{\psi_l}^*) \quad (3.17)$$

$$e(\Psi_{l,0}^*/R_{l,5}^*, \Gamma_l^*) = e(g, g) \quad (3.18)$$

$$e(\Psi_{l,1}^*, g) = e(g_1, \Psi_{l,0}^*) \quad (3.19)$$

$$e(\Psi_{l,2\ell}^*, g) = e(g_{2\ell}, \Psi_{l,0}^*). \quad (3.20)$$

It is immediate that Type I and Type II.a forgeries imply a forger against the AHO signature scheme and the proof is omitted.

Lemma 4 demonstrates that a Type II.b forgery necessarily contradicts the ℓ -FlexDHE assumption. This completes the proof since σ^* cannot constitute a successful misidentification attack without being a Type I or a Type II forgery. $\square$

Lemma 4 *The advantage of any Type II.b forger $\mathcal{A}$ is at most*

$$\mathbf{Adv}_{\mathcal{A}}^{\text{mis-id-II.b}}(\lambda) \leq \mathbf{Adv}^{\ell\text{-FlexDHE}}(\lambda)$$

where $\ell = \log N$ and N denotes the maximal number of users.

Proof. In the reduction the challenger $\mathcal{B}$ of ℓ -FlexDHE takes as input an instance $(g, g_1, \dots, g_\ell, g_{\ell+2}, \dots, g_{2\ell}) \in \mathbb{G}^{2\ell}$. To generate the group public key $\mathcal{Y}$ it follows exactly the specification of the **Setup** algorithm with the difference that, instead of computing ck as per step 3 of the algorithm, it defines $ck = (g_1, \dots, g_\ell, g_{\ell+2}, \dots, g_{2\ell}) \in \mathbb{G}^{2\ell-1}$ using its input and gives $\mathcal{Y} := (g, pk_{\text{AHO}}^{(0)}, pk_{\text{AHO}}^{(1)}, ck, \mathbf{f}, \tilde{\varphi}, (U, V), \Sigma)$ to the Type II.b forger $\mathcal{A}$.

Throughout the game, the adversary can adaptively invoke the Q_{pub} , $Q_{\text{a-join}}$, Q_{revoke} , Q_{read} , and Q_{keyOA} oracles. Since $\mathcal{B}$ knows $\mathcal{S}_{\text{GM}} = (sk_{\text{AHO}}^{(0)}, sk_{\text{AHO}}^{(1)})$ and $\mathcal{S}_{\text{OA}} = (\beta_1, \beta_2)$, it can faithfully answer all adversarial queries. The game ends with the adversary outputting a forgery σ^* for which the committed variables $C_{v_i}^*$, $(R_{l,2}^*, R_{l,3}^*, R_{l,4}^*, R_{l,5}^*)$, $(\Psi_{l,0}^*, \Psi_{l,1}^*, \Psi_{l,2\ell}^*)$ and $(\Gamma_l^*, W_{\phi_l}^*, W_{\psi_l}^*)$ satisfy relations (3.16)-(3.20) although σ^* opens to some user $i^* \in U^a \cap \mathcal{R}_{t^*}$.

Note that $(R_{l,1}^*, R_{l,2}^*, R_{l,3}^*, R_{l,4}^*, R_{l,5}^*)$ must be of the form

$$(R_{l,1}^*, R_{l,2}^*, R_{l,3}^*, R_{l,4}^*, R_{l,5}^*) = (g^{t^*}, g_{\phi_l}, g_1^{\text{ID}(x_{k_l}^*)}, g_{\psi_l}, g^{\text{ID}(x_{k_l}^*)}), \quad (3.21)$$

for some $\phi_l, \psi_l \in \{1, \dots, \ell\}$ and some $\text{ID}(x_{k_l}^*), \text{ID}(x_{u_l}^*) \in \{1, \dots, 2N-1\}$ that $\mathcal{B}$ knows for having chosen them himself/herself at the latest Q_{revoke} -query. By hypothesis, σ^* contains a committed pair $(X^*, C_{v_i}^*)$ that was signed by $\mathcal{B}$ at some $Q_{\text{a-join}}$ -query. Then, $\mathcal{B}$ recalls $(I_1^*, \dots, I_\ell^*)$ such that $C_{v_i}^* = \prod_{\kappa=1}^\ell g_{\ell+1-\kappa}^{I_\kappa^*}$ from his/her interaction with $\mathcal{A}$ at that $Q_{\text{a-join}}$ -query. Since $i^* \in U^a \cap \mathcal{R}_{t^*}$, it must hold that either:

- $I_{\phi_l}^* \neq \text{ID}(x_{k_l}^*)$: In this case, relations (3.21) and (3.16) imply that

$$e(g_{\phi_l}, C_{v_i}^*) = e(g_1, g_\ell)^{\text{ID}(x_{k_l}^*)} \cdot e(g, W_{\phi_l}^*) \quad (3.22)$$

for values $\phi_l \in \{1, \dots, \ell\}$ and $\text{ID}(x_{k_l}^*) \in \{1, \dots, 2N-1\}$ that are available to $\mathcal{B}$. Since he/she also knows $(I_1^*, \dots, I_\ell^*)$ such that $C_{v_i}^* = \prod_{\kappa=1}^\ell g_{\ell+1-\kappa}^{I_\kappa^*}$, he/she can compute $W' = \prod_{\kappa=1, \kappa \neq \phi_l}^\ell g_{\ell+1-\kappa+\phi_l}^{I_\kappa^*}$ which satisfies

$$e(g_{\phi_l}, C_{v_i}^*) = e(g_1, g_\ell)^{I_{\phi_l}^*} \cdot e(g, W'). \quad (3.23)$$

By combining (3.22) and (3.23), we find $g_{\ell+1} = (W_{\phi_l}^*/W')^{1/(I_{\phi_l}^* - \text{ID}(x_{k_l}^*))}$ is computable by $\mathcal{B}$ and thus he/she solves an instance of the ℓ -DHE problem (which is not easier than ℓ -FlexDHE).

- $I_{\psi_l}^* = \text{ID}(x_{u_l}^*)$: In this situation, if we define $\varrho = \log_{g_1}(\Psi_{l,1}^*)$, relations (3.21) and (3.17)-(3.20) imply that

$$e(g_{\psi_l}, C_{v_i}^*) = e(g_1, g_\ell)^\varrho \cdot e(g, W_{\psi_l}^*) \quad (3.24)$$

$$g^{\varrho - I_{\psi_l}^*} \neq 1_{\mathbb{G}} \quad (3.25)$$

$$\Psi_{l,0}^* = g^\varrho \quad (3.26)$$

$$\Psi_{l,2\ell}^* = g_{2\ell}^\varrho \quad (3.27)$$

Also, similarly to the previous case, the challenger $\mathcal{B}$ can compute $W' = \prod_{\kappa=1, \kappa \neq \psi_l}^\ell g_{\ell+1-\kappa+\psi_l}^{I_\kappa^*}$ such that

$$e(g_{\psi_l}, C_{v_i}^*) = e(g_1, g_\ell)^{I_{\psi_l}^*} \cdot e(g, W'). \quad (3.28)$$

If we divide (3.24) by (3.28), we obtain the equality $e(g_1, g_\ell)^{\varrho - I_{\psi_l}^*} = e(g, W'/W_{\psi_l}^*)$, so that $W'/W_{\psi_l}^* = g_{\ell+1}^{\varrho - I_{\psi_l}^*}$. The triple

$$\left(\Psi_{l,0}^* \cdot g^{-I_{\psi_l}^*}, W'/W_{\psi_l}^*, \Psi_{l,2\ell}^* \cdot g_{2\ell}^{-I_{\psi_l}^*} \right) = \left(g^{\varrho - I_{\psi_l}^*}, g_{\ell+1}^{\varrho - I_{\psi_l}^*}, g_{2\ell}^{\varrho - I_{\psi_l}^*} \right)$$

thus forms a non-trivial solution to the ℓ -FlexDHE problem.

In either case, we observe that $\mathcal{B}$ solves either the given ℓ -FlexDHE instance or the potentially harder ℓ -DHE problem. $\square$

Security Against Framing Attacks

Theorem 10 (Non-frameability) *The scheme is secure against framing attacks assuming that: (i) the q_b -SDH assumption holds in $\mathbb{G}$, where q_b is the maximal number of $Q_{b\text{-join}}$ -queries; (ii) Σ is a strongly unforgeable one-time signature.*

Proof. As in [81], we consider two kinds of framing attacks that can be possibly mounted by a non-frameability adversary $\mathcal{A}$.

- **Type I attacks:** the adversary $\mathcal{A}$ generates a forgery $\sigma^* = (\text{VK}^*, \Upsilon_1^*, \Upsilon_2^*, \Upsilon_3^*, \Upsilon_4^*, \Upsilon_5^*, \Omega^*, \text{com}^*, \Pi^*, \sigma_{ots}^*)$ for which the one-time verification key VK^* was used by some honest group member $i \in U^b$ when answering a Q_{sig} -query.
- **Type II attacks:** $\mathcal{A}$ outputs a forgery $\sigma^* = (\text{VK}^*, \Upsilon_1^*, \Upsilon_2^*, \Upsilon_3^*, \Upsilon_4^*, \Upsilon_5^*, \Omega^*, \text{com}^*, \Pi^*, \sigma_{ots}^*)$ for which the one-time verification key VK^* was never used by Q_{sig} to answer a signing query on behalf of an honest user $i \in U^b$.

Type I attacks clearly defeat the security of the one-time signature. Lemma 5 shows that a Type II forgery would contradict the Strong Diffie-Hellman assumption. $\square$

Lemma 5 *The scheme is secure against framing attacks of Type II if the q_s -SDH problem is hard. More precisely, the advantage of any adversary after q_s Q_{sig} -queries and q_b $Q_{\text{b-join}}$ -queries is at most $\text{Adv}^{\text{fra-II}}(\lambda) \leq q_b \cdot \text{Adv}^{q_s\text{-SDH}}(\lambda)$.*

Proof. Let us assume that a PPT adversary $\mathcal{A}$ comes up with a forgery (M^*, σ^*) that opens to some honest user $i \in U^b$ who did not issue a signature containing the verification key VK^* . The same proof as in [81] shows that the Strong Diffie-Hellman assumption can be broken.

Given a q -SDH instance $(\tilde{g}, \tilde{g}^a, \dots, \tilde{g}^{(a^{q_s})}) \in \mathbb{G}^{q_s+1}$, the reduction $\mathcal{B}$ generates a set of q_s one-time signature keys pairs $(\text{SK}_i, \text{VK}_i) \leftarrow \mathcal{G}(\lambda)$ for $i = 1$ to q_s . Then, using the Boneh-Boyen techniques (see [21][Lemma 3.2]) it builds a generator g and a randomly distributed public value $X^\dagger = g^a$ – which implicitly defines $x^\dagger = \log_g(X^\dagger) = a$ – such that it knows $\{(g^{1/(a+\text{VK}_i)}, \text{VK}_i)\}_{i=1}^{q_s}$.

Next, using the newly generated g , $\mathcal{B}$ generates key pairs $\{(sk_{\text{AHO}}^{(b)}, pk_{\text{AHO}}^{(b)})\}_{b=0,1}$ for the AHO signature (note that group elements of $\{pk_{\text{AHO}}^{(b)}\}_{b=0,1}$ are computed as powers of g) and uses $pk_{\text{AHO}}^{(0)}, pk_{\text{AHO}}^{(1)}$ to form the group public key

$$\mathcal{Y} := (g, pk_{\text{AHO}}^{(0)}, pk_{\text{AHO}}^{(1)}, ck, \mathbf{f}, \vec{\varphi}, (U, V), \Sigma).$$

The underlying Groth-Sahai CRS $\mathbf{f} = (\vec{f}_1, \vec{f}_2, \vec{f}_3)$ is generated for the perfect soundness setting, *i.e.*, with $\vec{f}_1 = (f_1 = g^{\beta_1}, 1, g)$, $\vec{f}_2 = (1, f_2 = g^{\beta_2}, g)$ and $\vec{f}_3 = \vec{f}_1^{\xi_1} \cdot \vec{f}_2^{\xi_2}$, where $\xi_1, \xi_2 \xleftarrow{\$} \mathbb{Z}_p^*$.

If the adversary $\mathcal{A}$ decides to corrupt the group manager or the opening authority during the game, $\mathcal{B}$ can reveal $\mathcal{S}_{\text{GM}} = (sk_{\text{AHO}}^{(0)}, sk_{\text{AHO}}^{(1)})$ and $\mathcal{S}_{\text{OA}} = (\beta_1, \beta_2) = (\log_g(f_1), \log_g(f_2))$. At the outset of the game, $\mathcal{B}$ picks a random $j^* \xleftarrow{\$} \{1, \dots, q_b\}$ and interacts with the Type II forger $\mathcal{A}$ as follows.

- Q_{keyGM} -queries: if $\mathcal{A}$ decides to corrupt the group manager, $\mathcal{B}$ surrenders $\mathcal{S}_{\text{GM}} = (sk_{\text{AHO}}^{(0)}, sk_{\text{AHO}}^{(1)})$.
- $Q_{\text{b-join}}$ -queries: when $\mathcal{A}$, acting as a corrupted group manager, decides to introduce a new honest user i in the group, $\mathcal{B}$ starts interacting with $\mathcal{A}$

in an execution of **Join** and runs J_{user} on behalf of the honest user. The actions taken by $\mathcal{B}$ depend on the index $j \in \{1, \dots, q_b\}$ of the $Q_{\text{b-join}}$ -query.

- If $j \neq j^*$, $\mathcal{B}$ follows exactly the specification of J_{user} .
- If $j = j^*$, $\mathcal{B}$ sends the value $X^\dagger$ to J_{GM} at step 1 of **Join**. User j^* 's membership secret is implicitly defined to be the unknown exponent $\text{sec}_{j^*} = a$ of the q -SDH instance. In steps 2-5 of the join protocol, $\mathcal{B}$ proceeds like the real J_{user} algorithm. When **Join** terminates, $\mathcal{B}$ obtains a membership certificate $\text{cert}_{j^*} = (\text{ID}(v^*), X^\dagger, C_{v^*}, \sigma_{v^*})$.
- Q_{pub} -queries: can be treated as in the real game, by having the simulator return $\mathcal{Y}$.
- Q_{sig} -queries: when the adversary $\mathcal{A}$ asks user $i \in U^b$ to sign a message M , $\mathcal{B}$ can answer the query by running the real signature generation algorithm if $i \neq j^*$. Otherwise (namely, if $i = j^*$), $\mathcal{B}$ uses the next available pair $\{(g^{1/(a+\text{VK}_i)}, \text{VK}_i)\}_{i=1}^{q_s}$ to define $\sigma_{\text{VK}_i} = g^{1/(a+\text{VK}_i)}$. It also recalls user j^* 's membership certificate $\text{cert}_{j^*} = (\text{ID}(v^*), X^\dagger, C_{v^*}, \sigma_{v^*})$ that it obtained from the J_{GM} -executing adversary at the j^* -th $Q_{\text{b-join}}$ -query. Using σ_{VK_i} and cert_{j^*} , it can easily generate all signature components and sign them using the one-time private key SK_i .

Finally, $\mathcal{A}$ outputs a signature $\sigma^* = (\text{VK}^*, \Upsilon_1^*, \Upsilon_2^*, \Upsilon_3^*, \Upsilon_4^*, \Upsilon_5^*, \Omega^*, \text{com}^*, \Pi^*, \sigma_{\text{ots}}^*)$, for some message M^* , that opens to some user $i^* \in U^b$ who did not sign M^* . At this point, $\mathcal{B}$ halts and reports failure if it turns out that $X^\dagger \neq \Upsilon_3^* \cdot \Upsilon_1^{*-1/\beta_1} \cdot \Upsilon_2^{*-1/\beta_2}$ since, in this case, it was unfortunate when drawing the random index j^* . Still, with probability $1/q_b$, the signature σ^* opens to the user introduced at the j^* -th $Q_{\text{b-join}}$ -query and $(\Upsilon_1^*, \Upsilon_2^*, \Upsilon_3^*)$ does decrypt to X^* . In this situation, the perfect soundness of the proof system ensures that $\text{com}_{\sigma_{\text{VK}^*}}^*$ is a commitment to a group element $\sigma_{\text{VK}^*}^*$ such that $e(\sigma_{\text{VK}^*}^*, X^\dagger \cdot g^{\text{VK}^*}) = e(g, g)$. Since σ^* is a Type II forgery, $\mathcal{B}$ can use β_1, β_2 to compute a BBS decryption of $\text{com}_{\sigma_{\text{VK}^*}}^*$ and obtain a pair of the form $(\sigma_{\text{VK}^*}, \text{VK}^*) = (g^{1/(x+\text{VK}^*)}, \text{VK}^*)$. The latter eventually yields a solution $(\tilde{g}^{1/(x+\text{VK}^*)}, \text{VK}^*)$ to the initial q_s -SDH instance by performing a Euclidean division in the exponent as in [21]. $\square$

Anonymity

As for the anonymity property, it naturally relies on the DLIN assumption.

Theorem 11 (Anonymity) *The advantage of any anonymity adversary is at most*

$$\text{Adv}^{\text{anon}}(\mathcal{A}) \leq \text{Adv}^{\text{ots}}(\lambda) + 3 \cdot \text{Adv}^{\text{DLIN}}(\lambda),$$

where the first term is $\mathcal{A}$'s probability of breaking the strong unforgeability of the one-time signature.

The proof is identical to that of Theorem 8.

3.3.4 Efficiency of this second Scheme

Our technique allows making the most of our first approach of section 3.2 by reducing the size of membership certificates to a small constant: at the cost of lengthening signatures by a factor of only 1.5, we obtain membership certificates consisting of only 9 group elements and a small integer. For $N = 1000$, users' private keys are thus compressed by a multiplicative factor of several hundreds and this can only become more dramatic for larger groups. At the same time, the scheme here retains all the useful properties of the previous scheme: like the construction of Nakanishi *et al.* [117], it does not require users to update their membership certificates at any time but, unlike [117], our group public key size is $O(\log N)$. Our system still uses revocation lists of size $O(r)$, which is on par with Certificate Revocation Lists (CRLs) of standard PKIs. It is worth noting that RLs are *not* part of the group public key: verifiers only need to know the number of the latest revocation period and they should not bother to read RLs entirely.

Eventually, our novel approach yields revocable group signatures that become competitive with the regular CRL approach in PKIs: signature generation and verification have constant cost, signatures and membership certificates being of $O(1)$ -size while revocation lists have size $O(r)$.

3.4 Efficiency Comparison and Comments

This section compares pairing-based revocable group signatures with constant-size signatures where group members are stateless and do not update their membership certificate whenever a revocation occurs. Comparisons are given in terms of computational costs and the size (measured by the number of group elements) of public keys, membership certificates and revocation lists as functions of N , r and, in some cases, the number T of revocation epochs. By “constant”, we thus mean that the complexity only depends on the security parameter λ . The four first schemes rely on the random oracle model while the four last schemes are in the standart model.

Table 3.1: Comparison between pairing-based revocable group signatures

Schemes	Group public key size	Membership certificate size	Revocation list size	Signature cost	Verification cost	Revocation cost
NFHF1 [117]	$O(N)$	$O(1)$	$O(r)$	$O(1)$	$O(1)$	$O(r)$
NFHF2 [117]	$O(N^{1/2})$	$O(1)$	$O(r)$	$O(1)$	$O(1)$	$O(r)$
BS [28]	$O(1)$	$O(1)$	$O(r)$	$O(1)$	$O(r)$	$O(1)$
NF [118]	$O(T)^\diamond$	$O(1)$	$O(r)$	$O(1)$	$O(r)$	$O(r)$
LV [110]	$O(T)^\diamond$	$O(1)$	$O(r)$	$O(1)$	$O(r)$	$O(r)$
Thesis I	$O(\log N)$	$O(\log^3 N)$	$O(r)$	$O(\log N)^\dagger$	$O(1)$	$O(r \cdot \log N)$
LPY2 (CS)	$O(1)$	$O(\log N)$	$O(r \cdot \log(N/r))$	$O(1)$	$O(1)$	$O(r \cdot \log(N/r))$
Thesis II	$O(\log N)$	$O(1)$	$O(r)$	$O(1)$	$O(1)$	$O(r)$

N : max. number of users; r : number of revocations T : max. number of revocation periods

$^\diamond$ These schemes can be modified to have $O(1)$ -size group public keys.

† This complexity is only involved at the first signature of each revocation epoch.

As previously mentioned, among schemes where revocations require no update in unrevoked users' credentials, our last method (Thesis II in the table) seems asymptotically optimal. The only dependency on N appears in the group public key size, which is logarithmic and thus quite moderate. At the same time, it retains revocation lists of size $O(r)$ as in our first scheme (Thesis I in the table). In comparison with the latter, we also eliminate the $O(\log N)$ multiplicative factor in the revocation cost and the complexity of the signing algorithm in the worst case.

The joining protocol is also much more efficient in our scheme based on concise vector commitments as the group manager has to generate only one structure-preserving signature (computing C_v in step 2.a of the protocol is actually cheaper than a single exponentiation in $\mathbb{G}$), instead of $\log(N)$ in our HIBE-based scheme.

In the full version of our Crypto paper [108], we gave tradeoffs between the strength of the assumption and the efficiency: in these alternative constructions, the assumption is weakened at the expense of group public keys of size $O(\log^2 N)$ or $O(\lambda + \log^2 N)$. The table also contains an entry for our second scheme LPY (CS) described in the full version of [109] where the complete subtree method is used for the NNL framework.

Chapter 4

P-Homomorphic Signatures

Contents

4.1	Definition and Security Notions	104
4.1.1	Unified Framework	105
4.1.2	Variations on Context-Hiding	106
4.1.3	Separation Results	108
4.2	Subset Predicate Signatures	110
4.2.1	Intuition behind our Subset Predicate Signatures	111
4.2.2	Construction with Short Keys for Subset Predicates	112
4.2.3	Security of our Subset Predicate Signatures	115
4.3	Quotable Signatures	117
4.3.1	Intuition behind our Substring Predicate Signatures	117
4.3.2	Construction of Linear-Size Quotable Signatures	118
4.3.3	Security of our Substring Predicate Signatures	121
4.4	Linearly Homomorphic Signatures	123
4.4.1	Intuition behind our Subspace Predicate Signatures	124
4.4.2	A One-Time Linearly Homomorphic Construction	127
4.4.3	A Full-Fledged Construction for Linear Subspaces	129
4.4.4	A Completely Context-Hiding Construction	135

With the advent of fully homomorphic encryption [77], much attention has been paid to the problem of computing on encrypted data in the recent years. This also revived the interest of the research community in homomorphic signatures which allow computations on authenticated data.

Informally, a signer has a set of messages $\{m_i\}_{i=1}^k$ and generates a corresponding set of signatures $\{\sigma_i\}_{i=1}^k$ with $\sigma_i = \text{Sign}(\text{sk}, m_i)$ for each i . The signed dataset $\{(m_i, \sigma_i)\}_{i=1}^k$ is then archived on a remote server. Later on, the server can publicly compute $(m, \sigma) = \text{Evaluate}(\text{pk}, \{(m_i, \sigma_i)\}_{i=1}^k, f)$ such that $\text{Verify}(\text{pk}, m, \sigma) = 1$, where $m = f(m_1, \dots, m_k)$ for some function f .

In the last decade, the area was investigated by several lines of research: examples include homomorphic signatures for arithmetic functions but also redactable signatures and various other forms of algebraic signatures. Each type of homomorphic signatures was studied independently in their own security model.

Recently, Ahn *et al.* [6] defined a framework for computing on signed data. For a predicate P , their notion of P -homomorphic signature allows anyone who observes signatures on a message m to publicly derive signatures on messages m' such that $P(m, m') = 1$. This framework is geared towards capturing homomorphic signatures supporting quoting and redacting, arithmetic functions and more. Ahn *et al.* gave thorough definitions for the unforgeability of P -homomorphic signatures. Besides, they introduced a strong notion of privacy, called *strong context hiding*, that captures the infeasibility of linking a derived signature to the signature it was derived from. A scheme is said strongly context-hiding when a derived signature is statistically indistinguishable from a freshly generated signature, *even* when the original signature is available.

In this chapter, we first reconsider the definition of strong context-hiding security and point out a subtlety that arises in the context of randomizable signatures. While the definition of Ahn *et al.* aims at perfect indistinguishability, it only considers honestly generated original signatures. In specific schemes, signatures may satisfy the verification algorithm without being produced by the legitimate signing algorithm. This may be a concern in certain applications. To address this issue, we suggest other definitions of unlinkability and discuss the relations among them. In order to unify these definitions, we thus define a notion of *completely context-hiding* homomorphic signature, which requires statistical unlinkability and implies other context-hiding properties.

Then, in the next sections, we propose three schemes for P -homomorphic signatures with different predicates that all achieve the strongest definition of context-hiding notions. Our first construction handles P -homomorphic signatures for subset predicates with constant-size public keys. Afterwards, we put forward a homomorphic signature for quoting substring made of linear-size (in the number of symbols composing the string). Finally, we provide a solution for linearly-homomorphic signatures. All the P -homomorphic pairing-based signature schemes are structure-preserving in the sense that messages, signature components and public keys live in the group $\mathbb{G}$, as in ordinary signature schemes. Their security is proven in the standard model.

4.1 Definition and Security Notions

All the homomorphic signature schemes *i.e.* schemes that generate signatures supporting a specified kind of public malleability, were encompassed in a common framework by Ahn *et al.* in 2012. Here, we recall this unified framework before proposing some variants of the initial context-hiding (CH) properties. Beyond their practical interest, we will show that all these unlinkability features

are provably distinct.

4.1.1 Unified Framework

Authorized modifications on signatures are captured by predicates. Hence the name of *P-homomorphic signatures*.

Definition 26 ([6]) Let $\mathcal{M}$ be a message space and $2^{\mathcal{M}}$ be its powerset. Let $P : 2^{\mathcal{M}} \times \mathcal{M} \rightarrow \{0, 1\}$ be a predicate. A message m' is said **derivable** from $M \subset \mathcal{M}$ if $P(M, m') = 1$. As in [6], $P^i(M)$ is the set of messages derivable from $P^{i-1}(M)$, where $P^0(M) := \{m' \in \mathcal{M} \mid P(M, m') = 1\}$. Finally, $P^*(M) := \bigcup_{i=0}^{\infty} P^i(M)$ denotes the set of messages derivable from M by iterated derivation.

Definition 27 ([6]) A *P-homomorphic signature* for a predicate $P : 2^{\mathcal{M}} \times \mathcal{M} \rightarrow \{0, 1\}$ is a triple of algorithms (**Keygen**, **SignDerive**, **Verify**) such that:

Keygen(λ): takes as input a security parameter $\lambda \in \mathbb{N}$ and outputs a key pair (sk, pk) . As in [6], the private key sk is seen as a signature on the empty tuple $\varepsilon \in \mathcal{M}$.

SignDerive($\text{pk}, (\{\sigma_m\}_{m \in M}, M), m'$): is a possibly randomized algorithm that takes as input a public key pk , a set of messages $M \subset \mathcal{M}$, a corresponding set of signatures $\{\sigma_m\}_{m \in M}$ and a derived message $m' \in \mathcal{M}$. If $P(M, m') = 0$, it returns $\perp$. Otherwise, it outputs a derived signature σ'

Verify(pk, σ, m): is a deterministic algorithm that takes as input a public key pk , a signature σ and a message m . It outputs 0 or 1.

Note that the empty tuple $\varepsilon \in \mathcal{M}$ satisfies $P(\varepsilon, m) = 1$ for each $m \in \mathcal{M}$. Like [6], we define the algorithm **Sign**(pk, sk, m) that runs **SignDerive**($\text{pk}, (\text{sk}, \varepsilon), m$) and returns the resulting output. For any set $M = \{m_1, \dots, m_k\} \subset \mathcal{M}$, we define $\text{Sign}(\text{sk}, M) := \{\text{Sign}(\text{sk}, m_1), \dots, \text{Sign}(\text{sk}, m_k)\}$. In the same manner, $\text{Verify}(\text{pk}, M, \{\sigma_m\}_{m \in M}) = 1$ means that $\text{Verify}(\text{pk}, m, \sigma_m) = 1$ for each $m \in M$.

CORRECTNESS. It is mandated that, for all pairs $(\text{pk}, \text{sk}) \leftarrow \text{Keygen}(\lambda)$, for any set $M \subset \mathcal{M}$, any message $m' \in \mathcal{M}$ such that $P(M, m') = 1$, then, we have

- $\text{SignDerive}(\text{pk}, (\text{Sign}(\text{sk}, M), M), m') \neq \perp$.
- $\text{Verify}(\text{pk}, m', \text{SignDerive}(\text{pk}, (\text{Sign}(\text{sk}, M), M), m')) = 1$.

Definition 28 ([6]) A *P-homomorphic signature* (**Keygen**, **SignDerive**, **Verify**) is **unforgeable** if no probabilistic polynomial-time (PPT) adversary has non-negligible advantage in the following game:

1. The challenger $\mathcal{B}$ generates $(\text{pk}, \text{sk}) \leftarrow \text{Keygen}(\lambda)$ and gives pk to the adversary $\mathcal{A}$. $\mathcal{B}$ initializes two initially empty tables T and Q .

2. $\mathcal{A}$ adaptively interleaves the following queries.

- *Signing queries:* $\mathcal{A}$ chooses a message $m \in \mathcal{M}$. The challenger replies by choosing a handle h , runs $\sigma \leftarrow \text{Sign}(sk, m)$ and stores (h, m, σ) in a table T . The handle h is returned to $\mathcal{A}$.
- *Derivation queries:* $\mathcal{A}$ chooses a vector of handles $\vec{h} = (h_1, \dots, h_k)$ and a message $m' \in \mathcal{M}$. The challenger $\mathcal{B}$ retrieves the tuples $\{(h_i, m_i, \sigma_i)\}_{i=1}^k$ from T and returns $\perp$ if one of these does not exist. Otherwise, $\mathcal{B}$ defines $M := (m_1, \dots, m_k)$ and signature set $\{\sigma_m\}_{m \in M} = \{\sigma_1, \dots, \sigma_k\}$. If $P(M, m') = 1$, the challenger runs $\sigma' \leftarrow \text{SignDerive}(\text{pk}, (\{\sigma_m\}_{m \in M}, M), m')$, chooses a handle h' , stores (h', m', σ') in T and returns h' to $\mathcal{A}$.
- *Reveal queries:* $\mathcal{A}$ chooses a handle h . If no tuple of the form (h, m', σ') exists in T , the challenger returns $\perp$. Otherwise, it returns σ' to $\mathcal{A}$ and adds (m', σ') to the set Q .

3. $\mathcal{A}$ outputs a pair (σ', m') and wins if the following conditions hold.

- $\text{Verify}(\text{pk}, m', \sigma') = 1$.
- If $M \subset \mathcal{M}$ is the set of messages in Q , then $m' \notin P^*(M)$.

Definition 29 ([6]) A homomorphic signature $(\text{Keygen}, \text{Sign}, \text{SignDerive}, \text{Verify})$ is **strongly context-hiding** for the predicate P if, for all key pairs $(\text{pk}, \text{sk}) \leftarrow \text{Keygen}(\lambda)$, for all messages $M \subset \mathcal{M}^*$ and $m' \in \mathcal{M}$ such that $P(M, m') = 1$, the following two distributions are statistically close:

$$\begin{aligned} & \{(\text{sk}, \{\sigma_m\}_{m \in M} \leftarrow \text{Sign}(\text{sk}, M), \text{Sign}(\text{sk}, m'))\}_{\text{sk}, M, m'}, \\ & \{(\text{sk}, \{\sigma_m\}_{m \in M} \leftarrow \text{Sign}(\text{sk}, M), \text{SignDerive}(\text{pk}, (\{\sigma_m\}_{m \in M}, M), m'))\}_{\text{sk}, M, m'}. \end{aligned}$$

In [6] Ahn *et al.* showed that, if a scheme is strongly context-hiding, then Definition 28 can be simplified by removing the **SignDerive** and **Reveal** oracles and only providing the adversary with an ordinary signing oracle.

4.1.2 Variations on Context-Hiding

As we will see, specific constructions leave a gap between signatures accepted by the verification algorithm and those generated by the original signing procedure. For these schemes, a stronger definition than Definition 29 may be necessary in some situations.

To illustrate this, we first give an alternative definition which is almost identical to the computational security definition of [6][Appendix A]: the only difference is that, in the challenge phase, one of the signatures is supplied by the adversary instead of being honestly generated by the challenger. This modification is motivated by re-randomizable signatures. It allows adversaries who attempt to re-randomize one of the signatures obtained from the oracle in order to embed some subliminal information that would help them win the game.

Definition 30 A P -homomorphic signature $(\text{Keygen}, \text{Sign}, \text{SignDerive}, \text{Verify})$ is **weakly adaptively context-hiding** if no PPT adversary has non-negligible advantage in the following game:

1. The challenger $\mathcal{B}$ runs $(\text{sk}, \text{pk}) \leftarrow \text{Keygen}(\lambda)$ and gives the public-key pk to the adversary $\mathcal{A}$.
2. The adversary $\mathcal{A}$ adaptively interleaves queries exactly as in Definition 28.
3. The adversary $\mathcal{A}$ chooses a message set $M \subset \mathcal{M}$ together with a set of signatures $\{\sigma_m\}_{m \in M}$ as well as another message $m' \in \mathcal{M}$. If $P(M, m') = 0$ or $\text{Verify}(\text{pk}, M, \{\sigma_m\}_{m \in M}) = 0$, $\mathcal{B}$ returns $\perp$. Otherwise, the challenger flips a fair binary coin $\beta \xleftarrow{\$} \{0, 1\}$. If $\beta = 0$, $\mathcal{B}$ computes a derived signature $\sigma^* = \text{SignDerive}(\text{pk}, (\{\sigma_m\}_{m \in M}, M), m')$. If $\beta = 1$, $\mathcal{B}$ computes $\sigma^* = \text{Sign}(\text{sk}, m')$. In either case, σ^* is sent as a challenge to $\mathcal{A}$.
4. $\mathcal{A}$ is allowed to make another series of queries as in stage 2.
5. Eventually, $\mathcal{A}$ outputs a bit $\beta' \in \{0, 1\}$ and wins if $\beta' = \beta$. As usual, $\mathcal{A}$'s advantage is defined to be $\text{Adv}(\mathcal{A}) = |\Pr[\beta' = \beta] - 1/2|$.

The latter definition can be seen as an analogue of a definition of unlinkability given by Prabhakaran and Rosulek [125] for homomorphic encryption: both models account for adversarially-chosen original signatures or ciphertexts.

We will see that Definitions 29 and 30 do not imply each other. While incomparable, we believe that they both make sense in practice. For example, when it comes to concealing the path followed by packets in network coding signatures, Definition 30 ensures that each node only learns the last node visited by incoming packets, even if it colludes with another node far upstream.

Towards unifying previous definitions, we now simplify Definition 30 as follows. Instead of providing the adversary $\mathcal{A}$ with a signing oracle, $\mathcal{A}$ is directly given the private key at the beginning.

Definition 31 A P -homomorphic signature $(\text{Keygen}, \text{Sign}, \text{SignDerive}, \text{Verify})$ is **adaptively context-hiding** if no PPT adversary has non-negligible advantage in the following game:

1. The challenger $\mathcal{B}$ runs $(\text{sk}, \text{pk}) \leftarrow \text{Keygen}(\lambda)$ and hands (sk, pk) to $\mathcal{A}$.
2. The adversary $\mathcal{A}$ chooses a message set $M \subset \mathcal{M}$ together with a set of signatures $\{\sigma_m\}_{m \in M}$ as well as another message $m' \in \mathcal{M}$. If $P(M, m') = 0$ or $\text{Verify}(\text{pk}, M, \{\sigma_m\}_{m \in M}) = 0$, $\mathcal{B}$ returns $\perp$. Otherwise, the challenger flips a fair binary coin $\beta \xleftarrow{\$} \{0, 1\}$. If $\beta = 0$, $\mathcal{B}$ computes a derived signature $\sigma^* = \text{SignDerive}(\text{pk}, (\{\sigma_m\}_{m \in M}, M), m')$. If $\beta = 1$, $\mathcal{B}$ computes $\sigma^* = \text{Sign}(\text{sk}, m')$. In either case, σ^* is sent as a challenge to $\mathcal{A}$.
3. Eventually, $\mathcal{A}$ outputs a bit $\beta' \in \{0, 1\}$ and wins if $\beta' = \beta$. As usual, $\mathcal{A}$'s advantage is defined to be $\text{Adv}(\mathcal{A}) = |\Pr[\beta' = \beta] - 1/2|$.

While the latter definition seems sufficient for many applications, it still does not imply Definition 29 and we may want signatures to be unlinkable in the statistical sense. The resulting stronger definition implies both Definition 31 and Definition 29 and goes as follows.

Definition 32 A *P-homomorphic signature* $(\text{Keygen}, \text{Sign}, \text{SignDerive}, \text{Verify})$ is **completely context-hiding** if, for all pairs $(\text{pk}, \text{sk}) \leftarrow \text{Keygen}(\lambda)$, all messages $M \subset \mathcal{M}^*$ and $m' \in \mathcal{M}$ such that $P(M, m') = 1$, for all $\{\sigma_m\}_{m \in M}$ such that $\text{Verify}(\text{pk}, M, \{\sigma_m\}_{m \in M}) = 1$, the following distributions are statistically close

$$\begin{aligned} & \{(\text{sk}, \{\sigma_m\}_{m \in M}, \text{Sign}(\text{sk}, m'))\}_{\text{sk}, M, m'}, \\ & \{(\text{sk}, \{\sigma_m\}_{m \in M}, \text{SignDerive}(\text{pk}, (\{\sigma_m\}_{m \in M}, M), m'))\}_{\text{sk}, M, m'}. \end{aligned}$$

In all schemes based on the dual system approach [9, 79], the existence of an alternative distribution of acceptable signatures makes it seemingly impossible to satisfy the above definition. In these schemes, the combination of strong (i.e., Definition 29) and adaptive context-hiding security thus appears as the best we can hope for. For this reason, we chose to present Definition 31 first instead of directly working with Definition 32.

Definition 32 assumes honestly generated keys (sk, pk) . It can be strengthened by allowing the adversary to generate a pair (sk, pk) of its own. In the random oracle model, the construction of [25] is easily seen to satisfy such a stronger definition (if we assume that all public keys live in a cyclic group which is part of common public parameters) because it has unique signatures. In the standard model, we do not know of any scheme that would be secure in that sense.

In Sections 4.2, 4.3 and 4.4 we satisfy Definition 32 with our homomorphic signatures for subset predicates, substring predicates and subspace predicates.

4.1.3 Separation Results

Definition 32 of completely context-hiding P-homomorphic signatures trivially implies the other context-hiding (CH) notions. More precisely, it is easy to see that Definition 32 implies Definition 31 which in turn implies Definition 30 and that Definition 32 also implies Definition 29. Assuming the existence of a completely context-hiding P-homomorphic signature scheme Π_0 we will show that these implications are all strict by modifying the scheme Π_0 and gradually weakening its security. Furthermore, this will allow us to show that Definitions 31 and 30 are incomparable relatively to Definition 29. The next sections show that the completely context-hiding level of privacy is not so hard to achieve.

Let us suppose that $\Pi_0 := (\text{Keygen}_0, \text{Sign}_0, \text{SignDerive}_0, \text{Verify}_0)$ is an unforgeable scheme which satisfies Definition 32 with secret key sk_0 and signatures σ_0 . In order to prove the announced results we extend Π_0 into schemes Π_i , for $i = 1, 2, 3$, with $\text{sk}'_i = \text{sk}_0 \parallel \text{sk}_i$ and $\sigma'_i = \sigma_0 \parallel \sigma_i$, and show that Π_1 is strongly

CH but not weakly adaptively CH, that Π_2 is weakly adaptive CH but not adaptively CH, and finally, that Π_3 is adaptively CH but not strongly CH. In our Asiacrypt paper [10] we made these separations using existing linearly homomorphic schemes in composite-order pairings. Nevertheless this other approach has the advantage of exhibiting practical attacks in known randomizable schemes and illustrates that none of them are even adaptively context-hiding. The method here is less practical or natural but it is also more general and straightforward.

Separating Strong and Weak-Adaptive Context-Hiding

Let us consider Π_1 where $\text{sk}_1 := (a)$ for a random scalar a . A signature σ'_1 on a message $m \in \mathcal{M}$ is the signature σ_0 on m concatenated with $\sigma_1 = g^a$ where g is an element of the public key of Π_0 . The new verification algorithm simply ignores the σ_1 component and checks σ_0 with Verify_0 . To derive a signature on a message m' derivable from $M \subset \mathcal{M}$ with valid signatures $\{(\sigma_0(m), \sigma_1(m))\}_{m \in M}$, the new algorithm for deriving signatures returns (σ_0, σ_1) where $\sigma_0 = \text{SignDerive}_0(\text{pk}, (\{\sigma_0(m)\}_{m \in M}, M), m')$ and σ_1 is $\sigma_1(m)$ of the first m of the list M .

It is easy to see that the construction Π_1 remains unforgeable and is strongly context-hiding in the sense of Definition 29. Indeed, the signing algorithm is honestly run in the first distribution of Definition 29. This implies that, for any message set $M \subset \mathcal{M}$, the underlying σ_1 will have the same value no matter if the second signature is produced by Sign_1 or SignDerive_1 . Further, since Π_0 is completely context-hiding by assumption, the distributions are still statistically closed. After all, we only add a same constant element in honest distributions.

However, the scheme does not satisfy Definition 30. Indeed, in step 2, the adversary can first invoke the signing oracle on k occasions to obtain signatures for some set $M = \{m_1, \dots, m_k\}$ of his/her choice. If we denote by $\{\sigma_m\}_{m \in M}$ the resulting signatures, the adversary replaces the $\sigma_1(m)$ components by another element h and obtains a new set $\{\sigma'_m\}_{m \in M}$ of valid signatures. The adversary $\mathcal{A}$ can then choose a random message $m' \in \mathcal{M}$ such that $P(M, m') = 1$ and send $((M, \{\sigma'_m\}_{m \in M}), m')$ to the challenger. The latter returns a challenge signature $\sigma^* = (\sigma_0^*, \sigma_1^*)$ on m' and $\mathcal{A}$ can immediately figure out whether σ_1^* is h or g^a . The adversary wins the game by respectively returning $\beta' = 0$ or $\beta' = 1$.

Separating Weak-Adaptive and Adaptive Context-Hiding

Let us consider Π_2 where $\text{sk}_2 := (\mu, \nu)$ such that the three elements $g, h = g^\mu$ and f^ν augment the public key. A signature (σ_0, σ_2) on m is obtained by running $\text{Sign}_0(\text{sk}_0, m)$ and padding a BBS encryption of 1, i.e. $\sigma_2 = (h^r, f^s, g^{r+s})$ for random scalars r, s . Again, validity of signatures are verified by ignoring σ_2 and checking the validity of σ_0 as in Π_0 . Deriving signatures is performed from the σ_0 components by running SignDerive_0 . Then, the resulting signature is padded with a random triple $\sigma_2 \leftarrow (u, v, w)$.

The scheme Π_2 remains unforgeable since sk_2 is independent of sk_0 . Since Π_0 is completely CH, the only usefull information the adversary could embed in valid signatures $\{\sigma_m\}_{m \in M}$ on M in the challenge phase of Definition 30 and Definition 31 are overwritten either by a random triple, if $\beta = 0$, or by a fresh BBS encryption of 1, if $\beta = 1$. When $\mathcal{A}$ does not have sk'_2 , winning the game implies breaking the underlined DLin assumption and if the secret key is available to $\mathcal{A}$, it can check whether $w = u^{1/\mu}v^{1/\nu}$. The adversary against adaptive CH wins the game with overwhelming probability.

Separating Adaptive and Strong Context-Hiding

The scheme Π_3 is the same as Π_2 except that the decryption key of the BBS encryption is *not* included in the signing secret key, $\text{sk}_3 = \epsilon$ is the empty string. Consequently, the adversary $\mathcal{A}$ against the adaptive context-hiding security is exactly in the situation of the adversary against the weak adaptive context-hiding security of scheme Π_2 . Under DLin, the scheme Π_3 satisfies Definition 31. However, the distributions of Definition 29 are no more statistically closed as in Π_0 . The indistinguishability is only computational now.

We made the additional assumption of DLin to show the separation between all the context-hiding properties. It is not a drawback since our completely context-hiding pairing-based signatures are all proven secure assuming that the DLin problem is intractable.

4.2 Subset Predicate Signatures

In this section, we use the malleability properties of Groth-Sahai proofs (see Section 2.5.3) to construct a homomorphic signature for subset predicates. The main advantage over the approach of Ahn *et al.* [6] is that we obtain constant-size¹ public keys in the standard model. In the standard model, the CP-ABE approach of Ahn *et al.* is currently limited to provide linear-size public keys in the maximal length of signed messages.

This limitation could be avoided using a ciphertext-policy adaptation of the unbounded key-policy ABE system of [105]. However, the ABE construction of [105] is only known to be selectively secure. Conceivably, such a scheme can be obtained by extending the techniques of [105]. Meanwhile, Okamoto-Takashima proposed such a fully secure ABE but it does not seem to support delegation invariance. This property is necessary to achieve strong context-hiding privacy when the ABE is turned into a subset predicate signature by applying the Naor-like transform of Ahn *et al.*. The details are beyond the scope of the thesis.

¹By “constant”, we mean that it only depends on the security parameter and not on the number of messages to be signed.

Nevertheless, the resulting system would probably encounter some difficulties when it comes to obtaining complete context-hiding security. Indeed, both constructions are based on dual system techniques which allow the generation of several secret key distributions in a single setup. Transposed in subset predicate signature, it means that the different distributions of signing secret keys (which corresponds to the master secret key of the ABE scheme) enables computing “dishonest” valid signatures of distinct distribution. Inherently, this method does not seem to satisfy the requirements of Definition 32 since the different distributions are only computationally indistinguishable.

In contrast, our scheme is proved completely context-hiding and fully (as opposed to selective-message) secure. It also allows for messages of unbounded (but polynomial) length.

4.2.1 Intuition behind our Subset Predicate Signatures

In homomorphic signatures for subset predicates, the message space $\mathcal{M}$ can be defined as the set of tuples $\mathcal{M} := \Sigma^*$, where Σ is a set of words. The predicate P is defined in such a way that, for any polynomials n and n' , we have

$$\begin{aligned} P(\{m_1, \dots, m_n\}, \{m'_1, \dots, m'_{n'}\}) &= 1 \\ \iff (n' \leq n) \bigwedge_{j=1}^{n'} (m'_j \in \{m_1, \dots, m_n\}). \end{aligned}$$

The intuition of the scheme begins with the following naive construction, based on any digital signature that only works when privacy is not a concern. The public key of the scheme is a standard digital signature key pair (sk, pk) . When a message $\text{Msg} = \{m_1, \dots, m_n\}$ must be signed, the signer generates a fresh public key (sk', pk') , certifies pk' by computing $\sigma_{pk'} \leftarrow \text{Sign}(sk, pk')$ and returning $(pk', \sigma_{pk'}, \{\sigma_i = \text{Sign}(sk', m_i)\}_{i=1}^n)$. This simple construction immediately allows signature derivations for subset predicates. Moreover, since each signed set of words Msg involves a different public key pk' , there is no way to generate a signature on a message Msg^* that mixes words from two distinct signed messages $\text{Msg}_1, \text{Msg}_2$. However, this construction is trivially not context-hiding. To achieve the latter property, instead of leaving pk' and $\{\sigma_i\}_{i=1}^{n'}$ appear in the clear within signatures, we let them appear in committed form and appeal to non-interactive witness indistinguishable (NIWI) arguments of knowledge of these signatures and keys. Then, the randomizability properties of Groth-Sahai proofs (which are recalled in Section 2.5.3) come in handy to obtain the desired privacy properties.

To realize the above idea, we work with Waters signature and the structure-preserving signature of Abe *et al.* (see both in Section 2.2.2) because they make it possible to work with *linear* pairing product equations. As observed in [70], these equations have proofs that only depend on the randomness of Groth-Sahai commitments and not on the committed witnesses or on the constant

of the target group of the equation. In the **SignDerive** algorithm, this allows updating some of the witnesses in such a way that the old proof remains valid.

4.2.2 Construction with Short Keys for Subset Predicates

In the following notations, we use the coordinate-wise pairing $E : \mathbb{G} \times \mathbb{G}^3 \rightarrow \mathbb{G}_T^3$ defined for GS-proofs of linear equations such that, for any element $h \in \mathbb{G}$ and any vector $\vec{g} = (g_1, g_2, g_3)$, we have $E(h, \vec{g}) = (e(h, g_1), e(h, g_2), e(h, g_3))$. We also recall the notations $\iota(X) = (1_{\mathbb{G}}, 1_{\mathbb{G}}, X) \in \mathbb{G}^3$ for $X \in \mathbb{G}$ and its analogue for $Y \in \mathbb{G}_T$, $\tilde{\iota}_T(Y) = (1_{\mathbb{G}_T}, 1_{\mathbb{G}_T}, Y) \in \mathbb{G}_T^3$.

Keygen(λ): given a security parameter $\lambda \in \mathbb{N}$, chooses bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$. Then, perform as follows.

1. Generates a Groth-Sahai CRS $\mathbf{f} = (\vec{f}_1, \vec{f}_2, \vec{f}_3)$ for the perfect witness indistinguishability setting. Namely, chooses $\vec{f}_1 = (f_1, 1, g)$, $\vec{f}_2 = (1, f_2, g)$, and $\vec{f}_3 = \vec{f}_1^{\xi_1} \cdot \vec{f}_2^{\xi_2} \cdot \iota(g)^{-1}$, with $f_1, f_2 \xleftarrow{\$} \mathbb{G}$, $\xi_1, \xi_2 \xleftarrow{\$} \mathbb{Z}_p$.
2. Generates a key pair $(sk_{\text{AHO}}, pk_{\text{AHO}})$ for the AHO signature in order to sign messages consisting of a single group element. This key pair is $sk_{\text{AHO}} = (\alpha_a, \alpha_b, \gamma_z, \delta_z, \gamma_1, \delta_1)$ and

$$pk_{\text{AHO}} = \left(G_r, H_r, G_z = G_r^{\gamma_z}, \right. \\ \left. H_z = H_r^{\delta_z}, G_1 = G_r^{\gamma_1}, H_1 = H_r^{\delta_1}, A, B \right).$$

3. Generates parameters for the Waters signature. Namely, chooses group elements $h \xleftarrow{\$} \mathbb{G}$, and $(u_0, u_1, \dots, u_L) \xleftarrow{\$} \mathbb{G}^{L+1}$. These are used to implement a hash function $H_{\mathbb{G}} : \{0, 1\}^L \rightarrow \mathbb{G}$ such that, for any string $m = m[1] \dots m[L] \in \{0, 1\}^L$, $H_{\mathbb{G}}(m) = u_0 \cdot \prod_{i=1}^L u_i^{m[i]}$.

The public key is defined to be

$$pk := \left((\mathbb{G}, \mathbb{G}_T), g, \mathbf{f}, pk_{\text{AHO}}, h, \{u_i\}_{i=0}^L \right)$$

and the private key is $sk = sk_{\text{AHO}}$. The public key defines $\Sigma = \{0, 1\}^L$.

Sign(sk, Msg): on input of a message $\text{Msg} = \{m_i\}_{i=1}^n$, where $m_i \in \{0, 1\}^L$ for each i , and the private key $sk = sk_{\text{AHO}}$, performs as follows.

1. Chooses a new public key $X = g^x$ for Waters signatures, with $x \xleftarrow{\$} \mathbb{Z}_p$. Generates a Groth-Sahai commitment $\vec{C}_X = \iota_{\mathbb{G}}(X) \cdot \vec{f}_1^{r_X} \cdot \vec{f}_2^{s_X} \cdot \vec{f}_3^{t_X}$, with $r_X, s_X, t_X \xleftarrow{\$} \mathbb{Z}_p$.
2. Generates an AHO signature $(\theta_1, \dots, \theta_7) \in \mathbb{G}^7$ on the group element $X \in \mathbb{G}$. Then, for each $j \in \{1, 2, 5\}$, generates Groth-Sahai commitments $\vec{C}_{\theta_j} = \iota_{\mathbb{G}}(\theta_j) \cdot \vec{f}_1^{r_{\theta_j}} \cdot \vec{f}_2^{s_{\theta_j}} \cdot \vec{f}_3^{t_{\theta_j}}$. Finally, generates NIWI

proofs $\vec{\pi}_{\text{AHO},1}, \vec{\pi}_{\text{AHO},2} \in \mathbb{G}^3$ that committed variables $(X, \theta_1, \theta_2, \theta_5)$ satisfy

$$\begin{aligned} A \cdot e(\theta_3, \theta_4)^{-1} &= e(G_z, \theta_1) \cdot e(G_r, \theta_2) \cdot e(G_1, X) \\ B \cdot e(\theta_6, \theta_7)^{-1} &= e(H_z, \theta_1) \cdot e(H_r, \theta_5) \cdot e(H_1, X) \end{aligned} \quad (4.1)$$

These proofs are obtained as

$$\begin{aligned} \vec{\pi}_{\text{AHO},1} &= (G_z^{-r_{\theta_1}} G_r^{-r_{\theta_2}} G_1^{-r_X}, G_z^{-s_{\theta_1}} G_r^{-s_{\theta_2}} G_1^{-s_X}, G_z^{-t_{\theta_1}} G_r^{-t_{\theta_2}} G_1^{-t_X}) \\ \vec{\pi}_{\text{AHO},2} &= (H_z^{-r_{\theta_1}} H_r^{-r_{\theta_5}} H_1^{-r_X}, H_z^{-s_{\theta_1}} H_r^{-s_{\theta_5}} H_1^{-s_X}, H_z^{-t_{\theta_1}} H_r^{-t_{\theta_5}} H_1^{-t_X}) \end{aligned}$$

- For each $i \in \{1, \dots, n\}$, generates a Waters signature $(\sigma_{i,1}, \sigma_{i,2})$ on the word $m_i \in \{0, 1\}^L$ by computing $(\sigma_{i,1}, \sigma_{i,2}) = (h^x \cdot H_{\mathbb{G}}(m_i)^{\chi_i}, g^{\chi_i})$ for a randomly chosen $\chi_i \xleftarrow{\$} \mathbb{Z}_p$. Then, generates a Groth-Sahai commitment $\vec{C}_{\sigma_{i,1}} = \iota_{\mathbb{G}}(\sigma_{i,1}) \cdot \vec{f}_1^{r_{i,1}} \cdot \vec{f}_2^{s_{i,1}} \cdot \vec{f}_3^{t_{i,1}}$, with $r_{i,1}, s_{i,1}, t_{i,1} \xleftarrow{\$} \mathbb{Z}_p$, and a NIWI proof $\pi_{W,i}$ that $(X, \sigma_{i,1})$ satisfy

$$e(H_{\mathbb{G}}(m_i), \sigma_{i,2}) = e(X, h)^{-1} \cdot e(\sigma_{i,1}, g). \quad (4.2)$$

This proof is obtained as $\pi_{W,i} = (h^{r_X} \cdot g^{-r_{i,1}}, h^{s_X} \cdot g^{-s_{i,1}}, h^{t_X} \cdot g^{-t_{i,1}})$.

- Returns the signature

$$\sigma = \left(\vec{C}_X, \{\vec{C}_{\theta_j}\}_{j \in \{1,2,5\}}, \{\theta_j\}_{j \in \{3,4,6,7\}}, \vec{\pi}_{\text{AHO},1}, \vec{\pi}_{\text{AHO},2}, \{\vec{C}_{\sigma_{i,1}}, \sigma_{i,2}, \pi_{W,i}\}_{i=1}^n \right). \quad (4.3)$$

Note that proofs $\vec{\pi}_{\text{AHO},1}, \vec{\pi}_{\text{AHO},2}$ and $\{\pi_{W,i}\}_i$ only depend on the randomness used in commitments and not on the committed values or on the left-hand side members of pairing-product equations (4.1) and (4.2).

SignDerive(pk, Msg, Msg', σ): given pk, $\text{Msg} = \{m_i\}_{i=1}^n$ and $\text{Msg}' = \{m'_i\}_{i=1}^{n'}$, returns $\perp$ if there exists $i \in \{1, \dots, n'\}$ such that $m'_i \notin \{m_i\}_{i=1}^n$. Otherwise, parses σ as in (4.3). For each $i \in \{1, \dots, n'\}$, lets $\rho(i) \in \{1, \dots, n\}$ be the index such that $m'_i = m_{\rho(i)}$. Then, for each $i \in \{1, \dots, n'\}$, does the following.

- Re-randomizes the commitment $\vec{C}_X$ and the proofs $\vec{\pi}_{\text{AHO},1}, \vec{\pi}_{\text{AHO},2}, \{\pi_{W,i}\}_i$ accordingly. Lets $\vec{C}'_X, \vec{\pi}'_{\text{AHO},1}, \vec{\pi}'_{\text{AHO},2}$, and $\{\pi'_{W,i}\}_i$ be the randomized commitment and proofs. Note that in all of these commitments and proofs (r_X, s_X, t_X) have been updated consistently.
- Re-randomizes $\{\vec{C}_{\theta_j}\}_{j \in \{2,5\}}$ and $\{\theta_j\}_{j \in \{3,4,6,7\}}$ by choosing $\varrho_2, \varrho_5, \mu, \nu$ and computing

$$\begin{aligned} \vec{C}'_{\theta_2} &= \vec{C}_{\theta_2} \cdot \iota_{\mathbb{G}}(\theta_4)^{\varrho_2} & \theta'_3 &= (\theta_3 \cdot G_r^{-\varrho_2})^{1/\mu} & \theta'_4 &= \theta_4^\mu, \\ \vec{C}'_{\theta_5} &= \vec{C}_{\theta_5} \cdot \iota_{\mathbb{G}}(\theta_7)^{\varrho_5} & \theta'_6 &= (\theta_6 \cdot H_r^{-\varrho_5})^{1/\nu} & \theta'_7 &= \theta_7^\nu. \end{aligned}$$

We note that, although the committed values inside $\vec{C}'_{\theta_2}, \vec{C}'_{\theta_5}$ have changed, the proofs $\pi'_{\text{AHO},1}, \pi'_{\text{AHO},2}$ are still valid for the new committed values. Then, computes $\{\vec{C}''_{\theta_j}\}_{j \in \{1,2,5\}}$ by re-randomizing the commitments $\vec{C}_{\theta_1}, \{\vec{C}'_{\theta_j}\}_{j \in \{2,5\}}$ and re-randomizes the proofs $\pi'_{\text{AHO},1}, \pi'_{\text{AHO},2}$ again. Lets $\vec{\pi}''_{\text{AHO},1}, \vec{\pi}''_{\text{AHO},2}$ be the re-randomized proofs.

3. For each $i \in \{1, \dots, n'\}$, chooses $\chi'_i \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\vec{C}'_{\sigma_{\rho(i),1}} = \vec{C}_{\sigma_{\rho(i),1}} \cdot \iota_{\mathbb{G}}(H_{\mathbb{G}}(m_{\rho(i)})^{\chi'_i}), \quad \sigma'_{\rho(i),2} = \sigma_{\rho(i),2} \cdot g^{\chi'_i}.$$

Even though the committed value inside $\vec{C}'_{\sigma_{\rho(i),1}}$ has changed, $\vec{\pi}'_{W,\rho(i)}$ remains a valid proof that the updated committed value $\sigma'_{\rho(i),1}$ satisfies $e(X, h) \cdot e(H_{\mathbb{G}}(m_{\rho(i)}), \sigma'_{\rho(i),2}) = e(\sigma'_{\rho(i),1}, g)$. The commitment $\vec{C}'_{\sigma_{\rho(i),1}}$ is then re-randomized and the proof $\vec{\pi}'_{W,\rho(i)}$ is re-randomized accordingly. Lets $\vec{C}''_{\sigma_{\rho(i),1}}$ and $\vec{\pi}''_{W,\rho(i)}$ denote the new commitment and proof.

4. Returns the signature

$$\sigma' = \left(\vec{C}'_X, \{\vec{C}''_{\theta_j}\}_{j \in \{1,2,5\}}, \{\theta'_j\}_{j \in \{3,4,6,7\}}, \right. \\ \left. \vec{\pi}''_{\text{AHO},1}, \vec{\pi}''_{\text{AHO},2}, \{\vec{C}''_{\sigma_{\rho(i),1}}, \sigma'_{\rho(i),2}, \vec{\pi}''_{W,\rho(i)}\}_{i=1}^{n'} \right). \quad (4.4)$$

Verify(pk, Msg, σ): given pk, σ and $\text{Msg} = \{m_i\}_{i=1}^n$, parses σ as per (4.3).

1. Returns 0 if $\vec{\pi}_{\text{AHO},1} = (\pi_1, \pi_2, \pi_3)$ and $\vec{\pi}_{\text{AHO},2} = (\pi_4, \pi_5, \pi_6)$ do not satisfy, with $t_A = \iota_T(A) \cdot E(\theta_3, \iota_{\mathbb{G}}(\theta_4))^{-1}$ and $t_B = \iota_T(B) \cdot E(\theta_6, \iota_{\mathbb{G}}(\theta_7))^{-1}$,

$$t_A = E(G_z, \vec{C}_{\theta_1}) \cdot E(G_r, \vec{C}_{\theta_2}) \cdot E(G_1, \vec{C}_X) \cdot \prod_{j=1}^3 E(\pi_j, \vec{f}_j) \\ t_B = E(H_z, \vec{C}_{\theta_1}) \cdot E(H_r, \vec{C}_{\theta_5}) \cdot E(H_1, \vec{C}_X) \cdot \prod_{j=1}^3 E(\pi_{j+3}, \vec{f}_j).$$

2. Returns 1 if, for each i , $\vec{\pi}_{W,i} = (\pi_{W,i,1}, \pi_{W,i,2}, \pi_{W,i,3})$ satisfies

$$E(h, \vec{C}_X) \cdot E(H_{\mathbb{G}}(m_i), \iota(\sigma_{i,2})) = E(g, \vec{C}_{\sigma_{i,1}}) \cdot \prod_{j=1}^3 E(\pi_{W,i,j}, \vec{f}_j).$$

Otherwise, returns 0.

From an efficiency point of view, the scheme is asymptotically optimal with constant-size public (and private) keys and signatures consisting of $7n + 22$ group elements for messages of length n .

4.2.3 Security of our Subset Predicate Signatures

This section shows that the above homomorphic signature scheme for subset predicate reaches the highest level of context hiding privacy. Afterwards, the P-homomorphic signatures is proved unforgeable in the sense of Definition 28.

Theorem 12 *The scheme is completely context-hiding.*

Sketch of proof. The security of the scheme in the sense of Definition 32 follows from the fact that, when the CRS $\mathbf{f} = (\vec{f}_1, \vec{f}_2, \vec{f}_3)$ is generated as in step 1 of the key generation algorithm, commitments $\vec{C}_X$ are always perfectly hiding and proofs are perfectly witness-indistinguishable.

Moreover, the components $\{\theta_j\}_{j \in \{3,4,6,7\}}$ of AHO signatures are perfectly re-randomized by the signature derivation algorithm and the same is true for $\{\sigma_{i,2}\}_{i=1}^n$. As for Groth-Sahai proofs $\pi_{\text{AHO},1}, \pi_{\text{AHO},2}$ and $\{\pi_{W,i}\}_{i=1}^n$, they are also perfectly re-randomized. It comes that the output of `SignDerive` is always distributed like a fresh signature. $\square$

Theorem 13 *The scheme is unforgeable if the DLIN and q -SFP assumptions hold in $\mathbb{G}$.*

Proof. Having shown that the scheme is completely context-hiding (and thus secure in the sense of Definition 29), we can prove unforgeability using a simplified definition where the adversary only interacts with a signing oracle (see [6]). The proof uses a sequence of three games where, for each $i \in \{0, 1, 2\}$, we denote by S_i the event that the adversary $\mathcal{A}$ produces a successful forgery at the end of `Gamei`.

Game₀: This game is the real game so $\mathcal{A}$'s advantage is $\Pr[S_0]$.

Game₁: We modify the generation of the public key and choose $\mathbf{f} = (\vec{f}_1, \vec{f}_2, \vec{f}_3)$ as a Groth-Sahai CRS for the perfect soundness setting, where even an unbounded adversary cannot prove false statements. More precisely, the challenger $\mathcal{B}$ sets up $\vec{f}_1 = (f_1, 1, g)$, $\vec{f}_2 = (1, f_2, g)$ and $\vec{f}_3 = \vec{f}_1^{\xi_1} \cdot \vec{f}_2^{\xi_2}$, with $f_1 = g^{\phi_1}$ and $f_2 = g^{\phi_2}$, for randomly chosen $\phi_1, \phi_2, \xi_1, \xi_2 \xleftarrow{\$} \mathbb{Z}_p$. It is easy to build a DLIN distinguisher that bridges between `Game0` and `Game1`. This implies that, under the DLIN assumption, this modification does not significantly affect $\mathcal{A}$'s behaviour. We can thus write $|\Pr[S_1] - \Pr[S_0]| \leq \text{Adv}^{\text{DLIN}}(\mathcal{B})$. Moreover, from `Game1` onwards, all Groth-Sahai commitments are perfectly binding and their content can be extracted using (ϕ_1, ϕ_2) by performing a BBS decryption (see Section 2.3.3 for details).

Game₂: In this game, $\mathcal{B}$ makes use of the values $(\phi_1, \phi_2) = (\log_g(f_1), \log_g(f_2))$ that were defined in `Game1`. When $\mathcal{A}$ outputs a forgery σ^* , the challenger $\mathcal{B}$ uses (ϕ_1, ϕ_2) to extract X^* from the Groth-Sahai commitment $\vec{C}_X^*$

contained in σ^* (recall that, due to the modification introduced in Game_1 , $\tilde{C}_X^*$ is now an extractable commitment). We raise a failure event, called F_2 , and let the challenger $\mathcal{B}$ abort if the extracted X^* was never used in any signing query. Clearly, any occurrence of F_2 immediately contradicts the unforgeability of the AHO signature and thus the q -SFP assumption. We can thus write $|\Pr[S_2] - \Pr[S_1]| \leq \Pr[F_2] \leq \mathbf{Adv}^{q\text{-SFP}}(\mathcal{B})$.

In Game_2 , we can easily prove that, conditioned on $\neg F_2$, event S_2 only occurs with negligible probability if the Diffie-Hellman assumption holds. Let $(\sigma^*, \text{Msg}^* = \{m_1^*, \dots, m_{n^*}^*\})$ denote $\mathcal{A}$'s forgery. If F_2 does not occur, we know that X^* , which is extracted from the commitment $\tilde{C}_X^*$ contained in σ^* , *did* appear in some signing query. Let $j \in \{1, \dots, q\}$ denote the index of that query $\text{Msg}_j = \{m_{j,1}, \dots, m_{j,n_j}\}$. If σ^* is a successful forgery, we know that $\text{Msg}^* \not\subseteq \text{Msg}_j$. In other words, there exists $\ell \in \{1, \dots, n^*\}$ such that $m_\ell^* \notin \text{Msg}_j$. This allows constructing a forger $\mathcal{B}'$ breaking the unforgeability of Waters signatures with probability $\Pr[S_2 | \neg F_2]/q$.

Specifically, the forger $\mathcal{B}'$ of Game_3 receives as input a public key $(X^\dagger, h, (u_0, u_1, \dots, u_L))$ for Waters signatures. To generate the public key pk of the system, it generates an AHO key pair $(sk_{\text{AHO}}, pk_{\text{AHO}})$ of its own at the beginning of the game. It also sets up $\mathbf{f} = (\vec{f}_1, \vec{f}_2, \vec{f}_3)$ as an extractable Groth-Sahai CRS. It finally chooses a random index $j^* \xleftarrow{\$} \{1, \dots, q\}$ as a guess that the extracted $X^* \in \mathbb{G}$ will coincide with the value X appearing in the j^* -th signing query. At each query $\text{Msg}_j = \{m_{j,1}, \dots, m_{j,n_j}\}$ such that $j \neq j^*$, $\mathcal{B}'$ answers the query faithfully, by setting up $X = g^x$ for a randomly chosen $x \xleftarrow{\$} \mathbb{Z}_p$ which is used to generate Waters signatures as in the real signing algorithm. If $j = j^*$, $\mathcal{B}'$ computes $\tilde{C}_X$ using its challenge Waters public key $X^\dagger$ at step 1 of the signing algorithm. At step 3 of the signing algorithm, $\mathcal{B}'$ obtains $\{(\sigma_{i,1}, \sigma_{i,2})\}_{i=1}^{n_j}$ by successively invoking its signing oracle for messages $m_{j^*,1}, \dots, m_{j^*,n_{j^*}}$. At the end of the game, with probability $1/q$, the element X^* extracted from $\tilde{C}_X^*$ in $\mathcal{A}$'s forgery happens to be $X^* = X^\dagger$. By hypothesis, $\text{Msg}^* = \{m_1^*, \dots, m_{n^*}^*\}$ contains an element m_ℓ^* that $\mathcal{B}'$ did not query to its challenger at the j^* -th signing query. From σ^* , $\mathcal{B}'$ can thus extract a valid signature $(\sigma_{\ell,1}^*, \sigma_{\ell,2}^*)$ on m_ℓ^* and break the security of Waters signatures. The result of [138] implies that $\Pr[S_2 | \neg F_2] \leq 8 \cdot q^2 \cdot (L+1) \cdot \mathbf{Adv}^{\text{CDH}}(\mathcal{B}')$.

When collecting probabilities, we find

$$\Pr[S_0] \leq \mathbf{Adv}^{\text{DLIN}}(\mathcal{B}) + 2 \cdot \mathbf{Adv}^{q\text{-SFP}}(\mathcal{B}) + 8 \cdot q^2 \cdot (L+1) \cdot \mathbf{Adv}^{\text{CDH}}(\mathcal{B}'').$$

Since the DLIN and q -SFP assumptions both imply the hardness of CDH in the same group, the statement of the theorem follows. $\square$

We note that, if one of the structure-preserving signatures of Abe *et al.* [1] is used in place of the AHO signature, it is easy to build a scheme relying on the DLIN assumption only.

4.3 Quotable Signatures

Given a signature on a message m , quotable signatures allow the public derivation of signatures on any substring of m . Ahn *et al.* [6] gave a system where signatures have quasi-linear size: for a message consisting of n symbols, each signature contains $O(n \log n)$ group elements². Their construction is known to be only strongly context-hiding (in the sense of Definition 29) and selectively unforgeable in the random oracle model. It was argued that their scheme can be modified so as to be proved fully unforgeable in the standard model using the dual system encryption technique of Waters [139] (or, more precisely, its signature analogue [79]). The latter inherently involves two distinct distributions of signatures satisfying the verification algorithm. The very existence of an alternative distribution of valid signatures implies that the resulting system can hardly be completely context-hiding.

The contribution of this section is a quotable signature scheme whose design principle is very different from Ahn *et al.*. The new scheme is proved fully unforgeable in the standard model and also turns out to be the first completely context-hiding quotable signature. Moreover, it improves upon the worst-case efficiency of Ahn *et al.* scheme in that a n -symbol message can be signed using $O(n)$ group elements.

Our construction builds on the AHO structure-preserving signature which is used to sign individual message symbols. An important property of the AHO structure-preserving signature is that certain signature components can serve as a commitment to the message. Our quotable signature exploits this property to link signatures on individual symbols: each symbol is signed with a commitment to the next symbol. Quotable signatures are then obtained as a sequence of perfectly hiding commitments to these underlying signatures and non-interactive randomizable arguments of their validity.

Beyond its asymptotically shorter signatures, our scheme also allows signing messages of arbitrary length using a constant-size public key. In contrast, the Ahn *et al.* construction requires the key generation algorithm to define a logarithmic bound on the maximal number of symbols in messages to be signed.

4.3.1 Intuition behind our Substring Predicate Signatures

In quotable signatures, given a signature on some message, signatures should only be possible to derive on arbitrary substrings of the original message. The message space $\mathcal{M}$ is also defined as the set of strings $\mathcal{M} := \Sigma^*$, where Σ is a set of symbols. The predicate P is defined by $P(\text{Msg}_1, \text{Msg}_2) = 1$ whenever Msg_2 is a substring of Msg_1 and 0 otherwise.

The scheme bears resemblance to the homomorphic signature for subset predicates of the previous section which also builds on structure-preserving

²In the signature derivation algorithm of Ahn *et al.*, two kinds of signatures can be produced. Apart from Type I signatures, which are distributed as original signatures, Type II signatures have $O(\log n)$ -size signatures but cannot be quoted any further.

signatures. In fact, the construction is itself a structure-preserving quotable signature as it allows signing sequences of group elements.

We actually use a variant of the unbounded AHO signature scheme which allows signing messages of arbitrary length with a public key of fixed size. In [4], this is achieved by taking advantage of the property called “signature binding” (proved in [4, Lemma 3]), which informally says that signature components (θ_3, θ_6) can be used as a commitment to the message: namely, given only the public key and access to a signing oracle, unless the scheme is existentially forgeable under chosen-message attacks, it is infeasible to come up with two *distinct* messages $(M_1, \dots, M_n), (M'_1, \dots, M'_n)$ with corresponding valid signatures $\sigma = (\theta_1, \dots, \theta_7)$ and $\sigma' = (\theta'_1, \dots, \theta'_7)$ such that $\theta_3 = \theta'_3$ and $\theta_6 = \theta'_6$. This remains true *even* if $(M_1, \dots, M_n)$ and $(M'_1, \dots, M'_n)$ are both submitted to the signing oracle during the game. Using this observation, a basic signature scheme where the message space is $\mathbb{G}^3$ can be turned into an “unbounded” structure-preserving signature, where the signer can sign messages of arbitrary length. The idea is to use signature components $\{(\theta_{i,3}, \theta_{i,6})\}_{i=1}^n$ to link adjacent message blocks together: each block $m_i \in \mathbb{G}$ is signed along with the $(\theta_{i-1,3}, \theta_{i-1,6})$ components of the signature on the previous block $m_{i-1} \in \mathbb{G}$. In our scheme, we proceed in the same way but, unlike Abe *et al.* [4], we do not encode the total number of blocks within the message. This modification allows anyone to quote signatures by removing portions of the chain in its extremities. In order to prevent illegal combinations of two different chains, the signer processes the last block m_n of each message $(m_1, \dots, m_n)$ by signing it with a pair of random group elements $(\tilde{\theta}_3, \tilde{\theta}_6)$ which are part of the private key. This allows us to prove security using the same arguments as Abe *et al.*

For the sake of privacy, the components of $\{\sigma_i\}_{i=1}^n$ are not explicitly given out but only appear within perfectly hiding Groth-Sahai commitments accompanied with appropriate NIWI arguments. At each signature derivation, commitments and NIWI arguments are suitably re-randomized.

An important difference with our construction for subset predicates is that underlying AHO signatures entirely appear in committed form. The reason is that using $(\theta_{i,3}, \theta_{i,6})$ in the chaining process prevents their re-randomization. For this reason, they also have to be committed so that we need to work with quadratic pairing-product equations.

4.3.2 Construction of Linear-Size Quotable Signatures

In the following, we use the notation presented in Section 2.5.3 for the DLin-based Groth-Sahai proof system. We quickly sum up that, when $X \in \mathbb{G}$ (resp. $t \in \mathbb{G}_T$), the notation $\iota(X)$ (resp. $\iota_T(t)$) will be used to denote the vector $(1, 1, X) \in \mathbb{G}^3$ (resp. the 3×3 matrix containing t in position $(3, 3)$ and $1_{\mathbb{G}_T}$ everywhere else). Finally, we also use a symmetric bilinear map $F : \mathbb{G}^3 \times \mathbb{G}^3 \rightarrow \mathbb{G}_T^9$ such that, for any two vectors $\vec{X} = (X_1, X_2, X_3) \in \mathbb{G}^3$ and $\vec{Y} = (Y_1, Y_2, Y_3) \in \mathbb{G}^3$, $F(\vec{X}, \vec{Y}) = \tilde{F}(\vec{X}, \vec{Y})^{1/2} \cdot \tilde{F}(\vec{Y}, \vec{X})^{1/2}$, where the non-commutative mapping $\tilde{F} : \mathbb{G}^3 \times \mathbb{G}^3 \rightarrow \mathbb{G}_T^9$ sends $(\vec{X}, \vec{Y})$ onto the matrix

$\tilde{F}(\vec{X}, \vec{Y})$ of entry-wise pairings (*i.e.*, containing $e(X_i, Y_j)$ in its entry (i, j)).

Keygen(λ): given a security parameter $\lambda \in \mathbb{N}$, chooses a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, g, e)$ of prime order $p > 2^\lambda$.

1. Chooses a Groth-Sahai CRS $\mathbf{f} = (\vec{f}_1, \vec{f}_2, \vec{f}_3)$ for the perfect WI setting. More precisely, chooses $\vec{f}_1 = (f_1, 1, g)$, $\vec{f}_2 = (1, f_2, g)$, and $\vec{f}_3 = \vec{f}_1^{\xi_1} \cdot \vec{f}_2^{\xi_2} \cdot (1, 1, g)^{-1}$, with $f_1, f_2, g \xleftarrow{\$} \mathbb{G}$, $\xi_1, \xi_2 \xleftarrow{\$} \mathbb{Z}_p$.
2. Generates a key pair $(sk_{\text{AHO}}, pk_{\text{AHO}})$ for the AHO signature in order to sign messages consisting of three group elements. This key pair consists of $sk_{\text{AHO}} = (\alpha_a, \alpha_b, \gamma_z, \delta_z, \{\gamma_i, \delta_i\}_{i=1}^3)$ and

$$pk_{\text{AHO}} = \left(G_r, H_r, G_z = G_r^{\gamma_z}, \right. \\ \left. H_z = H_r^{\delta_z}, \{G_i = G_r^{\gamma_i}, H_i = H_r^{\delta_i}\}_{i=1}^3, A, B \right).$$

3. Chooses two uniformly random group elements $\tilde{\theta}_3, \tilde{\theta}_6 \xleftarrow{\$} \mathbb{G}$.

The public key consists of $\mathbf{pk} := ((\mathbb{G}, \mathbb{G}_T), \mathbf{f}, pk_{\text{AHO}})$ whereas the private key is defined to be $\mathbf{sk} = (sk_{\text{AHO}}, (\tilde{\theta}_3, \tilde{\theta}_6))$. The public key defines the set of symbols $\Sigma = \mathbb{G}$.

Sign($\mathbf{sk}, \text{Msg}$): given $\mathbf{sk} = (sk_{\text{AHO}}, (\tilde{\theta}_3, \tilde{\theta}_6))$ and a length- n message $\text{Msg} = (m_1, \dots, m_n) \in \mathbb{G}^n$, for some $n \in \text{poly}(\lambda)$ and where $m_i \in \mathbb{G}$ for each $i \in \{1, \dots, n\}$, performs as follows.

1. Defines $(\theta_{n+1,3}, \theta_{n+1,6}) = (\tilde{\theta}_3, \tilde{\theta}_6)$. Then, for $k \in \{3, 6\}$, computes Groth-Sahai commitments $\vec{C}_{\theta_{n+1,k}} = \iota(\theta_{n+1,k}) \cdot \vec{f}_1^{r_{\theta_{n+1,k}}} \cdot \vec{f}_2^{s_{\theta_{n+1,k}}} \cdot \vec{f}_3^{t_{\theta_{n+1,k}}}$.
2. For each $j = n$ down to 1, generates an AHO signature $(\theta_{j,1}, \dots, \theta_{j,7})$ in $\mathbb{G}^7$ on the message $(m_j, \theta_{j+1,3}, \theta_{j+1,6}) \in \mathbb{G}^3$. For each $k \in \{1, \dots, 7\}$ and $j \in \{1, \dots, n\}$, generates commitments $\vec{C}_{\theta_{j,k}} = \iota(\theta_{j,k}) \cdot \vec{f}_1^{r_{\theta_{j,k}}} \cdot \vec{f}_2^{s_{\theta_{j,k}}} \cdot \vec{f}_3^{t_{\theta_{j,k}}}$. Next, generate NIWI proofs $\vec{\pi}_{\text{AHO},j,1}, \vec{\pi}_{\text{AHO},j,2} \in \mathbb{G}^9$ that variables $(\theta_{j,1}, \theta_{j,2}, \theta_{j,3}, \theta_{j,4}, \theta_{j,5}, \theta_{j,6}, \theta_{j,7})$ satisfy, with $t_j^A = A \cdot e(G_1, m_j)^{-1}$ and $t_j^B = B \cdot e(H_1, m_j)^{-1}$,

$$\begin{aligned} t_j^A &= e(G_z, \theta_{j,1}) \cdot e(G_r, \theta_{j,2}) \cdot e(\theta_{j,3}, \theta_{j,4}) \cdot e(G_2, \theta_{j+1,3}) \cdot e(G_3, \theta_{j+1,6}) \\ t_j^B &= e(H_z, \theta_{j,1}) \cdot e(H_r, \theta_{j,5}) \cdot e(\theta_{j,6}, \theta_{j,7}) \cdot e(H_2, \theta_{j+1,3}) \cdot e(H_3, \theta_{j+1,6}) \end{aligned} \quad (4.5)$$

These equations are quadratic, so that $\{\vec{\pi}_{\text{AHO},j,1}, \vec{\pi}_{\text{AHO},j,2}\}_{j=1}^n$ consists of 9 group elements each.

3. Finally returns the signature

$$\sigma = \left(\{\vec{C}_{\theta_{n+1,k}}\}_{k \in \{3,6\}}, \{\{\vec{C}_{\theta_{j,k}}\}_{k=1}^7, \vec{\pi}_{\text{AHO},j,1}, \vec{\pi}_{\text{AHO},j,2}\}_{j=1}^n \right). \quad (4.6)$$

SignDerive(pk, Msg, Msg', σ): given the public key pk and two messages Msg = $(m_1, \dots, m_n) \in \mathbb{G}^n$ and Msg' = $(m'_1, \dots, m'_{n'}) \in \mathbb{G}^{n'}$, returns $\perp$ if Msg' is not a substring of Msg. Otherwise, there exists an index $i \in \{1, \dots, n - n' + 1\}$ such that $\text{Msg}' = (m'_1, \dots, m'_{n'}) = (m_i, \dots, m_{i+n'-1})$. Then, it parses σ as in (4.6) and, for each $i \in \{1, \dots, n'\}$, conducts the following steps.

1. Lets $\tilde{\sigma} = (\{\vec{C}_{\theta_{i+n',k}}\}_{k \in \{3,6\}}, \{\{\vec{C}_{\theta_{i+j,k}}\}_{k=1}^7, \vec{\pi}_{\text{AHO},i+j,1}, \vec{\pi}_{\text{AHO},i+j,2}\}_{j=0}^{n'-1})$ be the sub-signature.
2. Re-randomizes the commitments $\vec{C}'_{\theta_{i+j,k}} = \vec{C}_{\theta_{i+j,k}} \cdot \vec{f}_1^{r'_{\theta_{i+j,k}}} \cdot \vec{f}_2^{s'_{\theta_{i+j,k}}} \cdot \vec{f}_3^{t'_{\theta_{i+j,k}}}$ for $j = 0$ to $n' - 1$ and $k = 1$ to 7. Likewise, computes re-randomized versions $\{\vec{C}'_{\theta_{i+n',k}}\}_{k \in \{3,6\}}$ of $\{\vec{C}_{\theta_{i+n',k}}\}_{k \in \{3,6\}}$. Finally, re-randomizes the proofs $\{\vec{\pi}_{\text{AHO},i+j,1} = (\vec{\pi}_{i+j,1}, \vec{\pi}_{i+j,2}, \vec{\pi}_{i+j,3})\}_{j=0}^{n'-1}$ and $\{\vec{\pi}_{\text{AHO},i+j,2} = (\vec{\pi}_{i+j,4}, \vec{\pi}_{i+j,5}, \vec{\pi}_{i+j,6})\}_{j=0}^{n'-1}$ as suggested in [13] (see equation 2.20).
3. The signature

$$\sigma' = \left(\{\vec{C}'_{\theta_{i+n',k}}\}_{k \in \{3,6\}}, \{\{\vec{C}'_{\theta_{i+j,k}}\}_{k=1}^7, \vec{\pi}'_{\text{AHO},i+j,1}, \vec{\pi}'_{\text{AHO},i+j,2}\}_{j=0}^{n'-1} \right). \quad (4.7)$$

is returned.

Verify(pk, Msg, σ): given pk, a signature σ and a message Msg = $(m_1, \dots, m_n)$ in $\mathbb{G}^n$, parses σ as per (4.6) and performs as follows. For $j = 1$ to n , returns 0 if $\vec{\pi}_{\text{AHO},j,1} = (\vec{\pi}_{j,1}, \vec{\pi}_{j,2}, \vec{\pi}_{j,3})$ and $\vec{\pi}_{\text{AHO},j,2} = (\vec{\pi}_{j,4}, \vec{\pi}_{j,5}, \vec{\pi}_{j,6})$ do not satisfy the equations below where $t_j^A = A \cdot e(G_1, m_j)^{-1}$ and $t_j^B = B \cdot e(H_1, m_j)^{-1}$. Otherwise, it returns 1.

$$\begin{aligned} \iota_T(t_j^A) &= F(\iota(G_z), \vec{C}_{\theta_{j,1}}) \cdot F(\iota(G_r), \vec{C}_{\theta_{j,2}}) \cdot F(\vec{C}_{\theta_{j,3}}, \vec{C}_{\theta_{j,4}}) \\ &\quad \cdot F(\iota(G_2), \vec{C}_{\theta_{j+1,3}}) \cdot F(\iota(G_3), \vec{C}_{\theta_{j+1,6}}) \cdot \prod_{k=1}^3 F(\vec{\pi}_{j,k}, \vec{f}_k) \quad (4.8) \\ \iota_T(t_j^B) &= F(\iota(H_z), \vec{C}_{\theta_{j,1}}) \cdot F(\iota(H_r), \vec{C}_{\theta_{j,5}}) \cdot F(\vec{C}_{\theta_{j,6}}, \vec{C}_{\theta_{j,7}}) \\ &\quad \cdot F(\iota(H_2), \vec{C}_{\theta_{j+1,3}}) \cdot F(\iota(H_3), \vec{C}_{\theta_{j+1,6}}) \cdot \prod_{k=1}^3 F(\vec{\pi}_{j,k+1}, \vec{f}_k). \end{aligned}$$

Unlike the scheme of Ahn *et al.* [6], the above system allows signing arbitrarily long messages with a public key of constant size whereas theirs requires setting a logarithmic bound on the length of signed messages at key generation. The signature length is asymptotically optimal: a n -symbol message can be signed using $39n + 6$ group elements.

On the other hand, we lose a useful feature of the construction in Ahn *et al.* which allows the derivation algorithm to produce two kinds of derived signatures: when the message m' consists of ℓ symbols, Type I signatures contain $O(\ell \log \ell)$ group elements and support subsequent quoting. Alternatively, the

quoting algorithm can derive a much shorter Type II signature which comprises $O(\log \ell)$ elements, but cannot be quoted any further. In our scheme, the quoter can only produce Type I signatures and does not have the same flexibility as in [6]. It would be interesting to retain this feature while keeping linear-size Type I signatures.

4.3.3 Security of our Substring Predicate Signatures

We now turn to the security of the scheme and first observe that it is clearly completely context-hiding due to the use of a witness-indistinguishable Groth-Sahai CRS.

Theorem 14 *The above quotable signature scheme is completely context-hiding.*

Sketch of proof. Each signature only consists of perfectly hiding commitments and perfectly NIWI arguments which can be perfectly re-randomized at each derivation. $\square$

The unforgeability relies on the DLIN assumption and the security properties of AHO signatures, as established by Theorem 15.

Theorem 15 *The scheme is existentially unforgeable against chosen-message attacks under the $(q \cdot L + 1)$ -SFP and DLIN assumptions, where q denotes the maximal number of signing queries and L is the maximal number of symbols per signing query.*

Proof. Since the scheme is completely context-hiding, we only need to prove unforgeability using the simpler definition where the adversary $\mathcal{A}$ only has a signing oracle. The proof uses a sequence of games where, for each i , S_i stands for the event that $\mathcal{A}$ produces a valid forgery in Game_i .

Game₀: This game is the real one. We denote by S_0 the event that the adversary $\mathcal{A}$ manages to output a successful forgery. Obviously, $\mathcal{A}$'s advantage is $\Pr[S_0]$.

Game₁: We change the generation of the public key and set up $\mathbf{f} = (\vec{f}_1, \vec{f}_2, \vec{f}_3)$ as a perfectly sound Groth-Sahai CRS. Concretely, the only change is that the challenger $\mathcal{B}$ chooses $\vec{f}_3$ in the span of $\vec{f}_1 = (f_1, 1, g)$ and $\vec{f}_2 = (1, f_2, g)$, where $f_1 = g^{\phi_1}$ and $f_2 = g^{\phi_2}$, for random chosen $\phi_1, \phi_2 \xleftarrow{\$} \mathbb{Z}_p$. Signing queries are answered as in Game_0 , using the private key $(sk_{\text{AHO}}, (\tilde{\theta}_3, \tilde{\theta}_6))$ and generating NIWI arguments faithfully. Under the DLIN assumption, this change should not significantly affect $\mathcal{A}$'s behaviour and we have $|\Pr[S_1] - \Pr[S_0]| \leq \text{Adv}^{\text{DLIN}}(\mathcal{B})$. Note that the reduction is immediate as $\mathcal{B}$ does not need the trapdoor (ϕ_1, ϕ_2) at any time. In Game_1 , perfectly hiding Groth-Sahai commitments (and NIWI arguments) are traded for perfectly binding commitments (and perfectly sound proofs).

Game₂: This game is identical to Game 1 except that we bring a conceptual change in the generation of sk . Instead of merely choosing $(\tilde{\theta}_3, \tilde{\theta}_6)$ at random, the challenger $\mathcal{B}$ picks a uniformly random group element $\tilde{m} \xleftarrow{\$} \mathbb{G}$ and computes an AHO signature $\{\tilde{\theta}_k\}_{k=1}^7$ on the “dummy” message $(\tilde{m}, 1, 1)$. The resulting $(\tilde{\theta}_3, \tilde{\theta}_6)$ are included in the private key sk whereas $\tilde{m}$ and $\{\tilde{\theta}_k\}_{k \in \{1,2,4,5,7\}}$ are retained by $\mathcal{B}$. We argue that this change does not alter $\mathcal{A}$ ’s view whatsoever since $(\tilde{\theta}_3, \tilde{\theta}_6)$ have the same distribution either way. Indeed, in Game₂, they remain uniformly distributed in $\mathbb{G}^2$ and statistically independent of the message $\tilde{m}$ and other signature components. We have $\Pr[S_2] = \Pr[S_1]$.

In Game₂, $\mathcal{B}$ uses the values $(\phi_1, \phi_2) = (\log_g(f_1), \log_g(f_2))$ that were defined in Game₁. When $\mathcal{A}$ outputs a forgery σ^* on a message $(m_1^*, \dots, m_{n^*}^*)$, $\mathcal{B}$ uses (ϕ_1, ϕ_2) to extract $(\theta_{n^*+1,3}^*, \theta_{n^*+1,6}^*)$ as well as a sequence of AHO signatures $\{\sigma_j^* = (\theta_{j,1}^*, \dots, \theta_{j,7}^*)\}_{j=1}^{n^*}$ from the Groth-Sahai commitments contained in σ^* . The perfect soundness of $\{\tilde{\pi}_{\text{AHO},j,1}^*, \tilde{\pi}_{\text{AHO},j,2}^*\}_{j=1}^{n^*}$ guarantees that extracted values $(m_1^*, \dots, m_{n^*}^*)$, $\{\sigma_j^*\}_{j=1}^{n^*}$ and $(\theta_{n^*+1,3}^*, \theta_{n^*+1,6}^*)$ satisfy equations (4.5).

In Game₂, we can prove that event S_2 occurs with negligible probability if the $(q \cdot L + 1)$ -SFP assumption holds. Indeed, if $\mathcal{A}$ is successful in Game₃, $\{\sigma_j^* = (\theta_{j,1}^*, \dots, \theta_{j,7}^*)\}_{j=1}^{n^*}$ is a sequence of valid AHO signatures on the messages $\{(m_j^*, \theta_{j+1,3}^*, \theta_{j+1,6}^*)\}_{j=1}^{n^*}$ but $(m_1^*, \dots, m_{n^*}^*)$ is not a subsequence involved in any of the signing queries. We can thus distinguish two situations.

Case A. There exists $j^\dagger \in \{1, \dots, n\}$ such that $\mathcal{B}$ never had to sign $(m_{j^\dagger}^*, \theta_{j^\dagger+1,3}^*, \theta_{j^\dagger+1,6}^*)$ in any signing query.

Case B. The messages $\{(m_j^*, \theta_{j+1,3}^*, \theta_{j+1,6}^*)\}_{j=1}^{n^*}$ were all signed by $\mathcal{B}$ at some point of the game but not all of them were involved in the same query. This covers the case of an adversary mixing substrings of two different messages for which it received signatures.

In Case A, it is immediate that $\mathcal{A}$ necessarily broke the chosen-message security of the AHO signature: the reduction $\mathcal{B}$ simply outputs $(m_{j^\dagger}^*, \theta_{j^\dagger+1,3}^*, \theta_{j^\dagger+1,6}^*)$ and the signature $\sigma_{j^\dagger}^*$.

We are thus left with Case B for which we know that $(m_1^*, \theta_{2,3}^*, \theta_{2,6}^*)$ was involved in the κ -th signing query $\text{Msg}_\kappa = (m_{\kappa,1}, \dots, m_{\kappa,n_\kappa})$, for some integers $\kappa \in \{1, \dots, q\}$ and $n_\kappa \in \{1, \dots, L\}$. Let $\{(\theta_{\kappa,j,1}, \dots, \theta_{\kappa,j,7})\}_{j=1}^{n_\kappa}$ be the AHO signatures that were used to answer the κ -th signing query. Let also $t \in \{1, \dots, n_\kappa\}$ be such that $(m_{\kappa,t}, \theta_{\kappa,t+1,3}, \theta_{\kappa,t+1,6}) = (m_1^*, \theta_{2,3}^*, \theta_{2,6}^*)$.

We now define j^* to be the largest index in $\{1, \dots, n^* - 1\}$ such that

$$(m_{\kappa,t+j^*-1}, \theta_{\kappa,t+j^*,3}, \theta_{\kappa,t+j^*,6}) = (m_{j^*}^*, \theta_{j^*+1,3}^*, \theta_{j^*+1,6}^*).$$

At this step, we further consider two sub-cases of Case B:

Case $t + j^* < n_\kappa + 1$: Since $m_{\kappa,t+j^*} \neq m_{j^*+1}$ or $(\theta_{\kappa,t+j^*+1,3}, \theta_{\kappa,t+j^*+1,6}) \neq (\theta_{j^*+2,3}^*, \theta_{j^*+2,6}^*)$, the signature binding property of the AHO signature is

broken since we have two distinct messages whose signatures share the same $\theta_{j^*+1,3}^*, \theta_{j^*+1,6}^*$ components. As implied by the results of [4], this contradicts the $(q \cdot L + 1)$ -SFP assumption since $\mathcal{B}$ computes at most $q \cdot L + 1$ AHO signatures.

Case $t + j^* = n_\kappa + 1$: We have $(\theta_{j^*+1,3}^*, \theta_{j^*+1,6}^*) = (\theta_{\kappa, n_\kappa+1,3}, \theta_{\kappa, n_\kappa+1,6}) = (\tilde{\theta}_3, \tilde{\theta}_6)$, which means that $(m_{j^*}^*, \theta_{j^*+1,3}^*, \theta_{j^*+1,6}^*)$ was the message of an “end-of-chain” signature produced by $\mathcal{B}$. In other words, this is a forgery where $(m_1^*, \dots, m_{n^*}^*)$ is a super-string of $(m_{\kappa,t}, \dots, m_{\kappa, n_\kappa})$. In this case, thanks to the modification introduced in Game₂, $\mathcal{B}$ knows $\{\tilde{\theta}_k\}_{k \in \{1,2,4,5,7\}}$ as well as a dummy message $\tilde{m}$ such that $(\tilde{\theta}_1, \dots, \tilde{\theta}_7)$ is a valid AHO signature on $(\tilde{m}, 1, 1)$. With overwhelming probability, we obtain distinct messages $(\tilde{m}, 1, 1)$ and $(m_{j^*+1}^*, \theta_{j^*+2,3}^*, \theta_{j^*+2,6}^*)$ that share the same signature components $(\theta_{j^*+1,3}^*, \theta_{j^*+1,6}^*) = (\tilde{\theta}_3, \tilde{\theta}_6)$. Indeed, the pair $(\tilde{\theta}_3, \tilde{\theta}_6)$ is statistically independent of the dummy message $\tilde{m}$, which was uniformly chosen in $\mathbb{G}$. It comes that we can only have $m_{j^*+1}^* = \tilde{m}$ by pure chance.

In Case B, the signature binding property of AHO signatures is thus broken either way and we can eventually write $\Pr[S_2] \leq 2 \cdot \mathbf{Adv}^{(q \cdot L + 1)\text{-SFP}}(\mathcal{B})$, where the factor 2 accounts for the fact that the reduction has to guess beforehand which of Case A or Case B will come about. Depending on this guess, $\mathcal{B}$ undertakes to either attack the standard unforgeability of AHO signatures or, alternatively, break their signature-binding property. In either case, $\mathcal{B}$ answers $\mathcal{A}$ ’s queries by invoking the signing oracle in its interaction with the appropriate challenger.

Putting the above altogether, we find the upper bound

$$\Pr[S_0] \leq \mathbf{Adv}^{\text{DLIN}}(\mathcal{B}) + 2 \cdot \mathbf{Adv}^{(q \cdot L + 1)\text{-SFP}}(\mathcal{B})$$

on the forger’s advantage. $\square$

4.4 Linearly Homomorphic Signatures

Linearly homomorphic signatures are an important class of homomorphic signatures for arithmetic functions, whose study was initiated by Boneh, Freeman, Katz and Waters [25]. While initially motivated by applications to network coding [25], they are also useful in proofs of storage [7, 8] or in verifiable computation mechanisms when it comes to authenticate servers’ computations on outsourced data (see, *e.g.*, [6]).

In this Section, we put forth the notion of linearly homomorphic structure-preserving signatures. Of course, linearly homomorphic signatures and structure-preserving signatures have been defined before, but the combination of the earlier notions is useful and non-trivial. These signature schemes function exactly like ordinary homomorphic signatures with the additional restriction that signatures and messages only consist of (vectors of) group elements whose discrete

logarithms may not be available. We describe three constructions and prove their security under established complexity assumptions in symmetric bilinear groups.

As in all SPS systems, the structure-preserving property makes it possible to efficiently prove knowledge of a homomorphic signature on a committed vector. However, in the next parts of the thesis, we describe applications of linearly homomorphic SPS beyond their compatibility with the Groth-Sahai techniques.

In our PKC paper [11], we also suggest a completely context-hiding linearly homomorphic signatures for vectors of scalars. The unforgeability holds if the DLin problem and the flex-DH problem are both hard in $\mathbb{G}$. To shorten the presentation we only consider schemes that lead to structure-preserving signatures for this highest level of unlinkability.

4.4.1 Intuition behind our Subspace Predicate Signatures

Before presenting the intuition of the linearly homomorphic schemes described in the following, we rephrase the security notions of Section 4.1 in the context of linear subspace predicates for pairing-based instantiations.

Let $(\mathbb{G}, \mathbb{G}_T)$ be a bilinear configuration of groups of prime order p over which a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is efficiently computable. We consider linearly homomorphic signatures for which the message space $\mathcal{M}$ consists of pairs $\mathcal{M} := \mathcal{T} \times \mathbb{G}^n$, for some $n \in \mathbb{N}$, where $\mathcal{T}$ is a tag space. The linear subspace predicate is defined by

$$\begin{aligned} P\left((\tau, \vec{M}), \{(\tau_i, \vec{M}_i)\}_{i=1}^\ell\right) &= 1 \\ \iff (\forall i \in \{1, \dots, \ell\} : \tau = \tau_i) \wedge \vec{M} &\in \text{span}\langle \vec{M}_1, \dots, \vec{M}_\ell \rangle. \end{aligned}$$

We remark that, in the applications considered in the thesis, tags do not need to be group elements. We thus allow them to be arbitrary strings.

Definition 33 *A linearly homomorphic structure-preserving signature scheme over $(\mathbb{G}, \mathbb{G}_T)$ consists of a quadruple of efficient algorithms $\Sigma = (\text{Keygen}, \text{Sign}, \text{SignDerive}, \text{Verify})$ for which the message space is $\mathcal{M} := \mathcal{T} \times \mathbb{G}^n$, for some $n \in \text{poly}(\lambda)$ and some set $\mathcal{T}$, and with the following specifications.*

Keygen(λ, n): *is a randomized algorithm that takes in a security parameter $\lambda \in \mathbb{N}$ and an integer $n \in \text{poly}(\lambda)$ denoting the dimension of vectors to be signed. It outputs a key pair (pk, sk) and the description of a tag (i.e., a file identifier) space $\mathcal{T}$.*

Sign($\text{sk}, \tau, \vec{M}$): *is a possibly probabilistic algorithm that takes as input a private key sk , a file identifier $\tau \in \mathcal{T}$ and a vector $\vec{M} \in \mathbb{G}^n$. It outputs a signature $\sigma \in \mathbb{G}^{n_s}$, for some $n_s \in \text{poly}(\lambda)$.*

SignDerive($\mathbf{pk}, \tau, \{(\omega_i, \sigma^{(i)})\}_{i=1}^\ell$): is a (possibly probabilistic) signature derivation algorithm. It takes as input a public key $\mathbf{pk}$, a file identifier τ as well as ℓ pairs $(\omega_i, \sigma^{(i)})$, each of which consists of a weight $\omega_i \in \mathbb{Z}_p$ and a signature $\sigma^{(i)} \in \mathbb{G}^{n_s}$ on $\vec{M}_i$. The output is a signature $\sigma \in \mathbb{G}^{n_s}$ on the vector $\vec{M} = \prod_{i=1}^\ell \vec{M}_i^{\omega_i}$.

Verify($\mathbf{pk}, \tau, \vec{M}, \sigma$): is a deterministic algorithm that takes in a public key $\mathbf{pk}$, a file identifier $\tau \in \mathcal{T}$, a signature σ and a vector $\vec{M}$. It outputs 1 if σ is deemed valid and 0 otherwise.

Correctness is expressed by imposing that, for all security parameters $\lambda \in \mathbb{N}$, all integers $n \in \text{poly}(\lambda)$ and all triples $(\mathbf{pk}, \mathbf{sk}, \mathcal{T}) \leftarrow \text{Keygen}(\lambda, n)$, the following holds:

1. For all $\tau \in \mathcal{T}$ and all n -vectors $\vec{M}$, if $\sigma = \text{Sign}(\mathbf{sk}, \tau, \vec{M})$, then we have $\text{Verify}(\mathbf{pk}, \tau, \vec{M}, \sigma) = 1$.
2. For all $\tau \in \mathcal{T}$, any $\ell > 0$ and any set of triples $\{(\omega_i, \sigma^{(i)}, \vec{M}_i)\}_{i=1}^\ell$, if $\text{Verify}(\mathbf{pk}, \tau, \vec{M}_i, \sigma^{(i)}) = 1$ for each $i \in \{1, \dots, \ell\}$, then

$$\text{Verify}(\mathbf{pk}, \tau, \prod_{i=1}^\ell \vec{M}_i^{\omega_i}, \text{SignDerive}(\mathbf{pk}, \tau, \{(\omega_i, \sigma^{(i)})\}_{i=1}^\ell)) = 1$$

A slight difference with the correctness of Definition 27 is that, in item 2 here, the **SignDerive** algorithm does not only take in signatures output by **Sign**. More generally, it preserves the validity of signatures no matter how they have been generated.

Security. In linearly homomorphic signatures, we use the same definition of unforgeability as in our Asiacrypt full paper [10, Appendix 2]. This definition implies security in the stronger model used by Freeman [69] since the adversary can interleave signing queries for individual vectors belonging to distinct subspaces. Moreover, file identifiers can be chosen by the adversary (which strengthens the definition of [25]) and are not assumed to be uniformly distributed. As a result, a file identifier can be a low-entropy, easy-to-remember string such as the name of the dataset's owner.

Definition 34 A linearly homomorphic SPS scheme $\Sigma = (\text{Keygen}, \text{Sign}, \text{Verify})$ is secure if no PPT adversary has non-negligible advantage in the game below:

1. The adversary $\mathcal{A}$ chooses an integer $n \in \mathbb{N}$ and sends it to the challenger who runs $\text{Keygen}(\lambda, n)$ and obtains $(\mathbf{pk}, \mathbf{sk})$ before sending $\mathbf{pk}$ to $\mathcal{A}$.
2. On polynomially-many occasions, $\mathcal{A}$ can interleave the following kinds of queries.

- *Signing queries:* $\mathcal{A}$ chooses a tag $\tau \in \mathcal{T}$ and a vector $\vec{M} \in \mathbb{G}^n$. The challenger picks a handle h and computes $\sigma \leftarrow \text{Sign}(\text{sk}, \tau, \vec{M})$. It stores $(h, (\tau, \vec{M}, \sigma))$ in a table T and returns h .
 - *Derivation queries:* $\mathcal{A}$ chooses a vector of handles $\vec{h} = (h_1, \dots, h_k)$ and a set of coefficients $\{\omega_i\}_{i=1}^k$. The challenger retrieves the tuples $\{(h_i, (\tau, \vec{M}_i), \sigma^{(i)})\}_{i=1}^k$ from T and returns $\perp$ if one of these does not exist or if there exists $i \in \{1, \dots, k\}$ such that $\tau_i \neq \tau$. Otherwise, $\mathcal{B}$ runs $\sigma' \leftarrow \text{SignDerive}(\text{pk}, \tau, \{(\omega_i, \sigma^{(i)})\}_{i=1}^k)$ and computes $\vec{M}' = \prod_{i=1}^k \vec{M}_i^{\omega_i}$. $\mathcal{B}$ also chooses a handle h' , stores $(h', (\tau, \vec{M}'), \sigma')$ in T and returns h' to $\mathcal{A}$.
 - *Reveal queries:* $\mathcal{A}$ chooses a handle h . If no tuple of the form $(h, (\tau, \vec{M}), \sigma')$ exists in T , the challenger returns $\perp$. Otherwise, $\mathcal{B}$ returns σ' to $\mathcal{A}$ and adds $((\tau, \vec{M}), \sigma')$ to the set Q .
3. $\mathcal{A}$ outputs an identifier τ^* , a signature σ^* and a vector $\vec{M}^* \in \mathbb{G}^n$. The adversary $\mathcal{A}$ wins if $\text{Verify}(\text{pk}, \tau^*, \vec{M}^*, \sigma^*) = 1$ and one of the conditions below is satisfied:

- (Type I): $\tau^* \neq \tau_i$ for any entry $(\tau_i, \cdot)$ in Q and $\vec{M}^* \neq (1_{\mathbb{G}}, \dots, 1_{\mathbb{G}})$.
- (Type II): $\tau^* = \tau_i$ for $k_i > 0$ entries $(\tau_i, \cdot)$ in Q and $\vec{M}^* \notin V_i$, where V_i denotes the subspace spanned by all vectors $\vec{M}_1, \dots, \vec{M}_{k_i}$ for which an entry of the form $(\tau^*, \vec{M}_j)$, with $j \in \{1, \dots, k_i\}$, appears in Q .

$\mathcal{A}$'s advantage is its probability of success taken over all coin tosses.

In our first scheme, we will consider a weaker notion of *one-time* security. In this notion, the adversary is limited to obtain signatures for only *one* linear subspace. In this case, there is no need for file identifiers and we assume that all vectors are assigned the identifier $\tau = \varepsilon$.

In the following, the adversary will be said **independent** if

- For any given tag τ , it is restricted to only query signatures on linearly independent vectors.
- Each vector is only queried at most once.

Non-independent adversaries are not subject to the above restrictions. It will be necessary to consider these adversaries in our construction of non-malleable commitments in Chapter 5. Nevertheless, security against independent adversaries suffices for many applications – including encrypted cloud storage – since the signer can always append unit vectors to each newly signed vector.

At first, one may wonder how Definition 34 can be satisfied at all given that the challenger may not have an efficient way to check whether the adversary is

successful. Indeed, in cryptographically useful discrete-logarithm-hard groups $\mathbb{G}$, deciding whether vectors $\{\vec{M}_i\}_i$ of $\mathbb{G}^n$ are linearly dependent is believed to be difficult when $n > 2$. However, it may be possible using some trapdoor information embedded in $\mathbf{pk}$, especially if the adversary additionally outputs signatures on $\{\vec{M}_i\}_i$.

In some applications, it makes sense to consider a weaker attack model where, in the case of Type II attacks, the adversary is only deemed successful if it can output a convincing proof that its target vector $\vec{M}^*$ is indeed independent of the vectors that were signed for the tag τ^* . The proof can be either a NIZK proof or, alternatively, a vector in the kernel of the matrix whose rows are the vectors that were signed for τ^* . We call such an adversary a **targeting** adversary.

Intuition of our Constructions

Our first scheme's starting point is the one-time (regular) SPS scheme of Abe *et al.* [4]. By removing certain public key components, we obtain the desired linear homomorphism, and prove the security using information-theoretic arguments as in [4, Appendix C]. The key observation here is that, as long as the adversary does not output a signature on a linear combination of previously signed vectors, it will be unable to sign its target vector in the same way as the reduction would, because certain private key components will remain perfectly hidden.

Our initial scheme inherits the one-time restriction of the Abe *et al.* scheme in that only one linear subspace can be safely signed with a given public key. Nevertheless, we can extend it to build a full linearly homomorphic SPS system. To this end, we suitably combine our first scheme with Waters signatures (see Section 2.2.2). Here, Waters signatures are used as a resting ground for fresh random exponents which are introduced in each signed vector and help us refresh the state of the system and apply each time the same argument as in the one-time scheme. While this “full-fledged” linearly homomorphic scheme is enough in several situations it does not reach sufficient context hiding security for others. Then, we also present techniques to turn the scheme into a fully randomizable one, where a derived signature has the same distribution as a directly signed message. However, its security stands against independent targeting adversaries.

4.4.2 A One-Time Linearly Homomorphic Construction

As a warm-up, we begin by describing a one-time homomorphic signature, where a given public key allows signing only *one* linear subspace. In the description hereunder, since only one linear subspace can be signed, no file identifier τ is used. We thus set τ to be the empty string ε in all algorithms.

Keygen(λ, n): given a security parameter λ and the dimension $n \in \mathbb{N}$ of the subspace to be signed, chooses bilinear group $(\mathbb{G}, \mathbb{G}_T)$ of prime order

$p > 2^\lambda$. Then, chooses generators $h, g_z, g_r, h_z \xleftarrow{\$} \mathbb{G}$. Pick $\chi_i, \gamma_i, \delta_i \xleftarrow{\$} \mathbb{Z}_p$, for $i = 1$ to n . Then, for each $i \in \{1, \dots, n\}$, computes $g_i = g_z^{\chi_i} g_r^{\gamma_i}$, $h_i = h_z^{\chi_i} h^{\delta_i}$. The private key is $\text{sk} = \{\chi_i, \gamma_i, \delta_i\}_{i=1}^n$ while the public key is defined to be

$$\text{pk} = (g_z, h_r, h_z, h, \{g_i, h_i\}_{i=1}^n) \in \mathbb{G}^{2n+4}.$$

Sign($\text{sk}, \tau, (M_1, \dots, M_n)$): to sign a vector $(M_1, \dots, M_n) \in \mathbb{G}^n$ associated with the identifier $\tau = \varepsilon$ using $\text{sk} = \{\chi_i, \gamma_i, \delta_i\}_{i=1}^n$, computing the signature consists of $\sigma = (z, r, u) \in \mathbb{G}^3$, where

$$z = \prod_{i=1}^n M_i^{-\chi_i}, \quad r = \prod_{i=1}^n M_i^{-\gamma_i}, \quad u = \prod_{i=1}^n M_i^{-\delta_i}.$$

SignDerive($\text{pk}, \tau, \{(\omega_i, \sigma^{(i)})\}_{i=1}^\ell$): given the public key pk , a file identifier $\tau = \varepsilon$ and ℓ tuples $(\omega_i, \sigma^{(i)})$, parses each signature $\sigma^{(i)}$ as $\sigma^{(i)} = (z_i, r_i, u_i) \in \mathbb{G}^3$ for $i = 1$ to ℓ . Computes and returns the derived signature $\sigma = (z, r, u) = (\prod_{i=1}^\ell z_i^{\omega_i}, \prod_{i=1}^\ell r_i^{\omega_i}, \prod_{i=1}^\ell u_i^{\omega_i})$.

Verify($\text{pk}, \sigma, \tau, (M_1, \dots, M_n)$): given a signature $\sigma = (z, r, u) \in \mathbb{G}^3$, a vector $(M_1, \dots, M_n)$ and a file identifier $\tau = \varepsilon$, returns 1 if and only if $(M_1, \dots, M_n) \neq (1_{\mathbb{G}}, \dots, 1_{\mathbb{G}})$ and (z, r, u) satisfy

$$\begin{aligned} 1_{\mathbb{G}_T} &= e(g_z, z) \cdot e(g_r, r) \cdot \prod_{i=1}^n e(g_i, M_i), \\ 1_{\mathbb{G}_T} &= e(h_z, z) \cdot e(h, u) \cdot \prod_{i=1}^n e(h_i, M_i). \end{aligned} \tag{4.9}$$

The proof of security relies on the fact that, while the signing algorithm is deterministic, signatures are not unique. However, the reduction will be able to compute exactly one signature for each vector. At the same time, an adversary has no information about which specific signature the legitimate signer would compute on a vector outside the span of already signed vectors. Moreover, by obtaining two distinct signatures on a given vector, the reduction can solve a given SDP instance.

Theorem 16 *The scheme is unforgeable if the SDP assumption holds in $(\mathbb{G}, \mathbb{G}_T)$.*

Proof. We describe an algorithm $\mathcal{B}$ that takes as input a SDP instance $(g_z, g_r, h_z, h) \in \mathbb{G}^4$ and uses a forger $\mathcal{A}$ to find a triple (z, r, u) such that $e(g_z, z) \cdot e(g_r, r) = e(h_z, z) \cdot e(h, u) = 1_{\mathbb{G}_T}$.

To this end, $\mathcal{B}$ honestly runs the key generation algorithm using randomly chosen $\{(\chi_i, \gamma_i, \delta_i)\}_{i=1}^n$. Whenever $\mathcal{A}$ requests a signature on a vector $(M_1, \dots, M_n) \in \mathbb{G}^n$, $\mathcal{B}$ faithfully follows the specification of the signing algorithm. The

game ends with the adversary $\mathcal{A}$ outputting a vector $(M_1^*, \dots, M_n^*)$ with a valid signature (z^*, r^*, u^*) . At this point, $\mathcal{B}$ computes its own signature

$$(z^\dagger, r^\dagger, u^\dagger) = \left(\prod_{i=1}^n M_i^{*- \chi_i}, \prod_{i=1}^n M_i^{*- \gamma_i}, \prod_{i=1}^n M_i^{*- \delta_i} \right) \quad (4.10)$$

on $(M_1^*, \dots, M_n^*)$. We claim that, with overwhelming probability, $(z^\dagger, r^\dagger, u^\dagger) = (z^*/z^\dagger, r^*/r^\dagger, u^*/u^\dagger)$ is a non-trivial solution to the SDP instance.

To see this, we first note that a given public key has exponentially many corresponding private keys and $\mathbf{pk}$ perfectly hides the vector $(\chi_1, \dots, \chi_n)$. Moreover, for a given $\mathbf{pk}$, each message $(M_1, \dots, M_n)$ has an exponential number of valid signatures but the one produced by the signing algorithm is completely determined by $(\chi_1, \dots, \chi_n)$. We will see that, in $\mathcal{A}$'s view, guessing the value $z^\dagger$ of (4.10) amounts to inferring which vector $(\chi_1, \dots, \chi_n)$ the reduction $\mathcal{B}$ is using.

Throughout the game, $\mathcal{A}$ obtains signatures $\{(z_i, r_i, u_i)\}_{i=1}^{n-1}$ on at most $n-1$ linearly independent vectors of $\mathbb{G}^n$. If we consider discrete logarithms, these signatures only provide $\mathcal{A}$ with $n-1$ linearly independent equations because, for each triple (z_i, r_i, u_i) , z_i uniquely determines (r_i, u_i) . Taking into account the information revealed by $\{(g_i, h_i)\}_{i=1}^n$, we find that an unbounded adversary is presented with $3n-1$ linear equations in $3n$ unknowns. In $\mathcal{A}$'s view, since $(M_1^*, \dots, M_n^*)$ must be independent of previously signed vectors, predicting $z^\dagger$ is only possible with probability $1/p$. With probability $1-1/p$, we thus have $z^\dagger \neq z^*$, in which case $(z^\dagger, r^\dagger, u^\dagger)$ solves the SDP instance because $(z^\dagger, r^\dagger, u^\dagger)$ and (z^*, r^*, u^*) both satisfy the verification equations. $\square$

The scheme can be modified so as to work in asymmetric pairing configurations and the Double Pairing assumption (see DBP in Section 2.1.2). However, we need to work with the SDP assumption in the next section.

4.4.3 A Full-Fledged Construction for Linear Subspaces

Here, we upgrade our one-time construction to obtain a scheme allowing us to sign an arbitrary number of linear subspaces. Here, each file identifier τ consists of a L -bit string. The construction builds on the observation that, in the scheme of Section 4.4.2, signatures (z, r, u) could be re-randomized by computing $(z \cdot g_r^\theta, r \cdot g_z^{-\theta}, u \cdot h_z^{-\log_h(g_r) \cdot \theta})$, with $\theta \xleftarrow{\$} \mathbb{Z}_p$, if $h_z^{-\log_h(g_r)}$ were available. Since publicizing $h_z^{-\log_h(g_r)}$ would render the scheme insecure, our idea is to use Waters signatures as a support to introduce extra randomizers in the exponent.

In the construction, the u component of each signature can be seen as an aggregation of the one-time signature of Section 4.4.2 with a Waters signature $(h_z^{\log_h(g_r)} \cdot H_{\mathbb{G}}(\tau)^{-\rho}, h^\rho)$ on the tag τ .

Keygen(λ, n): given a security parameter λ and the dimension $n \in \mathbb{N}$ of the

subspace to be signed, chooses bilinear group $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$. Then, conducts the following steps.

1. Chooses $h \xleftarrow{\$} \mathbb{G}$ and $\alpha_z, \alpha_r, \beta_z \xleftarrow{\$} \mathbb{Z}_p$. Defines $g_z = h^{\alpha_z}$, $g_r = h^{\alpha_r}$ and $h_z = h^{\beta_z}$.
2. For $i = 1$ to n , picks $\chi_i, \gamma_i, \delta_i \xleftarrow{\$} \mathbb{Z}_p$ and computes the group elements $g_i = g_z^{\chi_i} g_r^{\gamma_i}$, $h_i = h_z^{\delta_i}$.
3. Chooses a random vector $\bar{\mathbf{w}} = (w_0, w_1, \dots, w_L) \xleftarrow{\$} \mathbb{G}^{L+1}$. The latter defines a hash function $H_{\mathbb{G}} : \{0, 1\}^L \rightarrow \mathbb{G}$ which maps file identifier $\tau = \tau[1] \dots \tau[L] \in \{0, 1\}^L$ to $H_{\mathbb{G}}(\tau) = w_0 \cdot \prod_{k=1}^L w_k^{\tau[k]}$.

The private key is $\mathbf{sk} = (h_z^{\alpha_r}, \{\chi_i, \gamma_i, \delta_i\}_{i=1}^n)$ while the public key consists of

$$\mathbf{pk} = (g_z, g_r, h_z, h, \{g_i, h_i\}_{i=1}^n, \bar{\mathbf{w}}) \in \mathbb{G}^{2n+4} \times \mathbb{G}^{L+1}.$$

Sign($\mathbf{sk}, \tau, (M_1, \dots, M_n)$): to sign a vector $(M_1, \dots, M_n) \in \mathbb{G}^n$ w.r.t. the file identifier τ using $sk = (h_z^{\alpha_r}, \{\chi_i, \gamma_i, \delta_i\}_{i=1}^n)$, chooses $\theta, \rho \xleftarrow{\$} \mathbb{Z}_p$ and outputs $\sigma = (z, r, u, v) \in \mathbb{G}^4$, where

$$\begin{aligned} z &= g_r^\theta \cdot \prod_{i=1}^n M_i^{-\chi_i} & r &= g_z^{-\theta} \cdot \prod_{i=1}^n M_i^{-\gamma_i} \\ u &= (h_z^{\alpha_r})^{-\theta} \cdot \prod_{i=1}^n M_i^{-\delta_i} \cdot H_{\mathbb{G}}(\tau)^{-\rho} & v &= h^\rho \end{aligned}$$

SignDerive($\mathbf{pk}, \tau, \{(\omega_i, \sigma^{(i)})\}_{i=1}^\ell$): given $\mathbf{pk}$, a file identifier τ and ℓ tuples $(\omega_i, \sigma^{(i)})$, parses $\sigma^{(i)}$ as $\sigma^{(i)} = (z_i, r_i, u_i, v_i) \in \mathbb{G}^4$ for $i = 1$ to ℓ . Then, chooses a random $\rho' \xleftarrow{\$} \mathbb{Z}_p$ and computes and returns $\sigma = (z, r, u, v)$, where $z = \prod_{i=1}^\ell z_i^{\omega_i}$, $r = \prod_{i=1}^\ell r_i^{\omega_i}$, $u = H_{\mathbb{G}}(\tau)^{-\rho'} \cdot \prod_{i=1}^\ell u_i^{\omega_i}$ and $v = h^{\rho'} \cdot \prod_{i=1}^\ell v_i^{\omega_i}$.

Verify($\mathbf{pk}, \sigma, \tau, (M_1, \dots, M_n)$): given a signature $\sigma = (z, r, u, v) \in \mathbb{G}^4$, a file identifier τ and a vector of group elements $(M_1, \dots, M_n)$, returns 1 if and only if $(M_1, \dots, M_n) \neq (1_{\mathbb{G}}, \dots, 1_{\mathbb{G}})$ and (z, r, u, v) satisfy

$$\begin{aligned} 1_{\mathbb{G}_T} &= e(g_z, z) \cdot e(g_r, r) \cdot \prod_{i=1}^n e(g_i, M_i), \\ 1_{\mathbb{G}_T} &= e(h_z, z) \cdot e(h, u) \cdot e(H_{\mathbb{G}}(\tau), v) \cdot \prod_{i=1}^n e(h_i, M_i). \end{aligned} \tag{4.11}$$

The security of the scheme against *non-independent* Type I adversaries is proved under the SDP assumption. In the case of Type II forgeries, we need to assume the adversary to be independent because, at some point, the simulator is only able to compute a signature for a unique value³ of θ .

³Note that this is not a problem since the signer can derive θ as a pseudorandom function of τ and $(M_1, \dots, M_n)$ to make sure that a given vector is always signed using the same θ .

Theorem 17 *The scheme is unforgeable against independent adversaries if the SDP assumption holds in $(\mathbb{G}, \mathbb{G}_T)$. Moreover, the scheme is secure against non-independent Type I adversaries.*

Proof. The result is proved by separately considering Type I and Type II forgeries. For simplicity, we first consider Type II adversaries as the case of Type I attacks will be simpler. Lemmas 6 and 7 show how to build an algorithm solving the SDP problem either way. $\square$

The proof of Lemma 6 uses Waters signatures as a handle to randomize signatures. Specifically, whenever the reduction is able to compute a Waters signatures $(h_z^{\alpha_r} \cdot H_{\mathbb{G}}(\tau)^{-\rho}, h^\rho)$ on the tag τ , it can inject a fresh extra randomizer $\theta \in \mathbb{Z}_p$ in the exponent for each vector associated with τ . By doing so, with non-negligible probability, the specific vector $(\chi_1, \dots, \chi_n)$ used by the reduction will remain completely undetermined from $\mathcal{A}$'s view.

Lemma 6 *For any Type II independent forger $\mathcal{A}$, there exists an algorithm $\mathcal{B}$ solving the SDP problem such that $\mathbf{Adv}(\mathcal{A}) \leq 8 \cdot q \cdot (L+1) \cdot (\mathbf{Adv}^{\text{SDP}}(\mathcal{B}) + \frac{1}{p})$, where q is the number of distinct tags appearing in signing queries.*

Proof. Let us assume that an independent adversary $\mathcal{A}$ can produce a Type II forgery with non-negligible advantage ε . Using $\mathcal{A}$, we build an algorithm $\mathcal{B}$ solving a SDP instance (g_z, g_r, h_z, h) with probability at least $\varepsilon / (8(q-1)(L+1))$. Algorithm $\mathcal{B}$ chooses $(w_0, w_1, \dots, w_L) \in \mathbb{G}^{L+1}$ in the same way as in the security proof of Waters signatures [138]. Namely, for any string $\tau \in \{0, 1\}^L$, the hash value $H_{\mathbb{G}}(\tau) = w_0 \cdot \prod_{i=1}^L w_i^{\tau[i]}$ can be expressed as $H_{\mathbb{G}}(\tau) = g_r^{J(\tau)} \cdot h^{K(\tau)}$ for certain integer-valued functions $J, K : \{0, 1\}^L \rightarrow \mathbb{Z}_p$ that remain internal to the simulation. They are further defined using the methodology of programmable hash functions [88] so that, for any distinct $\tau, \tau_1, \dots, \tau_q$, we have $J(\tau) = 0 \pmod p$ and $J(\tau_i) \neq 0 \pmod p$ for each $i \in \{1, \dots, q\}$ with non-negligible probability $\zeta = 1 / (8 \cdot q \cdot (L+1))$.

The remaining public key components are defined by setting $g_i = g_z^{\chi_i} g_r^{\gamma_i}$ and $h_i = h_z^{\chi_i} h^{\delta_i}$, with $\chi_i, \gamma_i, \delta_i \xleftarrow{\$} \mathbb{Z}_p$ for $i = 1$ to n , as in the real key generation algorithm.

Since $\mathcal{A}$ is a Type II forger, it is expected to produce a forgery $(\tau^*, \vec{M}^*, \sigma^*)$ for a tag τ^* that was used by $\mathcal{B}$ in some signing query but for which $\vec{M}^* \notin \text{span}(\vec{M}_1, \dots, \vec{M}_{n-1})$, where $\vec{M}_1, \dots, \vec{M}_{n-1}$ are the vectors of $\mathbb{G}^n$ that were associated with τ^* . We denote by $\tau_1, \dots, \tau_q$ the distinct adversarially-chosen tags involved in $\mathcal{A}$'s queries during the game. Note that, since $\mathcal{A}$ is a Type II adversary, we will have $\tau^* \in \{\tau_1, \dots, \tau_q\}$ at the end of the game. We also assume w.l.o.g. that exactly $n-1$ signing queries are made for each tag $\tau \in \{\tau_1, \dots, \tau_q\}$ during the game (otherwise, $\mathcal{B}$ can simulate signing queries for itself). During its interaction with $\mathcal{A}$, the reduction $\mathcal{B}$ answers Sign, SignDerive and Reveal queries as follows.

Signing queries: At each signing query $(\tau_j, \vec{M} = (M_1, \dots, M_n))$ involving the j -th distinct tag τ_j , $\mathcal{B}$ evaluates the function $J(\tau_j)$ and considers the following situations.

- If $J(\tau_j) \neq 0$, $\mathcal{B}$ picks $\rho, \theta \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\Theta_1 = H_{\mathbb{G}}(\tau_j)^{-\rho} \cdot (h_z)^{\frac{K(\tau_j)}{J(\tau_j)} \cdot \theta}, \quad \Theta_2 = h^\rho \cdot (h_z)^{\frac{-\theta}{J(\tau_j)}},$$

which can be written $(\Theta_1, \Theta_2) = (h_z^{-\theta \cdot \alpha_r} \cdot H_{\mathbb{G}}(\tau)^{-\tilde{\rho}}, h^{\tilde{\rho}})$ if we define $\tilde{\rho} = \rho - \frac{\theta \cdot \beta_z}{J(\tau_j)}$. Using (Θ_1, Θ_2) , $\mathcal{B}$ obtains a valid signature on the vector $(M_1, \dots, M_n)$ by computing $v = \Theta_2$ and

$$z = g_r^\theta \cdot \prod_{i=1}^n M_i^{-\chi_i}, \quad r = g_z^{-\theta} \cdot \prod_{i=1}^n M_i^{-\gamma_i}, \quad u = \Theta_1 \cdot \prod_{i=1}^n M_i^{-\delta_i}.$$

The signature $\sigma = (z, r, u, v)$ is not directly sent to $\mathcal{A}$ but assigned to a new handle h and stored in an entry $(h, (\tau_j, \vec{M}), \sigma)$ of the table T .

- If $J(\tau_j) = 0$, $\mathcal{B}$ picks $\rho \xleftarrow{\$} \mathbb{Z}_p$ and computes $v = h^\rho$ and

$$z = \prod_{i=1}^n M_i^{-\chi_i}, \quad r = \prod_{i=1}^n M_i^{-\gamma_i}, \quad u = H_{\mathbb{G}}(\tau_j)^{-\rho} \cdot \prod_{i=1}^n M_i^{-\delta_i},$$

which corresponds to a valid signature (z, r, u, v) on $(M_1, \dots, M_n)$ for which $\theta = 1$. Again, the challenger $\mathcal{B}$ chooses a handle h and stores $(h, (\tau, \vec{M}), (z, r, u, v))$ in the table T .

Derivation queries: Whenever $\mathcal{A}$ queries $((h_1, \dots, h_k), (\tau, \vec{M}'), \{\beta_i\}_{i=1}^k)$ to the SignDerive oracle, $\mathcal{B}$ returns $\perp$ if not all handles $h_1, \dots, h_k$ correspond to queries involving τ . Otherwise, let $\vec{M}_1, \dots, \vec{M}_k$ be the queried vectors. If $\vec{M}' \neq \prod_{i=1}^k \vec{M}_i^{\beta_i}$, $\mathcal{B}$ returns $\perp$. Otherwise, $\mathcal{B}$ answers the query in the same way as the real SignDerive oracle, by updating the table T .

Reveal queries: When $\mathcal{A}$ supplies a handle h , $\mathcal{B}$ returns $\perp$ if no entry of the form $(h, (\tau, \vec{M}), \cdot)$ exists in T . Otherwise, $\mathcal{B}$ returns the previously generated signature σ and adds $((\tau, \vec{M}), \sigma)$ in the list Q .

Forgery: Eventually, $\mathcal{A}$ outputs a Type II forgery $(\tau^*, \vec{M}^*, \sigma^*)$, where $\vec{M}^* = (M_1^*, \dots, M_n^*)$ and $\sigma^* = (z^*, r^*, u^*, v^*) \in \mathbb{G}^4$ satisfies the verification equation. At this point, $\mathcal{B}$ evaluates $J(\tau^*)$ and reports failure if $J(\tau^*) \neq 0$ or if the set $\{\tau_1, \dots, \tau_q\}$ contains at least two tags τ_{j_1}, τ_{j_2} such that $J(\tau_{j_1}) = J(\tau_{j_2}) = 0$. The same analysis as in [138] shows that, with probability $1/(8(q-1)(L+1))$, we have $J(\tau^*) = 0$ and $J(\tau_j) \neq 0$ for each $\tau_j \in \{\tau_1, \dots, \tau_q\} \setminus \{\tau^*\}$. We thus find that $\mathcal{B}$'s probability not to abort during the entire game is at least $1/(8(q-1)(L+1))$.

If $\mathcal{B}$ does not fail, we have $H_{\mathbb{G}}(\tau^*) = h^{K(\tau^*)}$, so that $\mathcal{B}$ can compute $v^\dagger = v^*$ and

$$z^\dagger = \prod_{i=1}^n M_i^{*- \chi_i} \quad r^\dagger = \prod_{i=1}^n M_i^{*- \gamma_i} \quad u^\dagger = v^{*-K(\tau^*)} \cdot \prod_{i=1}^n M_i^{*- \delta_i}. \quad (4.12)$$

We see that $(z^\dagger, r^\dagger, u^\dagger, v^\dagger)$ forms a valid signature on $(M_1^*, \dots, M_n^*)$ whose last component $v^\dagger$ coincides with that of $\mathcal{A}$'s forgery. Since $(z^\dagger, r^\dagger, u^\dagger, v^\dagger)$ and (z^*, r^*, u^*, v^*) both satisfy the verification equations, the triple

$$(z^\dagger, r^\dagger, u^\dagger) = \left(\frac{z^*}{z^\dagger}, \frac{r^*}{r^\dagger}, \frac{u^*}{u^\dagger} \right)$$

necessarily satisfies $e(g_z, z^\dagger) \cdot e(g_r, r^\dagger) = e(h_z, z^\dagger) \cdot e(h, u^\dagger) = 1_{\mathbb{G}_T}$. We are thus left with proving that $z^\dagger \neq 1_{\mathbb{G}}$ with all but negligible probability.

To do this, the key observation is that, in the desirable event

$$J(\tau^*) = 0 \quad \wedge \quad \bigwedge_{\tau_j \neq \tau^*} J(\tau_j) \neq 0, \quad (4.13)$$

the only information that $\mathcal{B}$ reveals about $(\chi_1, \dots, \chi_n)$ is contained in the z -components of signatures involving τ^* if $\mathcal{A}$ is a Type II adversary. Indeed, for each signing query $(\tau, \vec{M})$ such that $\tau \neq \tau^*$, $\mathcal{B}$ introduces in the signature a fresh random exponent $\theta \in_R \mathbb{Z}_p$ that does not appear anywhere else. This allows $\mathcal{B}$ not to leak anything about $(\chi_1, \dots, \chi_n)$ during these queries.

More precisely, let us first consider what an unbounded Type II adversary $\mathcal{A}$ can see. Throughout the game, $\mathcal{A}$ makes $n(q-1) + (n-1)$ signing queries since at most $n-1$ independent queries are allowed for the tag τ^* . Let us index these queries as $\{(\tau_j, \vec{M}_k = (M_{k,1}, \dots, M_{k,n}))\}_{j,k}$, with $j \in \{1, \dots, q\}$, and let $\{(z_{j,k}, r_{j,k}, u_{j,k}, v_{j,k})\}_{j,k}$ denote the answers in which $\mathcal{B}$ introduces $n(q-1)$ variables $\{\theta_{j,k}\}_{j \neq j^*, k \in \{1, \dots, n\}}$ in the exponent. Together with private key elements $\{(\chi_i, \gamma_i, \delta_i)\}_{i=1}^n$, we have a total of $3n + n(q-1) = 2n + nq$ unknowns. Each signature $(z_{j,k}, r_{j,k}, u_{j,k}, v_{j,k})$ provides $\mathcal{A}$ with at most one new linearly independent equation – recall that $(z_{j,k}, v_{j,k})$ uniquely determines $r_{j,k}, u_{j,k}$ while $v_{j,k}$ does not depend on $\theta_{j,k}$ or $\{(\chi_i, \gamma_i, \delta_i)\}_{i=1}^n$ – in addition to the $2n$ linear equations resulting from the public key elements $\{(g_i, h_i)\}_{i=1}^n$.

Overall, a Type II adversary $\mathcal{A}$ thus obtains $2n + nq - 1$ linear equations which is insufficient to solve a system of $2n + nq$ unknowns. Since $(M_1^*, \dots, M_n^*)$ is linearly independent of the vectors $\vec{M}_{j^*,1}, \dots, \vec{M}_{j^*,n-1}$ associated with τ^* , for $\mathcal{A}$, predicting the value $z^\dagger$ of (4.12) is equivalent to finding the missing piece equation that would determine $(\chi_1, \dots, \chi_n)$. With probability $1 - 1/p$, we thus have $z^\dagger \neq z^*$ as claimed. $\square$

Lemma 7 *A Type I forger $\mathcal{A}$ implies an algorithm $\mathcal{B}$ solving the SDP problem with non-negligible advantage. More precisely, we have $\mathbf{Adv}(\mathcal{A}) \leq 8 \cdot q \cdot (L+1) \cdot (\mathbf{Adv}^{\text{SDP}}(\mathcal{B}) + \frac{1}{p})$, where q is the number of distinct tags occurring in signing queries. Moreover, the statement holds even for non-independent adversaries.*

Proof. Let $\mathcal{A}$ be a Type I forger with non-negligible advantage ε . We show that it implies an algorithm $\mathcal{B}$ solving a SDP instance (g_z, g_r, h_z, h) with probability at least $\varepsilon/(8q(L+1))$.

Algorithm $\mathcal{B}$ begins by choosing $(w_0, w_1, \dots, w_L) \in \mathbb{G}^{L+1}$ as in the security proof of Waters signatures. This is done in such a way that, for any $\tau \in \{0, 1\}^L$, the hash value $H_{\mathbb{G}}(\tau)$ can be written $H_{\mathbb{G}}(\tau) = g_r^{J(\tau)} \cdot h^{K(\tau)}$ for the same functions $J, K : \{0, 1\}^L \rightarrow \mathbb{Z}_p$ as in the proof of Lemma 6. For any distinct $\tau, \tau_1, \dots, \tau_q$, we will thus have $J(\tau) = 0 \bmod p$ and $J(\tau_i) \neq 0 \bmod p$ for each $i \in \{1, \dots, q\}$ with non-negligible probability $\zeta = 1/(8 \cdot q \cdot (L+1))$.

Other public key components are defined by setting $g_i = g_z^{\chi_i} g_r^{\gamma_i}$ and $h_i = h_z^{\chi_i} h^{\delta_i}$, with $\chi_i, \gamma_i, \delta_i \xleftarrow{\$} \mathbb{Z}_p$ for $i = 1$ to n . During the game, $\mathcal{A}$'s queries are handled as follows.

Signing queries: At each signing query $(\tau_j, \vec{M} = (M_1, \dots, M_n))$ involving the j -th distinct tag τ_j , $\mathcal{B}$ aborts in the event that $J(\tau_j) = 0 \bmod p$. Otherwise, $\mathcal{B}$ picks $\rho, \theta \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\Theta_1 = H_{\mathbb{G}}(\tau_j)^{-\rho} \cdot (h_z)^{\frac{K(\tau_j)}{J(\tau_j)} \cdot \theta}, \quad \Theta_2 = h^\rho \cdot (h_z)^{\frac{-\theta}{J(\tau_j)}}.$$

Note that the above pair can be written $(\Theta_1, \Theta_2) = (h_z^{-\theta \cdot \alpha_r} \cdot H_{\mathbb{G}}(\tau)^{-\tilde{\rho}}, h^{\tilde{\rho}})$, where $\tilde{\rho} = \rho - \frac{\theta \cdot \beta_z}{J(\tau_j)}$. Using (Θ_1, Θ_2) , $\mathcal{B}$ obtains a well-formed signature on $(M_1, \dots, M_n)$ by computing $v = \Theta_2$,

$$z = g_r^\theta \cdot \prod_{i=1}^n M_i^{-\chi_i}, \quad r = g_z^{-\theta} \cdot \prod_{i=1}^n M_i^{-\gamma_i}, \quad u = \Theta_1 \cdot \prod_{i=1}^n M_i^{-\delta_i}.$$

The signature $\sigma = (z, r, u, v)$ is not directly returned to $\mathcal{A}$ but associated with a new handle h and stored in an entry $(h, (\tau_j, \vec{M}), \sigma)$ of the table T .

Derivation queries: When $\mathcal{A}$ queries $((h_1, \dots, h_k), (\tau, \vec{M}'), \{\beta_i\}_{i=1}^k)$ to the signature derivation oracle, $\mathcal{B}$ returns $\perp$ if not all handles $h_1, \dots, h_k$ correspond to queries involving τ . Otherwise, let $\vec{M}_1, \dots, \vec{M}_k$ be the queried vectors. If $\vec{M}' \neq \prod_{i=1}^k \vec{M}_i^{\beta_i}$, $\mathcal{B}$ returns $\perp$. Otherwise, $\mathcal{B}$ answers exactly like the real SignDerive oracle and updates the table T .

Reveal queries: When $\mathcal{A}$ queries the Reveal oracle with a handle h , $\mathcal{B}$ returns $\perp$ if no entry of the form $(h, (\tau, \vec{M}), \cdot)$ exists in T . Otherwise, $\mathcal{B}$ returns the previously computed signature σ — just like the actual Reveal oracle — and adds $((\tau, \vec{M}), \sigma)$ in the list Q .

Forgery: Eventually, $\mathcal{A}$ outputs a Type II forgery $(\tau^*, \vec{M}^*, \sigma^*)$, where $\vec{M}^* = (M_1^*, \dots, M_n^*)$ and $\sigma^* = (z^*, r^*, u^*, v^*) \in \mathbb{G}^4$ is a tuple satisfying the verification equation. At this step, $\mathcal{A}$ computes $J(\tau^*)$ and aborts if $J(\tau^*) \neq 0$. However, the same analysis as in [138] shows that, with probability $1/(8q(L+1))$, we have $J(\tau^*) = 0$ and $J(\tau_j) \neq 0$ for each $j \in \{1, \dots, q\}$.

If $\mathcal{B}$ does not fail, we have $H_{\mathbb{G}}(\tau^*) = h^{K(\tau^*)}$ and $\mathcal{B}$ can thus compute $v^\dagger = v^*$,

$$z^\dagger = \prod_{i=1}^n M_i^{\star - \chi_i}, \quad r^\dagger = \prod_{i=1}^n M_i^{\star - \gamma_i}, \quad u^\dagger = v^{\star - K(\tau^*)} \cdot \prod_{i=1}^n M_i^{\star - \delta_i}. \quad (4.14)$$

The quadruple $(z^\dagger, r^\dagger, u^\dagger, v^\dagger)$ forms a valid signature on $(M_1^*, \dots, M_n^*)$ whose last component is identical to that of $\mathcal{A}$'s forgery. Since $(z^\dagger, r^\dagger, u^\dagger, v^\dagger)$ and (z^*, r^*, u^*, v^*) both satisfy the verification equations, we find that

$$(z^\dagger, r^\dagger, u^\dagger) = \left(\frac{z^*}{z^\dagger}, \frac{r^*}{r^\dagger}, \frac{u^*}{u^\dagger} \right)$$

necessarily gives a non-trivial solution to the SDP instance with overwhelming probability.

Indeed, the same arguments as in the proof of Lemma 6 show that we can only have $z^\dagger \neq 1_{\mathbb{G}}$ with probability $1/p$. The reason is that, in each signing query, $\mathcal{B}$ introduces a new blinding exponent θ that does not appear anywhere else. For this reason, $\mathcal{B}$ never leaks any information about $(\chi_1, \dots, \chi_n)$ at any time and the element $z^\dagger$ is thus completely undetermined in $\mathcal{A}$'s view. $\square$

Since the signature component u cannot be publicly randomized, the scheme does not have fully randomizable signatures. In the next Section 4.4.4, we describe a fully randomizable variant. In applications like non-malleable commitments to group elements, the above scheme is sufficient anyway.

4.4.4 A Completely Context-Hiding Construction

We show that our scheme of Section 4.4.3 can be modified so as to become completely context-hiding in the sense of Definition 32. Namely, signatures produced by the SignDerive algorithm should be statistically indistinguishable from signatures freshly generated by Sign, even when the original signatures are given.

The difficulty is that, in the scheme of Section 4.4.3, we cannot re-randomize the underlying θ without knowing $h_z^{\alpha_r}$. To address this problem, it is tempting to include in each signature a randomization component of the form $(h_z^{\alpha_r} \cdot H_{\mathbb{G}}(\tau)^{-\zeta}, h^\zeta)$, for some $\zeta \in \mathbb{Z}_p$, which can be seen as a signature on the vector $(1_{\mathbb{G}}, \dots, 1_{\mathbb{G}})$. Unfortunately, the security proof ceases to go through as the reduction finds itself unable to generate a well-formed pair $(h_z^{\alpha_r} \cdot H_{\mathbb{G}}(\tau)^{-\zeta}, h^\zeta)$ at some step of its interaction with the adversary. Our solution actually consists in committing to the signature components that cannot be re-randomized and provide evidence that committed group elements satisfy the verification equations. This is achieved using Groth-Sahai non-interactive arguments on a perfectly witness-indistinguishable Groth-Sahai CRS, as in previous Sections for subset and substring predicates as well as our linearly homomorphic construction for vectors of scalars $(m_1, \dots, m_n) \in \mathbb{Z}_p^n$ [11]. A slight difference with the last contribution, however, is that signature components $(H_{\mathbb{G}}(\tau)^{-\rho}, h^{-\rho})$ are no longer used and replaced by the technique of Malkin *et al.* [115], which yields slightly shorter signatures.

Keygen(λ, n): given a security parameter λ and the dimension $n \in \mathbb{N}$ of the subspace to be signed, chooses bilinear group $(\mathbb{G}, \mathbb{G}_T)$ of order $p > 2^\lambda$. Then, does the following.

1. Chooses $h \xleftarrow{\$} \mathbb{G}$ and $\alpha_z, \alpha_r, \beta_z \xleftarrow{\$} \mathbb{Z}_p$. Defines $g_z = h^{\alpha_z}$, $g_r = h^{\alpha_r}$ and $h_z = h^{\beta_z}$.
2. For each $i \in \{1, \dots, n\}$, pick $\chi_i, \gamma_i, \delta_i \xleftarrow{\$} \mathbb{Z}_p$ and computes $g_i = g_z^{\chi_i} \cdot g_r^{\gamma_i}$, $h_i = h_z^{\chi_i} \cdot h^{\delta_i}$.
3. Generates $L + 1$ Groth-Sahai common reference strings by choosing $f_1, f_2 \xleftarrow{\$} \mathbb{G}$ and defining vectors $\vec{f}_1 = (f_1, 1, g) \in \mathbb{G}^3$, $\vec{f}_2 = (1, f_2, g) \in \mathbb{G}^3$ and $\vec{f}_{3,i} \xleftarrow{\$} \mathbb{G}^3$, for each $i \in \{0, \dots, L\}$.

The public key consists of

$$\text{pk} = \left(g_z, g_r, h_z, h, \{g_i, h_i\}_{i=1}^n, \mathbf{f} = (\vec{f}_1, \vec{f}_2, \{\vec{f}_{3,i}\}_{i=0}^L) \right)$$

while the private key is $\text{sk} = (h_z^{\alpha_r}, \{\chi_i, \gamma_i, \delta_i\}_{i=1}^n)$.

Sign($\text{sk}, \tau, (M_1, \dots, M_n)$): to sign a vector $(M_1, \dots, M_n) \in \mathbb{G}^n$ using the secret key $\text{sk} = (h_z^{\alpha_r}, \{\chi_i, \gamma_i, \delta_i\}_{i=1}^n)$ with the file identifier τ , conducts the following steps.

1. Chooses $\theta \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$z = g_r^\theta \cdot \prod_{i=1}^n M_i^{-\chi_i}, \quad r = g_z^{-\theta} \cdot \prod_{i=1}^n M_i^{-\gamma_i}, \quad u = h_z^{-\theta \cdot \alpha_r} \cdot \prod_{i=1}^n M_i^{-\delta_i}.$$

2. Using the bits $\tau[1] \dots \tau[L]$ of the file identifier $\tau \in \{0, 1\}^L$, defines the vector $\vec{f}_\tau = \vec{f}_{3,0} \cdot \prod_{i=1}^L \vec{f}_{3,i}^{\tau[i]}$ so as to assemble a Groth-Sahai CRS $\mathbf{f}_\tau = (\vec{f}_1, \vec{f}_2, \vec{f}_\tau)$.
3. Using $\mathbf{f}_\tau$, computes Groth-Sahai commitments

$$\begin{aligned} \vec{C}_z &= \iota(z) \cdot \vec{f}_1^{\nu_{z,1}} \vec{f}_2^{\nu_{z,2}} \vec{f}_\tau^{\nu_{z,3}}, \\ \vec{C}_r &= \iota(r) \cdot \vec{f}_1^{\nu_{r,1}} \vec{f}_2^{\nu_{r,2}} \vec{f}_\tau^{\nu_{r,3}}, \\ \vec{C}_u &= \iota(u) \cdot \vec{f}_1^{\nu_{u,1}} \vec{f}_2^{\nu_{u,2}} \vec{f}_\tau^{\nu_{u,3}}, \end{aligned}$$

to z , r and u , respectively. Using the randomness of these commitments, generates Groth-Sahai proofs $\vec{\pi}_1 = (\pi_{1,1}, \pi_{1,2}, \pi_{1,3}) \in \mathbb{G}^3$ and $\vec{\pi}_2 = (\pi_{2,1}, \pi_{2,2}, \pi_{2,3}) \in \mathbb{G}^3$ that elements (z, r, u) satisfy the linear verification equations $1_{\mathbb{G}_T} = e(g_z, z) \cdot e(g_r, r) \cdot \prod_{i=1}^n e(g_i, M_i)$ and $1_{\mathbb{G}_T} = e(h_z, z) \cdot e(h, u) \cdot \prod_{i=1}^n e(h_i, M_i)$. These proofs are obtained as

$$\begin{aligned} \vec{\pi}_1 &= (g_z^{-\nu_{z,1}} g_r^{-\nu_{r,1}}, g_z^{-\nu_{z,2}} g_r^{-\nu_{r,2}}, g_z^{-\nu_{z,3}} g_r^{-\nu_{r,3}}) \\ \vec{\pi}_2 &= (h_z^{-\nu_{z,1}} h^{-\nu_{u,1}}, h_z^{-\nu_{z,2}} h^{-\nu_{u,2}}, h_z^{-\nu_{z,3}} h^{-\nu_{u,3}}) \end{aligned}$$

and satisfy the following Groth-Sahai verification equations where we set $t_g^M = \prod_{i=1}^n E(g_i, \iota(M_i))^{-1}$ and $t_h^M = \prod_{i=1}^n E(h_i, \iota(M_i))^{-1}$,

$$\begin{aligned} t_g^M &= E(g_z, \vec{C}_z) \cdot E(g_r, \vec{C}_r) \cdot E(\pi_{1,1}, \vec{f}_1) \cdot E(\pi_{1,2}, \vec{f}_2) \cdot E(\pi_{1,3}, \vec{f}_\tau) \\ t_h^M &= E(h_z, \vec{C}_z) \cdot E(h, \vec{C}_u) \cdot E(\pi_{2,1}, \vec{f}_1) \cdot E(\pi_{2,2}, \vec{f}_2) \cdot E(\pi_{2,3}, \vec{f}_\tau). \end{aligned} \quad (4.15)$$

The signature consists of

$$\sigma = (\vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2) \in \mathbb{G}^{15}. \quad (4.16)$$

SignDerive(pk, τ , $\{(\omega_i, \sigma^{(i)})\}_{i=1}^\ell$): given the public key pk, a file identifier τ and ℓ tuples $(\omega_i, \sigma^{(i)})$, parses each signature $\sigma^{(i)}$ as a tuple of the form $\sigma^{(i)} = (\vec{C}_{z,i}, \vec{C}_{r,i}, \vec{C}_{u,i}, \vec{\pi}_{1,i}, \vec{\pi}_{2,i}) \in \mathbb{G}^{15}$ for $i = 1$ to ℓ . Otherwise, the derivation process proceeds in two steps.

1. Computes the commitments $\vec{C}_z = \prod_{i=1}^\ell \vec{C}_{z,i}^{\omega_i}$, $\vec{C}_r = \prod_{i=1}^\ell \vec{C}_{r,i}^{\omega_i}$ and $\vec{C}_u = \prod_{i=1}^\ell \vec{C}_{u,i}^{\omega_i}$, and the proofs $\vec{\pi}_1 = \prod_{i=1}^\ell \vec{\pi}_{1,i}^{\omega_i}$ and $\vec{\pi}_2 = \prod_{i=1}^\ell \vec{\pi}_{2,i}^{\omega_i}$.
2. With the fresh randomness used in step 1, re-randomizes the above commitments and proofs using their homomorphic property and returns the re-randomized version $\sigma = (\vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2)$.

Verify(pk, σ , τ , $(M_1, \dots, M_n)$): given a pair $(\tau, (M_1, \dots, M_n))$ and a purported signature σ parses the latter as $(\vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2)$. Then, returns 1 if and only if $(M_1, \dots, M_n) \neq (1_{\mathbb{G}}, \dots, 1_{\mathbb{G}})$ and equations (4.15) are satisfied.

The scheme is clearly completely context-hiding because signatures only consist of perfectly randomizable commitments and NIWI arguments.

As for the unforgeability of the scheme, the proof of the following theorem is along the lines of [115, Theorem 5]. However, we can only prove unforgeability in a weaker sense as we need to assume that the adversary is targeting. Namely, in the case of Type II attacks, the adversary must also output a proof that it actually broke the security of the scheme and that its vector $\vec{M}^* = (M_1^*, \dots, M_n^*) \in \mathbb{G}^n$ is indeed independent of the vectors for which it obtained signatures for the target tag τ^* .

If $\{\vec{M}_i = (M_{i,1}, \dots, M_{i,n})\}_{i=1}^m$ denote the linearly independent vectors that were signed for τ^* , the adversary could simply output a single test vector $\vec{W} = (W_1, \dots, W_n) \in \mathbb{G}^n$ such that $\prod_{j=1}^n e(M_j^*, W_j) \neq 1_{\mathbb{G}_T}$ and $\prod_{j=1}^n e(M_{i,j}, W_j) = 1_{\mathbb{G}_T}$ for each $i \in \{1, \dots, m\}$. This test guarantees that the adversary's output is a non-trivial Type II forgery.

Theorem 18 *The above scheme provides unforgeability against independent targeting adversaries if the DLIN assumption holds in $\mathbb{G}$.*

Proof. Since the scheme is completely context-hiding, we work with a simpler security definition where the adversary only interacts with a signing oracle. This suffices to guarantee security in the sense of Definition 34, as implied by the result of Ahn *et al.* [6]. The proof proceeds via a sequence of games. In each game, we denote by X_i the probability that the adversary $\mathcal{A}$ wins.

Game_{real} : This is the real game. When the adversary $\mathcal{A}$ terminates, the simulator outputs 1 if $\mathcal{A}$ is successful. We thus have $\Pr[X_{real}] = \mathbf{Adv}(\mathcal{A})$.

Game₀ : This game is identical to **Game_{real}** but we modify the generation of the public key. Namely, the vectors $(\vec{f}_1, \vec{f}_2, \{\vec{f}_{3,i}\}_{i=0}^L)$ are chosen by setting $\vec{f}_1 = (f_1, 1_{\mathbb{G}}, g)$ and $\vec{f}_2 = (1_{\mathbb{G}}, f_2, g)$, with $f_1, f_2 \xleftarrow{\$} \mathbb{G}$. As for $\{\vec{f}_{3,i}\}_{i=0}^L$, they are obtained as

$$\begin{aligned} \vec{f}_{3,0} &= \vec{f}_1^{\xi_{0,1}} \cdot \vec{f}_2^{\xi_{0,2}} \cdot \iota(g)^{\xi_{0,3}} \cdot \iota(g)^{\mu \cdot \zeta - \rho_0} \\ \vec{f}_{3,i} &= \vec{f}_1^{\xi_{i,1}} \cdot \vec{f}_2^{\xi_{i,2}} \cdot \iota(g)^{\xi_{i,3}} \cdot \iota(g)^{-\rho_i}, \quad i \in \{1, \dots, L\} \end{aligned} \quad (4.17)$$

with scalars $\mu \xleftarrow{\$} \{0, \dots, L\}$, $\xi_{0,1}, \xi_{1,1}, \dots, \xi_{L,1} \xleftarrow{\$} \mathbb{Z}_p$, $\xi_{0,2}, \xi_{1,2}, \dots, \xi_{L,2} \xleftarrow{\$} \mathbb{Z}_p$, $\xi_{0,3}, \xi_{1,3}, \dots, \xi_{L,3} \xleftarrow{\$} \mathbb{Z}_p$ and $\rho_0, \rho_1, \dots, \rho_L \xleftarrow{\$} \{0, \dots, \zeta - 1\}$, with $\zeta = 2q$ and where q is the number of distinct tags across all signing queries. Note that this change is only conceptual since $\{\vec{f}_{3,i}\}_{i=0}^L$ have the same distribution as in **Game_{real}**. We thus have $\Pr[X_0] = \mathbf{Adv}(\mathcal{A})$.

Game₁ : In this game, we first raise an event F_1 , which causes the simulator $\mathcal{B}$ to abort if it does *not* occur. Let $\tau_1, \dots, \tau_q$ be the distinct tags successively involved in $\mathcal{A}$'s queries throughout the game and let τ^* be the tag involved in $\mathcal{A}$'s forgery. We know that, for a Type II forger, $\tau^* \in \{\tau_1, \dots, \tau_q\}$ whereas $\tau^* \notin \{\tau_1, \dots, \tau_q\}$ for a Type I adversary. For each string $\tau \in \{0, 1\}^L$, we consider the function $J(\tau) = \mu \cdot \zeta - \rho_0 - \sum_{i=1}^L \rho_i \tau[i]$. We also define F_1 to be the event that

$$J(\tau^*) = 0 \quad \wedge \quad \bigwedge_{\tau_j \in \{\tau_1, \dots, \tau_q\} \setminus \{\tau^*\}} J(\tau_j) \neq 0.$$

We note that the exponents $\rho_0, \rho_1, \dots, \rho_L$ are independent of $\mathcal{A}$'s view: as a consequence, the simulator could equivalently define $\{\vec{f}_{3,i}\}_{i=0}^L$ first and only choose $\{\rho_i\}_{i=0}^L$ – together with values $\{\xi_{3,i}\}_{i=0}^L$ explaining the $\{\vec{f}_{3,i}\}_{i=0}^L$ – at the end of the game, when $\tau^*, \tau_1, \dots, \tau_q$ have been defined. In the case of a Type I attack, the same analysis as [138] (after the simplification of Bellare and Ristenpart [14]) shows that $\Pr[X_1 \wedge F_1] \geq \mathbf{Adv}(\mathcal{A})^2 / (27 \cdot q \cdot (L + 1))$.

This follows from the fact that, for any set of queries, a lower bound on the probability of event F_1 is $1/(2q(L + 1))$. In the case of Type II attacks, a lower bound on the probability of F_1 for any set of queries is given by $\eta \geq 1/(2(q-1)(L+1)) > 1/(2q(L+1))$. Indeed, after re-ordering, the set of queried tags can be written $\{\tau^*, \tau_1, \dots, \tau_{q-1}\}$ and, from the known results [138, 88] on the programmability of Waters' hash function, we know that the probability, taken over the choice of $(\mu, \rho_0, \dots, \rho_L)$, to have $J(\tau^*) = 0$ and $\bigwedge_{j=1}^{q-1} J(\tau_j) \neq 0$

for any distinct $\tau^*, \tau_1, \dots, \tau_q$ is at least $1/(2(q-1)(L+1)) > 1/(2q(L+1))$. In the following, we denote by F_i the counterpart of event F_1 in **Game_i**.

Game₂ : In this game, we modify the distribution of the public key. Namely, $\vec{f}_1 = (f_1, 1, g)$ and $\vec{f}_2 = (1, f_2, g)$ are chosen as before but, instead of generating the vectors $\{\vec{f}_{3,i}\}_{i=0}^L$ as previously, we choose them as

$$\begin{aligned} \vec{f}_{3,0} &= \vec{f}_1^{\xi_{0,1}} \cdot \vec{f}_2^{\xi_{0,2}} \cdot \iota(g)^{\mu \cdot \zeta - \rho_0}, \\ \vec{f}_{3,i} &= \vec{f}_1^{\xi_{i,1}} \cdot \vec{f}_2^{\xi_{i,2}} \cdot (1, 1, g)^{-\rho_i}, \quad i \in \{1, \dots, L\}, \end{aligned} \quad (4.18)$$

which amounts to setting $\xi_{0,3} = \xi_{1,3} = \dots = \xi_{L,3} = 0$. This change should not significantly affect $\mathcal{A}$'s behaviour if the DLIN assumption holds. More precisely, if events $X_1 \wedge F_1$ and $X_2 \wedge F_2$ occur with noticeably different probabilities in **Game₁** and **Game₂**, this contradicts the DLIN assumption. Concretely, consider a DLIN instance $(g, f_1, f_2, f_1^{\delta_1}, f_2^{\delta_2}, Z)$, where $\delta_1, \delta_2 \xleftarrow{\$} \mathbb{Z}_p$ and $Z = g^{\delta_1 + \delta_2}$ or $Z \in_R \mathbb{G}$. Using the random self-reducibility of DLIN, we can create $L+1$ independent DLIN instances by picking $\varphi_i, \phi_i, \psi_i \xleftarrow{\$} \mathbb{Z}_p$, for $i \in \{0, \dots, L\}$ and setting

$$\begin{aligned} \vec{f}_{3,0} &= ((f_1^{\delta_1})^{\varphi_0} f_1^{\phi_0}, (f_2^{\delta_2})^{\varphi_0} f_2^{\psi_0}, Z^{\varphi_0} \cdot g^{\phi_0 + \psi_0} \cdot \iota(g)^{\mu \cdot \zeta - \rho_0}), \\ \vec{f}_{3,i} &= ((f_1^{\delta_1})^{\varphi_i} f_1^{\phi_i}, (f_2^{\delta_2})^{\varphi_i} f_2^{\psi_i}, Z^{\varphi_i} \cdot g^{\phi_i + \psi_i} \cdot \iota(g)^{-\rho_i}), \quad i \in \{1, \dots, L\} \end{aligned}$$

If $Z \in_R \mathbb{G}$, $\{\vec{f}_{3,i}\}_{i=0}^L$ is distributed as in **Game₁**. If $Z = g^{\delta_1 + \delta_2}$, the distribution of $\{\vec{f}_{3,i}\}_{i=0}^L$ is the same as in (4.18). For this reason, we can write $|\Pr[X_2 \wedge F_2] - \Pr[X_1 \wedge F_1]| \leq \mathbf{Adv}^{\text{DLIN}}(\mathcal{A})$ as we assumed that the challenger $\mathcal{B}$ can always detect when a targeting adversary is successful.

Game₃ : In this game, we modify the treatment of signing queries. We note that, for a given message $(\tau, \vec{M} = (M_1, \dots, M_n))$, there is an exponential number of witnesses $(z, r, u) \in \mathbb{G}^3$ satisfying the verification equations

$$\begin{aligned} e(g_z, z) \cdot e(g_r, r) \cdot \prod_{i=1}^n e(g_i, M_i) &= 1_{\mathbb{G}_T}, \\ e(h_z, z) \cdot e(h, u) \cdot \prod_{i=1}^n e(h_i, M_i) &= 1_{\mathbb{G}_T}. \end{aligned} \quad (4.19)$$

Specifically, each $z \in_R \mathbb{G}$ determines a unique pair (r, u) for which (4.19) holds. However, in **Game₃**, the simulator $\mathcal{B}$ answers all signing queries using the witness (z, r, u) such that

$$z = \prod_{i=1}^n M_i^{-\chi_i}, \quad r = \prod_{i=1}^n M_i^{-\gamma_i}, \quad u = \prod_{i=1}^n M_i^{-\delta_i}.$$

Note that this amounts to choosing $\theta = 0$ at step 1 of the signing algorithm. Still, $\mathcal{B}$ has a valid witness for the statement to be proved. It thus assembles a Groth-Sahai CRS $\mathbf{f} = (\vec{f}_1, \vec{f}_2, \vec{f}_\tau)$ by computing $\vec{f}_\tau = \vec{f}_{3,0} \cdot \prod_{i=1}^L \vec{f}_{3,i}^{\tau[i]}$. Using

$\mathbf{f}$, it computes Groth-Sahai commitments $\vec{C}_z, \vec{C}_r, \vec{C}_u$ to z, r and u . Using the randomness of these commitments, it faithfully generates proofs $\vec{\pi}_1$ and $\vec{\pi}_2$ satisfying the verification equations (4.15).

We argue that this change does not affect $\mathcal{A}$'s view whatsoever. Indeed, if event F_3 occurs we have $J(\tau^*) = 0$ and $J(\tau_j) \neq 0$ for each $\tau_j \neq \tau^*$. Moreover, when $J(\tau_j)$, the Groth-Sahai CRS $(\vec{f}_1, \vec{f}_2, \vec{f}_{\tau_j})$ is a perfectly hiding Groth-Sahai CRS. This means that $\vec{C}_z, \vec{C}_r, \vec{C}_u$ are perfectly hiding commitments and proofs $(\vec{\pi}_1, \vec{\pi}_2)$ are perfectly witness-indistinguishable proofs. In other words, although the proofs $(\vec{\pi}_1, \vec{\pi}_2)$ are always generated using the witnesses (z, r, u) for which $\theta = 0$, their distribution does not depend on which specific witness is used.

In contrast, in the case of Type II attacks, signing queries involving τ^* , $(\vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2)$ reveal the underlying (z, r, u) in the information theoretic sense since $(\vec{f}_1, \vec{f}_2, \vec{f}_{\tau^*})$ is a perfectly binding CRS when $J(\tau^*) = 0$. However, at most $n - 1$ signing queries on linearly independent vectors $\vec{M}_j$ are made for the tag τ^* , so that $\mathcal{A}$ only obtains $n - 1$ linearly independent equations in the exponent. As a consequence, $\mathcal{A}$ does not obtain a sufficient amount of information to recognize that $\theta = 0$ in the underlying signatures. For this reason, we find that $\Pr[X_3 \wedge F_3] = \Pr[X_2 \wedge F_2]$.

In **Game₃**, we show that a successful forger $\mathcal{A}$ implies an algorithm $\mathcal{B}$ solving a given SDP instance (g_z, g_r, h_z, h) with non-negligible advantage, which contradicts the DLIN assumption.

Recall that, when the adversary $\mathcal{A}$ terminates, it outputs $(\tau^*, \vec{M}^*, \sigma^*)$, where $\vec{M}^* = (M_1^*, \dots, M_n^*)$ and $\sigma^* = (\vec{C}_z^*, \vec{C}_r^*, \vec{C}_u^*, \vec{\pi}_1^*, \vec{\pi}_2^*) \in \mathbb{G}^{15}$ satisfies the verification equations. At this point, if the event F_1 introduced in **Game₁** occurs, we must have $J(\tau^*) = 0$, which means that $\vec{f}_{\tau^*} = \vec{f}_{3,0} \cdot \prod_{i=1}^{L+1} \vec{f}_{3,i}^{\tau^*[i]}$ is in $\text{span}(\vec{f}_1, \vec{f}_2)$. This implies that $\vec{C}_z^*, \vec{C}_r^*$ and $\vec{C}_u^*$ are perfectly binding commitments. Moreover, using $(\log_g(f_1), \log_g(f_2))$, $\mathcal{B}$ can extract the underlying group elements $(z^*, r^*, u^*) \in \mathbb{G}^3$ by performing BBS decryptions of ciphertexts $(\vec{C}_z^*, \vec{C}_r^*, \vec{C}_u^*)$. Since $(\vec{\pi}_1^*, \vec{\pi}_2^*)$ are valid proofs for a perfectly sound Groth-Sahai CRS, the extracted elements (z^*, r^*, u^*) necessarily satisfy

$$\begin{aligned} 1_{\mathbb{G}_T} &= e(g_z, z^*) \cdot e(g_r, r^*) \cdot \prod_{i=1}^n e(g_i, M_i^*), \\ 1_{\mathbb{G}_T} &= e(h_z, z^*) \cdot e(h, u^*) \cdot \prod_{i=1}^n e(h_i, M_i^*). \end{aligned} \quad (4.20)$$

Having extracted (z^*, r^*, u^*) , $\mathcal{B}$ also computes

$$z^\dagger = \prod_{i=1}^n M_i^{*-x_i}, \quad r^\dagger = \prod_{i=1}^n M_i^{*-y_i}, \quad u^\dagger = \prod_{i=1}^n M_i^{*-z_i}, \quad (4.21)$$

so that $(z^\dagger, r^\dagger, u^\dagger)$ also satisfies (4.20). Since $(z^\dagger, r^\dagger, u^\dagger)$ and (z^*, r^*, u^*) both satisfy (4.20), the triple

$$(z^\dagger, r^\dagger, u^\dagger) = \left(\frac{z^*}{z^\dagger}, \frac{r^*}{r^\dagger}, \frac{u^*}{u^\dagger} \right)$$

necessarily satisfies $e(g_z, z^\dagger) \cdot e(g_r, r^\dagger) = e(h_z, z^\dagger) \cdot e(h, u^\dagger) = 1_{\mathbb{G}_T}$. To conclude the proof, we argue that $z^\dagger \neq 1_{\mathbb{G}}$ with all but negligible probability.

To do this, we remark that, if the event F_1 defined in **Game**₁ occurs, the only information that $\mathcal{B}$ leaks about $(\chi_1, \dots, \chi_n)$ resides in the unique signing query involving τ^* in the case of Type II attacks. Indeed, for all signing queries $(\tau, \vec{M})$ involving tags τ such that $\tau \neq \tau^*$, we have $J(\tau) \neq 0$ so that $(\vec{f}_1, \vec{f}_2, \vec{f}_\tau)$ is a perfectly hiding Groth-Sahai CRS, for which proofs $(\vec{\pi}_1, \vec{\pi}_2)$ and commitments are perfectly witness-indistinguishable. In other words, the signatures $(\vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2)$ for which $J(\tau) \neq 0$ leak nothing about $(\chi_1, \dots, \chi_n)$. In contrast, in the case of Type II attacks, signing queries involving τ^* , $(\vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2)$ reveal the underlying (z, r, u) in the information theoretic sense. However, at most $n - 1$ linearly independent vectors $\vec{M}_j$ are signed w.r.t. τ^* , so that $\mathcal{A}$ only obtains $n - 1$ linearly independent equations in the exponent for the unknowns $(\chi_1, \dots, \chi_n)$. As a consequence, we can apply the same arguments as in the proof of Theorem 16 and Lemma 6. With probability $1 - 1/p$, we thus have $z^\dagger \neq z^*$.

To recap, we find

$$\Pr[X_3 \wedge F_3] = \mathbf{Adv}^{\text{SDP}}(\mathcal{B}) \cdot \left(1 - \frac{1}{p}\right)^{-1}.$$

When putting the above altogether, we find

$$\frac{\mathbf{Adv}(\mathcal{A})^2}{27 \cdot q \cdot (L + 1)} \leq \mathbf{Adv}^{\text{SDP}}(\mathcal{B}) \cdot \left(1 - \frac{1}{p}\right)^{-1} + \mathbf{Adv}^{\text{DLIN}}(\mathcal{B}).$$

Since any SDP algorithm $\mathcal{B}_0$ yields a DLIN distinguisher $\mathcal{B}_1$ such that $\mathbf{Adv}^{\text{DLIN}}(\mathcal{B}_0) \geq 2 \cdot \mathbf{Adv}^{\text{SDP}}(\mathcal{B}_1)$, we find

$$\mathbf{Adv}(\mathcal{A}) \leq \sqrt{27 \cdot q \cdot (L + 1) \cdot \left[1 + \frac{1}{2} \cdot \left(1 - \frac{1}{p}\right)^{-1}\right] \cdot \mathbf{Adv}^{\text{DLIN}}(\mathcal{B})}$$

and the announced result follows. $\square$

Part III

Commitment & Proof System Primitives

Chapter 5

Non-Malleable Trapdoor Commitments to Group Elements

Contents

5.1	Definitions and Known Results	146
5.2	A Generic Constant-Size Construction	148
5.2.1	Intuition of the Generic Transformation	149
5.2.2	Template of Linearly Homomorphic SPS Scheme	149
5.2.3	Construction of Simulation-Sound Structure-Preserving Trapdoor Commitments	150
5.2.4	Security Proof of the Generic Transformation	152

We show that linearly homomorphic SPS schemes generically yield non-malleable [60] trapdoor commitments to group elements. We actually construct a simulation-sound trapdoor commitment [73] – a primitive known (by [73, 114]) to imply re-usable non-malleable commitments with respect to opening [55] – from any linearly homomorphic structure-preserving signatures in symmetric bilinear groups. To our knowledge, we thus obtain the first constant-size trapdoor commitments to group elements providing re-usable non-malleability with respect to opening. Previous non-interactive commitments to group elements were either malleable [84, 82] or inherently length-increasing [64]: if we disregard the trivial solution consisting of hashing the message first (which is not an option when we want to allow for efficient proofs of knowledge of an opening), no general technique has been known to date for committing to many group elements at once using a short commitment string.

In our simulation-sound commitments to group elements, the commitment generation technique appeals to the verification algorithm of the signature scheme, and proceeds by evaluating the corresponding pairing-product equa-

tions on the message, but using random group elements instead of actual signatures. The binding and simulation-binding properties, in turn, stem from the infeasibility of forging signatures while the signature homomorphism allows equivocating fake commitments when simulating the view of an adversary. It was already known how to build simulation-sound and non-malleable commitments [73, 114, 55, 75, 38] from signature schemes with efficient Σ protocols. Our method is, in fact, different and immediately yields length-reducing structure-preserving commitments to vectors without using Σ protocols.

This construction of non-malleable commitment schemes from linearly homomorphic signatures is somewhat unexpected considering that the terms “non-malleability” and “homomorphism” are antagonistic, and thus may be considered incompatible.

5.1 Definitions and Known Results

Trapdoor commitment schemes have been recalled in Section 2.4.1, however, we first define them again to make explicit the alternative distribution of “simulated” commitments that are equivocable using the trapdoor key. We also slightly modify their notations to bring a better uniformity in the following.

Definition 35 *A trapdoor commitment is given by a tuple of five efficient algorithms (Setup, Com, FakeCom, FakeOpen, Verify) where Com and Verify proceed as in an ordinary commitment and other algorithms proceed as follows,*

Setup: *is a randomized algorithm that takes as input a security parameter λ . It produces a public key pk and a trapdoor tk ,*

FakeCom: *is a randomized algorithm that takes as input a public key pk and the trapdoor tk . It outputs a fake commitment string $\widetilde{\text{com}}$ and some auxiliary information aux ,*

FakeOpen: *takes as input a fake commitment produced by FakeCom and the corresponding auxiliary information aux . It also takes as input a message Msg and the trapdoor tk and outputs a fake de-commitment $\widetilde{\text{dec}}$ such that $\text{Verify}(pk, \text{Msg}, \widetilde{\text{com}}, \widetilde{\text{dec}}) = 1$,*

where the trapdoor indistinguishability property implicitly holds:

$$\begin{aligned} D_{\text{real}} &:= \{(pk, tk) \leftarrow \text{Setup}(\lambda); (\text{com}, \text{dec}) \leftarrow \text{Com}(pk, \text{Msg}) : (pk, \text{Msg}, \text{com}, \text{dec})\} \\ &\approx D_{\text{fake}} := \{(pk, tk) \leftarrow \text{Setup}(\lambda); (\widetilde{\text{com}}, \text{aux}) \leftarrow \text{FakeCom}(pk, tk); \\ &\quad \widetilde{\text{dec}} \leftarrow \text{FakeOpen}(\text{aux}, tk, \widetilde{\text{com}}, \text{Msg}) : (pk, \text{Msg}, \widetilde{\text{com}}, \widetilde{\text{dec}})\}. \end{aligned}$$

In a SSTC, each commitment is labelled with a tag. The definition of [114] requires that, even if the adversary can see equivocations of commitments to possibly distinct messages for several tags $\text{tag}_1, \dots, \text{tag}_q$, it will not be able to break the binding property for a new tag $\text{tag} \notin \{\text{tag}_1, \dots, \text{tag}_q\}$.

Definition 36 ([114]) *A simulation-sound trapdoor commitment is given by a tuple of efficient algorithms $(\text{Setup}, \text{Com}, \text{FakeCom}, \text{FakeOpen}, \text{Verify})$ where the tuple $(\text{Setup}, \text{Com}, \text{Verify})$ forms a commitment scheme and $(\text{FakeCom}, \text{FakeOpen})$ are PPT algorithms with the following properties*

Trapdoor: *for any tag and any message Msg , the following distributions are computationally indistinguishable:*

$$D_{\text{fake}} := \{(pk, tk) \leftarrow \text{Setup}(\lambda); (\widetilde{\text{com}}, \text{aux}) \leftarrow \text{FakeCom}(pk, tk, \text{tag}); \\ \widetilde{\text{dec}} \leftarrow \text{FakeOpen}(\text{aux}, tk, \widetilde{\text{com}}, \text{Msg}) : (pk, \text{tag}, \text{Msg}, \widetilde{\text{com}}, \widetilde{\text{dec}})\}$$

$$D_{\text{real}} := \{(pk, tk) \leftarrow \text{Setup}(\lambda); (\text{com}, \text{dec}) \leftarrow \text{Com}(pk, \text{tag}, \text{Msg}) : \\ (pk, \text{tag}, \text{Msg}, \text{com}, \text{dec})\}$$

Simulation-sound binding: *for any PPT adversary $\mathcal{A}$, the following probability is negligible*

$$\Pr[(pk, tk) \leftarrow \text{Setup}(\lambda); (\text{com}, \text{tag}, \text{Msg}_1, \text{Msg}_2, \text{dec}_1, \text{dec}_2) \leftarrow \mathcal{A}^{\mathcal{O}_{tk, pk}}(pk) : \\ \text{Msg}_1 \neq \text{Msg}_2 \wedge \text{Verify}(pk, \text{tag}, \text{Msg}_1, \text{com}, \text{dec}_1) = 1 \\ \wedge \text{Verify}(pk, \text{tag}, \text{Msg}_2, \text{com}, \text{dec}_2) = 1 \wedge \text{tag} \notin Q],$$

where $\mathcal{O}_{tk, pk}$ is an oracle that maintains an initially empty set Q and operates as follows:

- On input $(\text{commit}, \text{tag})$, it runs $(\widetilde{\text{com}}, \text{aux}) \leftarrow \text{FakeCom}(pk, tk, \text{tag})$, stores $(\widetilde{\text{com}}, \text{tag}, \text{aux})$, returns $\widetilde{\text{com}}$ and adds tag in Q .
- On input $(\text{decommit}, \widetilde{\text{com}}, \text{Msg})$: if a tuple $(\widetilde{\text{com}}, \text{tag}, \text{aux})$ was previously stored, it computes $\widetilde{\text{dec}} \leftarrow \text{FakeOpen}(\text{aux}, tk, \text{tag}, \widetilde{\text{com}}, \text{Msg})$ and returns $\widetilde{\text{dec}}$. Otherwise, $\mathcal{O}_{tk, pk}$ returns $\perp$.

While our SSTC to group elements will be proved secure in the above sense, a *non-adaptive* flavour of simulation-sound binding security is sufficient for the construction of non-malleable commitments. Indeed, Gennaro used [75] such a relaxed notion to achieve non-malleability from similar-looking multi-trapdoor commitments. In the non-adaptive notion, the adversary has to choose the set of tags $\text{tag}_1, \dots, \text{tag}_\ell$ for which he/she wants to query the $\mathcal{O}_{tk, pk}$ oracle before seeing the public key pk .

We now recall the definition of independence for commitment schemes, which is known (see, e.g., [76] for a proof) to imply re-usable non-malleability with respect to opening.

Definition 37 ([53]) *A trapdoor commitment scheme $(\text{Setup}, \text{Com}, \text{FakeCom}, \text{FakeOpen}, \text{Verify})$ provides ℓ -independence if, for any PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$*

and any pair of ℓ -tuples $(\text{Msg}_1, \dots, \text{Msg}_\ell)$, $(\text{Msg}'_1, \dots, \text{Msg}'_\ell)'$, the following probability is a negligible function of the security parameter λ :

$$\begin{aligned}
& \Pr[(pk, tk) \leftarrow \text{Setup}(\lambda); R_1, \dots, R_\ell \xleftarrow{\$} \{0, 1\}^{\text{poly}(\lambda)}; \\
& \quad (\widetilde{\text{com}}_i, \text{aux}_i) \leftarrow \text{FakeCom}(pk, tk, R_i) \\
& \quad (st, \text{com}^*) \leftarrow \mathcal{A}_1(pk, \widetilde{\text{com}}_1, \dots, \widetilde{\text{com}}_\ell) \text{ with } \text{com}^* \notin \{\widetilde{\text{com}}_i\}_{i=1}^\ell \\
& \quad \text{dec}_i \leftarrow \text{FakeOpen}(\text{aux}_i, tk, \widetilde{\text{com}}_i, \text{Msg}_i) \quad \forall i \in \{1, \dots, \ell\} \\
& \quad \text{dec}'_i \leftarrow \text{FakeOpen}(\text{aux}_i, tk, \widetilde{\text{com}}_i, \text{Msg}'_i) \quad \forall i \in \{1, \dots, \ell\} \\
& \quad (\text{Msg}_1^*, \text{dec}_1^*) \leftarrow \mathcal{A}_2(st, pk, \text{Msg}_1, \text{dec}_1, \dots, \text{Msg}_\ell, \text{dec}_\ell) \\
& \quad (\text{Msg}_2^*, \text{dec}_2^*) \leftarrow \mathcal{A}_2(st, pk, \text{Msg}'_1, \text{dec}'_1, \dots, \text{Msg}'_\ell, \text{dec}'_\ell) : \\
& \quad \text{Msg}_1^* \neq \text{Msg}_2^* \wedge \text{Verify}(pk, \text{Msg}_1^*, \text{com}^*, \text{dec}_1^*) = 1 \\
& \quad \wedge \text{Verify}(pk, \text{Msg}_2^*, \text{com}^*, \text{dec}_2^*) = 1]
\end{aligned}$$

A trapdoor commitment is independent if it provides ℓ -independence for any arbitrary $\ell \in \text{poly}(\lambda)$.

It is known (see, e.g., [114]) that, when a SSTC scheme and a secure one-time signature are combined to build an ordinary commitment scheme, the simulation-sound binding property and the security of the one-time signature imply the notion of independence.

5.2 A Generic Constant-Size Construction

As noted in [82, 83], some applications require committing to group elements without knowing their discrete logarithms or destroying their algebraic structure by hashing them first. This section shows that linearly homomorphic SPS imply length-reducing non-malleable structure-preserving commitments to vectors of group elements.

Using the linearly homomorphic signature scheme of Section 4.4.3, we obtain the first length-reducing non-malleable structure-preserving trapdoor commitment. The resulting scheme is not *strictly*¹ structure-preserving (according to the terminology of [5]) because the commitment string lives in $\mathbb{G}_T$ rather than $\mathbb{G}$. Still, openings only consist of elements in $\mathbb{G}$, which makes it possible to generate efficient NIWI proofs that committed group elements satisfy certain properties. To our knowledge, the only known non-malleable commitment schemes whose openings only consist of group elements were described by Fischlin *et al.* [64]. However, these constructions cannot be length-reducing as they achieve universal composability [37, 39].

¹We recall that strictly structure-preserving commitments cannot be length-reducing, as shown by Abe *et al.* [5], so that our scheme is essentially the best we can hope for if we aim at short commitment strings.

5.2.1 Intuition of the Generic Transformation

Our schemes are obtained by first constructing simulation-sound trapdoor commitments (SSTC) – in the sense of Definition 36 – to group elements. SSTC schemes were first suggested by Garay, MacKenzie and Yang [73] as a tool for constructing universally composable zero-knowledge proofs [37]. MacKenzie and Yang subsequently gave a simplified security definition which suffices to provide non-malleability with respect to opening in the sense of the definition of re-usable non-malleable commitments [55].

The verification equations (4.11) of the linearly homomorphic signatures of Section 4.4.3 are given by

$$\begin{aligned} 1_{\mathbb{G}_T} &= e(g_z, z) \cdot e(g_r, r) \cdot \prod_{i=1}^n e(g_i, m_i), \\ 1_{\mathbb{G}_T} &= e(h_z, z) \cdot e(h, u) \cdot e(H_{\mathbb{G}}(\tau), v) \cdot \prod_{i=1}^n e(h_i, m_i), \end{aligned} \quad (5.1)$$

where $\sigma = (z, r, u, v) \in \mathbb{G}^4$ is a signature on the group elements $(m_1, \dots, m_n)$, for the file identifier τ . Instead, the key idea of our construction is to generate a commitment on $(M_1, \dots, M_n)$ associated with a tag τ by picking the group elements (z, r, u, v) at random and to evaluate the right-hand side of the above equations. Concretely, for random $\text{dec} := (Z, R, U, V) \xleftarrow{\$} \mathbb{G}^4$ the commitment $C = (C_A, C_B)$ satisfies

$$\begin{aligned} C_A &= e(g_z, Z) \cdot e(g_r, R) \cdot \prod_{i=1}^n e(g_i, M_i), \\ C_B &= e(h_z, Z) \cdot e(h, U) \cdot e(H_{\mathbb{G}}(\tau), V) \cdot \prod_{i=1}^n e(h_i, M_i). \end{aligned} \quad (5.2)$$

Clearly, using the secret key of the linearly homomorphic signature as a trapdoor key, we see how a signature on an appropriate vector of messages $(m_1, \dots, m_n)$ can equivocate the openings of $C = (C_A, C_B) \in \mathbb{G}_T^2$ by multiplication of the equations. Example (2.27) of Groth-Sahai proofs shows how easy it is to convince a verifier that you are able to open the commitment.

5.2.2 Template of Linearly Homomorphic SPS Scheme

We first observe that *any* constant-size linearly homomorphic structure-preserving signature necessarily complies with the template below. Indeed, in order to have a linear homomorphism, each verification equation necessarily computes a product of pairings which should equal $1_{\mathbb{G}_T}$ in a valid signature. In each pairing of the product, one of the arguments must be a message or signature component while the second argument is either part of the public key or an encoding of the file identifier.

For simplicity, the template is described in terms of symmetric pairings but generalizations to asymmetric configurations are possible.

Keygen(λ, n): given λ and the dimension $n \in \mathbb{N}$ of the vectors to be signed, chooses constants n_z, n_v, m . Among these, n_z and n_v will determine the signature length while m will be the number of verification equations. Then, it chooses $\{F_{j,\mu}\}_{j \in \{1, \dots, m\}, \mu \in \{1, \dots, n_z\}}, \{G_{j,i}\}_{i \in \{1, \dots, n\}, j \in \{j, \dots, m\}}$ in the group $\mathbb{G}$. The public key is

$$\text{pk} = (\{F_{j,\mu}\}_{j \in \{1, \dots, m\}, \mu \in \{1, \dots, n_z\}}, \{G_{j,i}\}_{i \in \{1, \dots, n\}, j \in \{j, \dots, m\}})$$

while sk contains information about the representation of public elements w.r.t. specific bases.

Sign($\text{sk}, \tau, (M_1, \dots, M_n)$): Outputs a tuple $\sigma = (Z_1, \dots, Z_{n_z}, V_1, \dots, V_{n_v}) \in \mathbb{G}^{n_z + n_v}$.

SignDerive($\text{pk}, \tau, \{(\omega_i, \sigma^{(i)})\}_{i=1}^\ell$): parses each signature $\sigma^{(i)}$ as $(Z_1^{(i)}, \dots, Z_{n_z}^{(i)}, V_1^{(i)}, \dots, V_{n_v}^{(i)})$ and computes for all $\mu \in \{1, \dots, n_z\}$ and $\nu \in \{1, \dots, n_v\}$

$$Z_\mu = \prod_{i=1}^\ell Z_\mu^{(i) \omega_i} \quad V_\nu = \prod_{i=1}^\ell V_\nu^{(i) \omega_i}.$$

After a possible extra re-randomization step, it outputs the derived signature $(Z_1, \dots, Z_{n_z}, V_1, \dots, V_{n_v})$.

Verify($\text{pk}, \sigma, \tau, (M_1, \dots, M_n)$): given a signature σ , a tag τ and $(M_1, \dots, M_n)$, returns 0 if $(M_1, \dots, M_n) = (1_{\mathbb{G}}, \dots, 1_{\mathbb{G}})$. Otherwise, parses the signature as $(Z_1, \dots, Z_{n_z}, V_1, \dots, V_{n_v}) \in \mathbb{G}^{n_z + n_v}$ and performs as follows.

1. For each $j \in \{1, \dots, m\}$ and $\nu \in \{1, \dots, n_v\}$, computes one-to-one² encodings $T_{j,\nu} \in \mathbb{G}$ of the tag τ as a group element.
2. Returns 1 if and only if $c_j = 1_{\mathbb{G}_T}$ for $j = 1$ to m , where

$$c_j = \prod_{\mu=1}^{n_z} e(F_{j,\mu}, Z_\mu) \cdot \prod_{\nu=1}^{n_v} e(T_{j,\nu}, V_\nu) \cdot \prod_{i=1}^n e(G_{j,i}, M_i) \quad (5.3)$$

In the following, we say that a linearly homomorphic SPS is *regular* if, for each file identifier τ , any non-trivial vector $(M_1, \dots, M_n) \neq (1_{\mathbb{G}}, \dots, 1_{\mathbb{G}})$ has a valid signature.

5.2.3 Construction of Simulation-Sound Structure-Preserving Trapdoor Commitments

Let $\Pi^{\text{SPS}} = (\text{Keygen}, \text{Sign}, \text{SignDerive}, \text{Verify})$ be a linearly homomorphic SPS. We construct a simulation-sound trapdoor commitment as follows.

²This condition can be relaxed to have collision-resistant deterministic encodings. Here, we assume injectivity for simplicity.

SSTC.Setup(λ, n): given the desired dimension $n \in \mathbb{N}$ of committed vectors, chooses public parameters $\mathbf{pp}$ for the linearly homomorphic SPS scheme. Then, it runs $\Pi^{\text{SPS}}.\text{Keygen}(\lambda, n)$ to obtain a public key

$$\mathbf{pk} = (\{F_{j,\mu}\}_{j \in \{1, \dots, m\}, \mu \in \{1, \dots, n_z\}}, \{G_{j,i}\}_{i \in \{1, \dots, n\}, j \in \{j, \dots, m\}}),$$

for some constants n_z, n_v, m , and a $\mathbf{sk}$. The commitment key is $\mathbf{pk} = \mathbf{pk}$ and the trapdoor $\mathbf{tk}$ consists of $\mathbf{sk}$. Note that the public key defines a signature space $\mathbb{G}^{n_z+n_v}$, for constants n_z and n_v .

SSTC.Com($\mathbf{pk}, \mathbf{tag}, (M_1, \dots, M_n)$): to commit to $(M_1, \dots, M_n) \in \mathbb{G}^n$ with respect to the tag $\mathbf{tag} = \tau$, chooses $(Z_1, \dots, Z_{n_z}, V_1, \dots, V_{n_v}) \xleftarrow{\$} \mathbb{G}^{n_z+n_v}$ in the signature space. Then, it runs step 1 of the verification algorithm and evaluates the right-hand side member of (5.3). Namely, it computes for all $j \in \{1, \dots, m\}$

$$c_j = \prod_{\mu=1}^{n_z} e(F_{j,\mu}, Z_\mu) \cdot \prod_{\nu=1}^{n_v} e(T_{j,\nu}, V_\nu) \cdot \prod_{i=1}^n e(G_{j,i}, M_i) \quad (5.4)$$

where $\{T_{j,\nu}\}_{j,\nu}$ forms an injective encoding of $\mathbf{tag} = \tau$ as a set of group elements. The commitment string is $\mathbf{com} = (c_1, \dots, c_m)$ whereas the decommitment is $\mathbf{dec} = (Z_1, \dots, Z_{n_z}, V_1, \dots, V_{n_v})$.

SSTC.FakeCom($\mathbf{pk}, \mathbf{tk}, \mathbf{tag}$): selects random $(\hat{M}_1, \dots, \hat{M}_n) \xleftarrow{\$} \mathbb{G}^n$ and proceeds like **SSTC.Com**. If $(\hat{\mathbf{com}}, \hat{\mathbf{dec}})$ denotes the resulting pair, the algorithm outputs $\widetilde{\mathbf{com}} = \hat{\mathbf{com}}$ and the auxiliary information $\mathbf{aux}$, which consists of the pair $\mathbf{aux} = ((\hat{M}_1, \dots, \hat{M}_n), \hat{\mathbf{dec}})$ for $\mathbf{tag} = \tau$.

SSTC.FakeOpen($\mathbf{aux}, \mathbf{tk}, \mathbf{tag}, \widetilde{\mathbf{com}}, (M_1, \dots, M_n)$): parses the commitment $\widetilde{\mathbf{com}}$ as $(\tilde{c}_1, \dots, \tilde{c}_m)$ and $\mathbf{aux}$ as $((\hat{M}_1, \dots, \hat{M}_n), (\hat{Z}_1, \dots, \hat{Z}_{n_z}, \hat{V}_1, \dots, \hat{V}_{n_v}))$. It first generates a linearly homomorphic signature on the vector of messages $(M_1/\hat{M}_1, \dots, M_n/\hat{M}_n)$ for the tag $\mathbf{tag} = \tau$. Namely, using the trapdoor $\mathbf{tk} = \mathbf{sk}$, it computes a signature $\sigma' = (Z'_1, \dots, Z'_{n_z}, V'_1, \dots, V'_{n_v}) \leftarrow \Pi^{\text{SPS}}.\text{Sign}(\mathbf{sk}, \tau, (M_1/\hat{M}_1, \dots, M_n/\hat{M}_n))$. Since σ' is a valid signature and $\mathbf{aux} = ((\hat{M}_1, \dots, \hat{M}_n), (\hat{Z}_1, \dots, \hat{Z}_{n_z}, \hat{V}_1, \dots, \hat{V}_{n_v}))$ satisfies

$$\tilde{c}_j = \prod_{\mu=1}^{n_z} e(F_{j,\mu}, \hat{Z}_\mu) \cdot \prod_{\nu=1}^{n_v} e(T_{j,\nu}, \hat{V}_\nu) \cdot \prod_{i=1}^n e(G_{j,i}, \hat{M}_i) \quad (5.5)$$

for all $j \in \{1, \dots, m\}$, for $\hat{\sigma} = (\hat{Z}_1, \dots, \hat{Z}_{n_z}, \hat{V}_1, \dots, \hat{V}_{n_v})$, the algorithm can run $(\tilde{Z}_1, \dots, \tilde{Z}_{n_z}, \tilde{V}_1, \dots, \tilde{V}_{n_v}) \leftarrow \text{SignDerive}(\mathbf{pk}, \tau, \{(1, \sigma'), (1, \hat{\sigma})\})$, and outputs $\widetilde{\mathbf{dec}} = (\tilde{Z}_1, \dots, \tilde{Z}_{n_z}, \tilde{V}_1, \dots, \tilde{V}_{n_v})$ which is a valid decommitment to the vector $(M_1, \dots, M_n)$ with respect to $\mathbf{tag} = \tau$.

SSTC.Verify($\mathbf{pk}, \mathbf{tag}, (M_1, \dots, M_n), \mathbf{com}, \mathbf{dec}$): parses $\mathbf{com}$ as $(c_1, \dots, c_m) \in \mathbb{G}_T^m$ and the de-commitment $\mathbf{dec}$ as $(Z_1, \dots, Z_{n_z}, V_1, \dots, V_{n_v}) \in \mathbb{G}^{n_z+n_v}$ (if these values do not parse properly, return 0). Then, it computes a one-to-one encoding $\{T_{j,\nu}\}_{j,\nu}$ of $\mathbf{tag} = \tau$. Returns 1 if relations (5.4) hold and 0 otherwise.

In the full version of our paper [106, Appendix 5], we extend the above construction so as to build simulation-sound trapdoor commitments to vectors from any linearly homomorphic signature that fits a certain template. As a result, we obtain a modular construction of constant-size non-malleable commitments to vectors which preserves the feasibility of efficiently proving properties about committed values. We also give an instantiation based on the CDH assumption to commit on a vector of scalars using our linearly homomorphic signature of [10] which is notably secure against Type I independent adversaries — as implicitly proved by [10, Lemma 8]. With a further optimization, this results in a commitment consisting of one target group element and two source group elements for openings. Unfortunately the details are beyond the scope of this thesis since we are focusing on structure-preserving application.

Applying the transformation from SSTC to NM commitments.

As stated in the previous section, a standard technique allows constructing a re-usable non-malleable commitment from a SSTC scheme as follows. To commit to Msg , the sender generates a key-pair (VK, SK) for a one-time signature and generates $(\text{com}, \text{dec}) \leftarrow \text{SSTC.Commit}(pk, \text{VK}, \text{Msg})$ using VK as a tag. The non-malleable commitment string is the pair (com, VK) and the opening is given by (dec, σ) , where σ is a one-time signature on com , so that the receiver additionally checks the validity of σ . This construction is known to provide independence (see Definition 37 of previous section) and thus non-malleability with respect to opening, as proved in [53, 76].

In our setting, we cannot compute σ as a signature of com , as it consists of $\mathbb{G}_T$ elements. However, we can rather sign the pair (Msg, dec) — whose components live in $\mathbb{G}$ — as long as it uniquely determines com . To this end, we can use the one-time structure-preserving of [4, Appendix C.1] since it allows signing messages of arbitrary length using a constant-size one-time public key. Like our scheme of Section 4.4.3, it relies on the SDP assumption and thus yields a non-malleable commitment based on this sole assumption. Alternatively, we can move σ in the commitment string (which thus consists of $(\text{com}, \text{VK}, \sigma)$), in which case the one-time signature does not need to be structure-preserving but has to be strongly unforgeable (as can be observed from the definition of independent commitments recalled in Definition 37 of the previous section) while the standard notion of unforgeability suffices in the first case.

5.2.4 Security Proof of the Generic Transformation

The security of the generic construction is stated by the following theorem.

Theorem 19 *Assuming that the underlying linearly homomorphic SPS is regular and secure against non-independent Type I adversaries, the above construction is a simulation-sound trapdoor commitment to group elements.*

Proof. We first observe that the commitment satisfies the trapdoor property if the homomorphic SPS is regular. Indeed, in the distribution D_{fake} , the

commitment $\widetilde{\text{com}}$ is obtained as

$$c_j = \prod_{\mu=1}^{n_z} e(F_{j,\mu}, \hat{Z}_\mu) \cdot \prod_{\nu=1}^{n_v} e(T_{j,\nu}, \hat{V}_\nu) \cdot \prod_{i=1}^n e(G_{j,i}, \hat{M}_i) \quad (5.6)$$

for all $j \in \{1, \dots, m\}$, where $(\hat{M}_1, \dots, \hat{M}_n) \in_R \mathbb{G}^n$ and for a uniformly random tuple $(\hat{Z}_1, \dots, \hat{Z}_{n_z}, \hat{V}_1, \dots, \hat{V}_{n_v}) \in_R \mathbb{G}^{n_z+n_v}$. We also know that, for any $(M_1, \dots, M_n) \neq (\hat{M}_1, \dots, \hat{M}_n)$, the vector $(M_1/\hat{M}_1, \dots, M_n/\hat{M}_n)$ has a valid signature $\sigma' = (Z'_1, \dots, Z'_{n_z}, V'_1, \dots, V'_{n_v})$, so that there exists

$$\widetilde{\text{dec}} = (\tilde{Z}_1, \dots, \tilde{Z}_{n_z}, \tilde{V}_1, \dots, \tilde{V}_{n_v}) = (Z'_1 \cdot \hat{Z}_1, \dots, Z'_{n_z} \cdot \hat{Z}_{n_z}, V'_1 \cdot \hat{V}_1, \dots, V'_{n_v} \cdot \hat{V}_{n_v})$$

that explains $\widetilde{\text{com}}$ as a commitment to $(M_1, \dots, M_n)$. Moreover, since the vector $(\hat{Z}_1, \dots, \hat{Z}_{n_z}, \hat{V}_1, \dots, \hat{V}_{n_v})$ was chosen uniformly in $\mathbb{G}^{n_z+n_v}$, $\widetilde{\text{dec}}$ is uniform among values $(\tilde{Z}_1, \dots, \tilde{Z}_{n_z}, \tilde{V}_1, \dots, \tilde{V}_{n_v})$ such that, for all $j \in \{1, \dots, m\}$

$$c_j = \prod_{\mu=1}^{n_z} e(F_{j,\mu}, \tilde{Z}_\mu) \cdot \prod_{\nu=1}^{n_v} e(T_{j,\nu}, \tilde{V}_\nu) \cdot \prod_{i=1}^n e(G_{j,i}, M_i). \quad (5.7)$$

In other words, the joint distribution of $(\widetilde{\text{com}}, \widetilde{\text{dec}})$ is the same as if it were obtained by choosing $(\tilde{Z}_1, \dots, \tilde{Z}_{n_z}, \tilde{V}_1, \dots, \tilde{V}_{n_v}) \xleftarrow{\$} \mathbb{G}^{n_z+n_v}$ and computing $\{c_j\}_{j=1}^m$ as per (5.7).

We now turn to the simulation-sound binding property and show that, if there exists a PPT adversary $\mathcal{A}$ who breaks this property with non-negligible advantage ε , there exists a non-independent Type I forger $\mathcal{B}$ against the signature scheme.

Concretely, our adversary $\mathcal{B}$ obtains a public key pk from his/her own challenger and sends the commitment key $pk = \text{pk}$ to $\mathcal{A}$. Whenever $\mathcal{A}$ sends a query $(\text{commit}, \text{tag})$ to the $\mathcal{O}_{tk, pk}$ oracle, $\mathcal{B}$ faithfully runs the **SSTC.FakeCom** algorithm and thus computes $\widetilde{\text{com}} = \{\tilde{c}_j\}_{j=1}^m$ according to (5.6) for randomly chosen $(\hat{M}_1, \dots, \hat{M}_n) \xleftarrow{\$} \mathbb{G}^n$, $\hat{\text{dec}} = (\hat{Z}_1, \dots, \hat{Z}_{n_z}, \hat{V}_1, \dots, \hat{V}_{n_v}) \xleftarrow{\$} \mathbb{G}^{n_z+n_v}$ and retains the information $\text{aux} = ((\hat{M}_1, \dots, \hat{M}_n), \hat{\text{dec}})$. When the oracle $\mathcal{O}_{tk, pk}$ subsequently receives a query of the form $(\text{decommit}, \widetilde{\text{com}}, (M_1, \dots, M_n))$, the reduction $\mathcal{B}$ invokes its own signing oracle on the input $(\text{tag}, (M_1/\hat{M}_1, \dots, M_n/\hat{M}_n))$. Upon receiving the resulting signature $(Z'_1, \dots, Z'_{n_z}, V'_1, \dots, V'_{n_v})$, $\mathcal{B}$ computes and returns $\widetilde{\text{dec}} = (\hat{Z}_1 \cdot Z'_1, \dots, \hat{Z}_{n_z} \cdot Z'_{n_z}, \hat{V}_1 \cdot V'_1, \dots, \hat{V}_{n_v} \cdot V'_{n_v})$.

Eventually, the adversary $\mathcal{A}$ outputs a commitment of his/her own $\text{com}^* = (c_1^*, \dots, c_m^*)$ along with valid openings $\text{dec} = (Z_1, \dots, Z_{n_z}, V_1, \dots, V_{n_v})$, $\text{dec}' = (Z'_1, \dots, Z'_{n_z}, V'_1, \dots, V'_{n_v})$ to distinct vectors $(M_1, \dots, M_n) \neq (M'_1, \dots, M'_n)$ for some tag tag^* that has never been used in *any* query to $\mathcal{O}_{tk, sk}$. Since both openings successfully pass the verification test, we find that

$$(Z_1/Z'_1, \dots, Z_{n_z}/Z'_{n_z}, V_1/V'_1, \dots, V_{n_v}/V'_{n_v})$$

forms a valid homomorphic signature on the vector $(M_1/M'_1, \dots, M_n/M'_n) \neq (1_{\mathbb{G}}, \dots, 1_{\mathbb{G}})$ for the identifier $\tau^* = \text{tag}^*$. By construction, τ^* was never the input of a signing query made by $\mathcal{B}$ to his/her own oracle. Consequently, $\mathcal{B}$ is indeed a Type I non-independent forger with advantage ε . $\square$

Chapter 6

Quasi-Adaptive Non Interactive Zero Knowledge Proofs

Contents

6.1	Definition and Security Notions	156
6.1.1	Quasi-Adaptive NIZK Proofs	157
6.1.2	Simulation Soundness and Relative Soundness	158
6.1.3	Quasi-Adaptive Language for Linear Subspaces	159
6.2	Unbounded Simulation-Sound Arguments	159
6.2.1	Realization for Linear Subspace Membership	159
6.2.2	Proof of the Simulation-Sound Argument	162
6.3	Single-Theorem Relatively Sound Arguments	167
6.3.1	Single-Theorem Linear Subspace Membership Proof	167
6.3.2	Security of the Relatively Sound Argument	168
6.4	Extensions for the k-Linear Assumption	172
6.5	Comparison of Linear Subspace Arguments	174

Cryptographic proof systems are central building blocks to turn protocols into *verifiable* systems without damaging their security. The description of Section 2.5 recalls that they consist of a preponderant place where verifiability and privacy meet each other. In particular, non-interactive zero-knowledge proofs play a fundamental role in the design of numerous cryptographic protocols. Unfortunately, until the breakthrough made by the Groth-Sahai techniques, detailed in (2.5.3), it was not known how to construct them efficiently without making use of the random oracle methodology. Initially, efficient methods employed the Fiat-Shamir heuristics.

As we saw in the first part, the Groth-Sahai methodology describes non-interactive witness-indistinguishable (NIWI) and zero-knowledge (NIZK) proof

systems for algebraic relations in groups equipped with a bilinear map. Since 2008, it has been the most efficient in constructing such proofs for algebraic relations in the standard model. For the important task of proving membership in linear subspaces, Jutla and Roy have quite recently given significantly more efficient proofs in the quasi-adaptive setting (QA-NIZK). The difference between ordinary NIZK systems is that the common reference string (CRS) may depend on the considered language in this model. However, a single simulator must work for all languages. Even though it is a relaxed model, they pointed out that QA-NIZK proofs are sufficient for many applications of Groth-Sahai proofs. Our constructions in the Part IV will further emphasize this observation.

Chapter 5 pointed out that structure-preserving signatures with linearly homomorphic properties have unexpected applications in the design of non-malleable structure-preserving commitments. In this chapter, we greatly extend their range of applications by demonstrating that they can surprisingly be used – albeit non-generically – in the design of strongly non-malleable primitives like simulation-sound proofs (Definition 18).

Concretely, we describe unbounded simulation-sound QA-NIZK proofs of *constant-size* for linear subspaces. The length of a proof does not depend on the number of equations or the number of variables, but only on the underlying assumption. Somewhat surprisingly, they are even asymptotically shorter than random-oracle-based proofs derived from Σ -protocols.

For applications like CCA2 security, where only one-time simulation-soundness is needed, we further optimize our proof system and obtain a relatively simulation-sound QA-NIZK proof system, as defined in [94]. Under the DLIN assumption (resp. the k -linear assumption), we achieve this result with only 4 (resp. $k + 2$) group elements!

6.1 Definition and Security Notions

In this section we first recall the novel notion of quasi-adaptive non-interactive zero knowledge (NIZK) proofs, of Jutla and Roy [95], for probability distributions on parametrized languages. It is quasi-adaptive in the sense that the common reference string (CRS) generator can generate the CRS depending on the language parameters. However, the simulation is required to be uniform, i.e., a single efficient simulator should work for the whole class of parametrized languages. Next, for simulation-sound proofs (Definition 18) we first recall the extended notion of *unbounded* simulation soundness of Sahai [130], and secondly the *relative* simulation soundness of Jutla and Roy [94] for the one-time notion but where proof systems generate a trapdoor key that allows private verification. Finally, in the quasi-adaptive setting we introduce our notations for distributions on languages that are linear subspaces of vector spaces.

6.1.1 Quasi-Adaptive NIZK Proofs

Quasi-Adaptive NIZK (QA-NIZK) proofs are NIZK proofs where the CRS is allowed to depend on the specific language for which proofs have to be generated. The CRS is divided into a fixed part Γ , produced by an algorithm K_0 , and a language-dependent part ψ . However, there should be a single simulator for the entire class of languages.

Let λ be a security parameter. For public parameters Γ produced by the algorithm K_0 , let $\mathcal{D}_\Gamma$ be a probability distribution over a collection of relations $\mathcal{R} = \{R_\rho\}$ with an associated language $\mathcal{L}_\rho = \{x \mid \exists w : R_\rho(x, w) = 1\}$ parametrized by a string ρ .

We consider proof systems where the prover and the verifier both take a label lbl as additional input. For example, this label can be the message-carrying part of an Elgamal-like encryption.

Definition 38 (Quasi-Adaptive NIZK Proofs) *A quadruple of efficient algorithms (K_0, K_1, P, V) is a QA-NIZK proof system for above relation $\mathcal{R}$ if there exists a PPT simulator (S_1, S_2) such that, for any PPT adversaries $\mathcal{A}_1, \mathcal{A}_2$ and $\mathcal{A}_3$, we have the following properties;*

Quasi-Adaptive Completeness:

$$\Pr[\Gamma \leftarrow K_0(\lambda); \rho \leftarrow D_\Gamma; \psi \leftarrow K_1(\Gamma, \rho); (x, w, \text{lbl}) \leftarrow \mathcal{A}_1(\Gamma, \psi, \rho); \\ \pi \leftarrow P(\psi, x, w, \text{lbl}) : V(\psi, x, \pi, \text{lbl}) = 1 \mid R_\rho(x, w) = 1] = 1,$$

Quasi-Adaptive Soundness:

$$\Pr[\Gamma \leftarrow K_0(\lambda); \rho \leftarrow D_\Gamma; \psi \leftarrow K_1(\Gamma, \rho); (x, \pi, \text{lbl}) \leftarrow \mathcal{A}_2(\Gamma, \psi, \rho) : \\ V(\psi, x, \pi, \text{lbl}) = 1 \wedge \neg(\exists w : R_\rho(x, w) = 1)] \in \text{negl}(\lambda),$$

Quasi-Adaptive Zero-Knowledge:

$$\Pr[\Gamma \leftarrow K_0(\lambda); \rho \leftarrow D_\Gamma; (\psi, \tau_{sim}) \leftarrow S_1(\Gamma, \rho) : \mathcal{A}_3^{S_2(\psi, \tau_{sim}, \dots)}(\Gamma, \psi, \rho) = 1] \\ \approx \Pr[\Gamma \leftarrow K_0(\lambda); \rho \leftarrow D_\Gamma; \psi \leftarrow K_1(\Gamma, \rho) : \mathcal{A}_3^{P(\psi, \dots)}(\Gamma, \psi, \rho) = 1],$$

where

- $P(\psi, \cdot, \cdot, \cdot)$ emulates the actual prover. It takes as input (x, w) and lbl and outputs a proof π if $(x, w) \in R_\rho$. Otherwise, it outputs $\perp$.
- $S_2(\psi, \tau_{sim}, \cdot, \cdot, \cdot)$ is an oracle that takes as input (x, w) and lbl . It outputs a simulated proof $S_2(\psi, \tau_{sim}, x, \text{lbl})$ if $(x, w) \in R_\rho$ and $\perp$ if $(x, w) \notin R_\rho$.

We assume that the CRS ψ contains an encoding of ρ , which is thus available to V . The definition of Quasi-Adaptive Zero-Knowledge requires a single simulator for the entire family of relations $\mathcal{R}$.

6.1.2 Simulation Soundness and Relative Soundness

It is often useful to have a property called *simulation-soundness* which requires the adversary to be unable to prove false statements even after having seen simulated proofs for possibly false statements.

Definition 39 (Simulation-Sound QA-NIZK Proofs) A QA-NIZF proof (K_0, K_1, P, V) for above relation $\mathcal{R}$ and simulator (S_1, S_2) is simulation-sound if for any PPT adversary $\mathcal{A}_4$, we have the following property:

Unbounded Simulation-Soundness:

$$\begin{aligned} & \Pr[\Gamma \leftarrow K_0(\lambda); \rho \leftarrow D_\Gamma; (\psi, \tau_{sim}) \leftarrow S_1(\Gamma, \rho); \\ & \quad (x, \pi, \text{lbl}) \leftarrow \mathcal{A}_4^{S_2(\psi, \tau_{sim}, \dots)}(\Gamma, \psi, \rho) : V(\psi, x, \pi, \text{lbl}) = 1 \\ & \quad \wedge \neg(\exists w : R_\rho(x, w) = 1) \wedge (x, \pi, \text{lbl}) \notin Q] \in \text{negl}(\lambda), \end{aligned}$$

where the adversary is allowed unbounded access to an oracle $S_2(\psi, \tau_{sim}, \dots)$ that takes as input statement-label pairs (x, lbl) (where x may be outside $\mathcal{L}_\rho$) and outputs simulated proofs $\pi \leftarrow S_2(\psi, \tau_{sim}, x, \text{lbl})$ before updating the set $Q = Q \cup \{(x, \pi, \text{lbl})\}$, which is initially empty.

In the weaker notion of one-time simulation-soundness, only one query to the S_2 oracle is allowed.

In some applications, one may settle for a weaker notion, called *relative soundness* by Jutla and Roy [94], which allows for more efficient proofs, especially in the single-theorem case. Informally, relatively sound proof systems involve both a public verifier and a private verification algorithm, which has access to a trapdoor. For hard languages, the two verifiers should almost always agree on any adversarially-created proof. Moreover, the private verifier should not accept a non-trivial proof for a false statement, even if the adversary has already seen proofs for false statements.

Definition 40 (Single-Theorem Relatively Sound QA-NIZK) A labelled single-theorem relatively sound QA-NIZK proof system is comprised of a quasi-adaptive labelled proof system (K_0, K_1, P, V) along with an efficient private verifier W and an efficient simulator (S_1, S_2) . Moreover, the following properties should hold for any PPT adversaries $(\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4)$:

Quasi Adaptive Relative Single-Theorem Zero-Knowledge:

$$\begin{aligned} & \Pr[\Gamma \leftarrow K_0(\lambda); \rho \leftarrow D_\Gamma; (\psi, \tau) \leftarrow S_1(\Gamma, \rho); (x, w, \text{lbl}, s) \leftarrow \mathcal{A}_1^{W(\psi, \tau, \dots)}(\Gamma, \psi, \rho); \\ & \quad \pi \leftarrow S_2(\psi, \rho, \tau, x, \text{lbl}) : \mathcal{A}_2^{W(\psi, \tau, \dots)}(\pi, s) = 1] \\ & \approx \Pr[\Gamma \leftarrow K_0(\lambda); \rho \leftarrow D_\Gamma; \psi \leftarrow K_1(\Gamma, \rho); (x, w, \text{lbl}, s) \leftarrow \mathcal{A}_1^{V(\psi, \dots)}(\Gamma, \psi, \rho); \\ & \quad \pi \leftarrow P(\psi, \rho, x, w, \text{lbl}) : \mathcal{A}_2^{V(\psi, \dots)}(\pi, s) = 1], \end{aligned}$$

Here, $\mathcal{A}_1$ is restricted to choosing (x, w) such that $R_\rho(x, w) = 1$;

Quasi Adaptive Relative Single-Theorem Simulation-Soundness:

$$\begin{aligned}
& \Pr[\Gamma \leftarrow \mathbf{K}_0(\lambda); \rho \leftarrow D_\Gamma; (\psi, \tau) \leftarrow S_1(\Gamma, \rho); \\
& \quad (x, \text{lbl}, s) \leftarrow \mathcal{A}_3^{W(\psi, \tau, \dots)}(\Gamma, \psi, \rho); \pi \leftarrow S_2(\psi, \rho, \tau, x, \text{lbl}) : \\
& \quad (x', \text{lbl}', \pi') \leftarrow \mathcal{A}_4^{W(\psi, \tau, \dots)}(s, \pi) : (x, \pi, \text{lbl}) \neq (x', \pi', \text{lbl}') \\
& \quad \wedge \nexists w' \text{ s.t. } R_\rho(x', w') = 1 \wedge W(\psi, \tau, x', \text{lbl}', \pi') = 1] \in \text{negl}(\lambda)
\end{aligned}$$

Note that the definition of relative simulation-soundness does not require the adversary to provide a witness but the definition of single-theorem zero-knowledge does.

6.1.3 Quasi-Adaptive Language for Linear Subspaces

In the following, vectors are always considered as row vectors unless stated otherwise. If $\mathbf{A} \in \mathbb{Z}_p^{t \times n}$ is a matrix, we denote by $g^{\mathbf{A}} \in \mathbb{G}^{t \times n}$ the matrix obtained by exponentiating g using the entries of $\mathbf{A}$.

We consider public parameters $\Gamma = (\mathbb{G}, \mathbb{G}_T, g)$ consisting of bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ with a generator $g \in \mathbb{G}$. We will consider languages $\mathcal{L}_\rho = \{g^{\vec{x} \cdot \mathbf{A}} \in \mathbb{G}^n \mid \vec{x} \in \mathbb{Z}_p^t\}$ that are parametrized by $\rho = g^{\mathbf{A}} \in \mathbb{G}^{t \times n}$, where $\mathbf{A} \in \mathbb{Z}_q^{t \times n}$ is a $t \times n$ matrix of rank $t < n$.

As Jutla and Roy, we assume that the distribution $\mathcal{D}_\Gamma$ is efficiently samplable: there exists a PPT algorithm which outputs a pair $(\rho, \mathbf{A})$ describing a relation R_ρ and its associated language $\mathcal{L}_\rho$ according to $\mathcal{D}_\Gamma$. One example of such a distribution is obtained by picking a uniform matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{t \times n}$ – which has full rank with overwhelming probability – and setting $\rho = g^{\mathbf{A}}$.

6.2 Unbounded Simulation-Sound Arguments

For membership of the row space of a $t \times n$ matrix, the Jutla and Roy's QA-NIZK proofs save $\Omega(t)$ group elements compared to Groth-Sahai. Here, we give QA-NIZK proofs made of a *constant* number of group elements – regardless of the number of equations or the number of variables – and additionally prove them *unbounded* simulation-sound. Unlike previous unbounded simulation-sound Groth-Sahai-based proofs, our construction does not involve quadratic pairing product equations and does not rely on a chosen-ciphertext-secure encryption scheme. Instead, we build on our structure-preserving linearly homomorphic signatures described in Section 4.4.

6.2.1 Realization for Linear Subspace Membership

In our unbounded simulation-sound proofs, each QA-NIZK proof can be seen as a Groth-Sahai NIWI proof of knowledge of a one-time linearly homomorphic

signature on the vector that allegedly belongs to the linear subspace. Here, the NIWI proof is generated for a Groth-Sahai CRS that depends on the verification key of a one-time signature, the private key of which is used to sign the entire proof so as to prevent re-randomizations. The reason why it provides unbounded simulation-soundness is that, with non-negligible probability, the CRS is perfectly hiding on all simulated proofs and extractable in the adversarially-generated fake proof. At a high level, the system can be seen as a two-tier proof system made of a non-malleable proof of knowledge of a malleable proof of membership.

Technical Intuition. Specifically, the language-dependent CRS ψ contains one-time linearly homomorphic signatures on the rows of the matrix $\rho \in \mathbb{G}^{t \times n}$. For each vector $\vec{v} \in \mathcal{L}_\rho$, the prover can use the witness $\vec{x} \in \mathbb{Z}_p^t$ to derive and prove knowledge of a one-time homomorphic signature (z, r, u) on $\vec{v}$. This signature (z, r, u) is already a QA-NIZK proof of membership but it does not provide simulation-soundness. To acquire this property, we follow [115] and generate a NIWI proof of knowledge of (z, r, u) for a Groth-Sahai CRS that depends on the verification key of an ordinary one-time signature, exactly as we did for the file identifier in our completely context-hiding variant of the homomorphic signature. The latter's private key is used to sign the NIWI proof so as to prevent unwanted proof manipulations. Using the private key of the homomorphic one-time signature as a trapdoor, the simulator is also able to create proofs for vectors $\vec{v} \notin \mathcal{L}_\rho$. Due to the use of perfectly NIWI proofs, these fake proofs do not leak more information about the simulation key than the CRS does. At the same time, the CRS can be prepared in such a way that, with non-negligible probability, it becomes perfectly binding on an adversarially-generated proof, which allows extracting a non-trivial signature on a vector $\vec{v} \notin \mathcal{L}_\rho$.

The Scheme. Like in the Jutla and Roy's scheme, our quasi-adaptive NIZK proof system (K_0, K_1, P, V) is a split CRS construction in that K_1 can be divided into two algorithms (K_{10}, K_{11}) . The first one K_{10} outputs some state information s and a first CRS $\mathbf{CRS}_2$ which is only used by the verifier and does not depend on the language $\mathcal{L}_\rho$. The second part K_{11} of K_1 inputs the state information s and the output of Γ of K_0 and outputs $\mathbf{CRS}_1$ which is only used by the prover.

$K_0(\lambda)$: chooses groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$ with $g \xleftarrow{\$} \mathbb{G}$. Then, outputs $\Gamma = (\mathbb{G}, \mathbb{G}_T, g)$. The dimensions (t, n) of the matrix $\mathbf{A} \in \mathbb{Z}_p^{t \times n}$ can be either fixed or part of the language, so that t, n can be given as input to the CRS generation algorithm K_1 .

$K_1(\Gamma, \rho)$: parses Γ as $(\mathbb{G}, \mathbb{G}_T, g)$ and ρ as a matrix $\rho = (G_{i,j}) \in \mathbb{G}^{t \times n}$.

1. Generates a key pair $(\mathbf{pk}_{rand}, \mathbf{sk}_{rand})$ for the randomizable signature of Section 4.4.4 in order to sign vectors of $\mathbb{G}^n$. Namely, chooses $g_z, g_r, h_z, h_u \xleftarrow{\$} \mathbb{G}$ and performs the following.

- a. For $i = 1$ to n , picks $\chi_i, \gamma_i, \delta_i \xleftarrow{\$} \mathbb{Z}_p$ and computes $g_i = g_z^{\chi_i} g_r^{\gamma_i}$ and $h_i = h_z^{\chi_i} h_u^{\delta_i}$.
- b. Generates $L + 1$ Groth-Sahai common reference strings, for some $L \in \text{poly}(\lambda)$. To this end, chooses $f_1, f_2 \xleftarrow{\$} \mathbb{G}$ and define $\vec{f}_1 = (f_1, 1, g) \in \mathbb{G}^3$, $\vec{f}_2 = (1, f_2, g) \in \mathbb{G}^3$. Then, picks $\vec{f}_{3,i} \xleftarrow{\$} \mathbb{G}^3$ for $i = 0$ to L .

Lets $\mathbf{sk}_{rand} = \{(\chi_i, \gamma_i, \delta_i)\}_{i=1}^n$ be the private key and the matching public key is

$$\mathbf{pk}_{rand} = \left(g_z, g_r, h_z, h_u, \{(g_i, h_i)\}_{i=1}^n, \mathbf{f} = (\vec{f}_1, \vec{f}_2, \{\vec{f}_{3,i}\}_{i=0}^L) \right).$$

2. On the vectors $\vec{\rho}_i = (G_{i1}, \dots, G_{in}) \in \mathbb{G}^n$ that form the rows of ρ , uses $\mathbf{sk}_{rand}$ to generate one-time linearly homomorphic signatures $\{(z_i, r_i, u_i)\}_{i=1}^t$. These are obtained as

$$(z_i, r_i, u_i) = \left(\prod_{j=1}^n G_{i,j}^{-\chi_j}, \prod_{j=1}^n G_{i,j}^{-\gamma_j}, \prod_{j=1}^n G_{i,j}^{-\delta_j} \right) \quad \forall i \in \{1, \dots, t\}.$$

3. Chooses a strongly unforgeable one-time signature $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ with verification keys consisting of L -bit strings.
4. The CRS $\psi = (\mathbf{CRS}_1, \mathbf{CRS}_2)$ consists of two parts which are defined as

$$\mathbf{CRS}_1 = \left(\rho, \mathbf{pk}_{rand}, \{(z_i, r_i, u_i)\}_{i=1}^t, \Sigma \right), \quad \mathbf{CRS}_2 = \left(\mathbf{pk}_{rand}, \Sigma \right),$$

while the simulation trapdoor τ_{sim} is $\mathbf{sk}_{rand} = \{(\chi_i, \gamma_i, \delta_i)\}_{i=1}^n$.

$P(\Gamma, \psi, \vec{v}, x, \text{lbl})$: given a candidate $\vec{v} \in \mathbb{G}^n$ and a witness $\vec{x} = (x_1, \dots, x_t) \in \mathbb{Z}_p^t$ such that $\vec{v} = g^{\vec{x} \cdot \mathbf{A}}$, generates a one-time signature key pair $(\text{SVK}, \text{SSK}) \leftarrow \mathcal{G}(\lambda)$ and does the following.

1. Using $\{(z_j, r_j, u_j)\}_{j=1}^t$, derives a one-time linearly homomorphic signature (z, r, u) on $\vec{v}$. Namely, computes $z = \prod_{i=1}^t z_i^{x_i}$, $r = \prod_{i=1}^t r_i^{x_i}$ and $u = \prod_{i=1}^t u_i^{x_i}$.
2. Using $\text{SVK} = \text{SVK}[1] \dots \text{SVK}[L] \in \{0, 1\}^L$, defines the vector $\vec{f}_{\text{SVK}} = \vec{f}_{3,0} \cdot \prod_{i=1}^L \vec{f}_{3,i}^{\text{SVK}[i]}$ and assembles a Groth-Sahai CRS $\mathbf{f}_{\text{SVK}} = (\vec{f}_1, \vec{f}_2, \vec{f}_{\text{SVK}})$. Using $\mathbf{f}_{\text{SVK}}$, generate commitments $\vec{C}_z, \vec{C}_r, \vec{C}_u$ to the components of $(z, r, u) \in \mathbb{G}^3$ along with NIWI proofs $(\vec{\pi}_1, \vec{\pi}_2)$ that $\vec{v}$ and (z, r, u) satisfy (4.9). Lets $(\vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2) \in \mathbb{G}^{15}$ be the resulting commitments and proofs.
3. Generates $\sigma = \mathcal{S}(\text{SSK}, (\vec{v}, \vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2, \text{lbl}))$ and outputs

$$\pi = (\text{SVK}, \vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2, \sigma) \tag{6.1}$$

$V(\Gamma, \psi, \vec{v}, \pi, \text{lbl})$: parses π as per (6.1) and returns 1 if (i) $\mathcal{V}(\text{SVK}, (\vec{v}, \vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2, \text{lbl}), \sigma) = 1$; (ii) $(\vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2)$ forms a valid NIWI proof for the CRS $\mathbf{f}_{\text{SVK}} = (\vec{f}_1, \vec{f}_2, \vec{f}_{\text{SVK}})$ (see (4.15) for the detailed verification equations). If either condition fails to hold, returns 0.

In order to simulate a proof for a given vector $\vec{v} \in \mathbb{G}^n$, the simulator uses $\tau_{\text{sim}} = \text{sk}_{\text{rand}}$ to generate a fresh one-time homomorphic signature on $\vec{v} \in \mathbb{G}^n$ and proceeds as in steps 2-3 of algorithm P.

The proof π only consists of 15 group elements and a one-time pair (SVK, σ) . Remarkably, its length does not depend on the number of equations n or the number of variables t . In comparison, Groth-Sahai proofs already require $3t+2n$ group elements in their basic form and become even more expensive when it comes to achieving unbounded simulation-soundness. The Jutla-Roy techniques [95] reduce the proof length to $2(n-t)$ elements – which only competes with our proofs when $t \approx n$ – but it is unclear how to extend them to get unbounded simulation-soundness without affecting their efficiency. Our CRS consists of $O(t+n+L)$ group elements against $O(t(n-t))$ in [95]. More detailed comparisons are given in Section 6.5 between proof systems based on the DLIN assumption.

Interestingly, the above scheme even outperforms Fiat-Shamir-like proofs derived from Σ -protocols which would give $\Theta(t)$ -size proofs here. The construction readily extends to rely on the k -linear assumption for $k > 2$. In this case, the proof comprises $(k+1)(2k+1)$ elements and its size thus only depends on k , as detailed in Appendix 6.4.

Moreover, the verification algorithm only involves *linear* pairing product equations whereas all known unbounded simulation-sound extensions of Groth-Sahai proofs require either quadratic equations or a linearization step involving extra variables.

We finally remark that, if we give up the simulation-soundness property, the proof length drops to $k+1$ group elements under the k -linear assumption.

6.2.2 Proof of the Simulation-Sound Argument

The security of the above scheme is stated by the following theorem.

Theorem 20 *The scheme is an unbounded simulation-sound QA-NIZK proof system if the DLIN assumption holds in $\mathbb{G}$ and Σ is strongly unforgeable.*

Proof. The quasi-adaptive completeness property follows directly from the correctness of the randomizable linearly homomorphic signature of Section 4.4.4. We thus focus on the quasi-adaptive zero-knowledge and quasi-adaptive unbounded simulation-soundness properties.

Quasi-Adaptive Zero-Knowledge. To prove this property we consider a sequence of two games which begins with a game where the adversary has

access to a real prover P on a real CRS ψ . In the second game, the adversary will be faced with a simulator (S_1, S_2) .

Game₁: is a game where the adversary $\mathcal{A}$ is given the description of the language $\mathcal{L}_\rho$ and is granted access to a real CRS ψ and an actual prover $P(\psi, \cdot, \cdot)$ which takes as input a vector $\vec{v}$ along with a witness $\vec{x} \in \mathbb{Z}_p^t$ such that $\vec{v} = g^{\vec{x} \cdot \mathbf{A}}$. At each invocation, the oracle outputs a genuine proof π by running the legal P algorithm. The adversary is allowed to query $P(\psi, \cdot, \cdot)$ a polynomial number of times and eventually outputs a bit $\beta \in \{0, 1\}$. We denote by S_1 the event that $\beta = 1$.

Game₂: This game is identical to **Game₁** with the difference that, when the $P(\psi, \cdot, \cdot)$ oracle is queried on a pair $(\vec{v}, \vec{x})$, it does not use the witness $\vec{x} \in \mathbb{Z}_p^t$ anymore at step 1 of the proving algorithm. Instead, it uses the private key $\text{sk}_{\text{rand}} = \{(\chi_i, \gamma_i, \delta_i)\}_{i=1}^n$ to compute a one-time signature $(z, r, u) = (\prod_{j=1}^n v_j^{-\chi_j}, \prod_{j=1}^n v_j^{-\gamma_j}, \prod_{j=1}^n v_j^{-\delta_j})$ on the vector $\vec{v} = (v_1, \dots, v_n) \in \mathbb{G}^n$. The remaining parts of π are generated as in the real $P(\psi, \cdot, \cdot)$ oracle in steps 2 and 3 of the proof generation algorithm. Although, the witness $\vec{x} \in \mathbb{Z}_p^t$ is never used, it is easy to see that (z, r, u) has exactly the same distribution as in **Game₂** if $\vec{v} \in \mathcal{L}_\rho$ (i.e., as long as $\vec{v} = g^{\vec{x} \cdot \mathbf{A}}$ for some $\vec{x} \in \mathbb{Z}_p^t$). We thus have $\Pr[S_2] = \Pr[S_1]$.

We define the simulator (S_1, S_2) by having S_1 generate the CRS ψ as in **Game₁** (so that ψ has the same distribution as the real CRS) and letting S_2 generate proofs without using the witnesses as in **Game₂**. It easily comes that the system is perfectly quasi-adaptive zero-knowledge as, for all language members $\vec{v} \in \mathbb{G}^n$, simulated proofs are distributed as real proofs.

Quasi-Adaptive Unbounded Simulation-Soundness. To prove this property, we proceed again with a sequence of games.

Game₁: is the real game where the adversary $\mathcal{A}$ is given the description of the language $\mathcal{L}_\rho$ and is granted access to a simulated CRS ψ and a simulated prover $S_2(\psi, \tau_{\text{sim}}, \cdot, \cdot)$ which takes as input a vector-label pair $(\vec{v}, \text{lbl})$ and returns a simulated proof π that $\vec{v} \in \mathcal{L}_\rho$. To generate $\rho \in \mathbb{G}^{t \times n}$ according to the distribution D_Γ , the challenger generates a matrix $\mathbf{A} \in \mathbb{Z}_p^{t \times n}$ with the appropriate distribution (recall that D_Γ is efficiently samplable by hypothesis) and computes $\rho = g^{\mathbf{A}}$. In addition, the challenger $\mathcal{B}$ computes a basis $\mathbf{W} \in \mathbb{Z}_p^{n \times (n-t)}$ of the right kernel of $\mathbf{A}$ and retains it for later use. The adversary is allowed to query the simulated prover $S_2(\psi, \tau_{\text{sim}}, \cdot, \cdot)$ on polynomially many occasions. The game ends with the adversary $\mathcal{A}$ outputting an element $\vec{v}^*$, a proof π^* and a label lbl^* . The adversary is deemed successful if $\vec{v}^* \notin \mathcal{L}_\rho$ (i.e., $\vec{v}^*$ is not in the row space of $\rho \in \mathbb{G}^{t \times n}$) but (π^*, lbl^*) is a valid proof. We denote by S_1 the latter event, which $\mathcal{B}$ can recognize by testing if $\vec{v} = (v_1, \dots, v_n) \in \mathbb{G}^n$ satisfies $\prod_{j=1}^n v_j^{w_{ji}} = 1_{\mathbb{G}}$.

for each column $\vec{w}_i^\top = (w_{1i}, \dots, w_{ni})^\top$ of $\mathbf{W}$. Indeed, the vectors $\mathbf{y} \in \mathbb{Z}_p^n$ in the row space of $\mathbf{A}$ are exactly those for which $\mathbf{y} \cdot \mathbf{W} = \vec{0}$.

Game₂: This game is identical to **Game₁** but the challenger $\mathcal{B}$ aborts if the adversary $\mathcal{A}$ outputs a fake proof π^* that recycles one of the one-time verification keys appearing in outputs of the $S_2(\psi, \tau_{sim}, \dots)$ oracle. Clearly, **Game₂** and **Game₁** are identical until the latter event occurs and this event contradicts the strong unforgeability of Σ : if q denotes the number of queries to $S_2(\psi, \tau_{sim}, \dots)$, a standard argument shows that $|\Pr[S_2] - \Pr[S_1]| \leq q \cdot \text{Adv}^{\text{uf-ots}}(\mathcal{B})$.

Game₃: This game is identical to **Game₂** but we modify the generation of pk_{rand} when the public key is set up. Namely, the vectors $(\vec{f}_1, \vec{f}_2, \{\vec{f}_{3,i}\}_{i=0}^L)$ are chosen by setting $\vec{f}_1 = (f_1, 1_{\mathbb{G}}, g)$ and $\vec{f}_2 = (1_{\mathbb{G}}, f_2, g)$ where $f_1, f_2 \xleftarrow{\$} \mathbb{G}$ are chosen at random. As for $\{\vec{f}_{3,i}\}_{i=0}^L$, they are obtained as

$$\begin{aligned} \vec{f}_{3,0} &= \vec{f}_1^{\xi_{0,1}} \cdot \vec{f}_2^{\xi_{0,2}} \cdot (1, 1, g)^{\xi_{0,3}} \cdot (1, 1, g)^{\mu \cdot \zeta - \rho_0} \\ \vec{f}_{3,i} &= \vec{f}_1^{\xi_{i,1}} \cdot \vec{f}_2^{\xi_{i,2}} \cdot (1, 1, g)^{\xi_{i,3}} \cdot (1, 1, g)^{-\rho_i}, \end{aligned} \quad i \in \{1, \dots, L\} \quad (6.2)$$

with $\mu \xleftarrow{\$} \{0, \dots, L\}$, $\xi_{0,1}, \xi_{1,1}, \dots, \xi_{L,1} \xleftarrow{\$} \mathbb{Z}_p$, $\xi_{0,2}, \xi_{1,2}, \dots, \xi_{L,2} \xleftarrow{\$} \mathbb{Z}_p$, $\xi_{0,3}, \xi_{1,3}, \dots, \xi_{L,3} \xleftarrow{\$} \mathbb{Z}_p$ and $\rho_0, \rho_1, \dots, \rho_L \xleftarrow{\$} \{0, \dots, \zeta - 1\}$, with $\zeta = 2(q + 1)$ and where q is the number of queries to the $S_2(\psi, \tau_{sim}, \dots)$ oracle. Note that this change is only conceptual since the distribution of $\{\vec{f}_{3,i}\}_{i=0}^L$ has not changed since **Game₂**. We thus have $\Pr[S_3] = \Pr[S_2]$.

Game₄: This game is like **Game₃** but we consider an event **Good** which causes the challenger $\mathcal{B}$ to abort if it does *not* occur. Let $\text{SVK}_1, \dots, \text{SVK}_q$ be the distinct one-time verification keys appearing in outputs of the S_2 oracle throughout the game. Let also SVK^* be the verification key involved in the fake proof π^* produced by $\mathcal{A}$. We know that $\text{SVK}^* \notin \{\text{SVK}_1, \dots, \text{SVK}_q\}$ unless the failure event introduced in **Game₂** occurs. For each verification key $\text{SVK} \in \{0, 1\}^L$, we consider the function $J(\text{SVK}) = \mu \cdot \zeta - \rho_0 - \sum_{i=1}^L \rho_i \cdot \text{SVK}[i]$, where $\{\rho_i\}_{i=0}^L$ are the values internally defined by the simulator in **Game₃**. We also define **Good** to be the event that

$$J(\text{SVK}^*) = 0 \quad \wedge \quad \bigwedge_{j \in \{1, \dots, q\}} J(\text{SVK}_j) \neq 0. \quad (6.3)$$

We remark that the random exponents $\rho_0, \rho_1, \dots, \rho_L$ are chosen independently of $\mathcal{A}$'s view: this means that the simulator could equivalently define $\{\vec{f}_{3,i}\}_{i=0}^L$ first and only choose $\{\rho_i\}_{i=0}^L$ – together with values $\{\xi_{3,i}\}_{i=0}^L$ explaining the $\{\vec{f}_{3,i}\}_{i=0}^L$ – at the very end of the game, when $\text{SVK}^*, \text{SVK}_1, \dots, \text{SVK}_q, \text{SVK}$ have been defined. The same analysis as [138] (using the simplifications of Bellare and Ristenpart [14, Theorem 3.1]) shows that $\Pr[S_4 \wedge \text{Good}] \geq \Pr[S_3]^2 / (27 \cdot (q + 1) \cdot (L + 1))$.

This follows from the fact that, for any set of queries, a lower bound on the probability of event **Good** is $1/(2q(L + 1))$. Indeed, from the known results [138, 88] on the programmability of Waters' hash function, we know

that the probability, taken over the choice of $(\mu, \rho_0, \dots, \rho_L)$, to meet the conditions (6.3) is at least $1/(2q(L+1))$.

Game₅: We modify again the way to compute $\mathbf{pk}_{rand}$ in the generation of the public key. Namely, the vectors $\vec{f}_1 = (f_1, 1_{\mathbb{G}}, g)$, $\vec{f}_2 = (1_{\mathbb{G}}, f_2, g)$ are chosen as before. However, instead of generating $\{\vec{f}_{3,i}\}_{i=0}^L$ as in Game₄, we set them as

$$\begin{aligned} \vec{f}_{3,0} &= \vec{f}_1^{\xi_{0,1}} \cdot \vec{f}_2^{\xi_{0,2}} \cdot (1, 1, g)^{\mu \cdot \zeta - \rho_0} \\ \vec{f}_{3,i} &= \vec{f}_1^{\xi_{i,1}} \cdot \vec{f}_2^{\xi_{i,2}} \cdot (1, 1, g)^{-\rho_i}, \quad i \in \{1, \dots, L\} \end{aligned} \quad (6.4)$$

which amounts to setting $\xi_{0,3} = \xi_{1,3} = \dots = \xi_{L,3} = 0$. Clearly, $\{\vec{f}_{3,i}\}_{i=0}^L$ are no longer uniform in the span of $(\vec{f}_1, \vec{f}_2, (1, 1, g))$. Still, this change should not be noticeable to $\mathcal{A}$ if the DLIN assumption holds in $\mathbb{G}$. Concretely, if the adversary wins (recall that the challenger can still detect this event using the matrix $\mathbf{W} \in \mathbb{Z}_p^{n \times (n-t)}$ as explained in Game₁) with substantially different probabilities in Game₅ and Game₄, we can construct a DLIN distinguisher $\mathcal{B}^{\text{DLIN}}$ in the group $\mathbb{G}$. This distinguisher uses the random self-reducibility of DLIN to construct many independent-looking instances from the same distribution out of a given instance. The distinguisher then runs $\mathcal{A}$ on input of the CRS that was generated using the DLIN instances and it eventually outputs 1 if the adversary wins. We can thus write $|\Pr[S_5 \wedge \text{Good}] - \Pr[S_4 \wedge \text{Good}]| \leq \mathbf{Adv}_{\mathcal{B}^{\text{DLIN}}}(\lambda)$.

In Game₅, we show that a successful adversary implies an algorithm $\mathcal{B}$ solving a given SDP instance (g_z, g_r, h_z, h_u) with non-negligible probability, which *a fortiori* breaks the DLIN assumption in $\mathbb{G}$.

By hypothesis, we know that $\mathcal{A}$ manages to create a proof

$$\pi^* = (\text{SVK}^*, \vec{C}_z^*, \vec{C}_r^*, \vec{C}_u^*, \vec{\pi}_1^*, \vec{\pi}_2^*, \sigma^*)$$

for a vector $\vec{v}^* = (v_1^*, \dots, v_n^*)$ outside the row space of ρ but $(\vec{C}_z^*, \vec{C}_r^*, \vec{C}_u^*, \vec{\pi}_1^*, \vec{\pi}_2^*) \in \mathbb{G}^{15}$ and σ^* satisfy the verification equations. At this point, if the event **Good** introduced in Game₄ occurs, we must have $J(\text{SVK}^*) = 0$, which implies that $\vec{f}_{\text{SVK}^*} = \vec{f}_{3,0} \cdot \prod_{i=1}^{L+1} \vec{f}_{3,i}^{\text{SVK}^*[i]}$ lies in $\text{span}(\vec{f}_1, \vec{f}_2)$. Consequently, $\vec{C}_z^*$, $\vec{C}_r^*$ and $\vec{C}_u^*$ are necessarily perfectly binding and extractable commitments. Using the discret log's $(\log_g(f_1), \log_g(f_2))$, algorithm $\mathcal{B}$ can thus extract the committed group elements $(z^*, r^*, u^*) \in \mathbb{G}^3$ by BBS-decrypting the ciphertexts $(\vec{C}_z^*, \vec{C}_r^*, \vec{C}_u^*)$. Since $(\vec{\pi}_1^*, \vec{\pi}_2^*)$ are perfectly sound Groth-Sahai proofs, the extracted elements (z^*, r^*, u^*) necessarily satisfy

$$\begin{aligned} 1_{\mathbb{G}_T} &= e(g_z, z^*) \cdot e(g_r, r^*) \cdot \prod_{i=1}^n e(g_i, v_i^*), \\ 1_{\mathbb{G}_T} &= e(h_z, z^*) \cdot e(h_u, u^*) \cdot \prod_{i=1}^n e(h_i, v_i^*). \end{aligned} \quad (6.5)$$

Having extracted (z^*, r^*, u^*) , $\mathcal{B}$ also computes

$$z^\dagger = \prod_{i=1}^n v_i^{*-x_i}, \quad r^\dagger = \prod_{i=1}^n v_i^{*-y_i}, \quad u^\dagger = \prod_{i=1}^n v_i^{*-d_i}, \quad (6.6)$$

so that $(z^\dagger, r^\dagger, u^\dagger)$ also satisfies (6.5). Since $(z^\dagger, r^\dagger, u^\dagger)$ and (z^*, r^*, u^*) both satisfy (6.5), the triple

$$(z^\dagger, r^\dagger, u^\dagger) = \left(\frac{z^*}{z^\dagger}, \frac{r^*}{r^\dagger}, \frac{u^*}{u^\dagger} \right)$$

necessarily satisfies $e(g_z, z^\dagger) \cdot e(g_r, r^\dagger) = e(h_z, z^\dagger) \cdot e(h_u, u^\dagger) = 1_{\mathbb{G}_T}$. To conclude the proof, we argue that $z^\dagger \neq 1_{\mathbb{G}}$ with overwhelming probability.

To do this, we observe that, if the event **Good** defined in **Game**₄ actually comes about, then $\mathcal{B}$ never leaks more information about $(\chi_1, \dots, \chi_n)$ than $\mathcal{A}$ can infer by just observing $\{(z_j, r_j, u_j)\}_{j=1}^t$ in the public key. Indeed, in this case we have $J(\text{SVK}^*) = 0$ and $J(\text{SVK}_j) \neq 0$ for each $j \in \{1, \dots, q\}$. This means that, in the simulated proofs returned by $\text{S}_2(\psi, \tau_{\text{sim}}, \cdot, \cdot)$, the proofs $(\vec{\pi}_1, \vec{\pi}_2)$ are perfectly witness-indistinguishable as they are generated for a perfectly hiding Groth-Sahai CRS. For these simulated proofs, the built-in homomorphic signatures $(\vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2)$ leak nothing about the specific vector $(\chi_1, \dots, \chi_n)$ used by $\mathcal{B}$. As a consequence, the same arguments as in [106, Theorem 1] show that $z^\dagger \neq z^*$ with probability $1 - 1/p$. Specifically, in the CRS, $\{(g_i, h_i)\}_{i=1}^n$ and $\{(z_i, r_i, u_i)\}_{i=1}^t$ provide the adversary with a system of $2n + t < 3n$ equations in $3n$ unknowns $\{(\chi_i, \gamma_i, \delta_i)\}_{i=1}^n$, which leaves $z^\dagger$ completely undetermined as long as $\vec{v}^*$ is linearly independent of the rows of $(G_{i,j})_{i,j}$. We thus find

$$\Pr[S_5 \wedge \text{Good}] = \text{Adv}_{\mathcal{B}}^{\text{SDP}}(\lambda) \cdot \left(1 - \frac{1}{p}\right)^{-1}.$$

In turn, in **Game**₅, $\mathcal{B}$ implies a PPT distinguisher $\mathcal{B}^{\text{DLIN}'}$ for the DLIN assumption such that we have the inequality $\Pr[S_5 \wedge \text{Good}] < \frac{1}{2} \cdot \text{Adv}_{\mathcal{B}^{\text{DLIN}'}}(\lambda) \cdot \left(1 - \frac{1}{p}\right)^{-1}$. If $\text{Adv}^{\text{DLIN}}(\lambda)$ denotes the maximal advantage of any PPT distinguisher against the DLIN assumption in $\mathbb{G}$, the probability of event $S_4 \wedge \text{Good}$ can be bounded as $\Pr[S_4 \wedge \text{Good}] \leq \frac{3}{2} \cdot \text{Adv}^{\text{DLIN}}(\lambda) \cdot \left(1 - \frac{1}{p}\right)^{-1}$ in **Game**₄. This in turn yields

$$\Pr[S_3] \leq 7 \cdot \sqrt{q \cdot (L + 1) \cdot \text{Adv}^{\text{DLIN}}(\lambda) \cdot \left(1 - \frac{1}{p}\right)^{-1}},$$

so that $\mathcal{A}$'s advantage in breaking the unbounded simulation-soundness of the system is at most

$$\text{Adv}^{\text{uss}}(\mathcal{A}) \leq \text{Adv}^{\text{suf-ots}}(\lambda) + 7 \cdot \sqrt{q \cdot (L + 1) \cdot \text{Adv}^{\text{DLIN}}(\lambda) \cdot \left(1 - \frac{1}{p}\right)^{-1}}.$$

□

We note that the above construction is not tightly secure as the gap between the simulation-soundness adversary's advantage and the probability to break the DLIN assumption depends on the number of simulated proofs obtained by the adversary. For applications like tightly secure public-key encryption [86], it would be interesting to modify the proof system to obtain tight security.

6.3 Single-Theorem Relatively Sound Arguments

In applications where single-theorem relatively sound NIZK proofs suffice, we can further improve the efficiency. Under the k -linear assumption, the proof length reduces from $O(k^2)$ elements to $O(k)$ elements. Under the DLIN assumption, each proof fits within 4 elements and only costs $2n + 6$ pairings to verify. In comparison, the verifier needs $2(n - t)(t + 2)$ pairing evaluations in [95].

6.3.1 Single-Theorem Linear Subspace Membership Proof

Even if relative soundness is a relaxed form of simulation-soundness, we need to add a private verifier in the system. As in [94], we achieve this property using smooth projective hash functions [50]. To this end, we need to encode the matrix $\rho \in \mathbb{G}^{t \times n}$ as a $2t \times (2n + 1)$ matrix.

The Scheme.

$K_0(\lambda)$: chooses groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$ with $g \xleftarrow{\$} \mathbb{G}$. Then, outputs $\Gamma = (\mathbb{G}, \mathbb{G}_T, g)$. Again, the dimensions of $\mathbf{A} \in \mathbb{Z}_p^{t \times n}$ can be either fixed or part of $\mathcal{L}_\rho$, so that t, n can be given as input to the CRS generation algorithm K_1 .

$K_1(\Gamma, \rho)$: parses Γ as $(\mathbb{G}, \mathbb{G}_T, g)$ and ρ as $\rho = (G_{ij})_{1 \leq i \leq t, 1 \leq j \leq n} \in \mathbb{G}^{t \times n}$ and performs the following.

1. Chooses two n -vectors $\vec{d} = (d_1, \dots, d_n) \xleftarrow{\$} \mathbb{Z}_p^n$ and $\vec{e} = (e_1, \dots, e_n) \xleftarrow{\$} \mathbb{Z}_p^n$ in order to define $\vec{W} = (W_1, \dots, W_t) = g^{\mathbf{A} \cdot \vec{d}^T} \in \mathbb{G}^t$ and $\vec{Y} = (Y_1, \dots, Y_t) = g^{\mathbf{A} \cdot \vec{e}^T} \in \mathbb{G}^t$. These will be used to define a projective hash function.
2. Generates a key pair $(\text{pk}_{ots}, \text{sk}_{ots})$ for the one-time linearly homomorphic signature of Section 4.4.2 in order to sign vectors in $\mathbb{G}^{2n+1}$. Lets $\text{pk}_{ots} = ((\mathbb{G}, \mathbb{G}_T), g_z, g_r, h_z, h_u, \{(g_i, h_i)\}_{i=1}^{2n+1})$ be the public key and lets $\text{sk}_{ots} = \{(\chi_i, \gamma_i, \delta_i)\}_{i=1}^{2n+1}$ be the corresponding private key.
3. Uses the secret key sk_{ots} to generate one-time linearly homomorphic signatures $\{(z_i, r_i, u_i)\}_{i=1}^{2t}$ on the independent vectors below, which are obtained from the rows of the matrix $\rho = (G_{i,j})_{1 \leq i \leq t, 1 \leq j \leq n}$.

$$\begin{aligned} \vec{H}_{2i-1} &= (G_{i,1}, \dots, G_{i,n}, Y_i, 1, \dots, 1) \in \mathbb{G}^{2n+1} \\ \vec{H}_{2i} &= (1, \dots, 1, W_i, G_{i,1}, \dots, G_{i,n}) \in \mathbb{G}^{2n+1} \end{aligned} \quad i \in \{1, \dots, t\}$$

4. Chooses a collision-resistant hash function $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p$.

5. The CRS ψ consists of a first part $\mathbf{CRS}_1$ that is only used by the prover and a second part $\mathbf{CRS}_2$ which is only used by the verifier. These are defined as

$$\begin{aligned}\mathbf{CRS}_1 &= \left(\rho, \text{pk}_{ots}, \vec{W}, \vec{Y}, \{(z_i, r_i, u_i)\}_{i=1}^{2t}, H \right), \\ \mathbf{CRS}_2 &= \left(\text{pk}_{ots}, \vec{W}, \vec{Y}, H \right).\end{aligned}$$

The simulation trapdoor τ_{sim} is sk_{ots} and the private verification trapdoor is $\tau_v = \{\vec{d}, \vec{e}\}$.

$P(\Gamma, \psi, \vec{v}, x, \text{lbl})$: given a candidate $\vec{v} \in \mathbb{G}^n$, a witness $\vec{x} = (x_1, \dots, x_t) \in \mathbb{Z}_p^t$ such that $\vec{v} = g^{\vec{x} \cdot \mathbf{A}}$ and a label lbl , computes $\alpha = H(\rho, \vec{v}, \text{lbl}) \in \mathbb{Z}_p$. Then, using $\{(z_i, r_i, u_i)\}_{i=1}^{2t}$, derives a one-time linearly homomorphic signature (z, r, u) on the vector $\vec{v} = (v_1, \dots, v_n, \pi_0, v_1^\alpha, \dots, v_n^\alpha) \in \mathbb{G}^{2n+1}$, where $\pi_0 = \prod_{i=1}^t (W_i^\alpha Y_i)^{x_i}$. Namely, computes and outputs the proof $\pi = (z, r, u, \pi_0) \in \mathbb{G}^4$, where

$$\begin{aligned}z &= \prod_{i=1}^t (z_{2i-1} \cdot z_{2i}^\alpha)^{x_i}, & r &= \prod_{i=1}^t (r_{2i-1} \cdot r_{2i}^\alpha)^{x_i}, \\ u &= \prod_{i=1}^t (u_{2i-1} \cdot u_{2i}^\alpha)^{x_i}, & \pi_0 &= \prod_{i=1}^t (W_i^\alpha Y_i)^{x_i}.\end{aligned}$$

$V(\Gamma, \psi, \vec{v}, \pi, \text{lbl})$: parses $\vec{v}$ as $(v_1, \dots, v_n) \in \mathbb{G}^n$ and π as $(z, r, u, \pi_0) \in \mathbb{G}^4$. Computes $\alpha = H(\rho, \vec{v}, \text{lbl})$ and returns 1 if and only if (z, r, u) is a valid signature on $\vec{v} = (v_1, \dots, v_n, \pi_0, v_1^\alpha, \dots, v_n^\alpha) \in \mathbb{G}^{2n+1}$. Namely, it should satisfy the equalities $1_{\mathbb{G}_T} = e(g_z, z) \cdot e(g_r, r) \cdot \prod_{i=1}^n e(g_i \cdot g_{i+n+1}^\alpha, v_i) \cdot e(g_{n+1}, \pi_0)$ and $1_{\mathbb{G}_T} = e(h_z, z) \cdot e(h_u, u) \cdot \prod_{i=1}^n e(h_i \cdot h_{i+n+1}^\alpha, v_i) \cdot e(h_{n+1}, \pi_0)$.

$W(\Gamma, \psi, \tau_v, \vec{v}, \pi, \text{lbl})$: given $\vec{v} = (v_1, \dots, v_n) \in \mathbb{G}^n$, parses π as $(z, r, u, \pi_0) \in \mathbb{G}^4$ and τ_v as $\{\vec{d}, \vec{e}\}$, with $\vec{d} = (d_1, \dots, d_n) \in \mathbb{Z}_p^n$ and $\vec{e} = (e_1, \dots, e_n) \in \mathbb{Z}_p^n$. Computes $\alpha = H(\rho, \vec{v}, \text{lbl}) \in \mathbb{Z}_p$ and returns 0 if the public verification test V fails. Otherwise, returns 1 if $\pi_0 = \prod_{j=1}^n v_j^{e_j + \alpha d_j}$ and 0 if not.

We note that, while the proving algorithm is deterministic, each statement has many valid proofs. However, finding two valid proofs for the same statement is computationally hard, as will be shown in the proof of Theorem 21. Indeed, the security proof inherits the argument used in the security of the underlying one-time linearly homomorphic signature in Section 4.4.2.

The scheme readily extends to rest on the k -linear assumption with $k > 2$. In this case, the proof requires $k + 2$ group elements – whereas combining the techniques of [94, 95] demands $k(n + 1 - t)$ elements per proof – and a CRS of size $O(k(n + t))$.

6.3.2 Security of the Relatively Sound Argument

The security of the above scheme is stated by the following theorem.

Theorem 21 *The above proof system is a relatively sound QA-NIZK proof system if the SDP assumption holds in $(\mathbb{G}, \mathbb{G}_T)$ and if H is a collision-resistant hash function.*

Proof. The completeness property is straightforward to verify. To establish the result, we separately prove the relative quasi-adaptive zero-knowledge and relative quasi-adaptive simulation-soundness properties.

Quasi-Adaptive Relative Zero-Knowledge. We consider a sequence of two games which begins with a game where the adversary has oracle access to the actual prover P and a public verifier on a real CRS ψ . In the last game, the adversary will be interacting with a simulator (S_1, S_2) and the private verifier.

Game₁: is a game where the adversary $\mathcal{A}$ is given the description ρ of the language $\mathcal{L}_\rho$ and a real CRS $\psi = (\mathbf{CRS}_1, \mathbf{CRS}_2)$. In addition, the adversary has access to a public verification oracle $V(\psi, \cdot, \cdot)$, even though it can run the verification algorithm by itself. This will be useful to show that the private verifier always agrees with the public one when it interacts with a PPT adversary. At some point, the adversary chooses a pair $(\vec{v}, |\mathbf{b}|)$ along with a witness $\vec{x} \in \mathbb{Z}_p^t$ such that $\vec{v} = g^{\vec{x} \cdot \mathbf{A}}$. The challenger replies by returning an actual proof π produced by running $P(\psi, \vec{v}, \vec{x}, |\mathbf{b}|)$. When $\mathcal{A}$ terminates, it outputs a bit $\beta \in \{0, 1\}$. We denote by S_1 the event that $\beta = 1$.

Game₂: is like Game₁ but the adversary's public verification oracle $V(\psi, \cdot, \cdot)$ is replaced by the private verification oracle $W(\psi, \tau_v, \cdot, \cdot)$. Since the private verification algorithm begins by running the public one, both games are clearly identical until $\mathcal{A}$ queries the verification oracle on input of a candidate $(\vec{v}, (z, r, u, \pi_0), |\mathbf{b}|)$ that is accepted by $V(\psi, \cdot, \cdot)$ but rejected by $W(\psi, \tau_v, \cdot, \cdot)$. If we call this event F_2 , we have $|\Pr[S_1] - \Pr[S_2]| \leq \Pr[F_2]$.

Claim 1.

The probability of event F_2 can be bounded as $\Pr[F_2] \leq \mathbf{Adv}^{\text{SDP}}(\lambda) + \frac{1}{p}$.

Proof of Claim 1. We first remark that event F_2 can only occur for a candidate $(\vec{v}, (z, r, u, \pi_0), |\mathbf{b}|)$ such that the vector $(v_1, \dots, v_n, \pi_0, v_1^\alpha, \dots, v_n^\alpha)$ is not in the span of $\{\vec{H}_{2i-1}, \vec{H}_{2i}\}_{i=1}^t$. Indeed, otherwise, there would exist $\vec{x} = (x_1, \dots, x_t) \in \mathbb{Z}_p^t$ such that $(v_1, \dots, v_n) = g^{\vec{x} \cdot \mathbf{A}}$ and $\pi_0 = \prod_{i=1}^t (W_i^\alpha Y_i)^{x_i}$. In this case, we would also have

$$g^{\vec{x} \cdot \mathbf{A} \cdot (\vec{e}^\top + \alpha \cdot \vec{d}^\top)} = \prod_{i=1}^t (W_i^\alpha Y_i)^{x_i},$$

and the private verifier $W(\psi, \tau_v, \cdot, \cdot)$ would accept the proof (z, r, u, π_0) .

It comes that the only way for the adversary to cause a divergence between $W(\psi, \tau_v, \cdot, \cdot)$ and $V(\psi, \cdot, \cdot)$ is to create a valid-looking one-time linearly homomorphic signature (z, r, u) on a vector outside $\text{span}(\{\vec{H}_{2i-1}, \vec{H}_{2i}\}_{i=1}^t)$.

The result of Section 4.4.2 shows that this occurs with probability at most $\Pr[F_2] \leq \mathbf{Adv}^{\text{SDP}}(\lambda) + 1/p$. ■

Game₃: This game is like Game₂ but, when the adversary outputs his triple $(\vec{v}, \vec{x}, \text{lbl})$, the challenger does not use the witness $\vec{x} \in \mathbb{Z}_p^t$ any longer. To simulate the proof for $(\vec{v}, \vec{x}, \text{lbl})$, he/she first computes $\alpha = H(\rho, \vec{v}, \text{lbl})$. Then, using the private vectors $\vec{d}, \vec{e} \in \mathbb{Z}_p^n$, he/she computes the element $\pi_0 = \prod_{j=1}^n v_j^{e_j + \alpha \cdot d_j}$ before using the private key $\text{sk}_{ots} = \{(\chi_i, \gamma_i, \delta_i)\}_{i=1}^{2n+1}$ to compute a one-time signature

$$(z, r, u) = \left(\prod_{j=1}^{2n+1} v_j^{-\chi_j}, \prod_{j=1}^{2n+1} v_j^{-\gamma_j}, \prod_{j=1}^{2n+1} v_j^{-\delta_j} \right)$$

on the vector $\vec{v} = (v_1, \dots, v_{2n+1})$, where $v_{n+1} = \pi_0$ and $v_{n+i+1} = v_i^\alpha$ for $i = 1$ to n . The resulting proof is easily seen to have the same distribution as in Game₂ when $\vec{v} \in \mathcal{L}_\rho$. We thus have $\Pr[S_3] = \Pr[S_2]$.

We define the simulator (S_1, S_2) by having S_1 generate the CRS ψ as in Game₃ (observe that it has not changed since Game₁) and S_2 generate the proof without using the witnesses as in Game₃. The verification oracle is implemented as in Game₂ and Game₃. It easily comes that the system is computationally quasi-adaptive relatively zero-knowledge if the SDP assumption holds.

Quasi-Adaptive Relative Simulation-Soundness. We have to prove that, even if the simulator (S_1, S_2) provides the adversary $\mathcal{A}$ with a simulated proof π for a pair $(\vec{v}, \text{lbl})$, where $\vec{v} \in \mathbb{G}^n$ may not be in $\mathcal{L}_\rho$, $\mathcal{A}$ will remain unable to produce a new proof $(\vec{v}^*, \pi^*, \text{lbl}^*) \neq (\vec{v}, \pi, \text{lbl})$ such that $\vec{v}^* \notin \mathcal{L}_\rho$.

To prove the result, we rely on the smoothness of the projective hash function and on a specific property of the one-time linearly homomorphic signature of Section 4.4.2: namely, unless the SDP assumption is false, it is hard to compute two distinct signatures on the same vector, even when the private key is available.

We thus proceed with a sequence of games where the first game is the actual game and the last one is a game where the adversary has statistically no advantage. In Game_{*i*}, we denote by S_i the event that the adversary wins.

Game₁: is the real game where the adversary $\mathcal{A}$ is given the description of $\mathcal{L}_\rho$, a simulated CRS ψ and access to a simulated prover $S_2(\psi, \tau_{sim}, \cdot, \cdot)$ which is queried only once. On this occasion, S_2 takes as input a vector $\vec{v}^\dagger \in \mathbb{G}^n$ and a label $\text{lbl}^\dagger$ and it produces a simulated proof $\pi^\dagger = (z^\dagger, r^\dagger, u^\dagger, \pi_0^\dagger)$ that $\vec{v}^\dagger \in \mathcal{L}_\rho$. To generate $\rho \in \mathbb{G}^{t \times n}$ according to the distribution D_Γ at the beginning of the game, the challenger computes $\rho = g^{\mathbf{A}}$ after having sampled a matrix $\mathbf{A} \in \mathbb{Z}_p^{t \times n}$ with the appropriate distribution (which is possible since D_Γ is efficiently samplable). Moreover, the challenger $\mathcal{B}$ computes a basis $\mathbf{W} \in \mathbb{Z}_p^{n \times (n-t)}$ of the right kernel of

A. The adversary is allowed to query $S_2(\psi, \tau_{sim}, \dots)$ exactly once and the private verifier $W(\psi, \tau_v, \dots)$ on polynomially many occasions. When $\mathcal{A}$ terminates, it outputs a triple $(\vec{v}^*, \pi^*, \text{lbl}^*)$. The adversary is successful if $\vec{v}^* \notin \mathcal{L}_\rho$ but (π^*, lbl^*) passes the private verification test and $(\vec{v}^*, \pi^*, \text{lbl}^*) \neq (\vec{v}^\dagger, \pi^\dagger, \text{lbl}^\dagger)$. We denote by S_1 the latter event. Note that $\mathcal{B}$ can recognize this event as it can test if $\vec{v}^* \notin \mathcal{L}_\rho$ by checking whether $\vec{v}^* = (v_1^*, \dots, v_n^*) \in \mathbb{G}^n$ satisfies $\prod_{j=1}^n v_j^{*w_{ji}} = 1_{\mathbb{G}}$ for each column $\vec{w}_i^\top = (w_{1i}, \dots, w_{ni})^\top$ of $\mathbf{W}$. Indeed, the vectors $\mathbf{y} \in \mathbb{Z}_p^n$ in the row space of $\mathbf{A}$ are exactly those for which $\mathbf{y} \cdot \mathbf{W} = \vec{0}$.

Game₂: In this game, we modify the behaviour of the private verification oracle $W(\psi, \tau_v, \dots)$. At each invocation (including the final invocation on the adversary's output $(\vec{v}^*, \pi^*, \text{lbl}^*)$) on input of a triple $(\vec{v}, \pi, \text{lbl})$, the modified private verification oracle outputs 0 if $(\vec{v}, \pi, \text{lbl}) \neq (\vec{v}^\dagger, \pi^\dagger, \text{lbl}^\dagger)$ but $H(\rho, \vec{v}, \text{lbl}) = H(\rho, \vec{v}^\dagger, \text{lbl}^\dagger)$. Clearly, **Game₁** and **Game₂** proceed identically until the latter event, called F_2 , occurs. We have $|\Pr[S_2] - \Pr[S_1]| \leq \Pr[F_2]$. Moreover, F_2 is unlikely to occur if H is a collision-resistant hash function: we have $\Pr[F_2] \leq \text{Adv}^{\text{CR}}(\lambda)$.

Game₃: This game is identical to **Game₂** with the following difference. When the private verification oracle $W(\psi, \tau_v, \dots)$ is run on input of the adversary's proof $(\vec{v}^*, \pi^*, \text{lbl}^*)$, it returns 0 in the event that $\pi^* = (z^*, r^*, u^*, \pi_0^*)$ is such that $(z^*, r^*, u^*) \neq (z^\dagger, r^\dagger, u^\dagger)$ but $(\vec{v}^*, \text{lbl}^*) = (\vec{v}^\dagger, \text{lbl}^\dagger)$. If we call F_3 the event that the private verification oracle $W(\psi, \tau_v, \dots)$ rejects a proof that would have been accepted in **Game₂**, we have $|\Pr[S_3] - \Pr[S_2]| \leq \Pr[F_3]$. Moreover, F_3 implies a breach in the SDP assumption. Indeed, if $\pi_0^* \neq \pi_0^\dagger$, $W(\psi, \tau_v, \dots)$ would not accept π^* in **Game₂** either, regardless of whether $(z^*, r^*, u^*) \neq (z^\dagger, r^\dagger, u^\dagger)$ or not. If $\pi_0^* = \pi_0^\dagger$, event F_3 provides the challenger with two distinct linearly homomorphic signatures (z^*, r^*, u^*) and $(z^\dagger, r^\dagger, u^\dagger)$ on the same vector $(v_1^*, \dots, v_n^*, \pi_0^*, v_1^{*\alpha^*}, \dots, v_n^{*\alpha^*})$ as we also have $\alpha^* = \alpha^\dagger$. As mentioned in Section 4.4.2 (and as can be easily observed from the verification equations), this would contradict the SDP assumption and we thus have $|\Pr[S_3] - \Pr[S_2]| \leq \text{Adv}^{\text{SDP}}(\lambda)$.

Game₄: In this game, we further modify the behaviour of $W(\psi, \tau_v, \dots)$ when it assesses the adversary's output $(\vec{v}^*, \pi^*, \text{lbl}^*)$. Using the basis $\mathbf{W}$ of the right kernel of $\mathbf{A}$, the challenger $\mathcal{B}$ first checks if $\vec{v}^* \notin \mathcal{L}_\rho$ and forces $W(\psi, \tau_v, \dots)$ to return 0 whenever this is the case. If we denote by F_4 the event that $W(\psi, \tau_v, \dots)$ rejects an adversarially-generated triple $(\vec{v}^*, \pi^*, \text{lbl}^*)$ that would have survived the private verification test of **Game₃**, we have $|\Pr[S_4] - \Pr[S_3]| \leq \Pr[F_4]$. Since $\Pr[S_4] = 0$ by construction, we are left with the task of bounding $\Pr[F_4]$.

Claim 2. The probability of event F_4 is at most $\Pr[F_4] \leq 1/(p - q)$, where q denotes the number of private verification queries.

Proof of Claim 2. The proof of the claim is a standard argument based on the smoothness of the projective hash function. If we consider the information

that $\mathcal{A}$ can infer about the private evaluation key $\tau_v = (\vec{d} = (d_1, \dots, d_n), \vec{e} = (e_1, \dots, e_n))$ by observing the CRS and the proof $\pi^\dagger = (z^\dagger, r^\dagger, u^\dagger, \pi_0^\dagger)$, it amounts to the first $2t + 1$ rows of the left-hand side member of the following linear system:

$$\begin{pmatrix} \mathbf{Y}^\top \\ \mathbf{W}^\top \\ \pi_0^\dagger \\ \pi_0^* \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{A} \\ \log(\vec{v}^\dagger) & \alpha^\dagger \cdot \log(\vec{v}^\dagger) \\ \log(\vec{v}^*) & \alpha^* \cdot \log(\vec{v}^*) \end{pmatrix} \cdot \begin{pmatrix} \mathbf{e}^\top \\ \mathbf{d}^\top \end{pmatrix} \quad (6.7)$$

Let us assume that $\vec{v}^\dagger, \vec{v}^* \notin \mathcal{L}_\rho$. Since the above $(2t + 2) \times 2n$ matrix has full rank when $\alpha^* \neq \alpha$ (which is the case unless the failure event F_1 of **Game**₁ occurs), we see that the only value of π_0^* that would trick the private verification oracle $W(\psi, \tau, \cdot, \cdot)$ of **Game**₃ into accepting π^* is completely independent of the information provided by the CRS and the simulated proof $\pi^\dagger$ for $\vec{v}^\dagger$. However, $\mathcal{A}$ can also take advantage of its private verification queries throughout the game. Without any verification query, $(\vec{e}, \vec{d})$ is constrained by the first $2t + 1$ rows of (6.7) to live in a subspace of dimension $2(n - t) - 1 \geq 1$ in $\mathbb{Z}_p^{2n}$, so that π_0^* has p equally likely values in $\mathcal{A}$'s view. Each private verification query provides $\mathcal{A}$ with an inequality, which allows it to rule out one candidate for the value of π_0^* that $W(\psi, \tau_v, \cdot, \cdot)$ would accept. After q queries, $\mathcal{A}$ is thus left with $p - q$ equally likely candidates for π_0^* . We thus find $\Pr[F_4] \leq 1/(p - q)$, as claimed. ■

Putting the above altogether, $\mathcal{A}$'s advantage in breaking the quasi-adaptive relative simulation-soundness property is bounded as

$$\mathbf{Adv}^{\text{rss}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{CR}}(\lambda) + \mathbf{Adv}^{\text{SDP}}(\lambda) + \frac{1}{2^\lambda - q}.$$

□

6.4 Extensions for the k -Linear Assumption

To instantiate our proof systems using the k -linear assumption with $k > 2$, we first need to extend the one-time linearly homomorphic structure-preserving signature of Section 4.4.2. To this end, we need to define the following assumption which is implied by the k -linear assumption in the same way as SDP is implied by DLIN.

Definition 41 *The Simultaneous k -wise Pairing (k -SDP) problem is, given a random tuple $(g_{1,z}, \dots, g_{k,z}, g_{1,r}, \dots, g_{k,r}) \in_R \mathbb{G}^{2k}$, to find a non-trivial vector $(z, r_1, \dots, r_k) \in \mathbb{G}^{k+1}$ such that, for $z \neq 1_{\mathbb{G}}$,*

$$e(g_{j,z}, z) \cdot e(g_{j,r}, r_j) = 1_{\mathbb{G}_T}, \quad \forall j \in \{1, \dots, k\}. \quad (6.8)$$

Given a k -linear instance $(g_{1,r}, \dots, g_{k,r}, g_{1,r}^{a_1}, \dots, g_{k,r}^{a_k}, \eta) \in \mathbb{G}^{2k+1}$, for any non-trivial tuple $(z, r_1, \dots, r_k)$ satisfying $e(g_{j,r}^{a_j}, z) \cdot e(g_{j,r}, r_j) = 1_{\mathbb{G}_T}$ for each $j \in$

$\{1, \dots, k\}$, we have

$$\eta = g^{\sum_{j=1}^k a_j} \iff e(g, \prod_{j=1}^k r_j) \cdot e(z, \eta) = 1_{\mathbb{G}_T}.$$

Hence, any algorithm solving k -SDP with non-negligible probability implies a k -linear distinguisher.

Under the k -SDP assumption, the one-time linearly homomorphic structure-preserving signature of Section 4.4.2 can be extended as follows.

Keygen(λ, n): given a security parameter λ and the dimension $n \in \mathbb{N}$ of vectors to be signed, chooses bilinear group $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$. For $j = 1$ to k , chooses generators $g_{j,z}, g_{j,r} \xleftarrow{\$} \mathbb{G}$. Then, for each $i = 1$ to n , $j = 1$ to k , picks $\chi_i \xleftarrow{\$} \mathbb{Z}_p$, $\gamma_{j,i} \xleftarrow{\$} \mathbb{Z}_p$ and computes $g_{j,i} = g_{j,z}^{\chi_i} g_{j,r}^{\gamma_{j,i}}$. The private key is $\text{sk} = (\{\chi_i, \{\gamma_{j,i}\}_{j=1}^k\}_{i=1}^n)$ while the public key is

$$\text{pk} = (\{g_{j,z}, g_{j,r}, \{g_{j,i}\}_{i=1}^n\}_{j=1}^k).$$

Sign($\text{sk}, (M_1, \dots, M_n)$): to sign $(M_1, \dots, M_n) \in \mathbb{G}^n$ using the secret key $\text{sk} = (\{\chi_i, \{\gamma_{j,i}\}_{j=1}^k\}_{i=1}^n)$, computes and outputs $\sigma = (z, r_1, \dots, r_k)$ in $\mathbb{G}^{k+1}$, where

$$z = \prod_{i=1}^n M_i^{-\chi_i}, \quad \text{and for all } j \in \{1, \dots, k\}: \quad r_j = \prod_{i=1}^n M_i^{-\gamma_{j,i}}.$$

SignDerive($\text{pk}, \{(\omega_i, \sigma^{(i)})\}_{i=1}^\ell$): given a public key pk and ℓ tuples $(\omega_i, \sigma^{(i)})$, where $\omega_i \in \mathbb{Z}_p$ for each i , parses $\sigma^{(i)}$ as $\sigma^{(i)} = (z_i, r_{i,1}, \dots, r_{i,k}) \in \mathbb{G}^{k+1}$ for $i = 1$ to ℓ . Then, computes and returns $\sigma = (z, r_1, \dots, r_k)$, where $z = \prod_{i=1}^\ell z_i^{\omega_i}$, $r_j = \prod_{i=1}^\ell r_{i,j}^{\omega_i}$ for $j = 1$ to k .

Verify($\text{pk}, \sigma, (M_1, \dots, M_n)$): given a signature $\sigma = (z, r_1, \dots, r_k) \in \mathbb{G}^{k+1}$ and $(M_1, \dots, M_n)$, returns 1 if and only if $(M_1, \dots, M_n) \neq (1_{\mathbb{G}}, \dots, 1_{\mathbb{G}})$ and, for each $j \in \{1, \dots, k\}$, the following equality holds:

$$1_{\mathbb{G}_T} = e(g_{j,z}, z) \cdot e(g_{j,r}, r_j) \cdot \prod_{i=1}^n e(g_{j,i}, M_i). \quad (6.9)$$

In order to adapt the unbounded simulation-sound proof system of Section 6.2, we need to commit to the components of $(z, r_1, \dots, r_k)$ and NIWI arguments showing that committed group elements satisfy the pairing product equations (6.9). Under the k -linear assumption, committing to a group element requires $k + 1$ group elements (see, *e.g.*, [36] for details) whereas each equation of the form (6.9) costs $k + 1$ elements to prove. Overall, we thus need $(k + 1)(2k + 1)$ group elements and a one-time verification key pair (SVK, σ) .

In the relatively-sound QA-NIZK proof of Section 6.3, the proof element π_0 remains unchanged and we simply need to replace the triple (z, r, u) by a one-time linearly homomorphic signature $(z, r_1, \dots, r_k)$. Hence, we only need $k + 2$ group elements.

6.5 Comparison of Linear Subspace Arguments

This section compares the various NIZK proofs of linear subspace membership based on the DLIN assumption. Comparisons are given in terms of CRS size, proof size and the number of pairing evaluations for the verifier.

In the table, we consider our basic proof system (without any form of simulation-soundness, where each proof is a one-time linearly homomorphic signature (z, r, u)), its unbounded simulation-sound variant and the relatively simulation-sound variant of Section 6.3. We compare these with the original Groth-Sahai proofs (the only system achieving perfect soundness here), their most efficient unbounded simulation-sound extensions due to Camenisch *et al.* [36] and the Jutla-Roy techniques [95] with and without relative soundness.

Table 6.1: Comparison between proof systems for linear subspaces

Proof systems	CRS size [◇] *	Proof length [◇]	Verification [†]
Groth-Sahai [84]	6	$3t + 2n$	$3n(t + 3)$
Jutla-Roy [95]	$4t(n - t) + 3$	$2(n - t)$	$2(n-t)(t+2)$
Jutla-Roy RSS [95] + [94]	$4t(n + 1 - t) + 3$	$2(n + 1 - t) + 1$	$2(n + 1 - t)(t + 2)$
Groth-Sahai USS [36]	18	$6t + 2n + 52^\ddagger$	$O(tn)$
Our basic QA-NIZK proofs	$2n + 3t + 4$	3	$2n + 2$
Our RSS QA-NIZK proofs	$4n + 8t + 6$	4	$2n + 6$
Our USS QA-NIZK proofs	$2n + 3t + 3L + 10$	$20^\ddagger$	$2n + 30$

n : num. of equations t : num. of variables L : length of a hashed one-time verif. key

[◇] These sizes are measured in terms of number of group elements.

* The description $\rho \in \mathbb{G}^{t \times n}$ of the language is not counted as being part of the CRS here.

[†] In number of pairings without considering batch verification techniques.

[‡] We consider instantiations using Groth's one-time signature [80], where verification keys and signatures consist of 3 group elements and two elements of $\mathbb{Z}_p$, respectively.

As can be observed in the table, our constructions all yield constant-size arguments. Moreover, the number of pairing evaluations is always independent of the number of variables t , which substantially speeds up the verification process when $t \approx n/2$.

We also note that randomized batch verification techniques can be used to drastically reduce the number of pairing computations. In our USS system, for example, the number of pairings drops to $n + 18$ if the two verification equations are processed together and further optimizations are possible.

Our common reference strings always fit within $O(t + n)$ group elements (with another $O(L)$ elements in the USS variant) and thus provide significant savings w.r.t. [95] when $t \approx n/2$.

Part IV

Encryption-Based Primitives

Chapter 7

Publicly Verifiable Threshold Encryptions

Contents

7.1	Definition and Security Notions	178
7.1.1	Threshold Keyed-Homomorphic Encryptions . . .	178
7.2	Adaptively Secure Threshold Cryptosystems . . .	180
7.2.1	Intuition behind our Publicly-Verifiable Scheme . .	180
7.2.2	DLIN-Based CCA Construction	181
7.2.3	Efficiency and k -Linear Extention.	183
7.2.4	Security of the CCA Threshold Cryptosystem . . .	184
7.3	Keyed Homomorphic Threshold Cryptosystems .	189
7.3.1	Intuition behind our Keyed-Homomorphic Scheme	189
7.3.2	DLIN-Based KH-CCA Construction	190
7.3.3	Security of the Keyed-Homomorphic cryptosystem	193

Threshold cryptography [56, 57, 29] avoids single points of failure by splitting keys into $n > 1$ shares which are held by servers in such a way that at least t out of n servers should contribute to private key operations. In (t, n) -threshold cryptosystems, an adversary breaking into up to $t - 1$ servers should not jeopardize the security of the system.

This chapter is devoted to building threshold encryption schemes with publicly verifiable ciphertexts that are adaptively secure against chosen-ciphertext attacks. More precisely, we apply our methods of Chapter 6 to design new and improved CCA2-secure encryption schemes. In particular, we build the first efficient threshold CCA-secure *keyed-homomorphic* encryption scheme, where homomorphic operations can only be carried out using a dedicated evaluation key.

7.1 Definition and Security Notions

Threshold cryptosystems were initially suggested in [29, 56, 57]. In the static corruption setting, several non-interactive CCA-secure threshold systems have been described in the random oracle model [133, 66] and in the standard model [23, 31, 140].

Adaptively secure distributed cryptosystems with chosen-ciphertext security were proposed in [91, 2] but they require some interaction during the decryption process. Non-interactive solutions were put forth in [112, 113] but, as we will see, they are less efficient than the solution proposed here.

Keyed-homomorphic encryption is a primitive, suggested by Emura *et al.* [61], where homomorphic ciphertext manipulations are only possible for a party holding a devoted evaluation key SK_h which, by itself, does not enable decryption. The scheme should provide IND-CCA2 security when the evaluation key is unavailable to the adversary and remain IND-CCA1 secure when SK_h is exposed. The results of [61] showed that CCA2-security does not rule out homomorphism when the capability to compute over encrypted data is restricted.

7.1.1 Threshold Keyed-Homomorphic Encryptions

This section recalls the notion of publicly-verifiable keyed-homomorphic threshold encryptions. Ordinary publicly-verifiable threshold encryptions are strict particular cases of this definition.

Definition 42 *A (t, N) -threshold keyed-homomorphic encryption scheme consists of the following algorithms.*

Keygen (λ, t, N) : takes as input a security parameter λ and integers $t, N \in \text{poly}(\lambda)$ (with $1 \leq t \leq N$), where N is the number of decryption servers and $t \leq N$ is the decryption threshold. It outputs $(PK, SK_h, \mathbf{VK}, \mathbf{SK}_d)$, where PK is the public key, SK_h is the homomorphic evaluation key, $\mathbf{SK}_d = (SK_{d,1}, \dots, SK_{d,N})$ is a vector of private key shares and $\mathbf{VK} = (VK_1, \dots, VK_N)$ is a vector of verification keys. For each i , the decryption server i is given the share $(i, SK_{d,i})$. The verification key VK_i will be used to check the validity of decryption shares generated using $SK_{d,i}$.

Encrypt (PK, M) : takes as input a public key PK and a plaintext M . It outputs a ciphertext C .

Ciphertext-Verify (PK, C) : takes as input a public key PK and a ciphertext C . It outputs 1 if C is deemed valid w.r.t. PK and 0 otherwise.

Share-Decrypt $(PK, i, SK_{d,i}, C)$: on input of a public key PK , a ciphertext C and a private-key share $(i, SK_{d,i})$, this (possibly randomized) algorithm outputs a special symbol $(i, \perp)$ if **Ciphertext-Verify** $(PK, C) = 0$. Otherwise, it outputs a decryption share $\mu_i = (i, \hat{\mu}_i)$.

Share-Verify(PK, VK_i, C, μ_i): takes in PK , the verification key VK_i , a ciphertext C and a purported decryption share $\mu_i = (i, \hat{\mu}_i)$. It outputs either 1 or 0. In the former case, μ_i is said to be a *valid* decryption share. We adopt the convention that $(i, \perp)$ is an invalid decryption share.

Combine($PK, \mathbf{VK}, C, \{\mu_i\}_{i \in S}$): takes in $(PK, \mathbf{VK}, C)$ and a t -subset $S \subset \{1, \dots, N\}$ with decryption shares $\{\mu_i\}_{i \in S}$. It outputs either a plaintext M or $\perp$ if $\{\mu_i\}_{i \in S}$ contains invalid shares.

Eval($PK, SK_h, C^{(1)}, C^{(2)}$): takes as input PK , the evaluation key SK_h and ciphertexts $C^{(1)}, C^{(2)}$. If **Ciphertext-Verify**($PK, C^{(j)}$) = 0 for some $j \in \{1, 2\}$, the algorithm returns $\perp$. Otherwise, it conducts a binary homomorphic operation over $C^{(1)}$ and $C^{(2)}$ and outputs a ciphertext C .

The above syntax assumes a trusted dealer. It consists of an extension of the one of ordinary threshold cryptosystems. By setting $SK_h = \varepsilon$ and discarding the evaluation algorithm, we obtain a classical threshold system.

Definition 43 *A threshold keyed-homomorphic public-key cryptosystem is secure against chosen-ciphertext attacks (or KH-CCA secure) if no PPT adversary has noticeable advantage in this game:*

1. The challenger runs **Keygen**(λ) to obtain a public key PK , vectors $\mathbf{SK}_d = (SK_{d,1}, \dots, SK_{d,N})$, $\mathbf{VK} = (VK_1, \dots, VK_N)$ and a homomorphic evaluation key SK_h . He/she gives PK and $\mathbf{VK}$ to the adversary $\mathcal{A}$ and keeps $(SK_h, \mathbf{SK}_d)$ to himself/herself. In addition, the challenge initializes a set $\mathcal{D} \leftarrow \emptyset$, which is initially empty.
2. The adversary $\mathcal{A}$ adaptively makes queries to the following oracles on multiple occasions:
 - *Corruption query*: at any time, $\mathcal{A}$ may decide to corrupt a decryption server. To this end, he/she specifies an index $i \in \{1, \dots, N\}$ and obtains the private key share $SK_{d,i}$.
 - *Evaluation query*: $\mathcal{A}$ can invoke the evaluation oracle **Eval**($SK_h, \cdot$) on a pair $(C^{(1)}, C^{(2)})$ of ciphertexts of his/her choice. If there exists $j \in \{1, 2\}$ such that **Ciphertext-Verify**($PK, C^{(j)}$) = 0, returns $\perp$. Otherwise, the oracle **Eval**($SK_h, \cdot$) computes and returns $C \leftarrow \mathbf{Eval}(SK_h, C^{(1)}, C^{(2)})$. In addition, if $C^{(1)} \in \mathcal{D}$ or $C^{(2)} \in \mathcal{D}$, it sets $\mathcal{D} \leftarrow \mathcal{D} \cup \{C\}$.
 - *Reveal query*: at any time, $\mathcal{A}$ may also decide to corrupt the evaluator by invoking the **RevHK** oracle on a unique occasion. The oracle responds by returning SK_h .
 - *Decryption query*: $\mathcal{A}$ can also invoke the partial decryption oracle on arbitrary ciphertexts C and indexes $i \in \{1, \dots, n\}$. If $C \in \mathcal{D}$ or if **Ciphertext-Verify**(PK, C) = 0, the oracle returns $\perp$. Otherwise, the oracle runs $\mu_i \leftarrow \mathbf{Share-Decrypt}(PK, i, SK_{d,i}, C)$ and returns the decryption share μ_i .

3. The adversary $\mathcal{A}$ chooses two equal-length messages M_0, M_1 and obtains $C^* = \mathbf{Encrypt}(PK, M_\beta)$ for some random bit $\beta \xleftarrow{\$} \{0, 1\}$. In addition, the challenger sets $\mathcal{D} \leftarrow \mathcal{D} \cup \{C^*\}$.
4. $\mathcal{A}$ makes further queries as in step 2 with some restrictions. Namely, $\mathcal{A}$ cannot corrupt more than $t - 1$ servers throughout the entire game. Moreover, if $\mathcal{A}$ chooses to obtain SK_h (via the $\mathbf{RevHK}$ oracle) at some point, no more post-challenge decryption query is allowed beyond that point.
5. $\mathcal{A}$ outputs a bit β' and is deemed successful if $\beta' = \beta$. As usual, $\mathcal{A}$'s advantage is measured as the distance $\mathbf{Adv}(\mathcal{A}) = |\Pr[\beta' = \beta] - \frac{1}{2}|$.

It is important to note that, even if $\mathcal{A}$ chooses to obtain SK_h immediately after having seen the public key PK , it still has access to the decryption oracle *before* the challenge phase. In other words, the scheme should remain IND-CCA1 if $\mathcal{A}$ is given PK and SK_h at the outset of the game. After the challenge phase, decryption queries are allowed until the moment when the adversary obtains SK_h .

In [61], Emura *et al.* suggested a weaker definition where the adversary is not allowed to query the evaluation oracle on derivatives of the challenge ciphertext. As a consequence, the set $\mathcal{D}$ is always the singleton $\{C^*\}$ after step 3. Actually, their realizations of chosen-ciphertext-secure keyed-homomorphic schemes only achieve this relaxed notion. In the following Section 7.3, we will stick to the stronger definition. Moreover, their schemes do not readily enable threshold decryption – as would be desirable in voting protocols – since valid ciphertexts are not publicly recognizable, which makes it harder to prove CCA security in the threshold setting.

7.2 Adaptively Secure Threshold Cryptosystems

As an application of our relatively sound QA-NIZK proofs, we present a new robust non-interactive threshold encryption scheme based on the DLIN assumption. This scheme proved secure against chosen-ciphertext attacks in the adaptive corruption setting [40, 67].

7.2.1 Intuition behind our Publicly-Verifiable Scheme

Consider the DLIN-based cryptosystem based on 1-universal hash proof systems where the ciphertext $(C_1, C_2, C_3, C_0) = (f^{\theta_1}, h^{\theta_2}, g^{\theta_1 + \theta_2}, M \cdot X_1^{\theta_1} \cdot X_2^{\theta_2})$ is decrypted as $M = C_0 \cdot C_1^{-x_1} C_2^{-x_2} C_3^{-x_0}$, where $(X_1, X_2) = (f^{x_1} g^{x_2}, h^{x_2} g^{x_0})$ is the public key and (x_1, x_2, x_0) is the private key. In [113], chosen-ciphertext security was achieved using a *publicly* verifiable one-time simulation-sound proof of well-formedness for (C_1, C_2, C_3) . In the security proof, the one-time simulation-soundness property guarantees that the adversary is unable to trick the decryption oracle into returning the decryption of an invalid ciphertext, by generat-

ing a fake proof for an invalid triple (C_1, C_2, C_3) . For this reason, the specific private key (x_1, x_2, x_0) used by the reduction remains perfectly hidden. Consequently, if the challenge ciphertext is computed by choosing a random tuple $(C_1, C_2, C_3) \in_R \mathbb{G}^3$ and computing $C_0 = M \cdot C_1^{x_1} \cdot C_2^{x_2} \cdot C_3^{x_0}$, the plaintext M is independent of the adversary's view. To prove adaptive security in the threshold setting, [113] took advantage of the fact that the private key (more precisely, all private key shares) is known to the reduction at all times in the Cramer-Shoup paradigm.

A similar approach was taken in¹ [94], where a different method was used to achieve a form of one-time simulation-soundness. In combination with relatively sound proofs [94], the techniques of Jutla and Roy [95] reduce the size of ciphertexts to 9 group elements under the DLIN assumption. Here, as applications of Section 4.4.2 suggested in our paper [106], we obtain shorter ciphertexts by using linearly homomorphic signatures. We include in the public key the verification key of a one-time linearly homomorphic SPS for $n = 3$ as well as signatures on $(f, 1_{\mathbb{G}}, g)$ and $(1_{\mathbb{G}}, h, g)$. This allows publicly deriving a homomorphic signature (z, r, u) on the vector $(C_1, C_2, C_3) = (f^{\theta_1}, h^{\theta_2}, g^{\theta_1+\theta_2})$ and each ciphertext consists of $(z, r, u, C_0, C_1, C_2, C_3)$. In the security proof, the signature (z, r, u) serves as evidence that (C_1, C_2, C_3) has the right form at each pre-challenge decryption query: in order to generate a proof for a false statement, the adversary has to break the security of the homomorphic signature, by deriving a signature on a vector (C_1, C_2, C_3) outside the span of $(f, 1, g)$ and $(1, h, g)$.

While this technique does provide IND-CCA1 security, the scheme remains malleable and thus vulnerable to post-challenge decryption queries. This is where the relatively sound proof system of Section 6.3 comes into play. By using (C_0, C_1, C_2, C_3) as a label in the relatively sound proof that (C_1, C_2, C_3) lives in $\text{span}((f, 1, g), (1, h, g))$, we can make sure that, with all but negligible probability, the reduction will never accept a proof for a malformed (C_1, C_2, C_3) after the challenge phase without breaking the DLIN assumption. The key idea of the techniques of [94] is to guarantee that the adversary will not be able to send a decryption query for which the private verifier and the public verifier disagree on the validity of (C_1, C_2, C_3) .

7.2.2 DLIN-Based CCA Construction

In the threshold setting, the construction can be seen as a DLIN-based version of the Cramer-Shoup encryption scheme [49] (which is identical to the scheme in [131]), where the ciphertext components (C_1, C_2, C_3) and the designated verifier proof C_4 are additionally signed using a homomorphic signature. The scheme goes as follows.

Keygen (λ, t, N) : chooses bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$ and do the following.

¹Although it was not mentioned in [94], relatively sound proofs can be used to acquire CCA2 security in the threshold setting as well, as will be emphasized later on.

1. Chooses $g, f, h \xleftarrow{\$} \mathbb{G}$.
2. Choose random $x_0, x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p$, $y_0, y_1, y_2 \xleftarrow{\$} \mathbb{Z}_p$ and $w_0, w_1, w_2 \xleftarrow{\$} \mathbb{Z}_p$ in order to compute $X_1 = f^{x_1} g^{x_0}$, $X_2 = h^{x_2} g^{x_0}$, $Y_1 = f^{y_1} g^{y_0}$, $Y_2 = h^{y_2} g^{y_0}$ and $W_1 = f^{w_1} g^{w_0}$, $W_2 = h^{w_2} g^{w_0}$.
3. Generates a Groth-Sahai CRS $(\vec{f}_1, \vec{f}_2, \vec{f}_3)$ to be used for proving the validity of decryption shares. Namely, chooses $f_1, f_2 \xleftarrow{\$} \mathbb{G}$ as well as $\phi_1, \phi_2 \xleftarrow{\$} \mathbb{Z}_p$ and defines vectors

$$\vec{f}_1 = (f_1, 1, g), \quad \vec{f}_2 = (1, f_2, g) \quad \vec{f}_3 = \vec{f}_1^{\phi_1} \cdot \vec{f}_2^{\phi_2} \cdot (1, 1, g).$$
4. Chooses random polynomials $P_1[Z], P_2[Z], P_0[Z] \in \mathbb{Z}_p[Z]$ of degree $t - 1$ such that $P_1(0) = x_1$, $P_2(0) = x_2$ and $P_0(0) = x_0$. Sets $X_{i,1} = f^{P_1(i)} g^{P_0(i)}$ and $X_{i,2} = h^{P_2(i)} g^{P_0(i)}$ for $i = 1$ to N (see section 2.1.4)
5. Chooses a collision-resistant hash function $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p$.
6. Generates a key pair for the one-time linearly homomorphic signature of Section 4.4.2 with $n = 7$. Lets $(g_z, g_r, h_z, h_u, \{(g_i, h_i)\}_{i=1}^7)$ be the public key and lets $\{(\chi_i, \gamma_i, \delta_i)\}_{i=1}^7$ be the corresponding private key.
7. Generates one-time homomorphic signatures $\{(Z_j, R_j, U_j)\}_{j=1}^4$ on the independent vectors

$$\vec{h}_1 = (f, 1, g, Y_1, 1, 1, 1) \in \mathbb{G}^7, \quad \vec{h}_2 = (1, h, g, Y_2, 1, 1, 1) \in \mathbb{G}^7,$$

$$\vec{h}_3 = (1, 1, 1, W_1, f, 1, g) \in \mathbb{G}^7, \quad \vec{h}_4 = (1, 1, 1, W_2, 1, h, g) \in \mathbb{G}^7.$$
8. Defines decryption key shares $\mathbf{SK} = (SK_1, \dots, SK_N)$ such that for each $i \in \{1, \dots, N\}$ we set $SK_i = (P_1(i), P_2(i), P_0(i)) \in \mathbb{Z}_p^3$. The vector $\mathbf{VK} = (VK_1, \dots, VK_N)$ of verification keys is defined as $VK_i = (X_{i,1}, X_{i,2}) \in \mathbb{G}^2$ for each $i \in \{1, \dots, N\}$. The public key is defined to be

$$PK = \left(g, \vec{f}_1, \vec{f}_2, \vec{f}_3, X_1, X_2, Y_1, Y_2, W_1, W_2, \right. \\ \left. g_z, g_r, h_z, h_u, \{g_i, h_i\}_{i=1}^7, \{(Z_j, R_j, U_j)\}_{j=1}^4, H \right).$$

Encrypt(M, PK): to encrypt a message $M \in \mathbb{G}$, conducts the following steps.

1. Chooses $\theta_1, \theta_2 \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$C_0 = M \cdot X_1^{\theta_1} \cdot X_2^{\theta_2}, \quad C_1 = f^{\theta_1}, \quad C_2 = h^{\theta_2}, \quad C_3 = g^{\theta_1 + \theta_2}$$

$$C_4 = (W_1^\alpha Y_1)^{\theta_1} (W_2^\alpha Y_2)^{\theta_2},$$
 where $\alpha = H(C_0, C_1, C_2, C_3) \in \mathbb{Z}_p$.
2. Constructs a linearly homomorphic signature (Z, R, U) on the vector $(C_1, C_2, C_3, C_4, C_1^\alpha, C_2^\alpha, C_3^\alpha) \in \mathbb{G}^7$. Namely, computes

$$Z = Z_1^{\theta_1} \cdot Z_2^{\theta_2} \cdot Z_3^{\theta_1 \cdot \alpha} \cdot Z_4^{\theta_2 \cdot \alpha}, \quad R = R_1^{\theta_1} \cdot R_2^{\theta_2} \cdot R_3^{\theta_1 \cdot \alpha} \cdot R_4^{\theta_2 \cdot \alpha},$$

$$U = U_1^{\theta_1} \cdot U_2^{\theta_2} \cdot U_3^{\theta_1 \cdot \alpha} \cdot U_4^{\theta_2 \cdot \alpha}.$$

3. Outputs the ciphertext

$$C = (C_0, C_1, C_2, C_3, C_4, Z, R, U) \in \mathbb{G}^8 \quad (7.1)$$

Ciphertext-Verify(PK, C): parses C as per (7.1) and computes the hash $\alpha = H(C_0, C_1, C_2, C_3) \in \mathbb{Z}_p$ and returns 1 if and only if

$$\begin{aligned} 1_{\mathbb{G}_T} &= e(g_z, Z) \cdot e(g_r, R) \cdot \prod_{i=1}^3 e(g_i \cdot g_{i+4}^\alpha, C_i) \cdot e(g_4, C_4), \\ 1_{\mathbb{G}_T} &= e(h_z, Z) \cdot e(h_u, U) \cdot \prod_{i=1}^3 e(h_i \cdot h_{i+4}^\alpha, C_i) \cdot e(h_4, C_4). \end{aligned}$$

Share-Decrypt(PK, i, SK_i, C): on inputs $SK_i = (P_1(i), P_2(i), P_0(i)) \in \mathbb{Z}_p^3$ and C , returns $(i, \perp)$ in the event that **Ciphertext-Verify**(PK, C) = 0. Otherwise, computes $\hat{\mu}_i = (\nu_i, \vec{C}_{P_1}, \vec{C}_{P_2}, \vec{C}_P, \pi_{\mu_i})$ which consists of a partial decryption $\nu_i = C_1^{P_1(i)} \cdot C_2^{P_2(i)} \cdot C_3^{P_0(i)}$, commitments $\vec{C}_{P_1}, \vec{C}_{P_2}, \vec{C}_{P_0}$ to exponents $P_1(i), P_2(i), P_0(i) \in \mathbb{Z}_p$ and a proof π_{ν_i} that these satisfy the equations

$$\nu_i = C_1^{P_1(i)} \cdot C_2^{P_2(i)} \cdot C_3^{P_0(i)}, \quad X_{i,1} = f^{P_1(i)} g^{P_0(i)}, \quad X_{i,2} = h^{P_2(i)} g^{P_0(i)}. \quad (7.2)$$

The commitments $\vec{C}_{P_1}, \vec{C}_{P_2}, \vec{C}_{P_0}$ and the proof π_{ν_i} are generated using the CRS $\mathbf{f} = (\vec{f}_1, \vec{f}_2, \vec{f}_3)$.

Share-Verify($PK, VK_i, C, (i, \hat{\mu}_i)$): parses the ciphertext C as $(C_0, C_1, C_2, C_3, C_4, Z, R, U)$ and VK_i as $(X_{i,1}, X_{i,2}) \in \mathbb{G}^2$. If $\hat{\mu}_i = \perp$ or $\hat{\mu}_i$ cannot be properly parsed as $(\nu_i, \vec{C}_{P_1}, \vec{C}_{P_2}, \vec{C}_{P_0}, \pi_{\mu_i})$, returns 0. Otherwise, returns 1 if π_{μ_i} is a valid proof. In any other situation, returns 0.

Combine($PK, \mathbf{VK}, C, \{(i, \hat{\mu}_i)\}_{i \in S}$): for each $i \in S$, parses the decryption share $\hat{\mu}_i$ as $(\nu_i, \vec{C}_{P_1}, \vec{C}_{P_2}, \vec{C}_P, \pi_{\mu_i})$ and returns $\perp$ if **Share-Verify**($PK, C, (i, \hat{\mu}_i)$) = 0. Otherwise, computes $\nu = \prod_{i \in S} \nu_i^{\Delta_{i,S}(0)}$, which equals $\nu = C_1^{x_1} \cdot C_2^{x_2} \cdot C_3^{x_0} = X_1^{\theta_1} \cdot X_2^{\theta_2}$ and in turn reveals $M = C_0/\nu$.

7.2.3 Efficiency and k -Linear Extension.

If each element has a 256-bit representation on BN curves [12] at the 128-bit security level, the ciphertext overhead amounts to 1792 bits. The DLIN-based scheme of [113] has a ciphertext overhead comprised of 14 group elements and a one-time signature with its verification key (or 4864 bits using Groth's one-time signature [80]). The results of Escala *et al.* [62] reduce this overhead to 3328 bits. The recent techniques of Jutla and Roy [94, 95] – which also work in the threshold setting although it was not explicitly stated in [94] – lead to ciphertexts comprised of 9 group elements under the DLIN assumption and

$3k + 3$ under the k -linear assumption. Under DLIN, we thus further compress ciphertexts by 11% while relying on the same assumption and retaining tight security².

Under the k -linear assumption, our improvement becomes more important as the ciphertext reduces to $2k + 4$ group elements. Specifically, we need $k + 1$ elements for the homomorphic signature of Section 6.4, another $k + 1$ elements to contain the k -linear instance, one element for the Cramer-Shoup-like proof π_0 and one element to carry the plaintext. This allows saving $k - 1$ group elements with respect to the techniques of [94, 95].

We believe this result to be of importance as these schemes can potentially serve as building blocks for protocols in the multi-linear setting [74, 47]. Indeed, the $(k - 1)$ -linear problem is easy in groups equipped with a k -linear map (as shown in, *e.g.*, [62]) but we can hope for instantiations where the k -linear assumption holds, as seems to be the case in [47].

From a computational standpoint, the validity of a ciphertext only requires to compute a product of 7 pairings. Under the DLIN assumption, the framework of [113] requires a product of 12 pairings in the ciphertext verification algorithm.

7.2.4 Security of the CCA Threshold Cryptosystem

We prove security in the sense of a definition which is identical to Definition 43 with the difference that there is no evaluation key SK_h , no evaluation oracle and no RevHK oracle.

As in the scheme of [94], the security proof makes use of the private verification algorithm while the scheme itself only uses the public verification algorithm.

While it would be possible to rely on the relative zero-knowledge and relative soundness properties of the proof system in a modular way, we obtain a better exact security via a direct analysis.

Theorem 22 *The above threshold cryptosystem is IND-CCA secure against adaptive corruptions assuming that: (i) H is collision-resistant; (ii) The DLIN assumption holds in $\mathbb{G}$. More precisely, the advantage of any PPT adversary $\mathcal{A}$ is at most*

$$\mathbf{Adv}(\mathcal{A}) \leq \mathbf{Adv}^{\text{CR-hash}}(\lambda) + 3 \cdot \mathbf{Adv}^{\text{DLIN}}(\lambda) + \frac{2q + 1}{2^\lambda - q}, \quad (7.3)$$

where q is the number of decryption queries and the first term of the right-hand side member accounts for the maximal advantage of any PPT collision-finding algorithm for H .

²Note that the techniques of Lewko [104] can be applied to the scheme of [112] to get a DLIN-based system where ciphertexts contain 7 group elements and a one-time key pair (SVK, σ) . However, the reduction involves a degradation factor proportional to the number of decryption queries.

Proof. The proof uses a sequence of games starting with the real attack game and ending with a game where the adversary $\mathcal{A}$ has no advantage. For each i , S_i stands for the event that the challenger $\mathcal{B}$ outputs 1 at the end of Game_i .

Game₁: is the real attack game. In details, the adversary is given the public key PK and the set of verification keys $\mathbf{VK} = (VK_1, \dots, VK_N)$. At each corruption query $i \in \{1, \dots, n\}$, the challenger $\mathcal{B}$ reveals the queried private key share $SK_i = (P_1(i), P_2(i), P_0(i))$. At each decryption query, $\mathcal{B}$ faithfully runs the real shared decryption algorithm. In the challenge phase, the adversary $\mathcal{A}$ chooses messages $M_0, M_1 \in \mathbb{G}$ and obtains $C^* = (C_0^*, C_1^*, C_2^*, C_3^*, C_4^*, Z^*, R^*, U^*)$ which is an encryption of M_β , for some random coin $\beta \xleftarrow{\$} \{0, 1\}$ flipped by $\mathcal{B}$. We can assume w.l.o.g. that (C_1^*, C_2^*, C_3^*) are computed at the beginning of the game as they do not depend on M_β .

In the second phase, $\mathcal{A}$ makes more adaptive queries under the restriction of not asking for a partial decryption of C^* or for more than $t - 1$ private key shares throughout the entire game. Eventually, $\mathcal{A}$ halts and outputs β' . At this point, $\mathcal{B}$ outputs 1 if $\beta = \beta'$ and 0 otherwise.

Game₂: This game is like Game_1 except that the challenger $\mathcal{B}$ halts and outputs a random bit in the event that, before the challenger phase, $\mathcal{A}$ queries the partial decryption oracle on a ciphertext $C = (C_0, C_1, C_2, C_3, C_4, Z, R, U)$ such that $(C_1, C_2, C_3) = (C_1^*, C_2^*, C_3^*)$. Since (C_1^*, C_2^*, C_3^*) are invisible to $\mathcal{A}$ until the challenge phase, this event can only occur with probability q/p , so that $|\Pr[S_2] - \Pr[S_1]| < q/p$.

Game₃: We introduce another failure event F_3 and let $\mathcal{B}$ halt and output a random bit if this event occurs. We define F_3 as the event that $\mathcal{A}$ makes a decryption query involving a valid ciphertext $C = (C_0, C_1, C_2, C_3, C_4, Z, R, U)$ such that $H(C_0, C_1, C_2, C_3) = H(C_0^*, C_1^*, C_2^*, C_3^*)$ but $(C_0^*, C_1^*, C_2^*, C_3^*) \neq (C_0, C_1, C_2, C_3)$.

We see that Game_3 and Game_2 are identical until event F_3 occurs, which would contradict the collision-resistance of H . We thus have $|\Pr[S_3] - \Pr[S_2]| \leq \Pr[F_3] \leq \text{Adv}^{\text{CR-hash}}(\lambda)$. In subsequent games, if we define the values $\alpha = H(C_0, C_1, C_2, C_3)$ and $\alpha^* = H(C_0^*, C_1^*, C_2^*, C_3^*)$, we will henceforth assume that $\alpha \neq \alpha^*$ for each decryption query $C = (C_0, C_1, C_2, C_3, C_4, Z, R, U)$.

Game₄: In this game, we modify the decryption oracle and reject all post-challenge decryption queries $(C_0, C_1, C_2, C_3, C_4, Z, R, U)$ for which we have $(C_0, C_1, C_2, C_3, C_4) = (C_0^*, C_1^*, C_2^*, C_3^*, C_4^*)$. Clearly Game_4 is identical to Game_3 until $\mathcal{B}$ rejects a ciphertext that would not have been rejected in Game_3 .

If we call the latter event F_4 , we find that $|\Pr[S_4] - \Pr[S_3]| \leq \Pr[F_4]$. Since F_4 necessarily implies $(Z, R, U) \neq (Z^*, R^*, U^*)$, any occurrence of F_4 necessarily provides $\mathcal{A}$ with two distinct signatures on the same vector $(C_1^*, C_2^*, C_3^*, C_4^*, C_1^{\alpha^*}, C_2^{\alpha^*}, C_3^{\alpha^*})$, which in turn breaks the SDP as-

sumption by the specific property of the linearly homomorphic signature (see Section 4.4.2). It comes that $\Pr[F_4] \leq \mathbf{Adv}^{\text{SDP}}(\mathcal{B})$.

Game₅: We modify the generation of $C^* = (C_0^*, C_1^*, C_2^*, C_3^*, C_4^*, Z^*, R^*, U^*)$ in the challenge phase. Specifically, instead of computing $C_0^* = M_\beta \cdot X_1^{\theta_1} \cdot X_2^{\theta_2}$ and $C_4^* = (W_1^{\alpha^*} Y_1)^{\theta_1} (W_2^{\alpha^*} Y_2)^{\theta_2}$, where $\theta_1 = \log_f(C_1^*)$ and $\theta_2 = \log_h(C_2^*)$, the challenger $\mathcal{B}$ now computes $C_0^* = M_\beta \cdot C_1^{x_1} \cdot C_2^{x_2} \cdot C_3^{x_0}$ and $C_4^* = C_1^{y_1 + \alpha^* w_1} \cdot C_2^{y_2 + \alpha^* w_2} \cdot C_3^{y_0 + \alpha^* w_0}$, with $\alpha^* = H(C_0^*, C_1^*, C_2^*, C_3^*)$. Likewise, instead of using the encryption exponents (θ_1, θ_2) to derive a one-time linearly homomorphic signature (Z^*, R^*, U^*) from the public key, the challenger $\mathcal{B}$ uses $\{(\chi_i, \gamma_i, \delta_i)\}_{i=1}^7$ and computes

$$Z^* = \prod_{i=1}^7 C_i^{x_i - \chi_i} \quad R^* = \prod_{i=1}^7 C_i^{x_i - \gamma_i} \quad U^* = \prod_{i=1}^7 C_i^{x_i - \delta_i}, \quad (7.4)$$

where $(C_5^*, C_6^*, C_7^*) = (C_1^{\alpha^*}, C_2^{\alpha^*}, C_3^{\alpha^*})$.

This change is only conceptual since C_0^* still equals $C_0^* = M_\beta \cdot X_1^{\theta_1} \cdot X_2^{\theta_2}$ and the distribution of (C_4^*, Z^*, R^*, U^*) has not changed either. We thus have $\Pr[S_5] = \Pr[S_4]$.

Game₆: Here, we modify the decryption oracle and make use of the exponents $(y_0, y_1, y_2, w_0, w_1, w_2)$ that were chosen by $\mathcal{B}$ during the key generation phase. Namely, the challenger $\mathcal{B}$ does not only reject all ciphertexts $(C_0, C_1, C_2, C_3, C_4, Z, R, U)$ such that (Z, R, U) does not form a valid signature on $(C_1, C_2, C_3, C_4, C_1^\alpha, C_2^\alpha, C_3^\alpha)$ but also rejects those for which

$$C_4 \neq C_1^{y_1 + \alpha \cdot w_1} \cdot C_2^{y_2 + \alpha \cdot w_2} \cdot C_3^{y_0 + \alpha \cdot w_0},$$

where $\alpha = H(C_0, C_1, C_2, C_3)$. We raise a failure event F_6 , which causes $\mathcal{B}$ to halt and output a random bit if it occurs. This event F_6 is defined to be the event that the adversary $\mathcal{A}$ queries the decryption oracle on a ciphertext that gets rejected in **Game₆** and would not have been rejected in **Game₅**. Since **Game₆** is identical to **Game₅** until F_6 occurs, we have

$$\Pr[S_6] = \Pr[S_6 \wedge \neg F_6] + \frac{1}{2} \cdot \Pr[F_6] = \Pr[S_5] + \frac{1}{2} \cdot \Pr[F_6].$$

At the same time, Lemma 8 shows that $\Pr[F_6] \leq \mathbf{Adv}^{\text{SDP}}(\lambda) + \frac{1}{p}$. We remark that a side-effect of this modified decryption oracle is that it now rejects all post-challenge decryption queries $(C_0, C_1, C_2, C_3, C_4, Z, R, U)$ such that $(C_0, C_1, C_2, C_3) = (C_0^*, C_1^*, C_2^*, C_3^*)$ but $C_4 \neq C_4^*$.

Since F_6 is always efficiently detectable by the challenger $\mathcal{B}$, we can build an efficient DLIN distinguisher if the probability of event F_6 increases when C_3^* is tampered with in the challenge ciphertext as in the next game.

Game₇: This game is identical to **Game₆** with one modification in the challenge ciphertext. Instead of setting $C_3^* = g^{\theta_1 + \theta_2}$, where $\theta_1 = \log_f(C_1^*)$

and $\theta_2 = \log_h(C_2^*)$, we choose it as $C_3^* \xleftarrow{\$} \mathbb{G}$. The linearly homomorphic signature (Z^*, R^*, U^*) is computed according to (7.4), as previously. Under the DLIN assumption in $\mathbb{G}$, this modification should have no noticeable impact on $\mathcal{A}$'s behaviour. In particular, we have $|\Pr[S_7] - \Pr[S_6]| \leq \mathbf{Adv}^{\text{DLIN}}(\lambda)$.

Game₈: We modify the partial decryption oracle and replace the non-interactive proofs contained in decryption shares $\hat{\mu}_i$ by simulated NIZK proofs. This entails to turn $(\vec{f}_1, \vec{f}_2, \vec{f}_3)$ into a perfectly hiding Groth-Sahai CRS (where $\vec{f}_3$ is in $\text{span}(\vec{f}_1, \vec{f}_2)$) and non-interactive proofs for multi-exponentiation equations are simulated using the trapdoor of the simulated CRS. Under the DLIN assumption, this change is not noticeable by $\mathcal{A}$ and we have $|\Pr[S_8] - \Pr[S_7]| \leq \mathbf{Adv}^{\text{DLIN}}(\lambda)$.

Game₉: In this game, we modify again the decryption oracle and make use of the discrete logarithms $\alpha_f = \log_g(f)$ and $\alpha_h = \log_g(h)$. Since we are done with the transition consisting in replacing C_3^* by a random element, we are free to use (α_f, α_h) from this point forward. We thus introduce a modification in the treatment of decryption queries $C = (C_0, C_1, C_2, C_3, C_4, Z, R, U)$. Now $\mathcal{B}$ rejects *all* ciphertexts C such that $C_3 \neq C_1^{1/\alpha_f} \cdot C_2^{1/\alpha_h}$. Otherwise, it answers as in Game₈.

If we define F_9 to be the event that $\mathcal{B}$ rejects a ciphertext which would not have been rejected in Game₈, we see that Game₉ and Game₈ are identical from $\mathcal{A}$'s view until F_9 occurs. Therefore it comes that

$$\Pr[S_9] \leq \Pr[S_9 \wedge \neg F_9] + \Pr[F_9] = \Pr[S_8] + \Pr[F_9].$$

The same arguments as in the proof of Cramer and Shoup show that $\Pr[F_9] \leq q/(p - q)$. More precisely, after i decryption queries, the adversary is left with $p - i$ equally likely candidates for the value of C_4 that would have been accepted by the private ciphertext validation algorithm. The probability that the i -th decryption query satisfies the test given that the first $i - 1$ queries have failed it, is thus at most $i/(p - i)$.

In Game₉, it is easy to see that $\mathcal{A}$ has no advantage whatsoever and we have $\Pr[S_9] = 1/2$. Indeed, in the challenge phase, we have $(C_1^*, C_2^*, C_3^*) = (f^{\theta_1}, h^{\theta_2}, g^{\theta_1 + \theta_2 + \theta_3})$, with $\theta_1, \theta_2, \theta_3 \in_R \mathbb{Z}_p$, so that C_0^* can be written as $C_0^* = M_\beta \cdot X_1^{\theta_1} \cdot X_2^{\theta_2} \cdot g^{\theta_3 \cdot x_0}$. The latter equality implies that, as long as $x_0 \in \mathbb{Z}_p$ is independent of $\mathcal{A}$'s view, so is the bit $\beta \in \{0, 1\}$.

We also note that, in Game₉, decryption shares $\hat{\mu}_i$ contain NIZK proofs that are simulated without using private key shares and thus leak no information about these. It comes that, as long as $\mathcal{A}$ does not corrupt more than $t - 1$ servers, the only possible way to infer information about $x_0 = P(0)$ is to make decryption queries on invalid ciphertexts (*i.e.*, for which (C_1, C_2, C_3) lies outside the span of $\vec{f}$ and $\vec{h}$).

We thus find

$$|\Pr[S_1] - \frac{1}{2}| < \mathbf{Adv}^{\text{CR-hash}}(\lambda) + 2 \cdot \mathbf{Adv}^{\text{DLIN}}(\lambda) + 2 \cdot \mathbf{Adv}^{\text{SDP}}(\lambda) + \frac{2q + 1}{p - q}.$$

Since any algorithm solving SDP immediately provides a DLIN distinguisher, we also have the inequality $\mathbf{Adv}^{\text{SDP}}(\lambda) \leq \frac{1}{2} \mathbf{Adv}^{\text{DLIN}}(\lambda)$, which yields

$$\left| \Pr[S_1] - \frac{1}{2} \right| < \mathbf{Adv}^{\text{CR-hash}}(\lambda) + 3 \cdot \mathbf{Adv}^{\text{DLIN}}(\lambda) + \frac{2q+1}{2^\lambda - q} \quad (7.5)$$

and the claimed result follows. $\square$

Lemma 8 *In Game₆, the probability of event F_6 is at most*

$$\Pr[F_6] \leq \mathbf{Adv}^{\text{SDP}}(\lambda) + 1/p.$$

Proof of the lemma. We show that, if event F_6 occurs with non-negligible probability ε in Game₆, there exists an efficient algorithm $\mathcal{B}$ that solves a SDP instance (g_z, g_r, h_z, h_u) with about the same probability. To this end, we first remark that F_6 can only occur for a decryption query $(C_0, C_1, C_2, C_3, C_4, Z, R, U)$ such that $(C_1, C_2, C_3, C_4, C_1^\alpha, C_2^\alpha, C_3^\alpha)$ is outside $\text{span}(\vec{h}_1, \vec{h}_2, \vec{h}_3, \vec{h}_4)$. Indeed, otherwise, there exist integers $\theta_1, \theta_2 \in \mathbb{Z}_p$ such that $(C_1, C_2, C_3, C_4) = (f^{\theta_1}, h^{\theta_2}, g^{\theta_1+\theta_2}, (W_1^\alpha Y_1)^{\theta_1} (W_2^\alpha Y_2)^{\theta_2})$, in which case we always have $C_4 = C_1^{y_1+\alpha w_1} \cdot C_2^{y_2+\alpha w_2} \cdot C_3^{y_0+\alpha w_0}$ and the rejection rule of Game₆ does not apply.

Using the technique of [106][Theorem 1], we show that event F_6 implies an algorithm solving the given SDP instance with nearly the same probability. Algorithm $\mathcal{B}$ begins by setting up $g_i = g_z^{\chi_i} g_r^{\gamma_i}$ and $h_i = h_z^{\chi_i} h_u^{\delta_i}$, with $\chi_i, \gamma_i, \delta_i \xleftarrow{\$} \mathbb{Z}_p$ for $i \in \{1, \dots, 7\}$. Other public key components are generated as in the real scheme and the public key is given to the adversary.

Throughout the game, the reduction $\mathcal{B}$ answers $\mathcal{A}$'s decryption queries in the same way as in Game₆. By hypothesis, $\mathcal{A}$ must query the decryption of a ciphertext $(C_0, C_1, C_2, C_3, C_4, Z, R, U)$ such that (Z, R, U) is a valid linearly homomorphic signature on the vector $(C_1, C_2, C_3, C_4, C_1^\alpha, C_2^\alpha, C_3^\alpha)$, where $\alpha = H(C_0, C_1, C_2, C_3)$, but $C_4 \neq C_1^{y_1+\alpha w_1} \cdot C_2^{y_2+\alpha w_2} \cdot C_3^{y_0+\alpha w_0}$, which implies that the vector is not in $\text{span}(\vec{h}_1, \vec{h}_2, \vec{h}_3, \vec{h}_4)$. When $\mathcal{B}$ detects this event, it defines $(C_5, C_6, C_7) = (C_1^\alpha, C_2^\alpha, C_3^\alpha)$ and computes its own signature

$$(Z^\dagger, R^\dagger, U^\dagger) = \left(\prod_{i=1}^7 C_i^{-\chi_i}, \prod_{i=1}^7 C_i^{-\gamma_i}, \prod_{i=1}^7 C_i^{-\delta_i} \right) \quad (7.6)$$

on $(C_1, C_2, C_3, C_4, C_5, C_6, C_7)$. We claim that, with overwhelming probability,

$$(Z^\dagger, R^\dagger, U^\dagger) = \left(\frac{Z}{Z^\dagger}, \frac{R}{R^\dagger}, \frac{U}{U^\dagger} \right)$$

is a non-trivial solution to the SDP instance since $Z^\dagger \neq 1_{\mathbb{G}}$ with all but negligible probability.

To see this, we first note that the vector $(\chi_1, \dots, \chi_7)$ is independent of $\mathcal{A}$'s view before the challenge phase. Hence, since $(C_1, C_2, C_3, C_4, C_5, C_6, C_7)$ is linearly independent of $(\vec{h}_1, \vec{h}_2, \vec{h}_3, \vec{h}_4)$, the adversary $\mathcal{A}$ can only predict $Z^\dagger$ (as it is computed in (7.6)) with negligible probability $1/p$. The probability $\Pr[F_6]$ can thus be bounded as $\Pr[F_6] \leq \mathbf{Adv}_{\mathcal{B}}^{\text{SDP}}(\lambda) + \frac{1}{p}$. $\square$

In the proof of the above theorem, the relative simulation-soundness property of the proof system described in Section 6.3.1 is notably used in the transition from Game_8 to Game_9 . In order to obtain a tighter reduction, we chose not to rely on this property in a modular way. In the modular approach, we would have to build an algorithm $\mathcal{B}^{rs}$ that contradicts this property using an adversary for which event F_9 occurs with non-negligible probability. This algorithm $\mathcal{B}^{rs}$ would have to interact with the relative soundness challenger for a *given* language $\rho \in \mathbb{G}^{2 \times 3}$ for which $\mathcal{B}^{rs}$ does *not* have the underlying matrix $\mathbf{A} \in \mathbb{Z}_p^{2 \times 3}$ of discrete logarithms. For this reason, $\mathcal{B}^{rs}$ would not be able to efficiently detect when F_9 occurs. To break the relative soundness property, $\mathcal{B}^{rs}$ would have to guess the decryption query for which this event occurs, which is only possible with probability $1/q$. In the exact security result (7.3), we would thus lose a multiplicative factor of $O(q)$.

7.3 Keyed Homomorphic Threshold Cryptosystems

As an application of our unbounded simulation-sound QA-NIZK proofs, we present the first keyed-homomorphic threshold encryption scheme based on the DLIN assumption and prove its security according to Definition 43.

7.3.1 Intuition behind our Keyed-Homomorphic Scheme

The use of linearly homomorphic signatures as publicly verifiable proofs of ciphertext validity in the Cramer-Shoup paradigm [49, 50] as in the intuition of the previous scheme only discusses non-adaptive (*i.e.*, CCA1) attacks. In the CCA2 case, a natural idea is to proceed as in our unbounded simulation-sound proof system and use the verification key of a one-time signature as the tag of a homomorphic signature: since cross-tag homomorphic operations are disallowed, the one-time signature will prevent illegal ciphertext manipulations after the challenge phase.

To obtain the desired keyed-homomorphic property, we use the simulation trapdoor of a simulation-sound proof system as the homomorphic evaluation key. This approach was already used by Emura *et al.* [61] in the context of designated verifier proofs. Here, publicly verifiable proofs are obtained from a homomorphic signature scheme of which the private key serves as an evaluation key: anyone equipped with this key can multiply two ciphertexts (or, more precisely, their built-in homomorphic components), generate a new tag and sign the resulting ciphertext using the private key of the homomorphic signature. Moreover, we can leverage the fact that the latter private key is always available to the reduction in the security proof of Theorem 18 of our randomizable homomorphic signature of Section 4.4.4. In the game of Definition 43, the simulator can thus hand over the evaluation key SK_h to the adversary upon request.

Emura *et al.* [61] gave constructions of KH-CCA secure encryption schemes based on hash proof systems [50]. However, these constructions are only known to provide a relaxed flavour of KH-CCA security where evaluation queries should not involve derivatives of the challenge ciphertext. The reason is that 2-universal hash proof systems [50] only provide a form of one-time simulation soundness whereas the model of Definition 43 seemingly requires unbounded simulation-soundness. Indeed, when the evaluation oracle is queried on input of a derivative of the challenge ciphertext in the security proof, the homomorphic operation may result in a ciphertext containing a vector outside the language $\mathcal{L}_\rho$. Since the oracle has to simulate a proof for this vector, each homomorphic evaluation can carry a proof for a potentially false statement. In some sense, the reduction is faced with a multi-challenge game as each output of the evaluation oracle can be seen as yet another challenge ciphertext. In this setting, our efficient unbounded simulation-sound QA-NIZK proof system comes in handy.

We still have to make sure that CCA1 security is always preserved, should the adversary obtain the evaluation key SK_h at the outset of the game. To this end, we include a second derived one-time homomorphic signature (Z, R, U) in the ciphertext without including its private key in SK_h .

7.3.2 DLIN-Based KH-CCA Construction

Keygen(λ, t, N): Chooses bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$. Then, does the following.

1. Picks $f, g, h \xleftarrow{\$} \mathbb{G}$, $x_0, x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p$ and sets $X_1 = f^{x_1} g^{x_0} \in \mathbb{G}$, $X_2 = h^{x_2} g^{x_0} \in \mathbb{G}$. Then, defines $\vec{f} = (f, 1, g) \in \mathbb{G}^3$ and $\vec{h} = (1, h, g) \in \mathbb{G}^3$.
2. Chooses random polynomials $P_1[Z], P_2[Z], P[Z] \in \mathbb{Z}_p[Z]$ of degree $t - 1$ such that $P_1(0) = x_1$, $P_2(0) = x_2$ and $P(0) = x_0$. For each $i \in \{1, \dots, N\}$, computes $VK_i = (Y_{i,1}, Y_{i,2})$ where $Y_{i,1} = f^{P_1(i)} g^{P(i)}$ and $Y_{i,2} = h^{P_2(i)} g^{P(i)}$ (see section 2.1.4 for details).
3. Chooses $f_{r,1}, f_{r,2} \xleftarrow{\$} \mathbb{G}$ in order to define vectors $\vec{f}_{r,1} = (f_{r,1}, 1, g)$, $\vec{f}_{r,2} = (1, f_{r,2}, g)$ and $\vec{f}_{r,3} = \vec{f}_{r,1}^{\phi_1} \cdot \vec{f}_{r,2}^{\phi_2} \cdot (1, 1, g)^{-1}$, where $\phi_1, \phi_2 \xleftarrow{\$} \mathbb{Z}_p$. These vectors will be used as a Groth-Sahai CRS for the generation of NIZK proofs showing the validity of decryption shares.
4. Chooses a strongly unforgeable one-time signature $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ with verification keys consisting of L -bit strings, for some $L \in \text{poly}(\lambda)$.
5. Generates a key pair for the one-time linearly homomorphic structure-preserving signature of Section 4.4.2 with $n = 3$. The public key is $\text{pk}_{ot} = (G_z, G_r, H_z, H_u, \{(G_i, H_i)\}_{i=1}^3)$ and the corresponding private key is $\text{sk}_{ot} = \{(\varphi_i, \vartheta_i, \varpi_i)\}_{i=1}^3$.
6. Generates one-time homomorphic signatures $\{(Z_j, R_j, U_j)\}_{j=1,2}$ on the vectors $\vec{f} = (f, 1, g)$ and $\vec{h} = (1, h, g)$. These signatures consist of $(Z_1, R_1, U_1) = (f^{-\varphi_1} g^{-\varphi_3}, f^{-\vartheta_1} g^{-\vartheta_3}, f^{-\varpi_1} g^{-\varpi_3})$ and $(Z_2, R_2, U_2) = (h^{-\varphi_2} g^{-\varphi_3}, h^{-\vartheta_2} g^{-\vartheta_3}, h^{-\varpi_2} g^{-\varpi_3})$ and erases sk_{ot} .

7. Generates a key pair $(\mathbf{pk}_{rand}, \mathbf{sk}_{rand})$ as in step 1 of the proof system in Section 6.2 with $n = 3$. Lets $\mathbf{sk}_{rand} = \{(\chi_i, \gamma_i, \delta_i)\}_{i=1}^3$ be the private key for which the corresponding public key is

$$\mathbf{pk}_{rand} = \left(g_z, g_r, h_z, h_u, \{(g_i, h_i)\}_{i=1}^3, \mathbf{f} = (\vec{f}_1, \vec{f}_2, \{\vec{f}_{3,i}\}_{i=0}^L) \right).$$

8. Uses $\mathbf{sk}_{rand}$ to generate one-time linearly homomorphic signatures $\{(z_j, r_j, u_j)\}_{j=1,2}$ on the independent vectors $\vec{f} = (f, 1, g) \in \mathbb{G}^3$ and $\vec{h} = (1, h, g) \in \mathbb{G}^3$. These are obtained as

$$\begin{aligned} (z_1, r_1, u_1) &= (f^{-\chi_1} g^{-\chi_3}, f^{-\gamma_1} g^{-\gamma_3}, f^{-\delta_1} g^{-\delta_3}), \\ (z_2, r_2, u_2) &= (h^{-\chi_2} g^{-\chi_3}, h^{-\gamma_2} g^{-\gamma_3}, h^{-\delta_2} g^{-\delta_3}). \end{aligned}$$

9. The public key is defined to be

$$\begin{aligned} PK &= \left(g, \vec{f}, \vec{h}, \vec{f}_{r,1}, \vec{f}_{r,2}, \vec{f}_{r,3}, X_1, X_2, \right. \\ &\quad \left. \mathbf{pk}_{ot}, \mathbf{pk}_{rand}, \{(Z_j, R_j, U_j)\}_{j=1}^2, \{(z_j, r_j, u_j)\}_{j=1}^2 \right). \end{aligned}$$

The evaluation key is $SK_h = \mathbf{sk}_{rand} = \{(\chi_i, \gamma_i, \delta_i)\}_{i=1}^3$ while the i -th decryption key share is defined to be $SK_{d,i} = (P_1(i), P_2(i), P(i))$. The vector of verification keys is defined as $\mathbf{VK} = (VK_1, \dots, VK_N)$, where $VK_i = (Y_{i,1}, Y_{i,2})$ for $i = 1$ to N .

Encrypt (M, PK) : to encrypt $M \in \mathbb{G}$, generate a one-time signature key pair $(\mathbf{SVK}, \mathbf{SSK}) \leftarrow \mathcal{G}(\lambda)$.

1. Chooses $\theta_1, \theta_2 \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$C_0 = M \cdot X_1^{\theta_1} \cdot X_2^{\theta_2}, \quad C_1 = f^{\theta_1}, \quad C_2 = h^{\theta_2}, \quad C_3 = g^{\theta_1 + \theta_2}.$$

2. Constructs a first linearly homomorphic signature (Z, R, U) on the vector $(C_1, C_2, C_3) \in \mathbb{G}^3$. Namely, computes $Z = Z_1^{\theta_1} \cdot Z_2^{\theta_2}$, $R = R_1^{\theta_1} \cdot R_2^{\theta_2}$ and $U = U_1^{\theta_1} \cdot U_2^{\theta_2}$.

3. Using $\{(z_j, r_j, u_j)\}_{j=1,2}$, derives another homomorphic signature (z, r, u) on (C_1, C_2, C_3) . Namely, computes $z = z_1^{\theta_1} \cdot z_2^{\theta_2}$, $r = r_1^{\theta_1} \cdot r_2^{\theta_2}$ and $u = u_1^{\theta_1} \cdot u_2^{\theta_2}$.

4. Using $\mathbf{SVK} = \mathbf{SVK}[1] \dots \mathbf{SVK}[L] \in \{0, 1\}^L$, defines the vector $\vec{f}_{\mathbf{SVK}} = \vec{f}_{3,0} \cdot \prod_{i=1}^L \vec{f}_{3,i}^{\mathbf{SVK}[i]}$ and assembles a Groth-Sahai CRS $\mathbf{f}_{\mathbf{SVK}} = (\vec{f}_1, \vec{f}_2, \vec{f}_{\mathbf{SVK}})$. Using $\mathbf{f}_{\mathbf{SVK}}$, generates commitments $\vec{C}_z, \vec{C}_r, \vec{C}_u$ to the components of $(z, r, u) \in \mathbb{G}^3$ along with proofs $(\vec{\pi}_1, \vec{\pi}_2)$ as in step 2 of the proving algorithm of Section 6.2. Lets $(\vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2) \in \mathbb{G}^{15}$ be the resulting NIWI proof.

5. Generates $\sigma = \mathcal{S}(\mathbf{SSK}, (C_0, C_1, C_2, C_3, Z, R, U, \vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2))$ and outputs

$$C = (\mathbf{SVK}, C_0, C_1, C_2, C_3, Z, R, U, \vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2, \sigma) \quad (7.7)$$

Ciphertext-Verify(PK, C): parses C as in (7.7). Returns 1 if and only if: (i) $\mathcal{V}(\text{SVK}, (C_0, C_1, C_2, C_3, Z, R, U, \vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2), \sigma) = 1$; (ii) $(Z, R, U) \in \mathbb{G}^3$ is a valid homomorphic signature on (C_1, C_2, C_3) ; (iii) $(\vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2) \in \mathbb{G}^{15}$ is a valid proof w.r.t. the CRS $(\vec{f}_1, \vec{f}_2, \vec{f}_{\text{SVK}})$ that committed (z, r, u) satisfy the relations (4.9) for the vector $(C_1, C_2, C_3) \in \mathbb{G}^3$. Here, we define $\vec{f}_{\text{SVK}} = \vec{f}_{3,0} \cdot \prod_{i=1}^L \vec{f}_{3,i}^{\text{SVK}[i]}$.

Share-Decrypt($PK, i, SK_{d,i}, C$): on inputs $SK_{d,i} = (P_1(i), P_2(i), P(i)) \in \mathbb{Z}_p^3$ and C , returns $(i, \perp)$ if **Ciphertext-Verify**(PK, C) = 0. Otherwise, computes $\hat{\mu}_i = (\nu_i, \vec{C}_{P_1}, \vec{C}_{P_2}, \vec{C}_P, \pi_{\mu_i})$ which consists of a partial decryption $\nu_i = C_1^{P_1(i)} \cdot C_2^{P_2(i)} \cdot C_3^{P(i)}$ as well as commitments $\vec{C}_{P_1}, \vec{C}_{P_2}, \vec{C}_P$ to exponents $P_1(i), P_2(i), P(i) \in \mathbb{Z}_p$ and a proof π_{ν_i} that these satisfy the linear equations in the variables

$$\nu_i = C_1^{P_1(i)} \cdot C_2^{P_2(i)} \cdot C_3^{P(i)}, \quad Y_{i,1} = f^{P_1(i)} g^{P(i)}, \quad Y_{i,2} = h^{P_2(i)} g^{P(i)}.$$

The commitments $\vec{C}_{P_1}, \vec{C}_{P_2}, \vec{C}_P$ and the proof π_{ν_i} are generated using the CRS $(\vec{f}_{r,1}, \vec{f}_{r,2}, \vec{f}_{r,3})$ (see Section 2.5.3 and Equation 2.28 for details). Then, returns $\mu_i = (i, \hat{\mu}_i)$.

Share-Verify($PK, VK_i, C, (i, \hat{\mu}_i)$): parses C as in (7.7) and VK_i as $(Y_{i,1}, Y_{i,2})$. If $\hat{\mu}_i = \perp$ or $\hat{\mu}_i$ cannot be parsed as $(\nu_i, \vec{C}_{P_1}, \vec{C}_{P_2}, \vec{C}_P, \pi_{\mu_i})$, returns 0. Otherwise, returns 1 if and only if π_{μ_i} is valid.

Combine($PK, \mathbf{VK}, C, \{(i, \hat{\mu}_i)\}_{i \in S}$): for each $i \in S$, parses the decryption share $\hat{\mu}_i$ as $(\nu_i, \vec{C}_{P_1}, \vec{C}_{P_2}, \vec{C}_P, \pi_{\mu_i})$ and returns $\perp$ if **Share-Verify**($PK, C, (i, \hat{\mu}_i)$) = 0. Otherwise, computes $\nu = \prod_{i \in S} \nu_i^{\Delta_{i,S}(0)}$, which equals $\nu = C_1^{x_1} \cdot C_2^{x_2} \cdot C_3^{x_0} = X_1^{\theta_1} \cdot X_2^{\theta_2}$ and in turn reveals $M = C_0/\nu$.

Eval($PK, SK_h, C^{(1)}, C^{(2)}$): parses SK_h as $\{(\chi_i, \gamma_i, \delta_i)\}_{i=1}^3$. For each $j \in \{1, 2\}$, parses $C^{(j)}$ as

$$C^{(j)} = (\text{SVK}^{(j)}, C_0^{(j)}, C_1^{(j)}, C_2^{(j)}, C_3^{(j)}, Z^{(j)}, R^{(j)}, \\ U^{(j)}, \vec{C}_z^{(j)}, \vec{C}_r^{(j)}, \vec{C}_u^{(j)}, \vec{\pi}_1^{(j)}, \vec{\pi}_2^{(j)}, \sigma^{(j)})$$

and returns $\perp$ if either $C^{(1)}$ or $C^{(2)}$ is invalid. Otherwise,

1. Computes $C_0 = \prod_{j=1}^2 C_0^{(j)}$, $C_1 = \prod_{j=1}^2 C_1^{(j)}$, $C_2 = \prod_{j=1}^2 C_2^{(j)}$ and $C_3 = \prod_{j=1}^2 C_3^{(j)}$ as well as $Z = \prod_{j=1}^2 Z^{(j)}$, $R = \prod_{j=1}^2 R^{(j)}$ and $U = \prod_{j=1}^2 U^{(j)}$.
2. Generates a new one-time signature key pair $(\text{SVK}, \text{SSK}) \leftarrow \mathcal{G}(\lambda)$. Using $SK_h = \{(\chi_i, \gamma_i, \delta_i)\}_{i=1}^3$, generates proof elements $\vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2$ on the vector (C_1, C_2, C_3) using the simulator of the proof system in Section 6.2 with the one-time verification key SVK .

3. Returns the derived ciphertext $C = (\text{SVK}, C_0, C_1, C_2, C_3, Z, R, U, \vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2, \sigma)$ where $\sigma = \mathcal{S}(\text{SSK}, (C_0, C_1, C_2, C_3, Z, R, U, \vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2))$.

In the next Section 7.3.3, we prove the KH-CCA security of the scheme assuming that Σ is a strongly unforgeable one-time signature and that the DLIN assumption holds in $\mathbb{G}$.

In some applications, it may be desirable to add an extra randomization step to the evaluation algorithm in order to make sure that derived ciphertexts will be indistinguishable from freshly generated encryption (similarly to [125]). It is straightforward to modify the scheme to obtain this property.

If the scheme is instantiated using Groth's one-time signature [80], the ciphertext consists of 25 elements of $\mathbb{G}$ and two elements of $\mathbb{Z}_p$. It is interesting to compare the above system with an instantiation of the same design principle using the best known Groth-Sahai-based unbounded simulation-sound proof [36][Appendix A.2], which requires 65 group elements in this specific case. With this proof system, we end up with 77 group elements per ciphertexts under the DLIN assumption (assuming that an element of $\mathbb{Z}_p$ has the same length as the representation of a group element). The above realization thus saves 50 group elements and compresses ciphertexts to 35% of their original length.

We note that it is possible to adapt the scheme to rely on the Symmetric eXternal Diffie-Hellman assumption in asymmetric pairings $e : \mathbb{G} \times \hat{\mathbb{G}} \rightarrow \mathbb{G}_T$. In this case, the ciphertext contains 9 elements of $\mathbb{G}$, 2 elements of $\hat{\mathbb{G}}$ and a one-time key pair (SVK, σ) . Using the one-time signature of [80], the ciphertext overhead amounts to 4096 bits on Barreto-Naehrig curves [12] if each element of $\mathbb{G}$ (resp. each element of $\hat{\mathbb{G}}$) has a 256-bit (resp. 512-bit) representation.

7.3.3 Security of the Keyed-Homomorphic cryptosystem

Instead of relying on the simulation-soundness of the proof system in a modular manner, our proof of KH-CCA security uses a direct security analysis in order to obtain a tighter reduction.

Theorem 23 *The threshold keyed-homomorphic cryptosystem of Section 7.3.2 provides KH-CCA security in the sense of Definition 4.3 assuming that: (i) Σ is a strongly unforgeable one-time signature; (ii) The DLIN assumption holds in $\mathbb{G}$. Concretely, the advantage of any PPT adversary $\mathcal{A}$ is at most*

$$\begin{aligned} \text{Adv}^{\text{kh-cca}}(\mathcal{A}) &< (q_e + 1) \cdot \text{Adv}^{\text{suf-ots}}(\lambda) + \frac{3}{2} \cdot \text{Adv}^{\text{DLIN}}(\lambda) \\ &\quad + 7 \cdot \sqrt{(q_e + 1) \cdot (L + 1) \cdot \text{Adv}^{\text{DLIN}}(\lambda) \cdot \left(1 - \frac{1}{p}\right)^{-1}} + \frac{1}{p}, \end{aligned}$$

where $\text{Adv}^{\text{suf-ots}}(\lambda)$ denotes $\mathcal{A}$'s probability to break the strong unforgeability of Σ and q_e is the maximal number of evaluation queries involving derivatives of the challenge ciphertext.

Proof. The proof uses a sequence of games starting with the real attack game and ending with a game where the adversary $\mathcal{A}$ has no advantage. For each i , we also denote by S_i the event that the challenger outputs 1 in **Game_i**.

Game₁: is the actual attack game with the only difference that the challenger $\mathcal{B}$ does not erase $\mathbf{sk}_{ot}$ during the key generation phase. In details, the adversary is given the public key PK and the set of verification keys $\mathbf{VK} = (VK_1, \dots, VK_N)$. If $\mathcal{A}$ decides to query the **RevHK** oracle at some point, $\mathcal{B}$ reveals SK_h . At each corruption query $i \in \{1, \dots, N\}$, $\mathcal{B}$ reveals the queried private key share $SK_i = (P_1(i), P_2(i), P(i))$. At each decryption query, $\mathcal{B}$ faithfully runs the real shared decryption algorithm. At each evaluation query, $\mathcal{A}$ supplies two ciphertexts $C^{(1)}, C^{(2)}$ which are processed by $\mathcal{B}$ as in the evaluation algorithm. We denote by $C_1^\dagger, \dots, C_{q_e}^\dagger$ the outputs of the $\text{Eval}(SK_h, \cdot)$ oracle when the latter is queried on a pair $(C^{(1)}, C^{(2)})$ such that $C^{(j)} \in \mathcal{D}$ for some $j \in \{1, 2\}$ (in other words, $\{C_l^\dagger\}_{l=1}^{q_e}$ are the results of evaluation queries which increase $|\mathcal{D}|$). We also denote by $\text{SVK}_1^\dagger, \dots, \text{SVK}_{q_e}^\dagger$ the one-time verification keys appearing in these ciphertexts and assume w.l.o.g. that they are chosen by $\mathcal{B}$ at the very beginning of the game.

When the first phase is over, the adversary $\mathcal{A}$ chooses two distinct messages $M_0, M_1 \in \mathbb{G}$ and obtains $C^* = (\text{SVK}^*, C_0^*, C_1^*, C_2^*, C_3^*, Z^*, R^*, U^*, \vec{C}_z^*, \vec{C}_r^*, \vec{C}_u^*, \vec{\pi}_1^*, \vec{\pi}_2^*, \sigma^*)$ which is an encryption of M_β , for some random coin $\beta \xleftarrow{\$} \{0, 1\}$ flipped by $\mathcal{B}$.

In the second phase, $\mathcal{A}$ makes more decryption, evaluation and corruption queries under the restriction of not asking for a partial decryption of a ciphertext in $\mathcal{D}$ or for more than $t - 1$ private key shares throughout the entire game. Eventually, $\mathcal{A}$ halts and outputs $\beta' \in \{0, 1\}$. The challenger $\mathcal{B}$ outputs 1 if and only if $\beta = \beta'$. We denote this event by S_1 .

Game₂: This game is identical to **Game₁** with the difference that the challenger $\mathcal{B}$ rejects all decryption queries involving ciphertexts $C = (\text{SVK}, C_0, C_1, C_2, C_3, Z, R, U, \vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2, \sigma)$ such that $\text{SVK} \in \{\text{SVK}^*, \text{SVK}_1^\dagger, \dots, \text{SVK}_{q_e}^\dagger\}$. It also returns $\perp$ at each evaluation query $(C^{(1)}, C^{(2)})$ for which there exists $j \in \{1, 2\}$ for which $C^{(j)}$ contains a verification key $\text{SVK}^{(j)}$ such that $\text{SVK}^{(j)} = \text{SVK}_l^\dagger$, for some $\text{SVK}_l^\dagger \in \{\text{SVK}^*, \text{SVK}_1^\dagger, \dots, \text{SVK}_{q_e}^\dagger\}$, but $C^{(j)} \neq C_l^\dagger$.

If we define F_2 to be the event that $\mathcal{B}$ rejects a ciphertext that would not have been rejected in **Game₁**, we see that **Game₂** and **Game₁** proceed identically until event F_2 occurs. We thus have the inequality $|\Pr[S_2] - \Pr[S_1]| \leq \Pr[F_2]$. Moreover, event F_2 would imply a breach in the strong unforgeability of the one-time signature. Indeed, since $\mathcal{A}$ is not allowed to query the partial decryption of any ciphertext in $\mathcal{D}$, a standard argument allows proving the inequality $|\Pr[S_2] - \Pr[S_1]| \leq \Pr[F_2] \leq (q_e + 1) \cdot \text{Adv}^{\text{suf-ots}}(\lambda)$. In the forthcoming games, we will assume that $\text{SVK} \notin \{\text{SVK}^*, \text{SVK}_1^\dagger, \dots, \text{SVK}_{q_e}^\dagger\}$ at each decryption query.

Game₃: We modify the generation of the challenge ciphertext C^* . Namely, instead of computing $C_0^* = M_\beta \cdot X_1^\theta \cdot X_2^\theta$, using the encryption exponents $\theta_1 = \log_f(C_1^*)$ and $\theta_2 = \log_h(C_2^*)$, $\mathcal{B}$ uses the private key (x_0, x_1, x_2) and computes $C_0^* = M_\beta \cdot C_1^{*x_1} \cdot C_2^{*x_2} \cdot C_3^{*x_0}$. Likewise, instead of using (θ_1, θ_2) to derive the triple (z^*, r^*, u^*) at step 3 of the encryption algorithm, $\mathcal{B}$ uses the simulation trapdoor $\mathbf{sk}_{rand} = \{(\chi_i, \gamma_i, \delta_i)\}_{i=1}^3$ of the USS proof system and computes

$$z^* = \prod_{i=1}^3 C_i^{*- \chi_i} \quad r^* = \prod_{i=1}^3 C_i^{*- \gamma_i} \quad u^* = \prod_{i=1}^3 C_i^{*- \delta_i}. \quad (7.8)$$

Finally, (Z^*, R^*, U^*) is generated using $\mathbf{sk}_{ot} = \{(\varphi_i, \vartheta_i, \varpi_i)\}_{i=1}^3$ as

$$Z^* = \prod_{i=1}^3 C_i^{*- \varphi_i} \quad R^* = \prod_{i=1}^3 C_i^{*- \vartheta_i} \quad U^* = \prod_{i=1}^3 C_i^{*- \varpi_i}.$$

Then, $\mathcal{B}$ conducts steps 4-6 as in the actual encryption algorithm. It is easy to see that this change does not modify $\mathcal{A}$'s view since C_0^* still equals $C_0^* = M_\beta \cdot X_1^{\theta_1} \cdot X_2^{\theta_2}$ and the distributions of (z^*, r^*, u^*) and (Z^*, R^*, U^*) remain the same as in **Game₂**. We thus have $\Pr[S_3] = \Pr[S_2]$.

Game₄: This game is identical to **Game₃** with a new modification in the challenge ciphertext. Namely, instead of setting $C_3^* = g^{\theta_1 + \theta_2}$, where $\theta_1 = \log_f(C_1^*)$ and $\theta_2 = \log_h(C_2^*)$, we choose it as $C_3^* \xleftarrow{\$} \mathbb{G}$. At the third step of the encryption algorithm, the linearly homomorphic signature (z^*, r^*, u^*) is computed according to (7.8), as previously. Under the DLIN assumption in $\mathbb{G}$, this modification should not significantly alter $\mathcal{A}$'s behaviour. In particular, we have $|\Pr[S_4] - \Pr[S_3]| \leq \mathbf{Adv}^{\text{DLIN}}(\lambda)$.

Game₅: From this point forward, we make explicit use of the discrete logarithms $\alpha_f = \log_g(f)$ and $\alpha_h = \log_g(h)$, which is allowed since we are done with the transition consisting in tampering with C_3^* in the challenge ciphertext. In **Game₅**, we first change the treatment of ciphertexts $C = (\text{SVK}, C_0, C_1, C_2, C_3, Z, R, U, \vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2, \sigma)$ involved in *pre-challenge* decryption and evaluation queries. Namely, $\mathcal{B}$ simply ignores the linearly homomorphic signatures contained in these ciphertexts and returns $\perp$ if $C_3 \neq C_1^{1/\alpha_f} \cdot C_2^{1/\alpha_h}$. Otherwise, it responds as in earlier games.

If we call F_5 the event that $\mathcal{B}$ rejects a ciphertext that would not have been rejected in **Game₄**, **Game₅** and **Game₄** are clearly identical until F_5 occurs, so that $|\Pr[S_5] - \Pr[S_4]| \leq \Pr[F_5]$. At the same time, Lemma 9 shows that $\Pr[F_5] \leq \frac{1}{2} \cdot \mathbf{Adv}^{\text{DLIN}}(\lambda) + 1/p$.

The proof of Lemma 9 implies that, even if $\mathcal{A}$ chooses to expose the evaluation key SK_h at the very beginning of the game, it should not be able to create valid-looking ill-formed ciphertexts before the challenge phase unless the DLIN assumption is false. This will help us prove that the scheme remains IND-CCA1 if SK_h is revealed to the adversary when the game begins.

Game₆: We now modify the treatment of *post-challenge* queries and introduce yet another event F_6 . In this game, the challenger $\mathcal{B}$ halts and outputs a random bit in the event that the adversary $\mathcal{A}$ manages to query the partial decryption oracle or the evaluation oracle on a ciphertext $C^\circ = (\text{SVK}^\circ, C_0^\circ, C_1^\circ, C_2^\circ, C_3^\circ, Z^\circ, R^\circ, U^\circ, \vec{C}_z^\circ, \vec{C}_r^\circ, \vec{C}_u^\circ, \vec{\pi}_1^\circ, \vec{\pi}_2^\circ, \sigma^\circ)$ where $\text{SVK}^\circ \notin \{\text{SVK}^*, \text{SVK}_1^\dagger, \dots, \text{SVK}_{q_e}^\dagger\}$ and $C_3^\circ \neq C_1^{\circ 1/\alpha_f} \cdot C_2^{\circ 1/\alpha_h}$ although $(\vec{C}_z^\circ, \vec{C}_r^\circ, \vec{C}_u^\circ, \vec{\pi}_1^\circ, \vec{\pi}_2^\circ)$ is a valid linearly homomorphic signature on the vector $(C_1^\circ, C_2^\circ, C_3^\circ)$. We say that C° is a *fatal* query in this case. Since Game₆ is identical to Game₅ until event F_6 occurs, we have $|\Pr[S_6] - \Pr[S_5]| \leq \Pr[F_6]$. Lemma 10 demonstrates that the DLIN assumption can be broken in the group $\mathbb{G}$ if event F_6 occurs with non-negligible probability. More precisely, Lemma 10 shows that

$$\Pr[F_6] \leq 7 \cdot \sqrt{(q_e + 1) \cdot (L + 1) \cdot \text{Adv}^{\text{DLIN}}(\lambda) \cdot \left(1 - \frac{1}{p}\right)^{-1}}.$$

Game₇: We now modify the partial decryption oracle and replace the non-interactive proofs contained in decryption shares $\hat{\mu}_i$ by simulated NIZK proofs. This entails to turn $(\vec{f}_1, \vec{f}_2, \vec{f}_3)$ into a perfectly hiding Groth-Sahai CRS (where $\vec{f}_3$ is in the span of $\vec{f}_1$ and $\vec{f}_2$) and non-interactive proofs for multi-exponentiation equations are simulated using the trapdoor of the simulated CRS. Under the DLIN assumption, this change is not noticeable to $\mathcal{A}$ and we have $|\Pr[S_7] - \Pr[S_6]| \leq \text{Adv}^{\text{DLIN}}(\lambda)$.

In Game₇, it is easy to see that $\mathcal{A}$ has no advantage whatsoever, so that $\Pr[S_7] = 1/2$. Indeed, in the challenge phase, we have $(C_1^*, C_2^*, C_3^*) = (f^{\theta_1}, h^{\theta_2}, g^{\theta_1 + \theta_2 + \theta_3})$, where $\theta_1, \theta_2, \theta_3 \in_R \mathbb{Z}_p$. This implies that C_0^* can be written as $C_0^* = M_\beta \cdot X_1^{\theta_1} \cdot X_2^{\theta_2} \cdot g^{\theta_3 \cdot x_0}$. The latter equality implies that, as long as $x_0 \in \mathbb{Z}_p$ remains independent of $\mathcal{A}$'s view, so does the challenger's bit $\beta \in \{0, 1\}$.

To see why $\mathcal{A}$ does not learn anything about $x_0 \in \mathbb{Z}_p$, we first note that the homomorphic evaluation key SK_h is independent of x_0 , so that homomorphic evaluation queries leak nothing about it. We also remark that, in Game₇, decryption shares $\hat{\mu}_i$ contain NIZK proofs that are simulated without using private key shares. Hence, as long as $\mathcal{A}$ does not corrupt more than $t - 1$ servers, the only possible way to infer information about $x_0 = P(0)$ is to trick the partial decryption oracle into accepting an invalid ciphertext. However, in Game₇, all invalid ciphertexts are explicitly rejected.

We thus find the announced result

$$\begin{aligned} |\Pr[S_1] - \frac{1}{2}| &< (q_e + 1) \cdot \text{Adv}^{\text{suf-ots}}(\lambda) + \frac{3}{2} \cdot \text{Adv}^{\text{DLIN}}(\lambda) \\ &\quad + 7 \cdot \sqrt{(q_e + 1) \cdot (L + 1) \cdot \text{Adv}^{\text{DLIN}}(\lambda) \cdot \left(1 - \frac{1}{p}\right)^{-1}} + \frac{1}{p}. \end{aligned}$$

□

Lemma 9 *In Game_5 , the probability of event F_5 is at most*

$$\Pr[F_5] \leq \frac{1}{2} \cdot \mathbf{Adv}^{\text{DLIN}}(\lambda) + 1/p.$$

Proof. We show that, if event F_5 occurs with non-negligible probability in Game_5 , there exists an efficient algorithm $\mathcal{B}$ that solves a SDP instance (G_z, G_r, H_z, H_u) with nearly the same probability. In turn, $\mathcal{B}$ implies a distinguisher for the DLIN assumption in $\mathbb{G}$.

Algorithm $\mathcal{B}$ generates public key components that are defined as in the actual scheme. In particular, $\mathcal{B}$ sets $G_i = G_z^{\varphi_i} G_r^{\vartheta_i}$ and $H_i = H_z^{\varphi_i} H_u^{\varpi_i}$ with $\varphi_i, \vartheta_i, \varpi_i \xleftarrow{\$} \mathbb{Z}_p$ for $i \in \{1, 2, 3\}$.

Throughout the game, the reduction $\mathcal{B}$ answers $\mathcal{A}$'s decryption queries in the same way as in Game_5 . Moreover, since $\mathcal{B}$ has generated $(\mathbf{sk}_{\text{rand}}, \mathbf{pk}_{\text{rand}})$ faithfully, it is able to consistently reveal the evaluation key SK_h in case $\mathcal{A}$ decides to corrupt the evaluator. If event F_5 occurs with non-negligible probability, we know that, before the challenge phase, $\mathcal{A}$ must query the partial decryption or the homomorphic evaluation of a ciphertext $C = (\mathbf{SVK}, C_0, C_1, C_2, C_3, Z, R, U, \vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2, \sigma)$ such that (Z, R, U) is a valid one-time linearly homomorphic signature on (C_1, C_2, C_3) although (C_1, C_2, C_3) is outside the span of $(f, 1, g)$ and $(1, h, g)$. When algorithm $\mathcal{B}$ detects this event (by observing that $C_3 \neq C_1^{1/\alpha_f} C_2^{1/\alpha_h}$), it computes its own signature

$$(Z^\dagger, R^\dagger, U^\dagger) = \left(\prod_{i=1}^3 C_i^{-\varphi_i}, \prod_{i=1}^3 C_i^{-\vartheta_i}, \prod_{i=1}^3 C_i^{-\varpi_i} \right) \quad (7.9)$$

on (C_1, C_2, C_3) . We claim that, with overwhelming probability,

$$(Z^\dagger, R^\dagger, U^\dagger) = \left(\frac{Z}{Z^\dagger}, \frac{R}{R^\dagger}, \frac{U}{U^\dagger} \right)$$

is a non-trivial solution to the SDP instance since $Z^\dagger \neq 1_{\mathbb{G}}$ with overwhelming probability.

Indeed, we remark that the vector $(\varphi_1, \varphi_2, \varphi_3)$ is independent of $\mathcal{A}$'s view before the challenge phase. Consequently, since (C_1, C_2, C_3) is linearly independent of $(f, 1, g)$ and $(1, h, g)$, the adversary $\mathcal{A}$ is only able to predict the value $Z^\dagger$ of (7.9) with probability $1/p$. Given that we also have the inequality $\mathbf{Adv}^{\text{SDP}}(\lambda) \leq \frac{1}{2} \cdot \mathbf{Adv}^{\text{DLIN}}(\lambda)$, we thus find $\Pr[F_5] \leq \frac{1}{2} \cdot \mathbf{Adv}^{\text{DLIN}}(\lambda) + \frac{1}{p}$ as claimed. $\square$

In the transition from Game_5 to Game_6 , we could rely on the unbounded simulation-soundness of the underlying QA-NIZK proof to argue that fatal decryption or evaluation queries only occur with negligible probability. To do this, we would have to build a reduction algorithm that interacts with a simulation-soundness challenger for a given matrix $\rho \in \mathbb{G}^{2 \times 3}$ and simultaneously emulates $\mathcal{A}$'s challenger in the KH-CCA game. Since the reduction would not have the matrix $\mathbf{A} \in \mathbb{Z}_p^{2 \times 3}$, it would have no way to detect fatal queries. Consequently, the reduction would have to guess this query, which would introduce an extra degradation factor in the reduction.

Lemma 10 *In Game_6 , the probability of event F_6 is at most*

$$\Pr[F_6] \leq 7 \cdot \sqrt{(q_e + 1) \cdot (L + 1) \cdot \mathbf{Adv}^{\text{DLIN}}(\lambda) \cdot \left(1 - \frac{1}{p}\right)^{-1}}.$$

Proof. The key argument of the proof is that, conditionally on a certain desirable event, the evaluation oracle $\text{Eval}(SK_h, \cdot)$ will never information-theoretically reveal its evaluation key SK_h .

Assuming that event F_6 occurs with non-negligible probability in Game_6 , we show that there exists a distinguisher for the DLIN assumption in $\mathbb{G}$. To this end, we consider a subsequence of games starting with Game_6 and ending with $\text{Game}_{6.2}$. For each $j \in \{0, 1, 2\}$, we define $F_{6,j}$ as the counterpart of event F_6 in $\text{Game}_{6,j}$ (note that $F_{6,j}$ is efficiently detectable for each j). We first show that, as long as the DLIN assumption holds, if $\Pr[F_6]$ is non-negligible, so is $\Pr[F_{6.2}]$.

Game_{6.0} : This game is identical to Game_6 but we modify the generation of pk_{rand} when the public key is set up. Namely, the vectors $(\vec{f}_1, \vec{f}_2, \{\vec{f}_{3,i}\}_{i=0}^L)$ are chosen by setting $\vec{f}_1 = (f_1, 1_{\mathbb{G}}, g)$ and $\vec{f}_2 = (1_{\mathbb{G}}, f_2, g)$ where $f_1, f_2 \xleftarrow{\$} \mathbb{G}$ are chosen at random. As for $\{\vec{f}_{3,i}\}_{i=0}^L$, they are obtained as

$$\begin{aligned} \vec{f}_{3,0} &= \vec{f}_1^{\xi_{0,1}} \cdot \vec{f}_2^{\xi_{0,2}} \cdot (1, 1, g)^{\xi_{0,3}} \cdot (1, 1, g)^{\mu \cdot \zeta - \rho_0} \\ \vec{f}_{3,i} &= \vec{f}_1^{\xi_{i,1}} \cdot \vec{f}_2^{\xi_{i,2}} \cdot (1, 1, g)^{\xi_{i,3}} \cdot (1, 1, g)^{-\rho_i}, \quad i \in \{1, \dots, L\} \end{aligned} \quad (7.10)$$

with $\mu \xleftarrow{\$} \{0, \dots, L\}$, $\xi_{0,1}, \xi_{1,1}, \dots, \xi_{L,1} \xleftarrow{\$} \mathbb{Z}_p$, $\xi_{0,2}, \xi_{1,2}, \dots, \xi_{L,2} \xleftarrow{\$} \mathbb{Z}_p$, $\xi_{0,3}, \xi_{1,3}, \dots, \xi_{L,3} \xleftarrow{\$} \mathbb{Z}_p$ and $\rho_0, \rho_1, \dots, \rho_L \xleftarrow{\$} \{0, \dots, \zeta - 1\}$, with $\zeta = 2(q_e + 1)$ and where q_e is the number of evaluation queries that increase the cardinality of $\mathcal{D}$. Note that this change is only conceptual since the distribution of $\{\vec{f}_{3,i}\}_{i=0}^L$ has not changed since Game_6 . We thus have $\Pr[F_{6.0}] = \Pr[F_6]$.

Game_{6.1} : This game is like $\text{Game}_{6.0}$ but we consider another event **Good** which causes the challenger $\mathcal{B}$ to abort and output a random bit if it does *not* occur. Let $\text{SVK}_1^\dagger, \dots, \text{SVK}_{q_e}^\dagger$ be the distinct one-time verification keys appearing in outputs of the $\text{Eval}(SK_h, \cdot)$ oracle when the latter is invoked on a ciphertext in $\mathcal{D}$. Let also SVK^* be the verification key involved in the challenge ciphertext C^* and let $\text{SVK}^\diamond$ be the one involved in the first fatal query $C^\diamond$. We know that $\text{SVK}^\diamond \notin \{\text{SVK}^*, \text{SVK}_1^\dagger, \dots, \text{SVK}_{q_e}^\dagger\}$. For each verification key $\text{SVK} \in \{0, 1\}^L$, we consider the function $J(\text{SVK}) = \mu \cdot \zeta - \rho_0 - \sum_{i=1}^L \rho_i \cdot \text{SVK}[i]$. We also define **Good** to be the event that

$$J(\text{SVK}^\diamond) = 0 \quad \wedge \quad J(\text{SVK}^*) \neq 0 \quad \wedge \quad \bigwedge_{j \in \{1, \dots, q_e\}} J(\text{SVK}_j^\dagger) \neq 0. \quad (7.11)$$

We remark that the random exponents $\rho_0, \rho_1, \dots, \rho_L$ are chosen independently of $\mathcal{A}$'s view: this means that the simulator could equivalently define $\{\vec{f}_{3,i}\}_{i=0}^L$ first and only choose $\{\rho_i\}_{i=0}^L$ – together with values

$\{\xi_{3,i}\}_{i=0}^L$ explaining the $\{\vec{f}_{3,i}\}_{i=0}^L$ – at the very end of the game, when $\text{SVK}^*, \text{SVK}_1^\dagger, \dots, \text{SVK}_{q_e}^\dagger, \text{SVK}^\diamond$ have been defined. The same analysis as [138] (using the simplifications of Bellare and Ristenpart [14]) shows that $\Pr[F_{6.1} \wedge \text{Good}] \geq \Pr[F_{6.0}]^2 / (27 \cdot (q_e + 1) \cdot (L + 1))$.

This follows from the fact that, for any set of queries, a lower bound on the probability of event **Good** is $1/(2(q_e + 1)(L + 1))$. Indeed, from the known results [138, 88] on the programmability of Waters' hash function, we know that the probability, taken over the choice of $(\mu, \rho_0, \dots, \rho_L)$, to meet the conditions (7.11) is at least $1/(2(q_e + 1)(L + 1))$.

Game_{6.2}: We modify again the way to compute pk_{rand} in the generation of the public key. Namely, the vectors $\vec{f}_1 = (f_1, 1_{\mathbb{G}}, g)$, $\vec{f}_2 = (1_{\mathbb{G}}, f_2, g)$ are chosen as before. However, instead of generating $\{\vec{f}_{3,i}\}_{i=0}^L$ as in **Game_{6.1}**, we set them as

$$\begin{aligned} \vec{f}_{3,0} &= \vec{f}_1^{\xi_{0,1}} \cdot \vec{f}_2^{\xi_{0,2}} \cdot (1, 1, g)^{\mu \cdot \zeta - \rho_0} \\ \vec{f}_{3,i} &= \vec{f}_1^{\xi_{i,1}} \cdot \vec{f}_2^{\xi_{i,2}} \cdot (1, 1, g)^{-\rho_i}, \quad i \in \{1, \dots, L\} \end{aligned} \quad (7.12)$$

which amounts to setting $\xi_{0,3} = \xi_{1,3} = \dots = \xi_{L,3} = 0$. Clearly, $\{\vec{f}_{3,i}\}_{i=0}^L$ are no longer uniform in the span of $(\vec{f}_1, \vec{f}_2, (1, 1, g))$. Still, this change should have no noticeable effect on $\mathcal{A}$ if the DLIN assumption holds in $\mathbb{G}$. Concretely, if a fatal decryption/evaluation query occurs with substantially different probabilities in **Game_{6.2}** and **Game_{6.1}**, we can construct a DLIN distinguisher $\mathcal{B}^{\text{DLIN}}$ in the group $\mathbb{G}$ (recall that the reduction can detect fatal queries using $\alpha_f = \log_g(f)$ and $\alpha_h = \log_g(h)$). This distinguisher uses the random self-reducibility of DLIN to construct many independent-looking instances from the same distribution out of a given instance. For this reason, we can write $|\Pr[F_{6.2} \wedge \text{Good}] - \Pr[F_{6.1} \wedge \text{Good}]| \leq \text{Adv}_{\mathcal{B}^{\text{DLIN}}}(\lambda)$.

In **Game_{6.2}**, we show that an occurrence of event $F_{6.2}$ implies an algorithm $\mathcal{B}$ solving a given SDP instance (g_z, g_r, h_z, h_u) with non-negligible probability, which *a fortiori* breaks the DLIN assumption in $\mathbb{G}$ as the latter is implied by SDP.

By hypothesis, we know that the adversary $\mathcal{A}$ somehow manages to produce a fatal decryption/evaluation query on a ciphertext $C^\diamond$ for which $(C_1^\diamond, C_2^\diamond, C_3^\diamond)$ is outside the span of $(f, 1_{\mathbb{G}}, g)$ and $(1_{\mathbb{G}}, h, g)$ but $(\vec{C}_z^\diamond, \vec{C}_r^\diamond, \vec{C}_u^\diamond, \vec{\pi}_1^\diamond, \vec{\pi}_2^\diamond) \in \mathbb{G}^{15}$ satisfies the verification equations. At this point, if the event **Good** introduced in **Game_{6.1}** occurs, we must have $J(\text{SVK}^\diamond) = 0$, which implies that $\vec{f}_{\text{SVK}^\diamond} = \vec{f}_{3,0} \cdot \prod_{i=1}^{L+1} \vec{f}_{3,i}^{\text{SVK}^\diamond[i]}$ lies in $\text{span}(\vec{f}_1, \vec{f}_2)$. Consequently, $\vec{C}_z^\diamond, \vec{C}_r^\diamond$ and $\vec{C}_u^\diamond$ are necessarily perfectly binding and extractable commitments. Using $(\log_g(f_1), \log_g(f_2))$, $\mathcal{B}$ can thus extract the committed group elements $(z^\diamond, r^\diamond, u^\diamond) \in \mathbb{G}^3$ by BBS-decrypting of the ciphertexts $(\vec{C}_z^\diamond, \vec{C}_r^\diamond, \vec{C}_u^\diamond)$. Since $(\vec{\pi}_1^\diamond, \vec{\pi}_2^\diamond)$ are perfectly sound

Groth-Sahai proofs, the extracted elements $(z^\diamond, r^\diamond, u^\diamond)$ necessarily satisfy

$$\begin{aligned} 1_{\mathbb{G}_T} &= e(g_z, z^\diamond) \cdot e(g_r, r^\diamond) \cdot \prod_{i=1}^3 e(g_i, C_i^\diamond), \\ 1_{\mathbb{G}_T} &= e(h_z, z^\diamond) \cdot e(h_u, u^\diamond) \cdot \prod_{i=1}^3 e(h_i, C_i^\diamond). \end{aligned} \quad (7.13)$$

Having extracted $(z^\diamond, r^\diamond, u^\diamond)$, $\mathcal{B}$ also computes

$$z^\dagger = \prod_{i=1}^3 C_i^{\diamond-\chi_i} \quad r^\dagger = \prod_{i=1}^3 C_i^{\diamond-\gamma_i} \quad u^\dagger = \prod_{i=1}^3 C_i^{\diamond-\delta_i}, \quad (7.14)$$

so that $(z^\dagger, r^\dagger, u^\dagger)$ also satisfies (7.13). Since $(z^\dagger, r^\dagger, u^\dagger)$ and $(z^\diamond, r^\diamond, u^\diamond)$ both satisfy (7.13), the triple

$$(z^\dagger, r^\dagger, u^\dagger) = \left(\frac{z^\diamond}{z^\dagger}, \frac{r^\diamond}{r^\dagger}, \frac{u^\diamond}{u^\dagger} \right)$$

satisfies $e(g_z, z^\dagger) \cdot e(g_r, r^\dagger) = e(h_z, z^\dagger) \cdot e(h_u, u^\dagger) = 1_{\mathbb{G}_T}$. To conclude the proof, we argue that $z^\dagger \neq 1_{\mathbb{G}}$ with overwhelming probability.

To do this, we observe that, if the event **Good** defined in **Game**_{6.1} actually comes about, then $\mathcal{B}$ never leaks any more information about (χ_1, χ_2, χ_3) than $\mathcal{A}$ can infer by just observing $\{(z_j, r_j, u_j)\}_{j=1}^2$ in the public key. Indeed, in this case we have $J(\text{SVK}^*) \neq 0$ and $J(\text{SVK}_j^\dagger) \neq 0$ for each $j \in \{1, \dots, q_e\}$. This means that, in the challenge ciphertext and all its homomorphic evaluations, the proofs $(\vec{\pi}_1, \vec{\pi}_2)$ are perfectly WI as they are generated for a perfectly hiding Groth-Sahai CRS. For these ciphertexts, the built-in homomorphic signatures $(\vec{C}_z, \vec{C}_r, \vec{C}_u, \vec{\pi}_1, \vec{\pi}_2)$ leak nothing about the specific vector $(\chi_1, \chi_2, \chi_3) \in \mathbb{Z}_p^3$ used by $\mathcal{B}$. As a consequence, we can apply the same arguments as in the proof of Lemma 9 when it comes to argue that $z^\dagger \neq z^\diamond$ with probability $1 - 1/p$. We thus find

$$\Pr[F_{6.2} \wedge \text{Good}] = \mathbf{Adv}_{\mathcal{B}}^{\text{SDP}}(\lambda) \cdot \left(1 - \frac{1}{p}\right)^{-1}.$$

In turn, $\mathcal{B}$ implies a PPT distinguisher $\mathcal{B}^{\text{DLIN}'}$ for the DLIN assumption such that we have the inequality $\Pr[F_{6.2} \wedge \text{Good}] < \frac{1}{2} \cdot \mathbf{Adv}_{\mathcal{B}^{\text{DLIN}'}}(\lambda) \cdot \left(1 - \frac{1}{p}\right)^{-1}$. If $\mathbf{Adv}^{\text{DLIN}}(\lambda)$ denotes the maximal advantage of any PPT distinguisher against the DLIN assumption in $\mathbb{G}$, the probability to have $F_{6.1} \wedge \text{Good}$ can be bounded as $\Pr[F_{6.1} \wedge \text{Good}] \leq \frac{3}{2} \cdot \mathbf{Adv}^{\text{DLIN}}(\lambda) \cdot \left(1 - \frac{1}{p}\right)^{-1}$ in **Game**_{6.1}. This eventually yields the stated result

$$\Pr[F_6] \leq 7 \cdot \sqrt{(q_e + 1) \cdot (L + 1) \cdot \mathbf{Adv}^{\text{DLIN}}(\lambda) \cdot \left(1 - \frac{1}{p}\right)^{-1}}.$$

□

Conclusion

This thesis brings an improvement of the security of cryptographic primitives supporting public verifiability. After a preliminary part devoted to recalling some basic concepts of public-key cryptography and some more advanced building blocks, the contributions of the thesis were divided into three other parts. The second part focused on primitives related to signatures, the third one concentrated on commitments and proof systems, and the last part was dealing with threshold encryption.

Part II put forth contributions in the areas of group signatures and P -homomorphic signatures. In Chapter 3 we built two *scalable* group signature schemes with revocation. Our new revocation techniques, borrowed from the tree-based NNL framework of broadcast encryption, allowed building revocable group signatures where all the complexities depending on the number of group members N are at most polylogarithmic in N . The first construction relies on hierarchical identity-based encryptions and preserves the constant-size signatures of previous constructions and even the constant-cost verification of constructions in the random oracle model. Based on concise vector-commitments and a slightly stronger assumption, the second construction increases the performances so that the only dependency in N is a linear size public key in $\log N$, which makes our scheme competitive with group signatures that do *not* support revocation mechanisms.

Chapter 4 first improved the security model of P -homomorphic signatures. The recent unified framework of Ahn *et al.* for computing on authenticated data expressed the need for a new privacy notion called *strong context-hiding*. Although this property requires statistical indistinguishability, we showed that it is not sufficient in the case of randomizable signatures. Indeed, their model only considered honestly-generated signatures while concrete schemes allow embedding in signatures subliminal information that might be used to detect how future signatures will be performed. To fill this gap and prevent unwanted traceability, we strengthened the model with our *complete context-hiding* property that takes into account adversarially-controlled signatures. In a second step, we built the first homomorphic signature schemes achieving this requirements for the subset predicates and quotable signatures. Interestingly these realizations are both structure-preserving (SP) and more efficient than existing solutions that are only strongly context-hiding (CH). Eventually, we put forward three constructions for linearly homomorphic SP signatures: a one-

time construction where only one vector subspace can be authenticated, a full-fledged construction but not context-hiding and finally a fully secure completely CH construction.

In Part III, we showed that our linearly homomorphic structure-preserving (LH-SP) signatures have unexpected applications in commitment schemes and non-interactive proof systems. In Chapter 5 we proposed the first *constant-size* simulation-sound and *non-malleable* (NM) commitments on vectors of group elements in bilinear groups. This solution was made possible by evaluating the verification equations of LH-SP signatures on random group elements instead of actual signatures. Even though the commitments lie in the target group, it is still possible to prove statements related to committed opening values and the committed messages. This construction is purely *generic* and actually does not need to rely on any context-hiding property. This allows us to obtain a better performance of the NM commitments when we instantiate them with our second LH-SP signatures.

In Chapter 6 we presented two quasi-adaptive NIZK arguments for proving membership in linear subspaces on bilinear groups. By allowing the common reference string (CRS) to depend on the specific language of which we need to prove membership, it is possible to build shorter proofs. A deeper exploitation of our LH-SP signatures enabled us to show how the context-hiding property can provide quasi-adaptive ZK when their secret keys are used as trapdoor keys. In the same way that set membership proofs in the standard model have a CRS composed of signatures on each set elements, here we can only dispose of signatures on *bases* of the vector subspace. This enables a prover to derive signatures and prove satisfiability of the verification equation.

Combining our one-time signatures with smooth projective hash functions gave the first *constant-size* single-theorem relatively sound QA-NIZK arguments for such subspace proofs. Moreover, combining our one-time signature with Groth-Sahai NIWI proofs in the same manner as we did to obtain our completely context-hiding signatures gave the first *constant-size* solutions to *unbounded simulation-sound* QA-NIZK arguments. This result paves the way toward efficient schemes in a multi-user setting like key-dependent-message secure encryptions where the reduction should simulate a polynomial number of challenge ciphertexts related to the secret keys.

Finally, Part IV looked at publicly verifiable threshold encryption schemes in Chapter 7. The application of our quasi-adaptive NIZK arguments enabled us to build the shortest CCA-secure threshold schemes where the above single-theorem argument provides both verifiability and adaptive security. On the other hand, the unbounded simulation-sound QA-NIZK argument in combination with a strongly unforgeable one-time signature allowed us to build the first fully adaptive keyed-homomorphic CCA threshold encryptions. Indeed, since the cryptosystem must remain secure when the adversary receives new ciphertexts for adversarially-chosen evaluation of the challenge ciphertext, the reduction has to produce an unbounded number of fake proofs. This scheme is also the first where the validity of ciphertexts is publicly verifiable.

All along this thesis we proposed new pairing-based constructions by composing old and new building blocks. Their security was proved in the standard model and most of them are structure-preserving, which might allow them to be easily used in turn in other anonymity-based protocols. All the presented schemes are highly efficient and each of them supports public verifiability.

Taking a step back, the group signature schemes are those that require the strongest assumptions of the thesis. It is possible to replace some building blocks by existing variants that rely on simple(r) assumptions but this will decrease the overall performance. Therefore, it would be useful to have new solutions with weaker assumptions and without degrading the efficiency. Nevertheless, the principle based on the concise vector commitments used to prove that users are non revoked is of value in itself and it will be interesting to see if it could be applied elsewhere.

All the other contributions only assume the hardness of the DLIN problem. Even though our constructions seem optimal in several metrics, it might be possible to build new schemes with tighter reductions to obtain an even better practicality with respect to the size of the parameters. Besides, the constant-size simulation-sound arguments should have a wide range of other applications. This direction seems very promising.

It was amazing to see how relaxing the unforgeability of structure-preserving signatures has found numerous non-trivial applications. More conceptually, homomorphism may be reconciled with adaptive security and, to some extent, non-malleability may emerge from malleability.

Bibliography

- [1] M. Abe, M. Chase, B. David, M. Kohlweiss, R. Nishimaki, and M. Ohkubo. Constant-size structure-preserving signatures: Generic constructions and simple assumptions. In Wang and Sako [137], pages 4–24. 2.2.2, 4.2.3
- [2] M. Abe and S. Fehr. Adaptively secure feldman vss and applications to universally-composable threshold cryptography. In Franklin [68], pages 317–334. 7.1
- [3] M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, 2010. 6, 2.2.2, 2.2.2, 2
- [4] M. Abe, K. Haralambiev, and M. Ohkubo. Signing on elements in bilinear groups for modular protocol design. *IACR Cryptology ePrint Archive*, 2010:133, 2010. 9, 2.2.2, 2.2.2, 2.2.2, 2, 4.3.1, 4.3.3, 4.4.1, 5.2.3
- [5] M. Abe, K. Haralambiev, and M. Ohkubo. Group to group commitments do not shrink. In Pointcheval and Johansson [124], pages 301–317. 1.2.2, 2.4.3, 5.2, 1
- [6] J. Ahn, D. Boneh, J. Camenisch, S. Hohenberger, A. Shelat, and B. Waters. Computing on authenticated data. In Cramer [52], pages 1–20. 1.2.1, 4, 26, 27, 4.1.1, 28, 29, 4.1.1, 4.1.2, 4.2, 4.2.3, 4.3, 4.3.2, 4.4, 4.4.4
- [7] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song. Provable data possession at untrusted stores. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM Conference on Computer and Communications Security*, pages 598–609. ACM, 2007. 4.4
- [8] G. Ateniese, S. Kamara, and J. Katz. Proofs of storage from homomorphic identification protocols. In Matsui [116], pages 319–333. 4.4
- [9] N. Attrapadung and B. Libert. Homomorphic network coding signatures in the standard model. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 17–34. Springer, 2011. 4.1.2

- [10] N. Attrapadung, B. Libert, and T. Peters. Computing on authenticated data: New privacy definitions and constructions. In Wang and Sako [137], pages 367–385. 1.2.1, 4.1.3, 4.4.1, 5.2.3
- [11] N. Attrapadung, B. Libert, and T. Peters. Efficient completely context-hiding quotable and linearly homomorphic signatures. In Kurosawa and Hanaoka [102], pages 386–404. 1.2.1, 4.4, 4.4.4
- [12] P. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford E. Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer, 2005. 7.2.3, 7.3.2
- [13] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham. Randomizable proofs and delegatable anonymous credentials. In Halevi [85], pages 108–125. 4.3.2
- [14] M. Bellare and T. Ristenpart. Simulation without the artificial abort: Simplified proof and improved concrete security for waters’ ibe scheme. In Joux [93], pages 407–424. 4.4.4, 6.2.2, 7.3.3
- [15] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993. 12
- [16] M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–153. Springer, 2005. 3.1, 3.1.2
- [17] Eli Biham, editor. *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*. Springer, 2003. 42, 73
- [18] M. Blum, A. de Santis, S. Micali, and G. Persiano. Noninteractive zero-knowledge. *SIAM J. Comput.*, 20(6):1084–1118, 1991. 16
- [19] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *STOC*, pages 103–112. ACM, 1988. 1.2.2, 16
- [20] D. Boneh and X. Boyen. Efficient selective-id secure identity-based encryption without random oracles. In Cachin and Camenisch [33], pages 223–238. 2.6.2
- [21] D. Boneh and X. Boyen. Short signatures without random oracles. In Cachin and Camenisch [33], pages 56–73. 8, 2.2.2, 3.2.3, 3.3.3
- [22] D. Boneh, X. Boyen, and E. Goh. Hierarchical identity based encryption with constant size ciphertext. In Cramer [51], pages 440–456. 2.1.2, 3, 2.6.2, 2.6.2, 3.2.1

- [23] D. Boneh, X. Boyen, and S. Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. In David Pointcheval, editor, *CT-RSA*, volume 3860 of *Lecture Notes in Computer Science*, pages 226–243. Springer, 2006. 7.1
- [24] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In Franklin [68], pages 41–55. 5, 2.3.3
- [25] D. Boneh, D. Freeman, J. Katz, and B. Waters. Signing a linear subspace: Signature schemes for network coding. In Jarecki and Tsudik [92], pages 68–87. 4.1.2, 4.4, 4.4.1
- [26] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275. Springer, 2005. 10, 2.1.2
- [27] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In Boyd [30], pages 514–532. 3.2.1, 3.2.1
- [28] D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In Vijayalakshmi Atluri, Birgit Pfizmann, and Patrick Drew McDaniel, editors, *ACM Conference on Computer and Communications Security*, pages 168–177. ACM, 2004. 3.1
- [29] Colin Boyd. A new multiple key cipher and an improved voting scheme. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *EUROCRYPT*, volume 434 of *Lecture Notes in Computer Science*, pages 617–625. Springer, 1989. 7, 7.1
- [30] Colin Boyd, editor. *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, volume 2248 of *Lecture Notes in Computer Science*. Springer, 2001. 27, 66
- [31] X. Boyen, Q. Mei, and B. Waters. Direct chosen ciphertext security from identity-based techniques. In Vijay Atluri, Catherine Meadows, and Ari Juels, editors, *ACM Conference on Computer and Communications Security*, pages 320–329. ACM, 2005. 7.1
- [32] X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In Cynthia Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 290–307. Springer, 2006. 3, 2.6.2
- [33] Christian Cachin and Jan Camenisch, editors. *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*. Springer, 2004. 20, 21, 114

- [34] J. Camenisch, R. Chaabouni, and A. shelat. Efficient protocols for set membership and range proofs. In Pieprzyk [123], pages 234–252. 3.2.1, 3.3.1
- [35] J. Camenisch, M. Kohlweiss, and C. Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In Jarecki and Tsudik [92], pages 481–500. 2.1.2
- [36] Jan Camenisch, Nishanth Chandran, and Victor Shoup. A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In Joux [93], pages 351–368. 6.4, 6.5, 6.1, 7.3.2
- [37] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145. IEEE Computer Society, 2001. 5.2, 5.2.1
- [38] R. Canetti, Y. Dodis, R. Pass, and S. Walfish. Universally composable security with global setup. In Salil P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 61–85. Springer, 2007. 5
- [39] R. Canetti and M. Fischlin. Universally composable commitments. In Kilian [98], pages 19–40. 5.2
- [40] R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Adaptive security for threshold cryptosystems. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 98–115. Springer, 1999. 7.2
- [41] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited (preliminary version). In Vitter [136], pages 209–218. 2.1.3
- [42] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In Biham [17], pages 255–271. 2.6.2, 20
- [43] Ran Canetti and Juan A. Garay, editors. *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, volume 8043 of *Lecture Notes in Computer Science*. Springer, 2013. 62, 106
- [44] D. Catalano and D. Fiore. Concise vector commitments and their applications to zero-knowledge elementary databases. *IACR Cryptology ePrint Archive*, 2011:495, 2011. 2.4.4, 2.4.4
- [45] J. Cathalo, B. Libert, and M. Yung. Group encryption: Non-interactive realization in the standard model. In Matsui [116], pages 179–196. 2.1.2, 7
- [46] D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT*, pages 257–265, 1991. 1.2.1

- [47] Jean-Sébastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 476–493. Springer, 2013. 7.2.3
- [48] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO*, volume 839 of *LNCS*, pages 174–187. Springer, 1994. 19
- [49] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 1998. 2.3.3, 7.2.2, 7.3.1
- [50] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Knudsen [100], pages 45–64. 2.3.3, 6.3.1, 7.3.1
- [51] Ronald Cramer, editor. *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*. Springer, 2005. 22, 96, 138
- [52] Ronald Cramer, editor. *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, volume 7194 of *Lecture Notes in Computer Science*. Springer, 2012. 6, 113
- [53] G. Di Crescenzo, Y. Ishai, and R. Ostrovsky. Non-interactive and non-malleable commitment. In Vitter [136], pages 141–150. 37, 5.2.3
- [54] I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT*, pages 418–430, 2000. 2.5.2
- [55] I. Damgård and J. Groth. Non-interactive and reusable non-malleable commitment schemes. In Lawrence L. Larmore and Michel X. Goemans, editors, *STOC*, pages 426–437. ACM, 2003. 5, 5.2.1
- [56] Y. Desmedt. Society and group oriented cryptography: A new concept. In Carl Pomerance, editor, *CRYPTO*, volume 293 of *Lecture Notes in Computer Science*, pages 120–127. Springer, 1987. 7, 7.1
- [57] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315. Springer, 1989. 7, 7.1
- [58] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 1.1.2

- [59] Y. Dodis and N. Fazio. Public key broadcast encryption for stateless receivers. In Joan Feigenbaum, editor, *Digital Rights Management Workshop*, volume 2696 of *Lecture Notes in Computer Science*, pages 61–80. Springer, 2002. [2.6.3](#)
- [60] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography (extended abstract). In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *STOC*, pages 542–552. ACM, 1991. [5](#)
- [61] K. Emura, G. Hanaoka, G. Ohtake, T. Matsuda, and S. Yamada. Chosen ciphertext secure keyed-homomorphic public-key encryption. In Kurosawa and Hanaoka [\[102\]](#), pages 32–50. [7.1](#), [7.1.1](#), [7.3.1](#)
- [62] A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. L. Villar. An algebraic framework for diffie-hellman assumptions. In Canetti and Garay [\[43\]](#), pages 129–147. [7.2.3](#)
- [63] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986. [2.5.2](#)
- [64] M. Fischlin, B. Libert, and M. Manulis. Non-interactive and re-usable universally composable string commitments with adaptive security. In Lee and Wang [\[103\]](#), pages 468–485. [1.2.2](#), [5](#), [5.2](#)
- [65] Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors. *Public Key Cryptography - PKC 2012 - 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings*, volume 7293 of *Lecture Notes in Computer Science*. Springer, 2012. [69](#), [94](#)
- [66] P.-A. Fouque and D. Pointcheval. Threshold cryptosystems secure against chosen-ciphertext attacks. In Boyd [\[30\]](#), pages 351–368. [7.1](#)
- [67] Y. Frankel, P. MacKenzie, and M. Yung. Adaptively-secure distributed public-key systems. In Jaroslav Nešetřil, editor, *ESA*, volume 1643 of *Lecture Notes in Computer Science*, pages 4–27. Springer, 1999. [7.2](#)
- [68] Matthew K. Franklin, editor. *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*. Springer, 2004. [2](#), [24](#), [75](#)
- [69] D. Freeman. Improved security for linearly homomorphic signatures: A generic framework. In Fischlin et al. [\[65\]](#), pages 697–714. [4.4.1](#)
- [70] G. Fuchsbaauer. Commuting signatures and verifiable encryption. In Paterson [\[121\]](#), pages 224–245. [2.5.3](#), [4.2.1](#)
- [71] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discrete Appl. Math.*, 156(16), September 2008. [7](#)

- [72] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985. 2.3.3
- [73] J. Garay, P. MacKenzie, and K. Yang. Strengthening zero-knowledge protocols using signatures. In Biham [17], pages 177–194. 1.2.2, 2.5.2, 5, 5.2.1
- [74] S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2013. 7.2.3
- [75] R. Gennaro. Multi-trapdoor commitments and their applications to proofs of knowledge secure under concurrent man-in-the-middle attacks. In Franklin [68], pages 220–236. 1.2.2, 5, 5.1
- [76] R. Gennaro and S. Micali. Independent zero-knowledge sets. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP (2)*, volume 4052 of *Lecture Notes in Computer Science*, pages 34–45. Springer, 2006. 5.1, 5.2.3
- [77] C. Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *STOC*, pages 169–178. ACM, 2009. 4
- [78] C. Gentry and A. Silverberg. Hierarchical id-based cryptography. In Yuliang Zheng, editor, *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer, 2002. 2.6.2, 2.6.2
- [79] M. Gerbush, A. Lewko, A. O’Neill, and B. Waters. Dual form signatures: An approach for proving security from static assumptions. In Wang and Sako [137], pages 25–42. 4.1.2, 4.3
- [80] J. Groth. Simulation-sound nzk proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 444–459. Springer, 2006. 2.2.2, 3.3.2, 6.1, 7.2.3, 7.3.2
- [81] J. Groth. Fully anonymous group signatures without random oracles. In Kaoru Kurosawa, editor, *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 164–180. Springer, 2007. 3.2.1, 3.2.2, 3.2.3, 3.2.3, 3.2.3, 3.3, 3.3.1, 3.3.2, 3.3.3, 3.3.3
- [82] J. Groth. Homomorphic trapdoor commitments to group elements. *IACR Cryptology ePrint Archive*, 2009:7, 2009. 5, 5.2
- [83] J. Groth. Efficient zero-knowledge arguments from two-tiered homomorphic commitments. In Lee and Wang [103], pages 431–448. 5.2
- [84] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, 2008. 2.5.3, 5, 3.2.3, 5, 6.1

- [85] Shai Halevi, editor. *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, volume 5677 of *Lecture Notes in Computer Science*. Springer, 2009. 13, 139
- [86] D. Hofheinz and T. Jager. Tightly secure signatures and public-key encryption. In Safavi-Naini and Canetti [128], pages 590–607. 6.2.2
- [87] D. Hofheinz, T. Jager, and E. Kiltz. Short signatures from weaker assumptions. In Lee and Wang [103], pages 647–666. 3.2.1
- [88] D. Hofheinz and E. Kiltz. Programmable hash functions and their applications. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 21–38. Springer, 2008. 3.2.1, 3.2.3, 4.4.3, 4.4.4, 6.2.2, 7.3.3
- [89] J. Horwitz and B. Lynn. Toward hierarchical identity-based encryption. In Knudsen [100], pages 466–481. 2.6.2
- [90] M. Izabachène, B. Libert, and D. Vergnaud. Block-wise p-signatures and non-interactive anonymous credentials with efficient attributes. In Liqun Chen, editor, *IMA Int. Conf.*, volume 7089 of *Lecture Notes in Computer Science*, pages 431–450. Springer, 2011. 2.1.2
- [91] Stanislaw Jarecki and Anna Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In Bart Preneel, editor, *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 221–242. Springer, 2000. 7.1
- [92] Stanislaw Jarecki and Gene Tsudik, editors. *Public Key Cryptography - PKC 2009, 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, CA, USA, March 18-20, 2009. Proceedings*, volume 5443 of *Lecture Notes in Computer Science*. Springer, 2009. 25, 35, 117
- [93] Antoine Joux, editor. *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*. Springer, 2009. 14, 36
- [94] C. Jutla and A. Roy. Relatively-sound nizks and password-based key-exchange. In Fischlin et al. [65], pages 485–503. 6, 6.1, 6.1.2, 6.3.1, 6.1, 7.2.1, 1, 7.2.3, 7.2.4
- [95] C. Jutla and A. Roy. Shorter quasi-adaptive nizk proofs for linear subspaces. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT (1)*, volume 8269 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2013. 6.1, 6.2.1, 6.3, 6.3.1, 6.5, 6.1, 6.5, 7.2.1, 7.2.3
- [96] A. Kiayias and M. Yung. Group signatures with efficient concurrent join. In Cramer [51], pages 198–214. 3.1.1

- [97] A. Kiayias and M. Yung. Secure scalable group signature with dynamic joins and separable authorities. *IJSN*, 1(1/2):24–45, 2006. [3.1](#), [3.1.1](#), [3.1.2](#), [3.1.2](#)
- [98] Joe Kilian, editor. *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*. Springer, 2001. [39](#), [119](#)
- [99] E. Kiltz. Chosen-ciphertext security from tag-based encryption. In *TCC'06*, volume 3876 of *LNCS*, pages 581–600. Springer, 2006. [2.6.1](#), [3.2.3](#), [5](#)
- [100] Lars R. Knudsen, editor. *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*. Springer, 2002. [50](#), [89](#)
- [101] S. Kunz-Jacques and D. Pointcheval. About the security of mti/c0 and mqv. In Roberto De Prisco and Moti Yung, editors, *SCN*, volume 4116 of *Lecture Notes in Computer Science*, pages 156–172. Springer, 2006. [3](#)
- [102] Kaoru Kurosawa and Goichiro Hanaoka, editors. *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings*, volume 7778 of *Lecture Notes in Computer Science*. Springer, 2013. [11](#), [61](#)
- [103] Dong Hoon Lee and Xiaoyun Wang, editors. *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*. Springer, 2011. [64](#), [83](#), [87](#)
- [104] A. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In Pointcheval and Johansson [[124](#)], pages 318–335. [2](#)
- [105] A. Lewko and B. Waters. Unbounded hibe and attribute-based encryption. In Paterson [[121](#)], pages 547–567. [4.2](#)
- [106] B. Libert, T. Peters, M. Joye, and M. Yung. Linearly homomorphic structure-preserving signatures and their applications. In Canetti and Garay [[43](#)], pages 289–307. [1.2.1](#), [1.2.2](#), [5.2.3](#), [6.2.2](#), [7.2.1](#), [7.2.4](#)
- [107] B. Libert, T. Peters, M. Joye, and M. Yung. Non-malleability from malleability: Simulation-sound quasi-adaptive nizk proofs and cca2-secure encryption from homomorphic signatures. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT*, *Lecture Notes in Computer Science*. Springer, 2014. [1.2.2](#), [1.2.3](#)

- [108] B. Libert, T. Peters, and M. Yung. Group signatures with almost-for-free revocation. In Safavi-Naini and Canetti [128], pages 571–589. 1.2.1, 3.4
- [109] B. Libert, T. Peters, and M. Yung. Scalable group signatures with revocation. In Pointcheval and Johansson [124], pages 609–627. 1.2.1, 3.3.2, 3.4
- [110] B. Libert and D. Vergnaud. Group signatures with verifier-local revocation and backward unlinkability in the standard model. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS*, volume 5888 of *Lecture Notes in Computer Science*, pages 498–517. Springer, 2009. 3.1
- [111] B. Libert and M. Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In Daniele Micciancio, editor, *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 499–517. Springer, 2010. 2.4.4, 2.4.4
- [112] B. Libert and M. Yung. Adaptively secure forward-secure non-interactive threshold cryptosystems. In Chuankun Wu, Moti Yung, and Dongdai Lin, editors, *Inscrypt*, volume 7537 of *Lecture Notes in Computer Science*, pages 1–21. Springer, 2011. 7.1, 2
- [113] B. Libert and M. Yung. Non-interactive cca-secure threshold cryptosystems with adaptive security: New framework and constructions. In Cramer [52], pages 75–93. 7.1, 7.2.1, 7.2.3
- [114] P. MacKenzie and K. Yang. On simulation-sound trapdoor commitments. In Cachin and Camenisch [33], pages 382–400. 5, 5.1, 36, 5.1
- [115] T. Malkin, I. Teranishi, Y. Vahlis, and M. Yung. Signatures resilient to continual leakage on memory and computation. In Yuval Ishai, editor, *TCC*, volume 6597 of *Lecture Notes in Computer Science*, pages 89–106. Springer, 2011. 4.4.4, 4.4.4, 6.2.1
- [116] Mitsuru Matsui, editor. *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, volume 5912 of *Lecture Notes in Computer Science*. Springer, 2009. 8, 45
- [117] T. Nakanishi, H. Fujii, Y. Hira, and N. Funabiki. Revocable group signature schemes with constant costs for signing and verifying. In Jarecki and Tsudik [92], pages 463–480. 3.3.4, 3.1
- [118] T. Nakanishi and N. Funabiki. Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps. In Bimal K. Roy, editor, *ASIACRYPT*, volume 3788 of *Lecture Notes in Computer Science*, pages 533–548. Springer, 2005. 3.1
- [119] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In Kilian [98], pages 41–62. 2.6.3

- [120] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In Harriet Ortiz, editor, *STOC*, pages 427–437. ACM, 1990. [2.3.1](#), [2.5.1](#)
- [121] Kenneth G. Paterson, editor. *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*. Springer, 2011. [70](#), [105](#), [140](#)
- [122] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, 1991. [2.4.3](#)
- [123] Josef Pieprzyk, editor. *Advances in Cryptology - ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7-11, 2008. Proceedings*, volume 5350 of *Lecture Notes in Computer Science*. Springer, 2008. [34](#)
- [124] David Pointcheval and Thomas Johansson, editors. *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*. Springer, 2012. [5](#), [104](#), [109](#)
- [125] M. Prabhakaran and M. Rosulek. Homomorphic encryption with cca security. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfssdóttir, and Igor Walukiewicz, editors, *ICALP (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 667–678. Springer, 2008. [4.1.2](#), [7.3.2](#)
- [126] C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer, 1991. [2.3.1](#)
- [127] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978. [1.1.2](#)
- [128] Reihaneh Safavi-Naini and Ran Canetti, editors. *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*. Springer, 2012. [86](#), [108](#)
- [129] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS*, pages 543–553. IEEE Computer Society, 1999. [2.5.1](#), [18](#)
- [130] Amit Sahai. Simulation-sound non-interactive zero knowledge. Technical report, IBM RESEARCH REPORT RZ 3076, 2001. [6.1](#)

- [131] H. Shacham. A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. *IACR Cryptology ePrint Archive*, page 74, 2007. [2.3.3](#), [7.2.2](#)
- [132] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979. [2.1.4](#)
- [133] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In Kaisa Nyberg, editor, *EUROCRYPT*, volume 1403 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 1998. [7.1](#)
- [134] I. Teranishi and K. Sako. k -times anonymous authentication with a constant proving cost. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 525–542. Springer, 2006. [3.2.1](#)
- [135] Y. Tsiounis and M. Yung. On the security of elgamal based encryption. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography*, volume 1431 of *Lecture Notes in Computer Science*, pages 117–134. Springer, 1998. [2.3.3](#)
- [136] Jeffrey Scott Vitter, editor. *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23–26, 1998*. ACM, 1998. [41](#), [53](#)
- [137] Xiaoyun Wang and Kazue Sako, editors. *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2–6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*. Springer, 2012. [1](#), [10](#), [79](#)
- [138] B. Waters. Efficient identity-based encryption without random oracles. In Cramer [[51](#)], pages 114–127. [2.2.2](#), [4.2.3](#), [4.4.3](#), [4.4.3](#), [4.4.4](#), [6.2.2](#), [7.3.3](#)
- [139] B. Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In Halevi [[85](#)], pages 619–636. [4.3](#)
- [140] H. Wee. Threshold and revocation cryptosystems via extractable hash proofs. In Paterson [[121](#)], pages 589–609. [7.1](#)