Lattice-based Protocols for Privacy

Doctoral Dissertation submitted to the
Faculty of Informatics of the Università della Svizzera Italiana
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

presented by
Cecilia Boschini

under the supervision of
Prof. Stefan Wolf and Dr. Jan Camenisch

March 2020

Dissertation Committee

Prof. Marc Langheinrich	Università della Svizzera Italiana, Switzerland
Prof. Evanthia Papadopoulou	Università della Svizzera Italiana, Switzerland
Dr. Stephan Krenn	AIT Austrian Institute of Technology, Austria
Dr. Krzysztof Pietrzak	Institute of Science and Technology, Austria

Dissertation accepted on 12th March 2020

Research Advisor

Prof. Stefan Wolf

Co-Advisor

Dr. Jan Camenisch

PhD Program Director

The PhD program Director *pro tempore*

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Cecilia Boschini
Lugano, 12th March 2020

To my parents, who made this possible.

The personal is political.

Carol Hanisch

Abstract

Privacy and control over data have become a public concern. Simultaneously, the increasing likelihood of the construction of a general purpose quantum computer has led companies and governments to demand for quantum safe alternatives to the protocols used today. New schemes have been elaborated, whose conjectured security against a quantum computer relies on the hardness to solve different mathematical problems, such as problems defined over lattices. However, while quantum-safe alternatives are known, they tend to output tokens whose size is too large to be considered practical. The goal of this dissertation is to address these concerns by building privacy-preserving signatures whose security is based on the hardness of solving some problems over ideal lattices, and whose token sizes are an improvement over the state of the art.

Our first result is a toolbox of primitives (signatures, commitment and NIZK proofs) that are composable and allow building privacy-preserving protocols, such as Anonymous Attribute Tokens. The core building block are non-interactive zero-knowledge proofs with relaxed extractability that we obtained extending the construction in Lyubashevsky [2012]. In a second work, we combine them with a verifiable encryption scheme to construct a group signature whose keys and signatures require less that 2MB of storage.

Finally, we give efficient statistical zero-knowledge proofs (SNARKs) for Module/Ring LWE and Module/Ring SIS relations, providing the remaining ingredient for building efficient cryptographic protocols from lattice-based hardness assumptions. We apply our approach to the example use case of partially dynamic group signatures and obtain a lattice-based group signature that protects users against corrupted issuers, and that produces signatures smaller than the state of the art.

The results contained in this dissertation were published at international conferences.

Acknowledgements

When I started the PhD, I definitely did not know what I was getting myself into. I was not expecting to meet so many great people along the way either. This is my attempt at thanking all of them.

I was so lucky as to work in two extremely stimulating environments, the IBM - Research Laboratory in Zürich and the Università della Svizzera Italiana in Lugano, with two great supervisors, Jan Camenisch and Stefan Wolf. I am really grateful to Jan for giving me this opportunity, and for pushing me to finish what I had started when I was tempted to give up. From him, I have learned the importance of clear communication and planning, and he has shown me the importance of knowing one's own value. Stefan gifted me with his positive attitude, endless support, and many good discussion about politics, and was kind enough to allow me to visit the Facoltà Indipendente di Gandria whenever I wanted. The time I spent there was always inspiring and energizing.

At IBM, I was part of the group managed by the kindest and most supportive manager, Mike Osborne. He gave me career opportunities I could have never imagined, and he always believed I could ace them all. Among my colleagues, Gregory Neven has taught me with endless patience both how a paper ought to be written and how to climb – any success in either field is thanks to him, any blunder is my fault. I am really thankful to Manu, Mark, and Isa for their loving support, and for never cease to push me forward during dark times, and to Arne, for all the endless discussions, all the climbing, and for having helped more than once to reorder the chaos in my head. I cannot mention all the people I have been surrounded by in these years, but they have been all important: my former IBM colleagues, my yoga teacher Susi and the IBM yoga group, the members of the research group at USI, my climbing crew, my flatmate Nadia, and all my friends in Switzerland, Italy, and around the world. Nobody survives a PhD without a great support system, and I definitely got one of the best.

Finally, this is for my wonderful family. They are the strong foundation that allows me to keep going. For this, and for more, thank you.

Contents

Figures

List of Changes requested by the Committee

- Standardized the structure of the titles of the sections.
- Changed references style.
- Changed Stephan Krenn's affiliation to "AIT Austrian Institute of Technology".
- Various stylistic improvements and typos corrections.
- Improved definition of notation $\langle \cdot, \cdot \rangle$ in Definition 3.2.
- Added mention of aborts in Figure 3.2 (and similar figures)
- Fixed estimate of $\bar{N}_s$ and $\bar{N}_{s,\infty}$ in Theorem 3.34.
- Changed condition " $(R^*, \mu^*) \in \mathbf{L}_p$ " to " $(cid, R^*, \mu^*) \in \mathbf{L}_p$ " in Experiment $\mathsf{Exp}^{\mathsf{unf}}_{\mathsf{A}}$ in Figure 4.1, and edited the oracles and proofs to reflect this change.
- Added the line " $(ipk^*, isk^*) \leftarrow \mathsf{IKGen}(apar)$ " to Experiment $\mathsf{Exp}^{\mathsf{anon-b}}_{\mathsf{A}}$ in Figure 4.2, and edited the oracles and proofs to reflect this change.
- Added " instantiated with a post-quantum, collision-resistant hash function." at the end of Section 2.2.4.
- Fixed the input order of P and V throughout the text to deal correctly with the random string.
- Added the line " $\mathcal{Q}_S \leftarrow \emptyset$ " in Experiment $\mathsf{Exp}^{g-\mathsf{euf-acma}}_{\mathsf{A}}(1^\lambda)$ in Definition 3.25.

- Changed the right side of the inequality from " $\nu(\lambda)$ " to " $1 - \nu(\lambda)$ " in Definition 4.1.
- Added acknowledgments.
- Fixed all figures captions.
- Centered figure 3.10.
- Added horizontal black line to separate the figures from their captions.
- Fixed bad boxes.
- Added this list.

Chapter 1

Introduction

The recent years have seen a rise in privacy awareness. Both governments (cf. Council of European Union [2016]) and users have started to question companies' and products' data management. Unfortunately, so far it seems that the policies and security infrastructure in place are not enough. Both the spreading of private and public surveillance systems and the poor design of security infrastructure have resulted in a constant leakage of users' data. As Snowden [2019] already highlighted, it is necessary that security expert design protocols with the users' privacy in mind.

In the meantime, the progress in building quantum computers have highlighted the need for cryptographic schemes that will withstand an attacker with a full scale quantum computer at their disposal. This has boosted research in *post-quantum* cryptographic schemes, i.e., schemes that run on a classical computer and are secure even if the adversary can exploit the computational power of a quantum computer. The security of such schemes is based on the hardness of solving problems on particular algebraic structures called lattices, which were introduced as a cryptographic tool by Ajtai [1996] [1]. However, while there are already ongoing standardization processes for the basic primitives (e.g., key encapsulation mechanisms and signatures [National Institute of Standards and Technology, 2016]), more complex protocols such as group signature [Chaum and van Heyst, 1991] do not have practical, quantum-safe instantiations.

An established and successful way to construct efficient privacy-enhancing cryptographic protocols is to suitably combine various primitives such as signatures, commitments, and encryption schemes with efficient zero-knowledge proofs. Examples of such constructions include blind signatures [Abe and Ohkubo,

[1] before this work, lattices were only known as a cryptanalytic tool [Coppersmith and Shamir, 1997].

2009; Fischlin, 2006], group signatures [Bellare et al., 2003; Kiayias and Yung, 2005], electronic cash [Chaum et al., 1990], direct anonymous attestation [Brickell et al., 2004], voting schemes [Hirt and Sako, 2000], adaptive oblivious transfer [Camenisch et al., 2007, 2011], and anonymous credentials [Belenkiy et al., 2008; Camenisch and Lysyanskaya, 2001]. One of the crucial building blocks is a signature scheme with efficient zero-knowledge proofs of knowledge of a signature on a hidden message. Typically, they involve one party to prove to another the knowledge of a valid signature by a trusted third party, while hiding the signature as well as parts of the signed message. Commitment schemes are also common ingredients, either as "glue" to bridge zero-knowledge proofs over different cryptographic primitives [Camenisch et al., 2016], or to facilitate zero-knowledge proofs by hiding the message or certain components of the signature [Ateniese et al., 2000; Camenisch and Lysyanskaya, 2003; Boneh et al., 2004].

In this thesis we investigate practical, post-quantum (in fact, lattice-based), privacy-preserving protocols that can be built by combining digital signatures [Diffie and Hellman, 1976] with non-interactive zero-knowledge proofs (NIZKs) [Blum et al., 1988]. With *practical* schemes, we mean schemes that have keys and that produce cryptographic artifacts which sizes are close to their classical counterparts. We gave ourselves as target sizes around 1 MB for both keys and tokens/signatures. While this is not quite as small as for classical signatures (cf. for example the group signature by Boneh and Boyen [2004], where the size of a signature is under 200 bytes), this is considerably lower than anything that was published before our works (cf. the overview in [Libert, Ling, Nguyen and Wang, 2016]).

The first step towards constructing privacy-preserving schemes is to construct a toolbox of lattice-based primitives that are efficiently composable. The fundamental component in this toolbox are non-interactive zero-knowledge proofs for lattice-based relations. Such primitives allow a user P to prove to a verifier V knowledge of a secret without revealing it, and are fundamental to protect the users' privacy in some scenarios, e.g., in authentication. Assume that P wants to prove to the verifier that it has a secret witness w for a public instance x, such that (x, w) satisfies some relation $\mathscr{R}$. The prover starts by sending a commitment value α to the verifier. Then, V sends back a challenge β, to which P replies with some response γ. The verifier accepts if (α, β, γ) satisfy some constraints. The proof is sound if the prover cannot run an accepting protocol without knowing a valid w; the probability that a prover outputs a proof π and a x such that π is accepted by V while P does not know w such that $(x, w) \in \mathscr{R}$ is called *soundness error*. Hence, a zero-knowledge proof has a soundness error

of at least $1/|challenges|$, as if the prover could correctly guess the challenge, it could generate a commitment and response so that the verifier would accept. Non-interactive proofs can be obtained from this interactive protocol through the Fiat-Shamir heuristic [Fiat and Shamir, 1987].

Most lattice-based zero-knowledge proofs are either derived from the identification scheme by Lyubashevsky [2012], which is a 3-round interactive protocol with binary challenges, or from Stern-type proofs [Stern, 1994], which allow for challenges in $\{0,1,2\}$. Because of the large soundness errors these proofs incur, they have to be repeated many times in parallel, which comes at a considerable cost in efficiency. Lyubashevsky's "Fiat-Shamir with Aborts" technique [Lyubashevsky, 2012] yields much more efficient proofs with large challenges, but these proofs have the disadvantage that they are "relaxed", in the sense that extracted witnesses are only guaranteed to lie in a considerably larger domain than the witnesses used to construct the proof.

We focus on group signatures (and anonymous attribute tokens, that can be used as building blocks to construct group signatures, cf. Section 5.2) as they are particularly useful in scenarios where remote devices need to be authenticated, but privacy imposes that individual devices can only be identified by a designated authority. Examples include government-issued electronic identity (eID) cards, where each issued smart card creates identity claims as signed statements about its attributes, without needing to fully identify its owner [Bichsel et al., 2009], or remote anonymous attestation of computing platforms, where devices prove which software they execute [Brickell et al., 2004].

1.1 Our Results

In our first approach, we consider a generalization of the NIZK proof obtainable from [Lyubashevsky, 2012] that we call *relaxed NIZK proofs*. We show that, despite the protocol only allows to prove knowledge of a pair (x,w) that satisfies an approximation of the target relation $\mathscr{R}$, this relaxation in the definition still allows to obtain privacy-preserving signatures that are comparable with the state of the art.

To understand where the problem lies, assume that a prover P wants to prove it has a valid signature σ w.r.t. a public verification key svk. Let $\mathscr{R}$ be the relation that defines a "valid" signature: given svk, a signature σ is valid if $(svk,\sigma) \in \mathscr{R}$. However, relaxed NIZK proofs only allows to prove knowledge of σ such that (svk,σ) is in a larger relation $\bar{\mathscr{R}} \supseteq \mathscr{R}$. The challenge is in how to modify the signature scheme so that a signature such that $(svk,\sigma) \in \bar{\mathscr{R}}$ can be still accepted

as valid, and that the expansion in the relation does not make it easier to forge signatures.

In Chapter 3, we provide a signature (Section 3.6) and a commitment scheme (Section 3.5) with efficient zero-knowledge proofs (Section 3.3) using Lyubashevsky's Fiat-Shamir with aborts technique. To be compatible with the "relaxed" extraction of such zero-knowledge proofs, we define "relaxed" signature and commitment schemes, in the sense that the verification algorithms accept messages, signatures, and openings that are never output by the honest signing or committing algorithms. By allowing exactly the relaxation induced by the extraction of zero-knowledge proofs, and by proving that our schemes remain secure under a suitably adapted notion in spite of that relaxation, we obtain efficient and securely composable zero-knowledge proofs for lattice-based primitives. We remark that such approach was already explored by Benhamouda et al. [2015]. In their work, they introduce a commitment scheme with relaxed requirement on valid openings, to allow a more efficient composition with the ZK proof, even if this is not explicit in the formal definition of the scheme.

In Chapter 4 we demonstrate the utility of our signature and commitment schemes in the construction of privacy-enhancing technologies by building two anonymous attribute token (AAT) schemes [Camenisch et al., 2012]. An AAT scheme enables users to obtain credentials with multiple attributes, so that they can selectively disclose these attributes to (one or more) verifiers in an unlinkable fashion. Some AAT also allow opening, i.e., the existence of an authority that can reveal which user produced a given token. We construct two schemes, one with and one without opening.

We suggest concrete parameter choices for our schemes that yield a secure yet efficient instantiation. We follow the approach of Alkim et al. [2016] and present different sets of parameters, ranging from conservative, quantum-safe choices to more liberal estimates that only guarantee classical security. Even in our most conservative analysis, assuming the hardness of RSIS and RLWE through a complexity leveraging argument, we obtain presentation token sizes less than 15 MB, which is well below the signature sizes of related lattice-based primitives [Libert, Ling, Nguyen and Wang, 2016]. In our least conservative analysis, assuming the hardness of two new interactive assumptions, we even obtain presentation tokens as small as 1.6 MB, which can be considered for practical use.

Finally, we explore the flexibility of our primitives by combining them with the relaxed verifiable encryption scheme by Lyubashevsky and Neven [2017] to construct AATs with opening (see Section 4.2); this type of AAT are important, as they can be easily modified to get a group signature. The resulting scheme cannot be considered efficient though. We only include it as a proof of concept.

This construction is improved in Section 5.3, where we give a construction for a dynamic signature that produces signatures of size < 1.8 MB. There are two main differences with the construction in Section 5.2. First, the way verifiable encryption is defined by Lyubashevsky and Neven results in the user having to encrypt more secret values than what is really necessary, resulting in an unnecessarily large ciphertext. To alleviate this, in Section 3.8.4 we revisit their definition and generalize it to a verifiable encryption scheme that encrypts only a function of the witness, as opposed to the full witness. This definition is of independent interest and may be applicable to many other scenarios.

The second difference has to do with the behavior of lattice-based proof protocols with non-binary challenges. Similarly to discrete logarithm-based proofs in groups of unknown order, lattice-based proof protocols introduce some slack: the witness that can be extracted from a proof is larger than the one that an honest prover uses and the difference in size, i.e., the slack, depends on the size of the challenges. Thus, on the one hand, one would want to choose a large challenge set in order to have a small soundness error and thus does not have to repeat the proof too many times. On the other hand, if the slack gets larger, one has to account for that by increasing parameters of the lattice. Unlike discrete logarithms-based proofs, however, the slack also increases if multiple statements are proved in the same proof. To deal with this, we investigate all NIZK proofs and tailor them by splitting statements and by selecting different sizes of challenges for different proofs. We believe that the insights into the proof mechanisms that we have gained can also be used to increase the efficiency for other schemes composed from lattice-based primitives.

We prove security of our AATs and of our group signature schemes in the Random Oracle Model. Analogously to the non-lattice-based world, where schemes under weak assumptions do exist [Bellare et al., 2003, 2005] but truly practical schemes typically require stronger assumptions [Ateniese et al., 2000; Boneh et al., 2005], we also prove our scheme secure under relatively strong assumptions. Namely, we use two interactive assumptions that can be interpreted in two different ways. Either one believes the interactive assumptions as stated, in which case we obtain a tight security reduction and the most efficient parameters for our scheme, or one sees our assumptions as being implied by the standard RSIS and RLWE assumptions through a complexity leveraging argument. In this case, the parameters need to be increased to compensate for the loose reduction.

The final step of our work was to adapt a pre-existing non-interactive zero-knowledge proof, Aurora [Ben-Sasson et al., 2019], to understand how efficient our approach was (Section 5.4.2). We obtained non-interactive zero knowledge (NIZK) proofs for Module/Ring LWE and Module/Ring SIS relations, that are

	Partially Dynamic	Anonymous	Traceable	Non-Frameable	Users	δ_{HRF}	Signature(MB)
del Pino et al. [2018]	✓	✓	✓		2^{80}	1.002	0.581
$\mathscr{G}$	✓	✓	✓		2^{26}	1.0007	0.3
$\mathscr{G}_{\text{full}}$	✓	✓	✓	✓	2^{26}	1.0007	1.3

Table 1.1. Comparison for around 90 bits of security.

72kB in size for 128 bits of security. From Aurora, our proofs inherit statistical zero-knowledge and soundness, post-quantum security, exact extractability (that is, the extraction guarantee is for the same relation as the protocol completeness), and transparent setup (no need for a trusted authority to generate the system parameters). Such proofs support algebraic circuits, and therefore can be combined with lattice based building blocks. We show that it is possible to combine this protocol with the ring version of Boyen's signature [Boyen, 2010] (cf. Section 5.4.1), to prove knowledge of a signature on a publicly known message, or knowledge of a valid pair message-signature, and an RLWE-based encryption scheme [Lyubashevsky et al., 2010], to prove knowledge of a valid decryption of a given ciphertext. To showcase their efficiency we construct a (partially) dynamic group signature (cf. Section 5.5), and we compare it with the most efficient NIZK-based group signature to date [del Pino et al., 2018] in Table 1.1. Differently from ours, the scheme by del Pino et al. does not protect honest users from framing attempts by corrupted issuers (the non-frameability property). Therefore, we compare it with two variants of our scheme: $\mathscr{G}$, that does not guarantee non-frameability, and $\mathscr{G}_{\text{full}}$, that also has non-frameability. To compare the security levels of the schemes we consider the Hermite Root Factors (denoted by δ_{HRF}); a smaller delta implies higher security guarantees. The $\mathscr{G}$ scheme outputs signatures of size less than 250 KB. For around the same number of bits of post quantum security, the group signature by del Pino et al. [2018] outputs signatures of size 581 KB. In Section 5.5 we will present $\mathscr{G}_{\text{full}}$. The $\mathscr{G}_{\text{full}}$ variant of our scheme outputs signatures of size less than 1.3 MB. Parameters will be discussed more in depth in Section 5.5.6. In both cases, the NIZK proof is of size less than 250 KB, improving upon the state of the art. The group signature is proven secure in the ROM under RSIS and RLWE. Security in the QROM follows also from Chiesa et al. [2019]; to achieve 128 bits of QROM security requires a three-fold increase in proof size.

Regarding implementation, our adaptation of Aurora produces a RLWE proof in around 40 seconds on a consumer laptop (cf. Section 5.4.3). In comparison, the scheme of del Pino et al. [2018] produces proofs in under a second. Nonetheless, we consider our NIZK and group signature a benchmark for evaluating efficiency claims for (existing and future) NIZK proofs for lattice relations. In particular, it shows what can be achieved using 'generic' tools.

1.2 Related and Concurrent Work

The only known lattice-based anonymous attribute token scheme [Camenisch et al., 2012] has presentation token sizes that are linear in the number of group members, and is therefore mainly a proof of concept. Our AAT scheme is the first that could be considered suitable for practical applications in a post-quantum world[2].

The early lattice-based group signature schemes [Gordon et al., 2010; Camenisch et al., 2012] have signature sizes that are linear in the number of group members and are therefore mainly proofs of concept, unsuitable for any practical application. Later schemes [Laguillaumie et al., 2013; Ling et al., 2015; Nguyen et al., 2015] are asymptotically more efficient with signature sizes being logarithmic in the number of users.

Making use of the advances in lattice-based signature schemes, a number of group signature schemes were proposed following the general construction approach we have outlined earlier [Laguillaumie et al., 2013; Libert, Ling, Nguyen and Wang, 2016; Libert, Mouhartem and Nguyen, 2016; Ling et al., 2015, 2017a; Xagawa and Tanaka, 2009]. These schemes use as proof of knowledge protocols either an adaptation of Stern's protocol [Stern, 1994] or the "single-bit-challenge" version of the lattice-based Fiat-Shamir protocol by Lyubashevsky [2012]. As these proofs have soundness error 2/3 and 1/2, respectively, they need to be repeated in parallel to be secure, resulting in group signature schemes that can hardly be considered practical. None of these scheme give concrete parameters, providing asymptotic efficiency analyses instead. The only exception by Libert, Ling, Nguyen and Wang [2016] is the most efficient scheme prior to ours, with signatures over 60 MB and public keys of 4.9 MB for a group size of only 2^{10} users for 80 bits of security.

In recent years, significant effort has been put towards the design of efficient lattice-based group signatures. In particular, fully dynamic group signatures [Bootle et al., 2016] relying on Stern-type proofs have been proposed [Ling et al., 2018, 2017b], and there has been a line of work on group signatures based on improved Schnorr proofs [del Pino et al., 2018]. Finally, a new, very interesting construction was published by Katsumata and Yamada [2019], that builds group signatures without using NIZK proofs in the standard model. Their construction is of a different form, and, in particular, sidesteps the problem of building NIZKs for lattices, hence we can only compare the signature lengths. Differently from ours, their signature sizes still depend linearly on the number of users (while

[2]We do not claim ours to be the first practical AAT. In fact, an AAT scheme based on discrete log is at the core of Microsoft's U-Prove [Paquin and Zaverucha, n.d.].

ours depend polylogarithmically on the number of users) when security is based on standard LWE/SIS. They are able to remove this dependency assuming subexponential hardness for SIS.

Regarding NIZK proofs for lattices, both Libert et al. [2018] and Baum et al. [2018] introduce ZK proof to prove knowledge of solutions of lattice problems that are linear in the length of the secret and in $\log \beta$ respectively (where β is the bound of the norm of the secret vector). Our scheme improves these in that the proof length depends *polylogarithmically* on the length of the secret vector and $\log \beta$. Moreover, we give concrete estimates for parameters that guarantee 128 bits of security. The lattice-based SNARK of [Gennaro et al., 2018] relies on the qDH assumption (among others), hence unlike our scheme this is not post-quantum secure, and needs a trusted setup, which prevents to use it to build group signatures with the non-frameability property.

1.3 Outline of this Dissertation

In this dissertation we merged the results presented in three publications, [Boschini et al., 2018b,a, 2020]. We start in Chapter 2 by introducing a bit of preliminaries about cryptography (Section 2.2) and lattices (Section 2.3). Then, in Chapter 3 we introduce our suite of relaxed primitives. In Chapter 4 we investigate how to construct Anonymous Attribute Tokens using the building blocks presented in the previous chapter. Finally, in Chapter 5 we present our three group signatures: one obtained (quite straightforwardly) from our AAT with opening (Section 5.2), one (Section 5.3) obtained combining our relaxed building blocks with a modified version of the verifiable encryption scheme by Lyubashevsky and Neven [2017], and the last one (Section 5.5) based on the adaptation of Aurora to lattice relations. We present our conclusions in Chapter 6.

Chapter 2

Preliminaries – Cryptography and Lattices

Cryptography was born as the branch of (first Mathematics, then) Computer Science focused on "the construction of schemes that will be robust against malicious attempts to make these schemes deviate from their prescribed functionality" [Goldreich, 2001, Section 1.4.1]. Originally, security mostly relied on obscurity: a cryptosystem was deemed to be secure as long as both the algorithm and the secret source of randomness (the secret key) were unknown to the adversary. This notion of security turned out to be too restrictive, and cryptographic research started aiming at minimizing the amount of information that needed to be private for the system to be secure. Hence, a new notion of security was introduced, called *provable security*, where attacking a scheme was proved to imply a solution to some computationally infeasible mathematical problem (the so-called hardness assumptions). This proved to be a powerful approach, allowing to split the theoretical definition and construction of protocols from the actual protocols instantiations (that depends on the choice of the hardness assumption underlying security).

This section aims to introduce the security model (Section 2.2) and hardness assumptions (Section 2.3) used in the main results in this thesis.

2.1 Notation

Let $\mathbb{R}$ denote the real numbers, $\mathbb{Z} = \{0, -1, +1, -2, +2, \ldots\}$ the integer numbers, and $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ the natural numbers. Let $\mathbb{N}^+ = \{n \in \mathbb{N} \ : \ n > 0\}$, and define $\mathbb{R}^+$ analogously. Let $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ for a prime q. Elements in $\mathbb{Z}_q$ are equivalence classes, and are denoted (with an abuse of notation) simply as $a \in$

$\mathbb{Z}_q$. The representative a of the equivalence class is chosen to be such that $a \in \left[-\frac{q-1}{2}, \frac{q-1}{2}\right] \subset \mathbb{Z}$. Equalities modulo q are denoted by either $a = b \bmod q$ or by the more compact $a \equiv_q b$.

Vectors and matrices are denoted by upper-case letters. Column vectors are denoted as $V = \left[v_1 \ ; \ \ldots \ ; \ v_n\right]$ and row vectors as $V = \left[v_1 \ \ldots \ v_n\right]$. Analogously, the horizontal concatenation of two matrices A, B is denoted by $[A\,B]$, and the vertical concatenation by $[A \ ; \ B]$. The Euclidean (l_2) norm of a vector $V = [v_1 \ \ldots \ v_n]$ is denoted by $\|V\|_2 = \sqrt{\sum_i v_i^2}$. Its infinity norm is $\|V\|_\infty = \max_i |v_i|$. Given a matrix $M = [M_1 \ \ldots \ M_k]$, the euclidean and infinity norms of the matrix are defined as $\|M\|_2 = \max_i \|M_i\|_2$ and $\|M\|_\infty = \max_i \|M_i\|_\infty$ respectively. The generic norm will be defined analogously using the symbol $\|\cdot\|$ with omitted subscript. Given two vectors V_1, V_2, their Euclidean distance is denoted by $d(V_1, V_2)$. The identity matrix of dimension m is $\mathbb{I}_m$. When S is a set, the number of its elements is denoted by $|S|$. Given a (bit) string x, $|x|$ denotes also the length of the string. The meaning of the notation will be usually clear from the context, but will be clarified when necessary.

Elements of a polynomial ring are denoted by bold letters. Consequently, vectors and matrices having components in a polynomial ring are denoted by bold, upper-case letters. With an abuse of notation, the identity matrix of dimension m with components in a polynomial ring is still denoted as $\mathbb{I}_m$.

Given two events A and B we denote by $\mathsf{Pr}[A \wedge B]$ the probability that both A and B happen, and by $\mathsf{Pr}[A : B]$ the probability that A happens given that B happened. To highlight the event A we sometimes write $\mathsf{Pr}_B[A]$ instead of $\mathsf{Pr}[A : B]$. Sampling and element x from a distribution $\mathscr{D}$ will be denoted as $x \xleftarrow{\$} \mathscr{D}$. If x is sampled from a uniform over a set A, we will abuse the notation and write $x \xleftarrow{\$} A$. With $x \leftarrow a$ we will denote that x is assigned the value a. When necessary, the uniform distribution over a set S is denoted by $\mathscr{U}(S)$. Given two probability distribution $\mathscr{A}$ and $\mathscr{B}$ we denote by $\mathscr{A} \approx_s \mathscr{B}$ (resp., $\mathscr{A} \approx_c \mathscr{B}$) the fact that they are statistically (resp., computationally) indistinguishable. We denote by log the logarithm with base 2.

Finally, when necessary we will use the Bachmann–Landau notation (or Big O notation) to estimate parameters. Moreover, to assess that a quantity is small and can be neglected, we will prove that it is smaller than a "negligible function" $\nu(\cdot)$, defined as follows.

Definition 2.1 (Negligible Function)**.** A function $\nu : \mathbb{N} \to [0, 1]$ is *negligible* if for all constants $c \in \mathbb{N}$ there exists $n_0 \in \mathbb{N}$ such that for all $n > n_0$ $\nu(n) \leq \frac{1}{n^c}$.

2.2 Cryptography

Establishing whether a protocol is secure entails two main steps: first, defining what "security" means, and then give a proof that such security property is actually achieved. Throughout this work a cryptographic protocol is defined to be secure if an adversary A is only able to break it in an unfeasible amount of time, and with negligible success probability, as it is customary in cryptography (cf. [Katz and Lindell, 2014, Section 3.1] for a more extensive dissertation). Hence, given a protocol P and some security property of P, a secure protocol is defined as follows.

Definition 2.2. A cryptographic scheme P is said to be (t, ϵ)-*secure* if every adversary that takes time at most $t = t(\lambda)$ to break the security property succeeds with probability at most $\epsilon = \epsilon(\lambda)$, where $\lambda \in \mathbb{N}$ is the *security parameter*.

The parameter λ can be set during the initialization of the protocol and it allows to fine-tune the security level of P depending on the scenario in which the protocol has to be used.

Definition 2.2 is still quite vague, in that it is not clear what an "adversary" actually is. A generic way to model an adversary (and the other parties involved in the protocol) is as a probabilistic algorithm[1]. In this framework, an efficient adversary against P is a probabilistic algorithm that runs in polynomial time (shortened in the following as PPT), i.e. such that $t(\lambda)$ is a polynomial in the variable λ. A negligible success probability ν is intended as being smaller than any inverse polynomial in λ: for every constant c, $\nu(\lambda) \leq \lambda^{-c}$ for some large enough λ.

Definition 2.3. An *probabilistic* algorithm A is an algorithm whose execution uses some randomness that is given as input. If A is a probabilistic algorithm, then by $\mathsf{A}(x;\rho)$ we denote the output distribution of A on input x and run with random coins ρ (we write $\mathsf{A}(x)$ when the random coins are uniformly random). Computing y through A on input x amounts to choose y from the distribution $\mathsf{A}(x)$, and it is denoted by $y \xleftarrow{\$} \mathsf{A}(x)$.

A probabilistic algorithm A runs in polynomial time if there exists a polynomial $p(\cdot)$ such that for all inputs $x \in \{0,1\}^*$, the computation of $\mathsf{A}(x)$ terminates in at most $p(|x|)$ steps (where $|\cdot|$ denotes the length of the input string x).

The main reason behind modeling the adversary as a PPT algorithm is related to *security proofs*. Indeed, this model allows to make statements about the secu-

[1] In this way we also avoid the awkwardness of assigning a gender to the entities involved in the protocols: as they are all algorithms, they will be always addressed using the *it* pronoun.

rity of a protocol independently of the computational model used. Security of a protocol P can be proved in two different ways.

- One approach consists in defining an ideal functionality, i.e., an ideal version of the protocol that satisfies the required security properties; a successful adversary against P is able to distinguish the ideal functionality from the real-world protocol P.
- The other approach is to define security through a security experiment: a successful adversary against the scheme is an adversary able to win the experiment (for example, an adversary able to distinguish between two ciphertexts generated by a public key encryption scheme).

The scheme is secure if, given a successful adversary A (either in distinguishing P from the ideal functionality, or in winning the security experiment), it is possible to construct a PPT algorithm B that solves a mathematical problem assumed to be hard (the hardness assumption) typically exploiting A as a black-box. The algorithm B interacts with A by simulating the honest parties in the protocol (corrupted parties are assumed to be in complete control of A). Hence, the view of A when interacting with P should be indistinguishable from the view of A when interacting with B. The fact that A has black-box access to an algorithm or function F is indicated by the notation A^{F}.

As PPT algorithms are closed under composition, the algorithm B that runs A as a sub-routine runs in polynomial-time too. Moreover, let $1/p(\lambda)$ be the probability that B solves the problem if A is successful (p is a polynomial). If $\epsilon(\lambda)$ is the success probability of A, then the success probability of B is $\epsilon(\lambda)/p(\lambda)$, and it is non-negligible if ϵ is. Therefore, B solves the mathematical problem in polynomial-time with non-negligible provability, thus breaking the hardness assumption. This implies that ϵ has to be negligible in the security parameter.

The security guarantee obtained in this way is called *computational security*, as it relies on assumptions about the computational intractability of mathematical problems. The algorithm B is usually called the *simulator* (as it simulates a honest execution of the protocol when interacting with A), and the entire process of solving the problem exploiting A is called a *(security) reduction*.

2.2.1 The Random Oracle Model and Cryptographic Hash Functions

In addition to the model laid out previously, the security proofs contained in this dissertation live in a framework called Random Oracle Model (ROM). The ROM

(introduced by Bellare and Rogaway [1993]) assumes the existence of a public, randomly-chosen function $H : \{0,1\}^* \to \{0,1\}^\ell$. To evaluate the function on a bit string x (called *query*) it is necessary to give x as input to an oracle (i.e., to *query* the oracle on x) that then returns $H(x)$. The oracle is assumed to be *coherent*, meaning that the oracle outputs the same bit string if queried twice on the same input.

Through this methodology it is possible to show that a protocol has no inherent flaws. Thus, the choice of the RO becomes fundamental, as (possible) weaknesses can only be introduced through the implementation of the RO (cf. [Canetti et al., 1998]). Given that ROs are not available in the real-world, they are usually instantiated with cryptographic hash functions, where the "unpredictability" property of the RO is captured by the requirement that the hahs function has to be "hard to invert". Such functions are compression functions widely used in cryptography, usually defined as a family of hashes indexed by a key s (that does not have to be kept secret).

Definition 2.4 (Hash function)**.** A family of *hash function* is a pair of PPT algorithms (HGen, H) such that:

- $\mathsf{HGen}(1^\lambda)$ takes as input the security parameter λ and outputs a key s (s is assumed to contain λ too).
- $H(\mu; s)$ takes as input a key s and a message $\mu \in \{0,1\}^*$ and outputs a string of fixed length $\ell(s)$ for some polynomial $\ell(\cdot)$.

A hash function is a *cryptographic* hash function if it is *collision-resistant*, i.e., if for all PPT algorithms A and integers $\lambda \in \mathbb{N}$ the following probability is negligible

$$\Pr[H(\mu; s) = H(\mu'; s) \ : \ s \leftarrow \mathsf{HGen}(1^\lambda),\ (\mu, \mu') \leftarrow \mathsf{A}(s)] \ .$$

Remark that a hash function has collisions by design, as by definition the size of the domain is larger than the size of the codomain. Hence, it is particularly important to choose the dimension ℓ of the codomain so that finding collision is infeasible. A good rule of thumb is to choose $\ell = O(2^{\lambda/2})$ to prevent birthday attacks (cf. [Katz and Lindell, 2014] for a more detailed explanation).

Finally, we recall the following lemma by Lyubashevsky [2016]. It states that if the input set of a deterministic function is larger than the set of its output, there exists a collision with non-negligible probability.

Lemma 2.5 (Lemma 2.11 in [Lyubashevsky, 2016])**.** *Let $h : X \to Y$ be a deterministic function where X and Y are finite sets and $|X| \geq 2^\lambda |Y|$. If x is chosen uniformly at random from X, with probability at least $1 - 2^{-\lambda}$ there exists another $x' \in X$ such that $h(x) = h(x')$.*

2.2.2 Remarks on the Security Model: ROM vs. QROM

The ROM is slightly less rigorous than the standard model. In fact, it is possible to build schemes that are provably secure in the ROM but for which any implementation of the random oracle results in insecure schemes (cf. [Canetti et al., 1998]). Hence, a security proof in this model it is considered more as a good indicator that the protocol will be secure if implemented carefully. The bright side of proving security in the ROM is that this additional assumption on the existence of a RO makes it very powerful[2]. In fact, for many schemes, including the Fiat-Shamir heuristic ([Fiat and Shamir, 1987]) that is widely used in the construction of privacy-preserving signatures, only security proofs that assume the existence of a random oracle are known (cf. [Bellare and Rogaway, 1993; Pointcheval and Stern, 1996; Faust et al., 2012]).

The validity of the model is again disputed in the post-quantum scenario, i.e., the adversarial scenario where the A has access to a quantum computer. The problem lies in how to model the adversary's access to the oracle. A simple model is to assume that the adversary has a quantum computer, but only classical access to the random oracle. This case is included in the ROM. Indeed, the only difference with the classical ROM is in the computational power of A. Hence, as long as the underlying hardness assumptions are hard to solve with a quantum computer, a security proof for a protocol P in the classical ROM implies that P is also secure in the new model.

A scenario that is considered more general is the Quantum Random Oracle Model (QROM, cf. [Boneh et al., 2011]), where the adversary is given *quantum access* to the oracle, meaning that there is a quantum communication channel between A and the RO. The rationale behind it is that random oracles are usually implemented with hash functions, that are publicly known. Hence, a quantum adversary can run the code of the hash function locally on a quantum superposition of inputs. It is reasonable to think that schemes that are proven secure in the ROM might not be secure in the QROM. In fact, there are some results that identify the requirement for a security proof in the ROM to be valid also in the QROM. For example, Boneh et al. [2011] showed that, a security reduction in the ROM also holds in the QROM if it is history-free. Unfortunately, many security reductions in the ROM require to reprogram a random oracle; these reductions are not history-free and it is not clear whether they translate to the QROM sce-

[2]Bellare and Rogway argued that "*This paradigm yields protocols much more efficient than standard ones while retaining many of the advantages of provable security [...] for problems including encryption, signatures, and zero-knowledge proofs.* " (cf. the abstract of [Bellare and Rogaway, 1993]).

nario. When confronted with the choice of the security model, we decided to use the Random Oracle Model. This is both for the sake of simplicity, and because at the time the papers were written it was not clear whether some of the schemes and transformations used in our work, in particular the Fiat-Shamir heuristic and the reprogramming technique used in the security proofs, were secure in the QROM (for more recent results about the security of the FS heuristic in the QROM cf. [Unruh, 2017; Don et al., 2019]).

2.2.3 Rewinding

Sometimes, in the reduction the simulator algorithm B has to rewind the adversary A to extract information comparing different executions. This extraction needs the behavior of the adversary in the two (or more) executions to be the same up until a fixed step (or steps), where the behavior path of the different execution of A fork. The probability that this event happens depends on both the random oracle(s) A has access to and on its internal randomness, and it is analyzed in a series of lemmas called Forking Lemmas, the first of which was introduced by Pointcheval and Stern [1996]. A more general version was presented by Bagherzandi et al. [2008], and can be found in the following.

Let IG be an input generator for A, let $f = (\rho, h_1, \ldots, h_{q_H})$ be A's random coins and random oracle responses, and let $f_j = (\rho, h_1, \ldots, h_j)$. An execution of A is successful if it returns a non-empty set of indices $J \subseteq \{1, \ldots, q_H\}$ and corresponding outputs $\{sig_j\}_{j \in J}$. Let Ω be the set of all f and let Ω_{in} be the set of f for which A is successful on input in; its success probability is $\varepsilon = \Pr[f \in \Omega_{in} : in \xleftarrow{\$} \mathsf{IG}, f \xleftarrow{\$} \Omega]$. Consider the following generalized forking algorithm:

Algorithm $\mathsf{GF}_{\mathsf{A}}(in)$:
$f \xleftarrow{\$} \Omega$
$(J, \{sig_j\}_{j \in J}) \leftarrow \mathsf{A}(in; f)$
If $J = \emptyset$ then halt
Let $J = \{j_1, \ldots, j_n\}$ such that $j_1 \leq \ldots \leq j_n$, $X \leftarrow \{(h_j, sig_j)\}_{j \in J}$, $X' \leftarrow \emptyset$
For $i = 1, \ldots, n$ do
  $j = j_i$, $succ_i \leftarrow 0$, $k_i \leftarrow 0$, $k_{\max} \leftarrow 8nq_H/\varepsilon \cdot \ln(8n/\varepsilon)$
  Repeat until $succ_i = 1$ or $k_i > k_{\max}$
    $k_i \leftarrow k_i + 1$
    $f' = (\rho', h'_1, \ldots, h'_{q_H}) \xleftarrow{\$} \{f' \in \Omega : f'_j = f_j\}$
    $(J', \{sig'_j\}_{j \in J'}) \leftarrow \mathsf{A}(in; f')$
    If $J' \neq \emptyset \wedge j_i \in J' \wedge h'_{j_i} \neq h_{j_i}$ then $X' \leftarrow X' \cup \{(h'_{j_i}, sig'_{j_i})\}$, $succ_i \leftarrow 1$
If $succ_1 = \ldots = succ_n = 1$ then return (X, X') else return $\perp$

Experiment $\mathsf{Exp}^{\mathsf{sufcma}}_{\mathsf{A}}(\lambda)$
$(otssk, otsvk) \leftarrow \mathsf{OTSGen}(1^\lambda)$
$msg^* \leftarrow \mathsf{A}_1(\lambda, otsvk)$
$ots^* \leftarrow \mathcal{O}_{OTS}(msg^*)$
$(msg', ots') \leftarrow \mathsf{A}_2(otsvk, (msg^*, ots^*))$
If $1 \leftarrow \mathsf{OTSVf}(otsvk, msg', ots')$
and $(msg', ots') \neq (msg^*, ots^*)$,
then return 1 else return 0.

Oracle $\mathcal{O}_{OTS}(msg)$
$ots \leftarrow \mathsf{OTSSign}(otssk, msg)$
Return ots .

Figure 2.1. Strong unforgeability experiment for OTS.

Lemma 2.6 (Generalized forking lemma [Bagherzandi et al., 2008])**.** *If algorithm* A *runs in time* t *and has success probability* ε*, then the forking algorithm* GF_{A} *runs in time* $t \cdot 8n^2 q_H/\varepsilon \cdot \ln(8n/\varepsilon)$ *and returns* (X, X') *with probability* $\hat{\varepsilon} \geq \varepsilon/8$.

2.2.4 One-Time Signature Scheme

A *One-Time Signature (OTS) scheme* for message set $\mathcal{M}$ is composed by a key generation algorithm $(otssk, otsvk) \leftarrow \mathsf{OTSGen}(1^\lambda)$, a signing algorithm $ots \leftarrow \mathsf{OTSSign}(otssk, msg)$ and a verification algorithm $b \leftarrow \mathsf{OTSVf}(otsvk, msg, ots)$, $b \in \{0,1\}$.

Correctness requires that for all security parameters $\lambda \in \mathbb{N}$ it holds that:

$$\mathsf{Pr}\left[1 \leftarrow \mathsf{OTSVf}(otsvk, msg, ots) \;:\; \begin{array}{c}(otssk, otsvk) \leftarrow \mathsf{OTSGen}(1^\lambda),\\ ots \leftarrow \mathsf{OTSSign}(otssk, msg)\end{array}\right] = 1 \ .$$

A OTS scheme is said to be strongly unforgeable under chosen-message attacks if a PPT adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ has negligible probability in winning the experiment $\mathsf{Exp}^{\mathsf{sufcma}}_{\mathsf{A}}(\lambda)$ in Figure 2.1.

In particular, a OTS that is still secure even against a quantum computer and that is used in our protocols is the Lamport signature [Lamport, 1979] instantiated with a post-quantum, collision-resistant hash function.

2.3 Lattices

After the publication of Shor's algorithm [Shor, 1994], lattice-based hardness assumptions emerged as a viable alternative to classical hardness assumptions

such as DLOG, in particular thanks to the works of Ajtai [1996]. Indeed, Ajtai showed a *worst-case to average-case reduction* for lattice problems, i.e., that if certain lattice problems are hard to solve in the worst-case, then they are also hard on average. He also gave a family of one-way functions from lattice-based hardness assumptions, thus proving that it is possible to build provably secure cryptographic schemes from these hardness assumptions. This seminal result combined with the (supposed) quantum-hardness of lattice problems gave a strong reason to design cryptography relying on lattice-based hardness assumptions. Lattices were first introduced as $\mathbb{Z}$-modules over $\mathbb{R}$, but in fact lattices can actually be built on general rings, and in particular from polynomial rings (*ideal lattices*, introduced by Micciancio [2002] as a more efficient way to build cryptographic schemes over lattices). In this section we recall definitions and results about lattices, the hard problems that can be defined over them and the cryptographic primitives necessary to construct the protocols in the following chapters.

2.3.1 Euclidean Lattices

Let $\mathbb{R}^n$, $n \in \mathbb{N}^+$ be the vector space over the reals w.r.t. standard componentwise sum $+ : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ and product by a scalar $c[v_1 \dots v_n] = [v_1 \dots v_n]c = [cv_1 \dots cv_n]$, where $c, v_1, \dots, v_n \in \mathbb{R}$. A lattice in geometry and group theory is a particular subgroup of $\mathbb{R}^n$ that is closed under multiplication by an integer.

Definition 2.7. A *(Euclidean) lattice* $\mathscr{L}$ in $\mathbb{R}^n$ is a $\mathbb{Z}$-module of $\mathbb{R}^n$ with commutative scalar multiplication, i.e.

- $(\mathscr{L}, +)$ is an Abelian group.
- $\mathscr{L}$ is closed under scalar multiplication, when the scalars are integer numbers,
$$\forall c \in \mathbb{Z},\ \forall V \in \mathscr{L}, \quad cV \in \mathscr{L}\ .$$
- the scalar multiplication satisfies the commutative and distributive property,
$$\forall c, d \in \mathbb{Z},\ \forall V, W \in \mathscr{L}, \quad \begin{array}{l} cV = Vc \\ c(V+W) = (V+W)c = cV + cW \\ cd(V) = c(dV) \end{array}\ .$$

The number n is called *dimension* of the lattice. If $\mathscr{L} \subseteq \mathbb{Z}^n$, the lattice is said to be an *integer* lattice.

There exists also an equivalent (cf. [Micciancio and Goldwasser, 2002]) constructive definition of lattice.

Definition 2.8. Let $B = [B_1 \ \dots \ B_k] \in \mathbb{Z}^{n\times m}$ be a matrix whose columns B_i are linearly independent vectors in $\mathbb{R}^n$. The lattice $\mathscr{L}(B)$ generated by B is the set of all the integer linear combinations of the vectors B_i:

$$\mathscr{L}(B) = \{X \in \mathbb{R}^n \ : \ X = BA \text{ where } A \in \mathbb{Z}^m\} \ .$$

B is called *basis* of the lattice, and the number m of linearly independent vectors in B is the *rank* of the lattice. If $m = n$ and the B_i's are linearly independent the lattice is a *full-rank* lattice.

The Gram-Schmidt (GS) orthogonalization of a full-rank basis B is $\tilde{B} = [\tilde{B}_1 \ \dots \ \tilde{B}_n]$ where

$$\tilde{\mathrm{B}}_i = B_i - \sum_{j=1}^{i-1} \frac{\langle B_i, \tilde{B}_j \rangle}{\|\tilde{B}_j\|_2} \tilde{B}_j$$

for the usual definition of Euclidean norm and scalar product. Remark that the GS basis $\tilde{B}$ of the basis B of the lattice $\mathscr{L}(B)$ is not automatically a basis of $\mathscr{L}(B)$. In general, a lattice can have multiple bases. In fact, a new basis of a lattice $\mathscr{L}(B)$ can be obtained multiplying B by a unimodular matrix $U \in \mathbb{Z}^{m\times m}$, i.e., such that $\det(U) = \pm 1$.

Theorem 2.9. *Let $\mathscr{L} = \mathscr{L}(B)$ be a lattice generated by a basis $B \in \mathbb{R}^{n\times m}$. A matrix $B' \in \mathbb{R}^{n\times m}$ is a basis for $\mathscr{L}$ if there exists a unimodular matrix $U \in \mathbb{Z}^{m\times m}$ such that $B = B'U$.*

The minimum norm of (the GS orthogonalization of) a basis of a lattice $\mathscr{L}$ is defined as $\tilde{\lambda}(\mathscr{L}) = \min_{B \text{ s.t. } \mathscr{L}(B)=\mathscr{L}} \|\tilde{B}\|_2$.

Given a vector $V \in \mathbb{Z}^n$, a *coset* $\mathscr{L} + V$ of a lattice $\mathscr{L}$ is the set $\{A + V\}_{A\in\mathscr{L}}$.

Let $A \in \mathbb{Z}^{n\times m}$, for $n \le m \le q$ where q is usually (but not necessarily) a prime. There are two particular lattices that are important in cryptography. For a (almost) uniformly random matrix $A \in \mathbb{Z}_q^{n\times m}$, $\mathscr{L}_q^{\perp}(A)$ and $\mathscr{L}_q(A)$ are the hard random lattices:

$$\begin{aligned} \mathscr{L}_q^{\perp}(A) &= \{X \in \mathbb{Z}^m \,|\, AX = 0_n \bmod q\} \subseteq \mathbb{Z}^m \\ \mathscr{L}_q(A) &= \{X \in \mathbb{Z}^n \,|\, X = A^T S \bmod q \text{ for some } S \in \mathbb{Z}_q^m\} \subseteq \mathbb{Z}^n \ , \end{aligned}$$

where 0_n denotes the column vector with all components equal to zero The importance of these lattices is directly linked to the hardness assumptions SIS and LWE (cf. Section 2.3.4).

2.3.2 Ideal Lattices

Let $\mathbb{Z}[\mathbf{x}]$ be the ring of polynomials with integer coefficients and $\mathbf{f} \in \mathbb{Z}[\mathbf{x}]$ be a monic, irreducible polynomial of degree n. Consider the quotient ring $R := \mathbb{Z}[\mathbf{x}]/\langle \mathbf{f} \rangle$. Elements in this ring can be represented with the standard set of representatives $\{\mathbf{g} \bmod \mathbf{f} : \mathbf{g} \in \mathbb{Z}[\mathbf{x}]\}$[3]. For an element $\mathbf{a} = \sum_{i=0}^{n-1} a_i \mathbf{x}^i$ in R, the standard norms are computed as $\|\mathbf{a}\|_1 = \sum_i |a_i|$, $\|\mathbf{a}\|_2 = \sqrt{\sum_i a_i^2}$ and $\|\mathbf{a}\|_\infty = \max |a_i|$. For a vector $\mathbf{S} = (\mathbf{s}_1, \ldots, \mathbf{s}_m) \in R^m$, the norm $\|\mathbf{S}\|_p$ is defined as the norm of the concatenation of the vectors of the coefficients of its components:

$$\|\mathbf{S}\|_p = \|[s_{1,0} \ \ldots \ s_{1,n-1} \ s_{2,0} \ \ldots \ s_{2,n-1} \ \ldots \ s_{m,0} \ \ldots \ s_{m,n-1}]\|_p \ ,$$
$$\text{where } p \in \{1, 2, \infty\} \text{ and } \mathbf{s}_i = \sum_{j=0}^{n-1} s_{i,j} \mathbf{x}^j \ .$$

A *small* element of the ring refers to a polynomial with small coefficient w.r.t. one of these norms depending on the context.

The vector of length m whose components are equal to $\mathbf{1}$ is denoted by $\mathbf{1}_m$, while $\mathbf{0}_{n \times m}$ denotes the $n \times m$ matrix with all components equal to the zero polynomial. The subset of polynomials with coefficients in $\{0, 1\} \subseteq R_q$ is denoted by $\mathcal{S}_1$.

Polynomials can be put in correspondence with vectors in $\mathbb{Z}^n$ with the standard *group homomorphism* $\mathrm{h} : R \to \mathbb{Z}^n$ that sends the polynomial $\mathbf{a} = \sum_{i=0}^{n-1} a_i x^i$ to the vector of the coefficients, $\mathrm{h}(\mathbf{a}) = [a_0 \ \ldots \ a_{n-1}]$. Remark that h is a group homomorphism w.r.t. componentwise addition, but it is not a ring homomorphism because it does not send the product to componentwise product, i.e., $\mathrm{h}(\mathbf{fg}) \neq \mathrm{h}(\mathbf{f})\mathrm{h}(\mathbf{g})$. Combining the standard set of representatives with the group homomorphism we obtain that R is isomorphic as an additive group to the integer lattice $\mathbb{Z}^n$. Not all integer lattices can be represented in this way.

Definition 2.10. An *ideal lattice* is an integer lattice $\mathscr{L}(B) \subset \mathbb{Z}^n$ such that $\mathscr{L}(\mathbf{B}) = \{\mathrm{h}(\mathbf{g}) \bmod \mathbf{f} \ : \ \mathbf{g} \in I\}$ for some monic polynomial $\mathbf{f}$ of degree n and some ideal $I \subseteq \mathbb{Z}[\mathbf{x}]/\langle \mathbf{f} \rangle$.

With an abuse of notation, both the integer lattice $\mathscr{L}(B)$ and polynomial ideal from which it is generated will be identified by $\mathscr{L}(B)$, and elements of the ideal I will be called "lattice elements".

[3] This is because $\mathbf{f}$ is monic: by the polynomial division algorithm, if the leading coefficient of $\mathbf{f}$ is invertible in the ring of coefficients, then there exists and are uniquely determined $\mathbf{q}$ and $\mathbf{r}$ such that for all $\mathbf{g} \in R$, $\mathbf{g} = \mathbf{fq} + \mathbf{r}$, $\deg(\mathbf{r}) < \deg(\mathbf{f})$, cf. Thm 1.1 on page 173 in [Lang, 2002].

For example, an element of an ideal lattice generated by $\mathbf{f}_1, \mathbf{f}_2$ is a polynomial $\mathbf{g} \in R$ such that $\mathbf{g} = \mathbf{f}_1 \cdot \mathbf{a}_1 + \mathbf{f}_2 \cdot \mathbf{a}_2$ for some $\mathbf{a}_1, \mathbf{a}_2 \in R$. If $\mathbf{f}$ is also irreducible, every ideal of R is isomorphic to a full-rank lattice of $\mathbb{Z}^n$, i.e. a lattice generated by n linearly independent vectors in $\mathbb{Z}^n$ (cf Lemma 3.2 in [Lyubashevsky and Micciancio, 2006]).

Remark 1. Another possible way to represent polynomials is through the NTT transformation [Cooley and Tukey, 1965; Lyubashevsky et al., 2010, 2013], that sends each polynomial to the vector of its evaluation over the n-th roots of unity. This transformation is a ring homomorphism w.r.t. componentwise addition and multiplication. We do not use this representation in our schemes, as we opted for the more intuitive polynomial-to-vector transform. However, this transform makes a difference when considering security reductions of lattice-based hardness assumptions (cf. Remark 5 in Section 2.3.4).

Consider the polynomial ring $R_q = \mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{f} \rangle$ for a prime q and irreducible $\mathbf{f}$ of degree n. Elements in the ring are polynomials of degree at most $n-1$ with coefficients in $[0, q-1]$ and operations between ring elements are done modulo q. Such ring contains exactly q^n elements, and it trivially holds that

$$\max_{\mathbf{a} \in R_q} \|\mathbf{a}\|_\infty = \frac{q-1}{2} \quad \text{and} \quad \max_{\mathbf{a} \in R_q} \|\mathbf{a}\|_2 = \frac{q-1}{2}\sqrt{n}$$

Let $\deg(\mathbf{a})$ be the degree of the polynomial $\mathbf{a}$.

Let $\mathbf{f} = \mathbf{x}^n + 1$. The group homomorphism h defined over R can be defined analogously over R_q, and it has a particular property when applied to a product of elements $\mathbf{a} \cdot \mathbf{b}$. Indeed, the product $\mathbf{ab}$ of the two polynomials $\mathbf{a} = \sum_{i=0}^{n-1} a_i \mathbf{x}^i$, $\mathbf{b} = \sum_{i=0}^{n-1} b_i \mathbf{x}^i$ corresponds in this representation to a matrix-polynomial product $A \cdot B$, where $B \in \mathbb{Z}_q^n$ is a column vector$[b_0; b_1; \ldots; b_{n-1}]$ and $A \in \mathbb{Z}_q^{n \times n}$ is a matrix whose columns are the column vectors corresponding to the polynomials $\mathbf{a}$, $\mathbf{xa}$, $\ldots$ $\mathbf{x}^{n-1}\mathbf{a}$:

$$A \cdot B = \begin{bmatrix} a_0 & -a_{n-1} & \ldots & -a_1 \\ a_1 & a_0 & \ldots & -a_2 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & \ldots & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \end{bmatrix} \bmod q \ .$$

Lattices over R_q can be defined similarly to lattices over R. In particular, given $\mathbf{A} \in R_q^{1 \times m}$ we can construct the hard cryptographic lattice $\mathscr{L}_q^\perp(\mathbf{A}) = \{\mathbf{V} \in (\mathbb{Z}[\mathbf{x}]/\langle \mathbf{f} \rangle)^m \mid \mathbf{AV} = \mathbf{0} \bmod q\} \subseteq R^m$. Consider the previously mentioned embedding h that maps a polynomial to the vector of its coefficients. Then $\mathscr{L}_q^\perp(\mathbf{A})$ can

also be seen as a nm-dimensional integer lattice over $\mathbb{Z}$. The lattice $\mathscr{L}_q(\mathbf{A})$ can be defined in the ring setting analogously.

With R_3 we denote the ring of polynomials with coefficients in $\mathbb{Z}_3 = \{0, \pm 1\}$. These element can put in correspondence with elements of the subset of R_q of polynomials with coefficients in $\{\pm 1, 0\}$ using the standard mapping h. We denote by $Inv(R_q)$ the set of all the invertible polynomials in R_q.

The rest of this section is dedicated to the analysis of the cyclotomic ring $R_q = \mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$ where $n = 2^d$, $q \equiv 5 \bmod 8$. The choice of this ring is motivate by both the result in Lemma 2.11, that guarantees that there are enough invertible elements in the ring, and by the hardness result by Lyubashevsky and Micciancio [2006].

The ring R_q has some very useful properties. First, for any $K|n$, it is possible to construct a particular subring of R_q as the subset of elements $\mathbf{a} \in R_q$ such that $\mathbf{a} = a_0 + a_1\mathbf{x}^{n/K} + a_2\mathbf{x}^{2n/K} + \ldots + a_{K-1}\mathbf{x}^{(K-1)n/K}$ and $a_i \in [0, p-1]$. Such set is denoted by $R_q^{(K)}$. Observe that $R_q^{(K)}$ is isomorphic to $\mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^K + 1\rangle$.

The choice of q strongly influences the number of invertible elements that can be found in the ring. For example, if q is such that $q \equiv 5 \bmod 8$, all the elements with small enough coefficient are guaranteed to be invertible. Depending on the choice of q, the ring R_q can be constricted so that it has a large set of invertible elements.

Lemma 2.11 ([Lyubashevsky and Neven, 2017, Lemma 2.2])**.** *Let $R_q = \mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$ where $n > 1$ is a power of 2 and q is a prime congruent to 5 mod 8. This ring has exactly $2q^{n/2} - 1$ elements without an inverse. Moreover, every non-zero polynomial $\mathbf{a}$ in R_q with $\|\mathbf{a}\|_\infty < \sqrt{q/2}$ has an inverse.*

There exist some straightforward bounds on the norm of the product of polynomials in this ring. They are summarized in the following lemma.

Lemma 2.12. *For $\mathbf{a}, \mathbf{b} \in R_q$ it holds:*

$$\|\mathbf{ab}\|_\infty \leq \min\{\|\mathbf{a}\|_\infty \|\mathbf{b}\|_1,\ \|\mathbf{a}\|_1 \|\mathbf{b}\|_\infty,\ (q-1)/2\}\ .$$

Moreover, let $\mathbf{a},\ \mathbf{b} \in R_q$ be such that $n\|\mathbf{a}\|_\infty \cdot \|\mathbf{b}\|_\infty \leq (q-1)/2$. Then we have that $\|\mathbf{ab}\|_2 \leq \|\mathbf{a}\|_2 \|\mathbf{b}\|_2 \sqrt{n}$ and $\|\mathbf{ab}\|_\infty \leq \|\mathbf{a}\|_\infty \|\mathbf{b}\|_\infty n \leq \frac{q-1}{2}$.

Proof. Let $\mathbf{a} = \sum_{i=0}^{n-1} a_i \mathbf{x}^i$ and $\mathbf{b} = \sum_{i=0}^{n-1} b_i \mathbf{x}^i$. The product $\mathbf{ab}$ can be represented as the matrix-polynomial product:

$$A \cdot B = \begin{bmatrix} a_0 & -a_{n-1} & \ldots & -a_1 \\ a_1 & a_0 & \ldots & -a_2 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & \ldots & a_0 \end{bmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \end{pmatrix} \bmod q\ .$$

Let A_j be the j-th row of A. The coefficient of $\mathbf{x}^j$ in the product $\mathbf{ab}$ is the scalar product $\langle A_j, B\rangle$. Moreover, observe that $\|\mathbf{a}\|_2 = \|A_1\|_2 = \|A_j\|_2$ for $j = 2,\ldots,n$, as the Euclidean norm is sign-invariant (the same holds for the infinity norm). From these observations and from the Cauchy-Schwarz inequality it follows that:

$$\|\mathbf{ab}\|_2 = \sqrt{\sum_{j=1}^{n} \langle A_j, B\rangle^2} \leq \sqrt{\sum_{j=1}^{n} \|A_j\|_2^2 \|B\|_2^2} = \|\mathbf{a}\|_2 \|\mathbf{b}\|_2 \sqrt{n} \ ,$$

where the Cauchy-Schwarz inequality holds as the hypothesis on the infinity norms of $\mathbf{a}$ and $\mathbf{b}$ guarantees that there is no rounding $\bmod q$ in the computation of the coefficients of $\mathbf{ab}$. The other two bounds follow from the observation that $\|AB\|_\infty = \max_j \langle A_j, B\rangle \leq \|\mathbf{a}\|_\infty \|\mathbf{b}\|_\infty n$, and

$$\|\mathbf{ab}\|_\infty \leq \max_j \langle A_j, B\rangle \leq \max_j \|\mathbf{a}\|_\infty \sum_{i=0}^{n-1} b_i \leq \min\{\|\mathbf{a}\|_\infty \|\mathbf{b}\|_1, (q-1)/2\}$$

as in the second to last equation the argument of the max function does not depend on j anymore (the bound $\|\mathbf{a}\|_1 \|\mathbf{b}\|_\infty$ can be obtained analogously). □

2.3.3 Probability Distributions over Lattices

Defining probability distributions over lattices requires some care, as lattices are discrete sets. When building cryptography relying on lattice problems, the probability distributions needed are mostly the discrete Gaussian and uniform distribution.

The uniform distribution can be trivially defined over a discrete set.

Definition 2.13. Let S be a discrete set. The *uniform probability distribution* $\mathcal{U}(S)$ over S is characterized by the density function $f : S \to [0,1]$ such that $f(x) = \frac{1}{|S|}$ for all $x \in S$.

When considering ideal lattices, sampling an element uniformly at random in $S \subseteq R_q$ is equivalent to sampling an element in $\mathrm{h}(S)$, i.e., in the subset $\mathrm{h}(S)$ of $\mathbb{Z}_q^n$ whose elements are obtained from elements of S through the mapping h. Hence, sampling a uniform random element $\mathbf{a} \in \mathcal{L} \subseteq R_q$ is equivalent to sample n uniformly random elements $a_i \in \mathbb{Z}_q$, $i = 0,\ldots,n-1$.

The definition of a discrete Gaussian is slightly more complicated. Indeed, the continuous Gaussian distribution with mean μ and standard deviation σ is

characterized by the density function[4] $\rho(x) = \frac{1}{\sigma} e^{-\pi(x-\mu)^2/\sigma^2}$. Starting from ρ, the discrete Gaussian distribution over a lattice is defined as follows.

Definition 2.14. Let $\mathcal{L} \subseteq \mathbb{R}^n$ be an Euclidean lattice. The *discrete Gaussian distribution* centered in $C \in \mathbb{R}^n$ with standard deviation $\sigma \in \mathbb{R}$ on a full-rank lattice $\mathcal{L}$ is characterized by the density function

$$\mathcal{D}_{\mathcal{L},C,\sigma}(V) = \frac{e^{-\frac{\pi\|V-C\|_2^2}{\sigma^2}}}{\sum_{U\in\mathcal{L}} e^{-\frac{\pi\|U-C\|_2^2}{\sigma^2}}}$$

for all $V \in \mathcal{L}$ and 0 on all the other points in the space.

The Gaussian distribution over the lattice $\mathbb{Z}^n$ with center C and standard deviation σ is denoted by $\mathcal{D}^n_{\mathbb{Z},C,\sigma}$ or $\mathcal{D}_{\mathbb{Z}^n,C,\sigma}$. We omit the center from the notation if the Gaussian is centered around the origin. The Gaussian distribution on a lattice $\mathcal{L}$ with standard deviation σ is denoted by $\mathcal{D}_{\mathcal{L},\sigma}$.

Theorem 4.1 in [Gentry et al., 2008] guarantees that if we have a basis B of $\mathcal{L}$ we can sample from $\mathcal{D}_{\mathcal{L},C,\sigma}$ if $\sigma > \|\tilde{B}\|_2\sqrt{\log(n)}$, n being the dimension of $\mathcal{L}$. Moreover, in the same paper the authors introduce a trapdoor to sample from the discrete Gaussian distribution. Indeed, assume we are given a matrix $A \in \mathbb{Z}^{n\times m}$, a target vector $U \in \mathbb{Z}^n$ and a basis B for $\mathcal{L}_q^{\perp}(A)$. First, we find an arbitrary vector R such that $AR = U \bmod q$. Then we sample a vector $V \sim \mathcal{D}_{\mathcal{L}_q^{\perp}(A),-R,\sigma}$ and set $S = R + V$, thus $AS = U \bmod q$ (as $AV = 0 \bmod q$). It is easy to verify also that $S \sim \mathcal{D}_{\mathbb{Z}^m,0_m,\sigma}$ conditioned on $AS = U \bmod q$. Let $\mathcal{D}^{\perp}_{A,U,\sigma}$ be the distribution of the vectors S such that $S \sim \mathcal{D}_{\mathbb{Z}^m,0_m,\sigma}$ conditioned on $AS = U \bmod q$. If $\sigma > \|\tilde{B}\|_2\sqrt{\log(n)}$ it is possible to sample from this distribution using a basis B of $\mathcal{L}^{\perp}(A)$ (cf. [Gentry et al., 2008; Brakerski et al., 2013]).

The output of the Gaussian distribution over the integers is hard to predict as long as the standard variation is large enough.

Lemma 2.15 ([Lyubashevsky, 2012, Lemma 4.4])**.** *Let $a > 0$, $a \in \mathbb{N}$. For all $Z \in \mathbb{Z}^a$, $\sigma \geq 3$ it holds that*[5]*:*

$$\mathcal{D}^a_{\mathbb{Z},\sigma}(Z) \leq 2^{-a} .$$

[4] This is not what is usually considered the standard probability density of a Gaussian distribution. However, the standard density function of a Gaussian with mean μ' and standard deviation σ' can be obtained from ρ by setting $\mu' = \mu$ and $\sigma' = \frac{\sigma}{\sqrt{2\pi}}$.

[5] The bound on the standard deviation is different due to our choice of using the scaled representation of the Gaussian distribution (cf. footnote 4).

The following lemma gives a bound on the norm of a vector sampled from a discrete Gaussian. Particular values for the variables k and m in the original lemmas have been chosen to ensure that the probabilities would be small enough. The second norm bound is obtained from the original lemma applying the union bound. For the sake of completeness we include also a bound on the generic ℓ_p norm of a Gaussian sample. Remark that it is necessary to adapt the original bound by Lyubashevsky due to our choice of using a scaled Gaussian instead of the standard Gaussian distribution (cf. footnote 4.)

Lemma 2.16 ([Banaszczyk, 1993, Lemma 1.5], [Lyubashevsky, 2012, Lemma 4.4], [Banaszczyk, 1995, Lemma 2.9], [Peikert, 2008, Lemma 3.1])**.** *Let* $a > 0$, $a \in \mathbb{N}$. *The following bounds hold:*

1. $\mathsf{Pr}_{S \xleftarrow{\$} \mathscr{D}_\sigma^a}[\|S\|_2 > 1.05\sigma\sqrt{a}] < (0.6326)^a$
2. $\mathsf{Pr}_{S \xleftarrow{\$} \mathscr{D}_\sigma^a}[\|S\|_\infty > 8\sigma] < a2^{-275}$
3. $\mathsf{Pr}_{S \xleftarrow{\$} \mathscr{D}_\sigma^a}\left[\|S\|_p > a^2 \cdot \frac{(2\Gamma(p/2+1))^{1/p}}{\sqrt{\pi}}\right] < a^{-p}$. [6]

For our applications it is enough that the bounds hold with non-negligible probability, as if the norm of a sample is too large, it is possible to discard it and sample a fresh one.

The following Theorem by Micciancio and Peikert [2012] shows how a (pseudo-)random vector $\mathbf{U}$, for which no trapdoor is known, can be extended into a pseudo-random vector $[\mathbf{U}\ \mathbf{V}]$ for which we will be able to sample from the Gaussian distribution $\mathscr{D}^\perp_{[\mathbf{U}\ \mathbf{V}+\mathbf{mG}],\mathbf{u},\sigma}$ for any invertible $\mathbf{m}$ and for some standard deviation σ.

Theorem 2.17 (adapted from [Micciancio and Peikert, 2012])**.** *Let* $\mathbf{A}$ *be a vector in* $R_q^{1\times\ell}$ *and* $\mathbf{X}$ *be a matrix in* $R_q^{\ell\times m}$. *Also let the* gadget matrix *be*

$$\mathbf{G} = \begin{bmatrix} 1 & \lceil q^{1/m} \rceil & \dots & \lceil q^{(m-1)/m} \rceil \end{bmatrix}.$$

Then for any invertible $\mathbf{m} \in R_q$, *there is an algorithm that can sample from the distribution* $\mathscr{D}^\perp_{[\mathbf{A}\ \mathbf{AX}+\mathbf{mG}],\mathbf{u},\sigma}$ *for any* $\sigma = s\omega(\sqrt{\log n})$, $s = O(\sqrt{n\log q})$ *for any* $\mathbf{u} \in R_q$.

[6] The inequality 3) is obtained from [Peikert, 2008, Lemma 3.1] setting $r = a^{(p+1)/p} \cdot \frac{(2\Gamma(p/2+1))^{1/p}}{\sqrt{\pi}}$ and using the bound $a^{(p+1)/p} \leq a^2\ \forall p \in \mathbb{N}, a \in \mathbb{N}$.

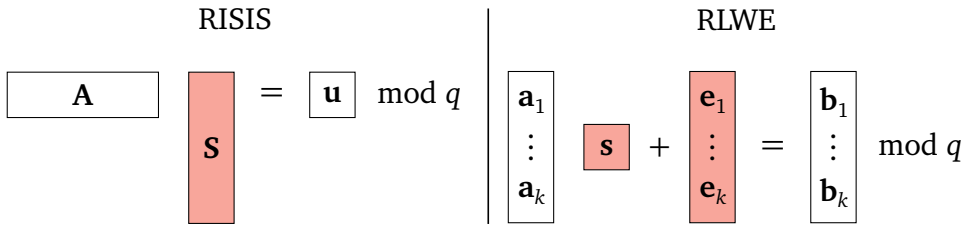

Figure 2.2. RISIS and RLWE relations. In red are highlighted the secrets and errors (which are not included in the instances of the problems).

Lemma 2.18 is a combination of the double-trapdoor idea by Agrawal et al. [2010], with the sampling procedure by Brakerski et al. [2013].

Lemma 2.18. *Suppose* $\mathbf{U} \in R_q^{1\times k}$ *and* $\mathbf{V} \in R_q^{1\times m}$ *are polynomial vectors, and* $\mathbf{B}_U$, $\mathbf{B}_{(U,V)}$ *are bases of* $\Lambda^\perp(\mathbf{U})$ *and* $\Lambda^\perp([\mathbf{U}\ \mathbf{V}])$ *respectively such that*

$$\|\tilde{\mathbf{B}}_U\|,\ \|\tilde{\mathbf{B}}_{(U,V)}\| < \sigma\sqrt{\pi/\ln(2n+4)}.$$

Then, there exists an algorithm $\mathsf{SampleD}(\mathbf{U},\mathbf{V},\mathbf{B},\mathbf{u},\sigma)$*, where* $\mathbf{B}$ *is either* $\mathbf{B}_U$ *or* $\mathbf{B}_{(U,V)}$*, that can efficiently sample from the distribution* $D^\perp_{[\mathbf{U}\ \mathbf{V}],\mathbf{u},\sigma}$ *for any* $\mathbf{u} \in R_q$.

2.3.4 Lattice-based Hardness Assumptions

The security of lattice-based cryptographic schemes is mainly based on two hardness assumptions: the Short Integer Solution (SIS) Problem and the Learning With Errors (LWE) Problem (cf. Figure 2.2).

SIS can be seen as a variant of the subset-sum problem over a particular group, and it allows building one-way and collision-resistant hash functions (cf. [Ajtai, 1996] and [Goldreich et al., 1996] respectively) and signatures ([Gentry et al., 2008; Cash et al., 2010; Boyen, 2010; Lyubashevsky, 2012; Micciancio and Peikert, 2012]). When Micciancio proposed to work with ideal lattices [Micciancio, 2002], he also presented a ring-based variant of SIS, the *generalized compact knapsack problem*, now known as the Ring-SIS (RSIS) problem.

Remark 2. We use a particular version of the RSIS assumption where the matrix $\mathbf{A}$ has the last component equal to $\mathbf{1}$, which is called the Hermite Normal Form. This variant was originally defined for integer lattices, and proved equivalent to the normal SIS instance [Buchmann and Lindner, 2009]. The HNF of RSIS can be proved equivalent to RSIS in an analogous way.

Definition 2.19. The RSIS problem asks given a vector $\mathbf{A} \in R_q^{1\times m}$ to find a vector $\mathbf{S} \in R_q^m$ such that $\mathbf{AS} = \mathbf{0} \bmod q$ and $\|\mathbf{S}\|_\infty \leq \beta$. The average-case RSIS distribution $\mathrm{RSIS}_{m,q,\beta}$ where $m = m(\lambda)$, $q = q(\lambda)$, $\beta = \beta(\lambda)$ is the ensemble over the instances $(q(\lambda), \mathbf{A}, \beta(\lambda))$ where $\mathbf{A} \xleftarrow{\$} R_q^{1\times m(\lambda)}$.

The *inhomogeneous* RSIS problem (RISIS) asks to find $\mathbf{S} \in R_q^m$ such that $\mathbf{AS} = \mathbf{u}$, and $\|\mathbf{S}\|_\infty \leq \beta$ given $\mathbf{A}$ and $\mathbf{u} \in R_q$. The average-case RISIS distribution $\mathrm{RISIS}_{m,q,\beta}$ where $m = m(\lambda)$, $q = q(\lambda)$, $\beta = \beta(\lambda)$ is the ensemble over the instances $(q(\lambda), \mathbf{A}, \mathbf{u}, \beta(\lambda))$ for uniformly random and independent $\mathbf{A} \xleftarrow{\$} R_q^{1\times m(\lambda)}$ and $\mathbf{u} \xleftarrow{\$} R_q$.

Interpreting this problem in terms of lattices, the (R)SIS problem is actually equivalent to finding a (sufficiently) short vector in $\Lambda_q^\perp(\mathbf{A})$, i.e., to solving the Shortest Vector Problem over the hard random lattice $\Lambda_q^\perp$.

LWE requires to distinguish "noisy linear equations" from truly random ones (cf. [Regev, 2005]), and is even more versatile, allowing in particular the construction of public-key encryption schemes [Regev, 2005; Micciancio and Peikert, 2012]. Over ideal lattices, the learning with error problem requires essentially to determine whether a pair $(\mathbf{a}, \mathbf{b}) \in R_q \times R_q$ is such that $\mathbf{b} \approx \mathbf{as}$ for some secret $\mathbf{s}$, or if it is composed by two uniformly random ring elements [Lyubashevsky et al., 2010].

Definition 2.20. The *RLWE distribution* $A_{\mathbf{s},\chi}$ outputs pairs $(\mathbf{a}, \mathbf{b}) \in R_q \times R_q$ such that $\mathbf{b} = \mathbf{as} + \mathbf{e} \bmod q$ for a uniformly random $\mathbf{a}$ in R_q and $\mathbf{e} \xleftarrow{\$} \chi$.

The (average-case) *$\mathrm{RLWE}_{k,\chi}$ decisional problem* (in the normal form) on ring R_q with distribution χ and k samples is to distinguish whether k pairs $(\mathbf{a}_1, \mathbf{b}_1), \ldots, (\mathbf{a}_k, \mathbf{b}_k)$ were sampled from $A_{\mathbf{s},\chi}$ for a random choice of $\mathbf{s} \xleftarrow{\$} \chi$ or from the uniform distribution over R_q^2.

The *$\mathrm{RLWE}_{k,\chi}$ search problem* on ring R_q with distribution χ is: given access to arbitrarily many samples from $A_{\mathbf{s},\chi}$ for some arbitrary $\mathbf{s} \in R_q$, find $\mathbf{s}$.

Remark 3. The version of RLWE in Definition 2.20 is the "normal form" of the problem, where the secret and the error are chosen from the same distribution (in the generic form, the secret is chosen uniformly at random in the ring). This version is as hard as the standard one (cf. Lemma 2.24 in the full version of [Lyubashevsky et al., 2013]).

The Learning with Error problem can be interpreted using hard random lattices as well: solving the search (R)LWE problem is equivalent to solving the Closest Vector Problem on $\Lambda_q(\mathbf{A})$, where $\mathbf{A} = [\mathbf{a}_1, \ \ldots, \ \mathbf{a}_k]$.

Remark 4. The definition of the *error distribution* χ is fundamental to the problem hardness, and has to be handled with extra care when working over rings. In the original definition of LWE such distribution was the the discrete Gaussian (over $\mathbb{Z}_q$, cf. [Gentry et al., 2008] for how to define it). When translating such problem to the polynomial rings setting, such distribution is not enough. In fact, the original security reductions for RLWE requires the error distribution to be sampled each time from a set of (non-spherical) Gaussian distributions (cf. [Lyubashevsky et al., 2010]). However, the result was expanded to include reductions for other error distributions, in particular the uniform distribution with constant number of samples and the Gaussian distribution with limited number of samples. The result is reported in Theorem 2.23.

The hardness of both assumptions can be put in relation with the hardness of finding a short element (i.e., a polynomial with small norm) in an ideal lattice in the ring R(cf. [Lyubashevsky and Micciancio, 2006]).

Definition 2.21. The approximate Shortest Vector Problem over rings $\mathrm{RSVP}^{\infty}_{\gamma}$ (resp. $\mathrm{RSVP}^{2}_{\gamma}$) asks, given an ideal lattice $\mathscr{L} \subseteq R$, to find an element $\mathbf{s} \in \mathscr{L}$ such that $\mathbf{s} \neq \mathbf{0}$ and $\|\mathbf{s}\|_\infty \leq \gamma \lambda_1^\infty(\mathscr{L})$ (resp. $\|\mathbf{s}\|_2 \leq \gamma \lambda_1^2(\mathscr{L})$), where $\lambda_1^\infty(\mathscr{L}) = \min_{\mathbf{x}\in\mathscr{L}} \|\mathbf{x}\|_\infty$ (resp. $\lambda_1^2(\mathscr{L}) = \min_{\mathbf{x}\in\mathscr{L}} \|\mathbf{x}\|_2$).

Remark 5. As observed already in [Lyubashevsky, 2016], the choice of the polynomial representation has an impact on norm values. In fact,there are polynomial rings where the fact that a polynomial has small norm when represented as the vector of its coefficients does not imply that its norm is small when represented through the NTT transform (cf. [Erdös, 1946]). Thus, reductions between hardness assumptions (such as the results in the rest of the section) might be less strong (or even meaningless) when switching from one representation to the other. However, it was shown that for polynomial rings $\mathbb{Z}_q[x]/\langle \mathbf{f} \rangle$ where $\mathbf{f} = x^n + \sum_i f_i x^i$ for small f_i, the expansion factor of the norm when switching from coefficient to NTT representation is small enough to guarantee that the reductions are still meaningful (cf. [Lyubashevsky and Micciancio, 2006]). Therefore, we can still point to those reductions to motivate our choice of hardness assumptions.

Lyubashevsky and Micciancio [2006] and Peikert and Rosen [2006] independently showed that solving an average-case instance of RSIS is at least as hard as solving SPP in the worst-case[7]. Lyubashevsky, Peikert and Regev then showed

[7]Remark that the hardness of RSIS implies the hardness of its inhomogeneous version. This can be seen for example from the original version of Theorem 2.22, where the RSIS problem is

a polynomial-time quantum reduction from RLWE with Gaussian error distribution to the shortest vector problem over ideal lattices (cf. [Lyubashevsky et al., 2010]). The results (adapted to the particular ring R_q used in this work) are included in the following.

Theorem 2.22 ([Lyubashevsky and Micciancio, 2006, Theorem 5.1])**.** *Consider the polynomial ring $R_q = \mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$, where $n = 2^d \in \mathbb{N}$, q is a prime. There is a polynomial-time reduction from $RSVP_\gamma^\infty$ to solving an instance of RSIS drawn from $RSIS_{m,q,\beta}$ where*

$$m > \frac{\log q}{\log \beta}\,, \qquad q > 6\beta m n^{1.5} \log n\,, \qquad \gamma = 36\beta m n \log^2 n\,.$$

Theorem 2.23 (Theorem 2.22, full-version of [Lyubashevsky et al., 2013])**.** *Consider the polynomial ring $R_q = \mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$, where $n = 2^d \in \mathbb{N}$, q is a prime. Let $\alpha = \alpha(n)$, and $q = q(n) \geq 2$, $q \equiv 1 \bmod 2n$ be a poly(n)-bounded prime such that $\alpha q \geq \omega(\sqrt{\log n})$. There exists a polynomial-time quantum reduction from $RSVP_\gamma^2$ for $\gamma = \tilde{O}(\sqrt{n}/\alpha)$ to $RLWE_{\ell-1,\chi}$, where $\ell > 1$ and $\chi = \mathscr{D}_\sigma$, $\sigma = \alpha(n\ell/\log(n\ell))^{1/4}$.*

The degradation in the number of samples comes from the choice of the normal form of RLWE (cf. Lemma 2.24 in the full version of [Lyubashevsky et al., 2013]).

Lyubashevsky et al. also observe that, in practice, if ℓ is a small constant, then it seems that one can safely take the distribution χ to be uniformly random in $\mathscr{S}_1$. This is insecure for polynomially large ℓ [Arora and Ge, 2011]. To estimate the hardness of the lattice problems for given parameters in the security reductions we use the root Hermite factor δ introduced by Gama and Nguyen [Gama and Nguyen, 2008]. An example of how to compute such parameter and how to interpret its value can be found in Section 4.1.3.

Finally, Langlois and Stehlé [2015] defined the multidimensional versions of RSIS and RLWE, called Module-SIS and Module-LWE respectively.

Definition 2.24 (Module-SIS)**.** Given a *matrix* of ring elements $\mathbf{A} \xleftarrow{\$} R_q^{m_1 \times m_2}$, the average-case Module-SIS problem $MSIS_{q(\lambda),\beta(\lambda)}$ asks to find a vector $\mathbf{S} \in R_q^{m_2}$ such that $\mathbf{AS} = \mathbf{0}$ and $\|\mathbf{S}\| \leq \beta$.

The inhomogeneous version can be defined analogously.

presented as the collision-resistance property of a family $\mathscr{H}$ of hash functions. The first preimage-resistance property of such family corresponds exactly to RISIS being hard. As collision-resistance implies first-preimage resistance, this implies that the hardness of RSIS implies the hardness of RISIS.

Definition 2.25 (Module-LWE)**.** The *MLWE* distribution $A_{\mathbf{S},\chi}$ outputs pairs $(\mathbf{A}, \mathbf{AS} + \mathbf{e} \bmod q) \in R_q^m \times R_q$, where the secret $\mathbf{S}$ is in R_q^m and the error $\mathbf{e}$ is drawn from a distribution χ.

The (average-case) $MLWE_{k,\chi}$ *decisional problem* (in the normal form) on ring R_q with distribution χ and k samples is to distinguish whether k pairs $(\mathbf{A}_1, \mathbf{b}_1), \ldots, (\mathbf{A}_k, \mathbf{b}_k)$ were sampled from $A_{\mathbf{S},\chi}$ for a random choice of $\mathbf{S} \xleftarrow{\$} R_q^m$ or from the uniform distribution over $R_q^m \times R_q$.

The $MLWE_{k,\chi}$ *search problem* on ring R_q with distribution χ is: given access to arbitrarily many samples from $A_{\mathbf{S},\chi}$ for some arbitrary $\mathbf{S} \in R_q^m$, find $\mathbf{S}$.

Solving any of these problems on the average can be reduced to the hardness of solving a generalization of a well-known lattice-problem (the Shortest Independent Vectors Problem) in the worst-case [Langlois and Stehlé, 2015].

Chapter 3

Relaxed Cryptographic Primitives

Privacy preserving protocols, and in particular privacy-preserving signatures, rely on a common core building block: *zero-knowledge proofs*. Such primitives allow a user to prove knowledge of a bit string (e.g., a lattice point, an attribute, etc.) without revealing it. The first step towards building efficient privacy-preserving signatures from lattice-based hardness assumptions is to obtain efficient zero-knowledge proofs for lattices.

In this chapter we present an approach to the problem that we called *Relaxed Cryptography* (introduced by Boschini et al. [2018b]) that led to the first lattice-based group signature with signatures of size less than 1.8 MB (built by Boschini et al. [2018a]). In fact, to be composed with other preexisting building blocks (such as digital signatures), the zero-knowledge proof obtained requires to change definitions and instantiations of such primitives. This chapter contains all the relaxed primitives needed for the construction of the privacy-preserving protocols in Chapters 4 and 5.

3.1 Zero-Knowledge Proofs

Let $L \subseteq \{0,1\}^*$ be a NP language identified by a relation $\mathscr{R} \subseteq \{0,1\}^*$, i.e., an instance x is in L if there exists a witness w such that $(x,w) \in \mathscr{R}$. The witness w is a *valid* witness for the instance x; valid witnesses are restricted to have length at most $p(|x|)$ for some polynomial $p(.)$. When building cryptographic protocols relations are usually required to be hard, i.e., it should be hard to find a witness for a given instance x, even when it is known that $x \in L$.

Definition 3.1 (Hard relation[1])**.** A relation $\mathscr{R}$ is said to be *hard* if

[1]Definitions in this section are taken from [Damgård, 2002].

- There exists a PPT algorithm RGen that on input 1^k outputs a pair $(x, w) \in \mathscr{R}$ s.t. $|x| = k$.
- For all PPT algorithms A the following probability is negligible:

$$\mathsf{Pr}_k\left[(x, w_A) \in \mathscr{R} \ : \ (x, w) \leftarrow \mathsf{RGen}(1^k),\ w_A \leftarrow \mathsf{A}(x)\right] .$$

Let P be an algorithm that wants to prove to a verifier V that it knows a valid witness w for a (public) instance x; assume they are both PPT algorithms.

Definition 3.2 (Proof System)**.** Let L be a language defined by relation $\mathscr{R}$. Let P and V be PPT algorithms, and let $\langle \mathsf{P}, \mathsf{V} \rangle$ denote their interaction, whose output is the bit output by the verifier. A protocol $\Pi = (\mathsf{P}, \mathsf{V})$ is a *proof system* if the following holds:

Completeness. $\forall x \in L$,

$$\mathsf{Pr}[1 \leftarrow \langle \mathsf{P}, \mathsf{V} \rangle(x)] \geq 1 - \nu_c(|x|) .$$

Soundness. $\forall x \notin L$, and for all (even unbounded) PPT algorithms A,

$$\mathsf{Pr}[1 \leftarrow \langle \mathsf{A}, \mathsf{V} \rangle(x)] \leq \nu_s(|x|) .$$

The quantity ν_s is called *soundness error*.

If A runtime is polynomially bounded, soundness is computational (as it relies on assuming some computation would take time longer than polynomial in the length of x).

A Σ-protocol [Cramer et al., 1994] is a particular 3-round proof system (cf. Figure 3.1). Both the prover and the verifier take as input random coins ρ. At the end of each computation, they output the result of the computation and a state (st_p for the prover, and st_v for the verifier), that contains the output of the computation and some additional information (such as the instance and witness in the case of the prover).

Definition 3.3 (Σ-protocol)**.** Let $\mathsf{P} = (\mathsf{P}_0, \mathsf{P}_1)$ be a prover protocol and $\mathsf{V} = (\mathsf{V}_0, \mathsf{V}_1)$ be a verifier protocol for relation $\mathscr{R}$ and language L. A protocol $\Sigma = ((\mathsf{P}_0, \mathsf{P}_1), (\mathsf{V}_0, \mathsf{V}_1))$ is a Σ-protocol if it is a 3-move protocol as shown in Figure 3.1, and it satisfies the following properties:

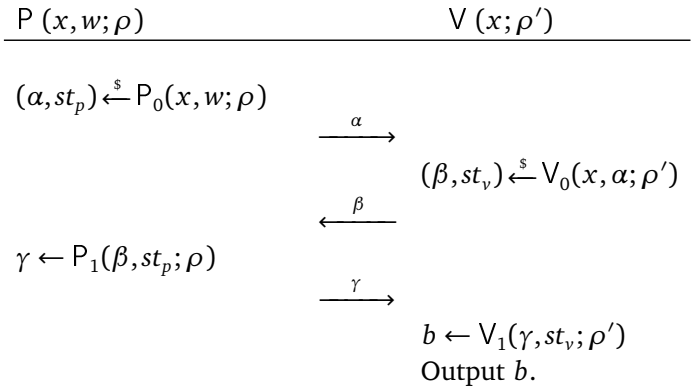

Figure 3.1. Structure of a Σ-protocol.

Correctness. For all $(x,w) \in \mathscr{R}$, if $\mathsf{P}(x,w;\rho)$ and $\mathsf{V}(x;\rho')$ follow the protocol, the verifier accepts but with negligible probability:

$$\Pr\left[b = 1 \ : \ \begin{array}{c} (\alpha, st_p) \leftarrow \mathsf{P}_0(x,w;\rho)\ ;\ (\beta, st_v) \xleftarrow{\$} \mathsf{V}_0(x,\alpha;\rho)\ ; \\ \gamma \leftarrow \mathsf{P}_1(\beta, st_p;\rho)\ ;\ b \leftarrow \mathsf{V}_1(\gamma, st_v) \end{array} \right] = 1 - \nu(|x|)\,,$$

where the probability is over the coins ρ of P_0 and the coins of V_0.

Honest-verifier zero knowledge (HVZK). There exists a polynomial-time algorithm Σ, called *zero-knowledge simulator,* such that for any distinguisher D and for any $(x,w) \in \mathscr{R}$,

$$\Pr\left[b' = b \ : \ \begin{array}{c} b \xleftarrow{\$} \{0,1\}\ ;\ (\alpha, st_p) \leftarrow \mathsf{P}_0(x,w;\rho)\ ; \\ (\beta, st_v) \xleftarrow{\$} \mathsf{V}_0(x,\alpha;\rho)\ ;\ \gamma \leftarrow \mathsf{P}_1(\beta, st_p;\rho)\ ; \\ \pi_0 \leftarrow (\alpha,\beta,\gamma)\ ;\ \pi_1 \leftarrow \Sigma(x)\ ;\ b' \xleftarrow{\$} \mathsf{D}(x,w,\pi_b) \end{array} \right] - \frac{1}{2}$$

is negligible.

Special Soundness. There exists a PPT extractor E that on input an instance x and a pair of accepting transcripts (α,β,γ), (α,β',γ') where $\beta \neq \beta'$ computes a witness w' such that $(x,w') \in \mathscr{R}$.

A Σ-protocol is trivially a proof system (special soundness implies soundness).

Remark 6. A Σ-protocol is in fact a proof of knowledge (cf. [Damgård, 2002]). In particular, if the soundness holds against a computationally unbounded prover, the protocol is called a *proof* of knowledge. If soundness only holds for a polynomially bounded prover, the protocol is called an *argument* of knowledge.

Sometimes it is not possible for the prover and the verifier to interact, as, for example, they might not be online at the same time. Hence, it becomes necessary fro the prover to be able to produce a convincing, still zero-knowledge proof without the help of V. This type of protocol is called a *non-interactive zero-knowledge proof*. Besides zero-knowledge, it requires that an adversary that sees multiple valid proofs cannot generate a proof of a false statement. This property is called *simulation soundness*, as it is specific to a simulator Σ for the prover to which the adversary has access. Such simulator is a stateful algorithm that can operate in two modes: $(h_i, st) \xleftarrow{\$} \Sigma(1, st, q)$ answers random oracle queries $H_c(q)$, while $(\pi, st) \xleftarrow{\$} \Sigma(2, st, x)$ simulates a NIZK proof π for x. Below, the oracle $\Sigma_1(q)$ returns the first outputs of $\Sigma(1, st, q)$, the oracle $\Sigma_2(x, w)$ returns the first output of $\Sigma(2, st, x)$ if $(x, w) \in \mathscr{R}$ and returns $\perp$ otherwise, and oracle $\Sigma_2'(x)$ returns the first output of $\Sigma(2, st, x)$ regardless whether $x \in L$ or not.

Definition 3.4 (NIZK proof system)**.** A *NIZK proof system* $(\mathsf{P}^{H_c}, \mathsf{V}^{H_c})$ in the Random Oracle Model for relation $\mathscr{R}$ and language L is couple of PPT algorithms with the following properties:

Correctness. For all $(x, w) \in \mathscr{R}$ it holds that

$$\Pr\left[b = 1 \ : \ \pi \leftarrow \mathsf{P}^{H_c}(x, w; \rho) \, , b \leftarrow \mathsf{V}^{H_c}(x, \pi) \right] \ = \ 1 - \nu(|x|) \, .$$

(Non-Interactive) Zero-Knowledge. There exists a PPT simulator Σ such that for all PPT distinguishers D the following quantity is negligible:

$$\left| \Pr\left[\mathsf{D}^{H_c(\cdot), \mathsf{P}^{H_c}(\cdot,\cdot)}(1^\lambda) = 1 \right] - \Pr\left[\mathsf{D}^{\Sigma_1(\cdot), \Sigma_2(\cdot,\cdot)}(1^\lambda) = 1 \right] \right| \, . \tag{3.1}$$

Simulation Soundness. There exists a PPT simulator Σ such that for all PPT adversaries A,

$$\Pr\left[\mathsf{V}^{\Sigma_1}(x^*, \pi^*) = 1 \wedge x^* \notin L \wedge (x^*, \pi^*) \notin \mathscr{Q} \ : \ (x^*, \pi^*) \xleftarrow{\$} \mathsf{A}^{\Sigma_1, \Sigma_2'}(1^\lambda) \right]$$

is negligible, where $\mathscr{Q}$ is the set of tuples (x, π) where A made a query $\Sigma_2(x)$ and obtained response π.

Finally, analogously to identification schemes, a Σ-protocol can be made non-interactive using the Fiat-Shamir (FS) transformation [Fiat and Shamir, 1987].

Definition 3.5 (Fiat-Shamir transform)**.** The FS transformation of an interactive protocol (P, V) is a non-interactive proof system $(\mathsf{P}^{H_c}, \mathsf{V}^{H_c})$ where:

- H_c is a random oracle with range equal to the space of the verifier's coins.

- The proving algorithm $\mathsf{P}^{H_c}(x, w; \rho)$ computes a proof $\pi = (\alpha, \beta, \gamma)$ by choosing random coins ρ and computing $(\alpha, st_p) \leftarrow \mathsf{P}_0(x, w; \rho)$, $\beta \leftarrow H_c(x, \alpha)$, and $\gamma \leftarrow \mathsf{P}_1(\beta, st_p; \rho)$.
- The verification algorithm $\mathsf{V}^{H_c}(x, \pi)$ checks that $H_c(x, \alpha) = \beta$ and $\mathsf{V}_1(x, \alpha, \beta, \gamma) = 1$.

Faust et al. [Faust et al., 2012, Theorem 2] proved that in the ROM the FS transform applied to a (public-coin) Σ-protocol with quasi-unique responses results in a NIZK proof system.

3.2 Previous Works: Schnorr-type Proofs for Lattice Problems

Fixed $n, m \in \mathbb{N}$, a prime q, and $N \in \mathbb{R}$, a typical hard relation over ideal lattices is of the following form:

$$\mathscr{R} = \mathscr{R}_{n,m_1,m_2,q,N} = \{(\mathbf{A}, \mathbf{U}; \mathbf{S}) \in R_q^{m_1 \times m_2} \times R_q^{m_1} \times R^{m_2} \mid \mathbf{AS} = \mathbf{U} \bmod q \ \wedge \ \|\mathbf{S}\| \leq N\} . \tag{3.2}$$

Indeed, with appropriately defined instance generator it is possible to prove that the relation $\mathscr{R}$ is in fact a hard relation under (Module-)RSIS. Remark that we intentionally have not specified the type of norm. This will be specified when instantiating each Σ-protocol.

Theorem 3.6 (Hard Relations over Ideal Lattice)**.** *Consider the polynomial ring $R_q = \mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$, where $n = 2^d \in \mathbb{N}$, q is a prime. The relation $\mathscr{R}$ is a (computationally) hard relation under the $RSIS_{m,q,\beta}$ assumption when $\mathscr{R} = \mathscr{R}_{\text{RSIS}} = \mathscr{R}_{n,1,m,q,\beta}$, and under the $MSIS_{q,\beta}$ assumption in the general case (when $\mathscr{R}_{n,m_1,m_2,q,\beta}$). If the parameters m, q, and β of $\mathscr{R}_{\text{RSIS}}$ satisfy the requirements of Theorem 2.22, the relation is hard under $RSVP_\gamma$ (where γ is set as in Theorem 2.22).*

Proof. The proof has two steps. First, it has to be proved that RSIS fits the description of a hard lattice relation (cf. Equation (3.2)). Second, it is necessary to show that there exists a polynomial-time algorithm that outputs instances for which is hard to find a witness (in polynomial-time). The proof of the final statement trivially follows from Theorem 2.22.

The relation $\mathscr{R}$ describing R(I)SIS is:

$$\mathscr{R}_{\text{RSIS}} = \mathscr{R}_{n,1,m,q,\beta} = \{(\mathbf{A}, \mathbf{u}; \mathbf{S}) \in R_q^{1 \times m} \times R_q \times R^m \mid \mathbf{AS} = \mathbf{u} \bmod q \ \wedge \ \|\mathbf{S}\| \leq \beta\} . \tag{3.3}$$

Generating an instance of $\mathscr{R}_{\text{RSIS}}$ in polynomial-time is possible using for example the following algorithm:

$\mathsf{RGen}_{\text{RSIS}}(1^\lambda)$:
$\mathbf{A} \xleftarrow{\$} R_q^m$
$\mathbf{S} \xleftarrow{\$} \{\mathbf{S} \in R^m \mid \|\mathbf{S}\| \leq \beta\}$
$\mathbf{u} = \mathbf{AS} \bmod q$.
Return $(\mathbf{A}, \mathbf{u}; \mathbf{S})$.

Given an instance $(\mathbf{A}, \mathbf{u}) \leftarrow \mathsf{RGen}_{\text{RSIS}}(1^\lambda)$, if there exists a PPT algorithm A that finds a witness $\mathbf{S}$ such that $(\mathbf{A}, \mathbf{u}; \mathbf{S}) \in \mathscr{R}_{n,m,q,\beta}$, then it is possible to exploit it to solve an average-case RSIS instance $\mathbf{A}$ by giving the $\mathscr{R}_{\text{RSIS}}$ instance $(\mathbf{A}, \mathbf{0})$ as input to A. As there is no requirement on the probability distribution of $\mathscr{R}_{\text{RISIS}}$ instances, A cannot distinguish between $(\mathbf{A}, \mathbf{u}) \xleftarrow{\$} \mathsf{RGen}_{\text{RISIS}}(1^\lambda)$ and a random instance $(\mathbf{A}, \mathbf{0})$ of RSIS.

The proof that the generic $\mathscr{R}_{n,m_1,m_2,q,\beta}$ relation is hard under MSIS is trivial. □

A zero-knowledge and sound proof for relation $\mathscr{R}$ in Equation (3.2) can be extracted from the lattice-based digital signature by Lyubashevsky [2012] (although in the original work it is not explicitly presented as such). The protocol was based on an identification scheme designed by Schnorr [1990] for the DLOG problem, and it is shown in Figure 3.2. The original version of the protocol is defined for integer lattices, but it can be easily adapted to ideal lattices exploiting the group homomorphism h (cf. Section 2.3.2).

Although the original scheme had a larger challenge set, we present it with a simple binary challenge set for the sake of clarity. The prover $\mathsf{P} = (\mathsf{P}_0, \mathsf{P}_1)$ and verifier $\mathsf{V} = (\mathsf{V}_0, \mathsf{V}_1)$ algorithms work as follows:

$(\mathbf{T}, st_p) \xleftarrow{\$} \mathsf{P}_0(x, w; \rho)$ **:** the algorithm samples $\mathbf{Y} \xleftarrow{\$} \mathscr{D}_{R,\sigma}^{m_2}$ (the value of σ will be set later), and sets $\mathbf{T} = \mathbf{AY} \bmod q$, then sets $st_p = (\mathbf{A}, \mathbf{U}, \mathbf{S}, \mathbf{Y})$ and sends $\mathbf{T}$ to V_0.

$(c, st_v) \xleftarrow{\$} \mathsf{V}_0(\mathbf{A}, \mathbf{U}, \mathbf{T}; \rho')$**:** upon receiving $\mathbf{T}$, the algorithm samples a random binary challenge $c \xleftarrow{\$} \{0, 1\}$, sets $st_v = (\mathbf{A}, \mathbf{U}, \mathbf{T}, c, B)$ and sends c to P_1.

$\mathbf{Z} \leftarrow \mathsf{P}_1(c, st_p; \rho)$**:** on input c and $st_p = (\mathbf{A}, \mathbf{U}, \mathbf{S}, \mathbf{Y})$ computes $\mathbf{Z} = \mathbf{Y} + c\mathbf{S}$, and sends $\mathbf{Z}$ to the verifier with probability $\min\left\{\frac{\mathscr{D}_{R,\sigma}^{m_2}(\mathbf{Z})}{M\mathscr{D}_{R,\sigma,c\mathbf{S}}^{m_2}(\mathbf{Z})}, 1\right\}$ for some constant $M > 0$ and standard variation $\sigma = \omega(T\sqrt{\log nm_2})$, T being a bound on the norm of $c\mathbf{S}$.

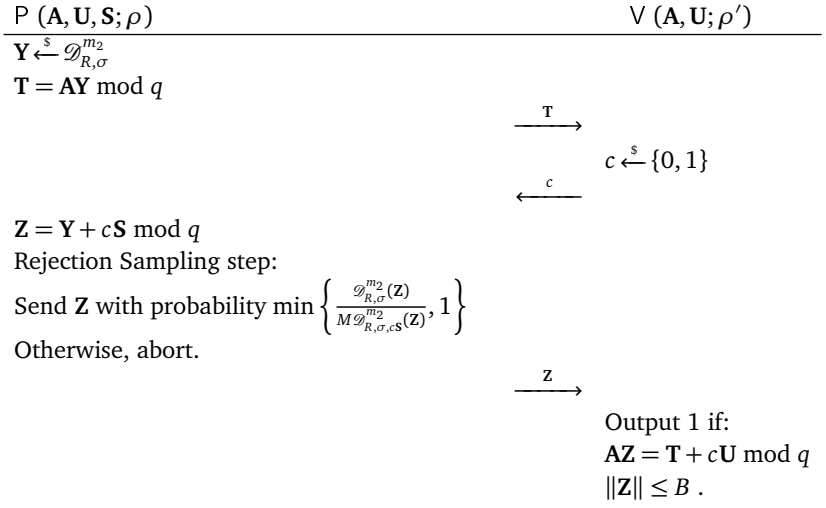

Figure 3.2. Σ-protocol for lattice-based relations.

$b \leftarrow \mathsf{V}_1(\mathbf{Z}; st_v)$**:** on input $\mathbf{Z}$ and $st_v = (\mathbf{A}, \mathbf{U}, \mathbf{T}, c, B)$ it outputs 1 if $\mathbf{AZ} = \mathbf{T} + c\mathbf{U}$ and $\|\mathbf{Z}\| \leq B$, where $B = B_2 = 1.05\sigma\sqrt{nm_2}$ if $\|.\| = \|.\|_2$ and $B = B_\infty = 8\sigma$ if $\|.\| = \|.\|_\infty$.

The rejection sampling step is the core of this scheme, as it allows to hide the dependency of the response on the witness. It is based on the following result.

Lemma 3.7 ([Lyubashevsky, 2012, Lemma 4.7, Theorem 4.6 and Lemma 4.5])**.** *Let $\mathcal{V}$ be an arbitrary set, and $h : \mathcal{V} \to \mathbb{R}$, $f : \mathbb{Z}^m \to \mathbb{R}$ be probability distributions. Let $g_V : \mathbb{Z}^m \to \mathbb{R}$ be a family of probability distributions indexed by all $V \in \mathcal{V}$ such that*

$$\exists M \in \mathbb{R} \text{ such that } \forall V \in \mathcal{V}, \mathsf{Pr}_{z \xleftarrow{\$} f}[M g_V(z) \geq f(z)] \geq 1 - \nu .$$

In particular, when $\forall V \in \mathcal{V}$, $\|V\| < T$ for a constant $T > 0$, $g_V = \mathcal{D}^m_{V,\sigma}$ and $f = \mathcal{D}^m_\sigma$ for some standard variation $\sigma = \omega(T\sqrt{\log m})$, it yields $\nu = 2^{-\omega(\log m)}$.

Then the distribution of the output of the following algorithm A

1. $V \xleftarrow{\$} h$
2. $Z \xleftarrow{\$} g_V$
3. *outputs* (V, Z) *with probability* $\min\left(\frac{f(Z)}{M g_V(Z)}, 1\right)$

is within statistical distance ν/M from the distribution of the following algorithm B*:*

1. $V \xleftarrow{\$} h$
2. $Z \xleftarrow{\$} f$
3. *output* (V, Z) *with probability* $1/M$.

Moreover, the probability that A *outputs something is at least* $(1-\nu)/M$.

More concretely, if $\mathcal{V} = \{X \in \mathbb{Z}^m \ : \ \|X\| \leq B\}$ *for a constant* $B > 0$, $\sigma = \alpha B$ *for* $\alpha > 0$, $g_V = \mathcal{D}^m_{\mathbb{Z},\sigma,V}$ *and* $f = \mathcal{D}^m_{\mathbb{Z},\sigma}$, *then* $M = \exp(12/\alpha + 1/(2\alpha^2))$, *the statistical distance between* A *and* B *is at most* $\frac{2^{-100}}{M}$, *and the probability that* A *outputs something is at least* $\frac{1-2^{-100}}{M}$.

To apply the results of this lemma, it is necessary that the standard deviation σ is a multiple of the bound β on the norm of the witness $\mathbf{S}$. In particular, we will set $\alpha = 12$ and $\sigma = 12\beta$ (so that the output probability is large enough).

Remark that we have intentionally stated the norm check w.r.t. a generic norm $\|.\|$. This will be useful later, as it allows to plug in the infinity norm or the Euclidean norm, depending on the necessity. The bound B on the norm of $\mathbf{Z}$ depends on the choice of the norm according to Lemma 2.16.

The following theorem summarizes the results by Lyubashevsky (adapted to the case of ideal lattices). We add a sketch of the proof to recall some techniques we need when working with the generalized version.

Theorem 3.8 (Summary of the results in [Lyubashevsky, 2012] applied to rings)**.** *The protocol* (P, V) *defined as before is a proof system with soundness error at least* $1/2$ *for the (hard) relation* $\mathcal{R}$ *in the ROM. The proof system has abort probability* $1 - \min\left\{\frac{\mathcal{D}^m_{R,\sigma}(\mathbf{Z})}{M\mathcal{D}^m_{R,\sigma,c\mathbf{S}}(\mathbf{Z})}, 1\right\}$. *Combined with the Fiat-Shamir transform, such proof system satisfies the (non-interactive) zero-knowledge property in Definition 3.4.*

Sketch. Correctness follows from Lemma 2.16. It is possible to obtain perfect correctness by allowing P to discard $\mathbf{Y}$ whenever $\|\mathbf{Y} + \mathbf{S}\| > B$. However, this would degrade the runtime of the prover.

To prove the rest of the properties, we start defining a simulator Σ. Such simulator operates in two modes, one to simulate the random oracle and the other one to simulate the protocol. It keeps a state *st* that contains the parameters of the proof and a list $\mathcal{Q}$ of all the queries to the random oracle.

$\Sigma(1, st, q)$**:** on input a query q to the random oracle H, it looks up q in the list $\mathcal{Q}$ of already queried values. If $\mathcal{Q}$ contains an entry (q, h), it outputs (h, st). Otherwise, it sets $h \xleftarrow{\$} \{0,1\}$, updates the list of queries $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(q, h)$, and outputs $(h, st\})$.

$\Sigma(2, st, x)$**:** on input $(\mathbf{A}, \mathbf{U})$, it samples $\mathbf{Z} \xleftarrow{\$} \mathcal{D}_{R,\sigma}^{m_2}$ (rejecting the samples that do not satisfy the norm bound; the probability that rejection is necessary is low though, thanks to Lemma 2.16), $c \xleftarrow{\$} \{0,1\}$, and sets $\mathbf{T} = \mathbf{AZ} - c\mathbf{U} \bmod q$. If there exists already an entry $(\mathbf{A}, \mathbf{U}, \mathbf{T}, h)$ in $\mathcal{Q}$ and $h \neq c$, the simulator discards $\mathbf{Z}, c, \mathbf{T}$ and starts over. Otherwise, it updates the list of queries setting $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{((\mathbf{A}, \mathbf{U}, \mathbf{T}), c)\}$ (if necessary), sets $\pi = (\mathbf{T}, c, \mathbf{Z})$, and it outputs (π, st).

Simulators Σ_1 and Σ_2' simply run $\Sigma(1, \cdot, \cdot)$ and $\Sigma(2, \cdot, \cdot)$ respectively. The simulator Σ_2 first checks that the instance $(\mathbf{A}, \mathbf{U}, \mathbf{S})$ it has been queried on is in the language, then runs $\Sigma(2, \mathbf{A}, \mathbf{U})$.

The *soundness error* is $1/2$ as a dishonest prover could guess the output c of H, sample $\mathbf{Z}$ from a Gaussian (rejecting the samples that do not satisfy the bound on the norm, like Σ_2 does), and set $\mathbf{T} = \mathbf{AZ} - c\mathbf{U} \bmod q$. If the dishonest prover guessed the challenge correctly (i.e., $c = H(\mathbf{A}, \mathbf{U}, \mathbf{T})$), the transcript $(\mathbf{T}, c, \mathbf{Z})$ is indistinguishable from a honestly generated one.

To prove the *zero-knowledge* property, we analyze the advantage of a PPT algorithm D in distinguishing the simulator from a honest prover. Thanks to rejection sampling (Lemma 3.7) the statistical distance between the distribution of the output π of Σ_2' and a honestly generated proof is $s2^{-\omega(\log nm_2)}/M$, where s is the number of queries made by the distinguisher to the proving oracle. The only way an adversary would be able to distinguish a simulated proof from an original one would be if Σ_2 aborts. This happens if the oracle answer to the query $(\mathbf{A}, \mathbf{U}, \mathbf{T})$ was already set different from h. The probability that the query was already made is computed in Lemma 5.3 in [Lyubashevsky, 2012], and is $s(s+o)2^{-nm_1+1}$, where again s is the number of queries made by the distinguisher to the proving oracle, and o is the number of queries made by the distinguisher to the random oracle. Hence, the advantage of a distinguisher in distinguishing the output of the simulator from honestly generated proofs is $s2^{-\omega(\log nm_2)}/M + s(s+o)2^{-nm_1+1}$. □

This protocol has two main limitations: a high soundness error, and no special soundness (nor simulation soundness). In fact, a soundness error of $1/2$ is usually too high, as a malicious prover can generate a proof for a false statement 50% of the times. However, to make it negligible it is enough to repeat the proof λ times to achieve a security of 2^λ. This creates big efficiency issues when this proof system is used in complex protocols, such as group signatures (cf. the discussion in Section 5.3.1).

The problem with special soundness is more complicated: indeed, given two transcripts having same commitments and different challenges, an extractor can

only get a witness $\mathbf{S}$ that satisfies the polynomial equation but has a larger norm than what is required by the original relation. To see this, consider two transcripts $(\mathbf{T}, c, \mathbf{Z})$ and $(\mathbf{T}, c', \mathbf{Z}')$ corresponding to the (public) instance $(\mathbf{A}, \mathbf{U})$ of $\mathscr{R}_{n,m_1,m_2,q,\beta}$. The transcripts are such that:

$$\begin{aligned}\mathbf{T} &= \mathbf{AZ} - c\mathbf{u} \bmod q \\ &= \mathbf{AZ}' - c'\mathbf{u} \bmod q \ ,\end{aligned}$$

hence,

$$\mathbf{A}(\mathbf{Z} - \mathbf{Z}') = (c' - c)\mathbf{U} \bmod q \ .$$

Given that $c, c' \in \{0,1\}$ and $c \neq c'$, the challenge difference is $|c' - c| = 1$. Assume w.l.o.g. that it is 1. Hence $\bar{\mathbf{Z}} = \mathbf{Z} - \mathbf{Z}'$ is a valid solution of the polynomial equation $\mathbf{AS} = \mathbf{U} \bmod q$ (where $\mathbf{S}$ is the unknown). However, the only guarantee the extractor gets about the norm of $\bar{\mathbf{Z}}$ is that $\|\bar{\mathbf{Z}}\| \leq \|\mathbf{Z}\| - \|\mathbf{Z}'\| \leq 2B$, where the bound B is set depending on the choice of the norm to $B_2 = 1.05\sigma\sqrt{mn}$ or $B_\infty = 8\sigma$ (both greater than β, as $\sigma = 12\beta$). Therefore, $\bar{\mathbf{Z}}$ is not a valid witness for $(\mathbf{A}, \mathbf{U})$ w.r.t. $\mathscr{R}_{n,m_1,m_2,q,\beta}$, but w.r.t the larger relation $\mathscr{R}_{n,m_1,m_2,q,2B}$.

Both these shortcomings are addressed the next sections. Section 3.3 contains a new definition of a relaxed version of Σ-protocol, where special soundness is relaxed so that Lyubashevsky's protocol satisfies it. Then, in Section 3.4 a more generic version of the proof system is introduced, with a larger challenge set to improve the soundness error.

3.3 Relaxed Non-Interactive Zero-Knowledge Proofs

We define *Relaxed Σ-protocol* and *Relaxed Non-Interactive Zero-Knowledge Proofs Systems* where the relaxed soundness definition guarantees the extraction of a witness from a wider language than the one used by an honest prover. Proofs with relaxed extracted notions have been used implicitly in previous work, e.g., for schemes based on discrete logarithms in group of unknown order [Camenisch et al., 2009; Camenisch and Lysyanskaya, 2003; Camenisch and Shoup, 2003; Xue et al., 2008] and some lattice-based schemes [Lyubashevsky, 2012; Benhamouda et al., 2015; Lyubashevsky and Neven, 2017].

Let $L \subseteq \{0,1\}^*$ be a NP language with witness relation $\mathscr{R}$, meaning $x \in L \iff \exists\ w \ : \ (x,w) \in \mathscr{R}$. Let $\bar{L} \supseteq L$ be a relaxed language with witness relation $\bar{\mathscr{R}} \supseteq \mathscr{R}$. We define *relaxed Σ-protocols* inspired by the original definitions by Cramer [1996] and Faust et al. [2012], but with a relaxed soundness condition that guarantees the extraction of a witness from $\bar{\mathscr{R}}$ rather than $\mathscr{R}$ (similar to Camenisch et al. [2009]).

Definition 3.9 (Relaxed Σ-protocol)**.** A *relaxed Σ-protocol* $\Sigma = (\mathsf{P}, \mathsf{V})$ for relations $(\mathscr{R}, \bar{\mathscr{R}})$ is a three-round public-coin interactive proof system where $\mathsf{P} = (\mathsf{P}_0, \mathsf{P}_1)$ and $\mathsf{V} = (\mathsf{V}_0, \mathsf{V}_1)$ are couples of PPT algorithms that, on top of the standard correctness and honest-verifier zero-knowledge (HVZK) properties recalled in Definition 3.3, satisfy the following property:

Relaxed Special Soundness. There exists an efficient algorithm E, called special extractor, that given two accepting conversations (α, β, γ) and $(\alpha, \beta', \gamma')$ for language member $\bar{x} \in \bar{L}$ where $\beta \neq \beta'$, computes $\bar{w} \leftarrow \mathsf{E}(\bar{x}, \alpha, \beta, \gamma, \beta', \gamma')$ such that $(\bar{x}, \bar{w}) \in \bar{\mathscr{R}}$.

Remark that relaxed Σ-protocols are relaxed proofs of knowledge, as the knowledge extractor extracts from P a pair (x, w) in $\bar{\mathscr{R}}$ (the proof is a straightforward adaptation to relaxed protocols of the proof of Theorem 1 in [Damgård, 2002]).

Similarly to standard Σ-protocols, a relaxed Σ-protocol (P, V) can be turned into a *relaxed* non-interactive zero-knowledge (NIZK) proof system $(\mathsf{P}^{H_c}, \mathsf{V}^{H_c})$ using the Fiat-Shamir transform [Fiat and Shamir, 1987] that computes the second round $\beta \leftarrow H_c(x, \alpha)$, where α is the first round of the proof and H_c is a random oracle.

Definition 3.10 (Relaxed NIZK)**.** A *relaxed NIZK proof system* $(\mathsf{P}^{H_c}, \mathsf{V}^{H_c})$ in the Random Oracle Model for relations $(\mathscr{R}, \mathscr{R}')$ is couple of PPT algorithms that satisfy the standard correctness and non-interactive zero-knowledge properties recalled in Definition 3.4, as well as the following soundness property:

Relaxed Simulation Soundness. There exists a PPT simulator $\Sigma = (\Sigma_1, \Sigma_2)$ that simulates random oracle responses (Σ_1) as well as NIZK proofs (Σ_2), including for members $x \notin L$, such that for all PPT adversaries A,

$$\Pr\left[\mathsf{V}^{\Sigma_1}(x^*, \pi^*) = 1 \wedge x^* \notin \bar{L} \wedge (x^*, \pi^*) \notin \mathscr{Q} \ : \ (x^*, \pi^*) \xleftarrow{\$} \mathsf{A}^{\Sigma}(1^\lambda)\right]$$

is negligible, where $\mathscr{Q}$ is the set of tuples (x, π) such that A queried x to Σ_2 and obtained π as response.

Faust et al. [2012] proved that the Fiat-Shamir transform of an HVZK Σ-protocol with *quasi-unique responses* yields an unbounded non interactive zero-knowledge protocol in the Random Oracle Model. Since the Σ-protocols we consider do not always have quasi-unique responses, we suggest an alternative construction from one-time signature (OTS) schemes (the definition of OTS can be found in Section 2.2.4).

Let (P,V) be an interactive protocol and $(\mathsf{OTSGen},\mathsf{OTSSign},\mathsf{OTSVf})$ a OTS scheme. We construct a non-interactive zero-knowledge (NIZK) proof system $(\mathsf{P}^{H_c},\mathsf{V}^{H_c})$ through the Fiat-Shamir transform using a random oracle H_c with range equal to the space of the verifier's coins. The proving algorithm $\mathsf{P}^{H_c}(x,w)$ computes a proof π by choosing random coins ρ, generating a OTS key pair $(otssk, otsvk) \leftarrow \mathsf{OTSGen}(1^\lambda)$, and computing $\alpha \leftarrow \mathsf{P}_0(x,w;\rho)$. It determines the challenge as $\beta \leftarrow H_c(x,\alpha,otsvk)$ and finally computes $\gamma \leftarrow \mathsf{P}_1(x,w,\alpha,\beta;\rho)$ and signs the whole transcript as $\sigma \leftarrow \mathsf{OTSSign}(otssk,(x,\alpha,\beta,\gamma))$. The proof is $\pi = ((\alpha, otsvk),\beta,(\gamma,\sigma))$. Verification $\mathsf{V}^{H_c}(x,\pi)$ checks that $\beta = H_c(x,\alpha,otsvk)$, that $\mathsf{V}_1(x,\alpha,\beta,\gamma) = 1$, and that $\mathsf{OTSVf}(otsvk,\sigma,(x,\alpha,\beta,\gamma)) = 1$. We first rephrase the non-triviality definition for identification schemes due to Abdalla et al. [2002] in terms of Σ-protocols.

Definition 3.11 (Non-trivial min-entropy of commitment)**.** Let λ be the security parameter, let $\mathsf{Coins}(\lambda)$ the set of coins used by the prover, and let $A(x,w) = \{\mathsf{P}_0(x,w;\rho) \ : \ \rho \leftarrow \mathsf{Coins}(\lambda)\}$ for any $(x,w) \in \mathscr{R}$. The min-entropy of (P,V) is defined as $\varepsilon(\lambda) = \min_{(x,w)\in\mathscr{R}} \log(1/\mu(x,w))$, where $\mu(x,w)$ is the maximum probability that the first round of the protocol takes on a particular value, i.e. $\mu(x,w) = \max_{\alpha\in\mathsf{A}(x,w)} \Pr[\mathsf{P}_0(x,w;\rho) = \alpha \ : \ \rho \leftarrow \mathsf{Coins}(\lambda)]$. We say that (P,V) is *non-trivial* if $\varepsilon(n) = \omega(\log(\lambda))$.

Adding non-triviality and a OTS to a relaxed Σ-protocol modified according to the Fiat-Shamir heuristic yields a NIZK proof. The proof follows closely the proofs of Theorem 1 and 2 in [Faust et al., 2012].

Theorem 3.12. *Let* (P,V) *be a non-trivial relaxed* Σ*-protocol for a NP language* L*. Let* H_c *be a hash function with range equal to the space of the verifier's coins, modeled as a random oracle and let* $(\mathsf{OTSGen},\mathsf{OTSSign},\mathsf{OTSVf})$ *be a strongly unforgeable one-time signature scheme. Then the proof system* $(\mathsf{P}^{H_c},\mathsf{V}^{H_c})$*, derived from* (P,V) *as described previously, is a relaxed NIZK in the Random Oracle Model.*

Proof. Correctness is straightforward. The proof of NIZK property is a modification of the proof of Theorem 1 in [Faust et al., 2012], and holds also for classical Σ-protocols. The simulator works as follows:

- To answer a query q to Σ_1, $\Sigma(1,st,q)$ samples a lookup table $\mathscr{T}_H$ kept in state st and returns $\mathscr{T}_H(q)$. If $\mathscr{T}_H(q)$ is not defined, it sets it to a fresh random value (of the appropriate length).

- To answer a query x to Σ_2', $\Sigma(2,st,x)$ calls the HVZK simulator of (P,V) to obtain the transcript (α,β,γ). Then, it generates the keys of the OTS

$(otssk, otsvk) \leftarrow \mathsf{OTSGen}(1^\lambda)$. Finally, it signs the transcript running $\sigma \leftarrow \mathsf{OTSSign}(ssk, (\alpha, \beta, \gamma))$ and it updates $\mathcal{T}_H$ so that $\beta = \mathcal{T}_H(x, \alpha, otsvk)$. If $\mathcal{T}_H$ was already defined on this input it returns $\perp$ and aborts.

By construction, the only case in which a distinguisher D can distinguish with non-negligible probability the simulation from the real-world protocol is when the simulator Σ'_2 aborts. This can happen only if $(x, \alpha, otsvk)$ was already queried to Σ_1. Given that the protocol is non-trivial, the probability of an abort is upper-bounded by $2^{-\varepsilon(\lambda)}$, that is negligible in λ. Hence:

$$\begin{aligned}
&\mathsf{Pr}\left[\mathsf{D}^{\Sigma_1(\cdot),\Sigma'_2(\cdot,\cdot)}(1^\lambda) = 1\right] = \\
&= \mathsf{Pr}\left[\mathsf{D}^{\Sigma_1(\cdot),\Sigma'_2(\cdot,\cdot)}(1^\lambda) = 1 \ : \ \Sigma'_2 \text{ aborts}\right]\mathsf{Pr}\left[\Sigma'_2 \text{ aborts}\right] + \\
&\quad + \mathsf{Pr}\left[\mathsf{D}^{\Sigma_1(\cdot),\Sigma'_2(\cdot,\cdot)}(1^\lambda) = 1 \ : \ \Sigma'_2 \text{ does not abort}\right]\mathsf{Pr}\left[\Sigma'_2 \text{ does not abort}\right] \\
&= 0 + \mathsf{Pr}\left[\mathsf{D}^{H_c(\cdot),\mathsf{P}^{H_c}(\cdot,\cdot)}(1^\lambda) = 1\right]\mathsf{Pr}(\Sigma'_2 \text{ does not abort}) \\
&\leq \mathsf{Pr}\left[\mathsf{D}^{H_c(\cdot),\mathsf{P}^{H_c}(\cdot,\cdot)}(1^\lambda) = 1\right](1 - 2^{-\varepsilon(n)}) \ .
\end{aligned}$$

Therefore the difference in (3.1) is bounded by $2^{-\varepsilon(\lambda)}$ that is negligible in λ.

Finally, we prove relaxed simulation soundness. Let A be an adversary that breaks the relaxed simulation soundness property with probability ϵ. Let Σ be the simulator previously described. At the end of the querying phase, the adversary outputs a valid forgery (x^*, π^*) that was not in the set $\mathcal{Q}$ of pairs queried to the simulator. Let the forged proof be $\pi^* = ((\alpha^*, otsvk^*), \beta^*, (\gamma^*, \sigma^*))$. Without loss of generality, we can assume that A queried $(x^*, \alpha^*, otsvk)$ to Σ_1.

Now, we can have two cases: either A learned α^* and $otsvk^*$ by querying x^* to the simulator Σ'_2, or it did not. We indicate by query the case in which A queried for x^* and $\overline{\mathsf{query}}$ the other case. Given that these two events are mutually exclusive, the probability that A wins the forgery game can be split in:

$$\mathrm{Pr}[\mathsf{A} \text{ wins}] = \mathrm{Pr}[\mathsf{A} \text{ wins } \wedge \ \mathsf{query}] + \mathrm{Pr}[\mathsf{A} \text{ wins } \wedge \ \overline{\mathsf{query}}] \ .$$

We treat each case separately. In particular, we build two reductions $\mathsf{B}_{\mathsf{s-unf}}$ and $\mathsf{B}_{\mathsf{sound}}$ that exploit A to break either the strong-unforgeability of the OTS or the relaxed soundness of the relaxed Σ-protocol.

Assume that A wins and query happens. The reduction $\mathsf{B}_{\mathsf{s-unf}}$ has access to an oracle $\mathcal{O}_{OTS}$ that on input a message outputs a pair of keys and a signature on the message. To break strong-unforgeability of the OTS scheme, $\mathsf{B}_{\mathsf{s-unf}}$ should output a different signature valid with respect to the same public key. To do that, it first guesses the query j' to Σ'_2 for which A will forge. Then, it works as follows:

Queries to Σ_1. It answers to queries to Σ_1 and fills the table $\mathcal{T}_H$ as the real simulator would have done.

Queries to Σ_2'. Upon receiving the j-th query:

- If $j \neq j'$, $\mathsf{B}_{\mathsf{s-unf}}$ answers and stores the pair query-answer in the list $\mathcal{Q}$ as the real simulator Σ would do.
- If $j = j'$, $\mathsf{B}_{\mathsf{s-unf}}$ generates the transcript $(\alpha', \beta', \gamma')$ as the real HVZK simulator of the relaxed Σ-protocol (P, V) would do. It triggers $\mathcal{O}_{OTS}$ to receive the key pair $(otssk', otsvk')$. Then, it queries $\mathcal{O}_{OTS}$ with $(x*, \alpha', \beta', \gamma')$ to obtain a signature σ'. $\mathsf{B}_{\mathsf{s-unf}}$ sets $\mathcal{T}_H(x^*, \alpha', otsvk') = \beta'$ and updates $\mathcal{Q}$ accordingly. If the value was already assigned, $\mathsf{B}_{\mathsf{s-unf}}$ aborts and outputs $\perp$. Otherwise, it stores $(x^*, (\alpha', otsvk', \beta', \gamma', \sigma'))$ in $Queries$ and outputs $\pi' = ((\alpha', otsvk'), \beta', (\gamma', \sigma'))$.

At the end of the querying phase, A outputs a valid forgery (x^*, π^*). Remark that, for it to be a valid forgery, it should hold $\pi' \neq \pi^*$.

The reduction $\mathsf{B}_{\mathsf{s-unf}}$ parses the response as (γ^*, σ^*), where σ^* is a signature on the message $msg^* = (x^*, \alpha^*, \beta^*, \gamma^*)$. Then it recovers from $\mathcal{Q}$ the query $(x^*, (\alpha', otsvk', \beta', \gamma', \sigma'))$. Let $msg' = (x^*, \alpha', \beta', \gamma')$. Given that query happened, it holds that $\alpha' = \alpha^*$ and $otsvk^8 = otsvk'$, and subsequently $\beta' = \beta^*$. Hence, we can have two cases:

- If $\gamma' = \gamma^*$ (i.e., if $msg' = msg^*$), then for π^* to be a valid forgery it should be $\sigma^* \neq \sigma'$. Hence (msg^*, σ) is a valid forgery.
- If $\gamma' \neq \gamma^*$ then σ^* is a signature on a different message with respect to the key $otsvk$. Hence (msg^*, σ^*) is again a valid forgery.

Finally, observe that as before the probability that $\mathsf{B}_{\mathsf{s-unf}}$ aborts when simulating Σ_2' is negligible as the scheme is non-trivial. Hence, $\mathsf{B}_{\mathsf{s-unf}}$ outputs (msg^*, σ^*) (and breaks strong-unforgeability) with probability $\mathsf{Pr}[\mathsf{A}\ \mathsf{wins} \wedge \mathsf{query}] \cdot (1 - 2^{-\varepsilon(\lambda)})\frac{1}{n_q}$, where n_q is the number of queries that A can ask to the simulator. Therefore it holds $\mathsf{Pr}[\mathsf{A}\ \mathsf{wins} \wedge \mathsf{query}] \sim \nu(\lambda)$.

Now, consider the case in which A wins and it did compute α^* or $otsvk^*$ by itself. We build $\mathsf{B}_{\mathsf{sound}}$ as follows. It first chooses an index $j' \in \{1, \ldots, n_q\}$ uniformly at random and then it implements the simulator as follows:

Queries to Σ_1. Upon receiving query $(x_j, \alpha_j, otsvk_j)$:

- If $j \neq j'$, $\mathsf{B}_{\mathsf{sound}}$ answers the query and fills the table $\mathcal{T}_H$ as the real simulator would have done.
- If $j = j'$, $\mathsf{B}_{\mathsf{sound}}$ runs the relaxed Σ-protocol with the honest verifier V for statement $x_{j'}$ with commitment $\alpha_{j'}, otsvk_{j'}$. Upon receiving the

challenge β from V, it programs $\mathscr{T}_H(x_j, \alpha_j, otsvk_{j'}) = \beta$. Then it outputs β as an answer to the query.

$\mathsf{B}_{\mathsf{sound}}$ answers as the real simulator would to the queries of type $(x, \mathfrak{a})$.

Queries to Σ_2'. To answer these queries, the reduction honestly executes the NIZK simulator. Upon receiving query x, $\mathsf{B}_{\mathsf{sound}}$ runs $\mathsf{OTSGen}(1^\lambda)$ to obtain a key pair $(otssk, otsvk)$. Then, it runs $\mathsf{OTSSign}(otssk, (x, \alpha, \beta, \gamma))$ to obtain a signature σ. $\mathsf{B}_{\mathsf{sound}}$ sets $\mathscr{T}_H(x, \alpha, otsvk) = \beta$ and updates $\mathscr{Q}$ accordingly. If the value was already assigned, $\mathsf{B}_{\mathsf{sound}}$ aborts and outputs $\perp$. Otherwise, it stores $(x, (\alpha, otsvk, \beta, \gamma, \sigma))$ in $\mathscr{Q}$ and outputs $\pi = ((\alpha, otsvk), \beta, (\gamma, \sigma))$.

When A outputs the forgery $(x^*, (\alpha^*, otsvk^*, \beta^*, \gamma^*, \sigma^*))$ the simulator parses it and sends γ^* to the V. The success probability is bounded by:

$$\Pr[\mathsf{B}_{\mathsf{sound}} \text{ wins}] = \Pr[\mathsf{A} \text{ wins } \wedge \overline{\mathsf{query}}] \frac{1}{n_q} \ .$$

Thus,

$$\Pr[\mathsf{A} \text{ wins } \wedge \overline{\mathsf{query}}] = \Pr[\mathsf{B}_{\mathsf{sound}} \text{ wins}] n_q \leq \nu(\lambda)$$

and hence we have that:

$$\epsilon = \Pr[\mathsf{A} \text{ wins}] = \Pr[\mathsf{A} \text{ wins } \wedge \mathsf{query}] + \Pr[\mathsf{A} \text{ wins } \wedge \overline{\mathsf{query}}] \leq \nu(\lambda) \ .$$

□

3.4 Relaxed Non-Interactive Zero-Knowledge Proofs for $\mathscr{R}_{\text{RSIS}}$

With the new relaxed definitions, the protocol By Lyubashevsky can now be proved to be a (relaxed) NIZK proof. We present a generalized version of Lyubashevsky's "Fiat-Shamir with aborts" technique [Lyubashevsky, 2009, 2012]. This yields a relaxed Σ-protocol for the languages $(L, \bar{L})$ associated to the following relations:

$$\mathscr{R} = \left\{ ([\mathbf{A} \ -\mathbf{U}], \mathbf{0}_{m+1}; [\mathbf{S}; \mathbf{1}]) \in R_q^{\ell \times (m+1)} \times R_q^{1 \times (m+1)} \times R^m \times \{\mathbf{1}\} : \begin{array}{c} \mathbf{AS} = \mathbf{U} \bmod q, \\ \|[\mathbf{S}; \mathbf{1}]\| \leq B \end{array} \right\} \subseteq \mathscr{R}_{n,\ell+1,m+1,q,B}$$

$$\bar{\mathscr{R}} = \left\{ ([\mathbf{A} \ -\mathbf{U}], \mathbf{0}_{m+1}; [\bar{\mathbf{S}}; \bar{\mathbf{c}}]) \in R_q^{\ell \times (m+1)} \times R_q^{1 \times (m+1)} \times R^m \times \bar{\mathscr{C}} : \begin{array}{c} \mathbf{A}\bar{\mathbf{S}} = \mathbf{U}\bar{\mathbf{c}} \bmod q, \\ \|[\bar{\mathbf{S}}; \bar{\mathbf{c}}]\| \leq \bar{B} \end{array} \right\} \subseteq \mathscr{R}_{n,\ell+1,m+1,q,\bar{B}}$$

for some positive constants $B, \bar{B}$ such that $B \leq \bar{B}$ (that will be indicated as $B_2, \bar{B}_2$ if the relation is defined w.r.t. the Euclidean norm, and $B_\infty, \bar{B}_\infty$ if the relation

is defined w.r.t. the infinity norm), and for a subset $\bar{\mathscr{C}} \subseteq R_q$ containing small elements $\bar{\mathbf{c}}$ (a more precise definition is be given later). The set $\bar{\mathscr{C}}$ is in fact the relaxed version of the challenge set used in the (relaxed) Σ-protocol. All of these parameters are denoted by *ppar*, and we always implicitly assume that they are given as input to both the prover and the verifier.

The generic Σ-protocol $\mathsf{r}\Sigma = (\mathsf{P}, \mathsf{V})$ (cf. Figure 3.3) is essentially Lyubashevsky's protocol (cf. Figure 3.2) where challenges are sampled from a larger challenge set $\mathscr{C} \subseteq \{\mathbf{c} \in R_q \,:\, \|\mathbf{c}\|_\infty \leq 1\}$ containing polynomials with coefficients in $\{0, \pm 1\}$. The algorithms $\mathsf{P} = (\mathsf{P}_0, \mathsf{P}_1)$ and $\mathsf{V} = (\mathsf{V}_0, \mathsf{V}_1)$ are described in the following. Let $w = [\mathbf{S}; \mathbf{1}]$ be a witness for the instance $x = ([\mathbf{A} \; -\mathbf{U}], \mathbf{0}_{m+1})$ of relation $\mathscr{R}$.

- The prover P_0 samples a masking vector $\mathbf{Y} \xleftarrow{\$} \mathscr{D}^m_{R,\sigma}$ (the value of the standard deviation σ is again derived from Lemma 3.7 and is addressed later), and sends $\mathbf{T} = \mathbf{AY} \bmod q$ to V.

- Next, V_0 samples a challenge $\mathbf{c} \in \mathscr{C}$ and sends it to P.

- The prover algorithm P_1 constructs $\mathbf{Z} = \mathbf{Y} + \mathbf{cS} \bmod q$ and, depending on rejection sampling (cf. Lemma 3.7), either aborts or sends it to V.

- The verifier accepts if $\mathbf{AZ} - \mathbf{cU} = \mathbf{T} \bmod q$ and $\|\mathbf{Z}\|_2 \leq 1.05\sigma\sqrt{nm} =: B_2$, $\|\mathbf{Z}\|_\infty \leq 8\sigma =: B_\infty$.

The zero-knowledge property is guaranteed by rejection sampling, and the choice of parameters should reflect that. To apply Lemma 3.7 the standard deviation to be $\sigma = 12T$, where T is a bound on the norm of $\mathbf{cS}$ obtained (for example) through Lemma 2.12 and $a > 0$. This guarantees that the prover outputs something with probability greater than $(1 - 2^{-100})/e^{289/288} \geq 0.36$ (cf. Theorem 4.6 in [Lyubashevsky, 2012]). The bounds on the norm of the witnesses for the relaxed relation $\bar{\mathscr{R}}$ are $\bar{B}_2 = 2B_2 = 2.1\sigma\sqrt{nm}$ and $\bar{B}_\infty = 2B_\infty = 16\sigma$. Finally, to guarantee that the extractor obtains elements in $\bar{L}$, the set $\bar{\mathscr{C}}$ should be such that $\{\mathbf{c}_1 - \mathbf{c}_2\}_{\mathbf{c}_1, \mathbf{c}_2 \in \mathscr{C}} \subseteq \bar{\mathscr{C}}$.

The choice of the challenge set $\mathscr{C}$ has to be handled with care, as it heavily influences the length of the proof transcript (i.e., the efficiency of the protocol), as the standard deviation σ depends on it. As we present protocols built from relaxed proofs with different challenge sets, we address how to choose the challenges in a separate section (cf. Section 3.4.1).

In the following theorem we prove that our protocol is in fact a non-trivial relaxed Σ-protocol.

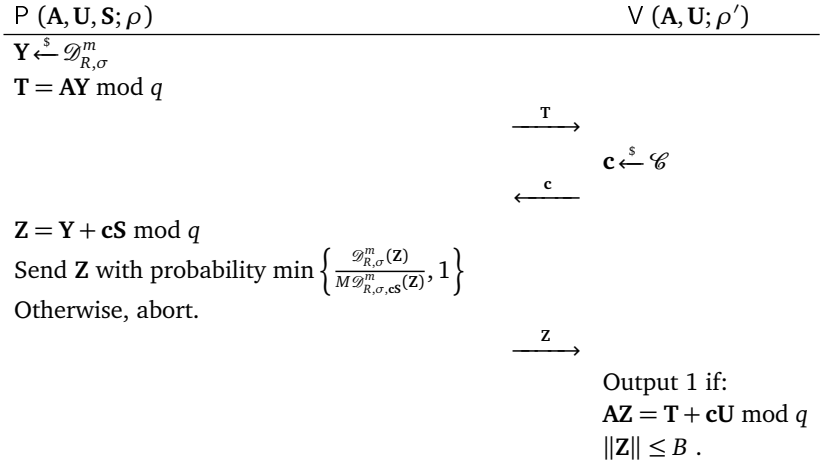

Figure 3.3. Generic Σ-protocol for lattice-based relations.

Theorem 3.13. *The protocol described above is a non-trivial relaxed Σ-protocol for relations $(\mathscr{R}, \bar{\mathscr{R}})$ with soundness error at least $\frac{1}{|\mathscr{C}|}$.*

Proof. Thanks to rejection sampling, $\mathbf{Z}$ is distributed as $\mathscr{D}^m_{R,\sigma}$. Hence, by Lemma 2.16, with high probability $\|\mathbf{Z}\|_2 \leq 1.05\sigma\sqrt{nm}$, $\|\mathbf{Z}\|_\infty \leq 8\sigma$, and correctness follows.

A simulator Σ for this proof can be constructed as it follows:

$\Sigma(\mathbf{A}, \mathbf{U}, \sigma, 1^\lambda)$:

Sample $\mathbf{Z} \xleftarrow{\$} \mathscr{D}^m_{R,\sigma}$ and $\mathbf{c} \xleftarrow{\$} \mathscr{C}$.

Set $\mathbf{T} := \mathbf{AZ} - \mathbf{cU} \bmod q$.

Output $(\mathbf{T}, \mathbf{c}, \mathbf{Z})$.

The distribution of the output is exactly the same of a honestly generated transcript, because rejection sampling guarantees that the distribution of honestly generated $\mathbf{Z}$ is at statistical distance $2^{-100}e^{-289/288}$ from $\mathscr{D}^m_{R,\sigma}$. From this simulator is also easy to see that a cheating prover is successful (i.e., can output a valid proof without knowing a witness $\mathbf{S}$) if it can correctly guess the challenge. Hence, the soundness error of this proof is at least $\frac{1}{|\mathscr{C}|}$.

To prove relaxed special soundness, we construct an extractor E. The extractor E receives as input a pair of valid transcripts $(\mathbf{T}, \mathbf{c}_1, \mathbf{Z}_1)$ and $(\mathbf{T}, \mathbf{c}_2, \mathbf{Z}_2)$, i.e., where $\mathbf{T} = \mathbf{AZ}_1 - \mathbf{c}_1\mathbf{U} = \mathbf{AZ}_2 - \mathbf{c}_2\mathbf{U}$ and the norms of $\mathbf{Z}_1, \mathbf{Z}_2$ satisfy the bounds stated by the verification algorithm. The extractor sets $\bar{\mathbf{S}} = \mathbf{Z}_1 - \mathbf{Z}_2$ and $\bar{\mathbf{c}} = \mathbf{c}_1 - \mathbf{c}_2$ It yields that $\mathbf{A}\bar{\mathbf{Z}} - \bar{\mathbf{c}}\mathbf{U} = \mathbf{0}_{1\times\ell}$. Then it outputs the pair $(\bar{\mathbf{S}}, \bar{\mathbf{c}})$. The norm of $\bar{\mathbf{S}}$ is bounded by $\|\bar{\mathbf{S}}\|_2 \leq \|\mathbf{Z}_1\|_2 + \|\mathbf{Z}_2\|_2 \leq 2B_2 = \bar{B}_2$ and $\|\bar{\mathbf{S}}\|_\infty \leq \|\mathbf{Z}_1\|_\infty + \|\mathbf{Z}_2\|_\infty \leq 2B_\infty = \bar{B}_\infty$, while $\bar{\mathbf{c}} \in \bar{\mathscr{C}}$ by definition of $\bar{\mathscr{C}}$. Therefore, $(\bar{\mathbf{S}}, \bar{\mathbf{c}}) \in \bar{\mathscr{R}}$.

Finally, we address non triviality. To prove it, it is enough to prove that

$$\max_{\mathbf{T}\in R_q^m} \mathsf{Pr}[\mathbf{AY} = \mathbf{T} \bmod q \ : \ \mathbf{Y} \xleftarrow{\$} \mathscr{D}_{R,\sigma}^m] \leq \frac{1}{2^{n-1}} \ , \tag{3.4}$$

as $n = O(\lambda)$. To do that, we exploit the min entropy of the Gaussian distribution. Indeed, consider the following probability, for fixed $\mathbf{A}$ and $\mathbf{T}$:

$$\mathsf{Pr}\left[\mathbf{AY} = \mathbf{T} \bmod q \ : \ \mathbf{Y} \xleftarrow{\$} \mathscr{D}_{R,\sigma}^m\right] \leq \max_i \mathsf{Pr}\left[\sum_{j=1}^m \mathbf{a}_{i,j}\mathbf{y}_j = \mathbf{t}_i \bmod q \ : \ \mathbf{Y} \xleftarrow{\$} \mathscr{D}_{R,\sigma}^m\right] ,$$

where by $\mathbf{a}_{i,j}$ and $\mathbf{t}_i$ we denote the components of $\mathbf{A}$ and $\mathbf{T}$ respectively. The inequality follows from the fact that, given two events E_1 and E_2, it holds

$$\mathsf{Pr}[E_1 \wedge E_2] \leq \max_{i=1,2} \mathsf{Pr}[E_i] \ .$$

We analyze the probability that a linear combination of polynomials $\mathbf{y}_j$ sampled from a Gaussian distribution (as defined in Section 2.3.3) with fixed polynomials $\mathbf{a}_j$ results in a given value $\mathbf{t}$:

$$\begin{aligned}
&\mathsf{Pr}\left[\sum_{j=1}^m \mathbf{a}_j\mathbf{y}_j = \mathbf{t} \bmod q \ : \ \mathbf{Y} \xleftarrow{\$} \mathscr{D}_{R,\sigma}^m\right] = \mathsf{Pr}\left[\mathbf{a}_1\mathbf{y}_1 = \mathbf{t} - \sum_{j=2}^m \mathbf{a}_j\mathbf{y}_j \bmod q \ : \ \mathbf{Y} \xleftarrow{\$} \mathscr{D}_{R,\sigma}^m\right] \\
&\leq \max_{\mathbf{v}\in R_q} \mathsf{Pr}\left[\mathbf{ay} = \mathbf{v} \bmod q \ : \ \mathbf{y} \xleftarrow{\$} \mathscr{D}_{R,\sigma}^m\right] \\
&= \max_{V\in\mathbb{Z}_q^n} \mathsf{Pr}\left[\begin{bmatrix} a_0 & -a_{n-1} & \dots & -a_1 \\ a_1 & a_0 & \dots & -a_2 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & \dots & a_0 \end{bmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{n-1} \end{pmatrix} \bmod q \ : \ Y \xleftarrow{\$} \mathscr{D}_{\mathbb{Z},\sigma}^n\right] \\
&\leq \max_{V\in\mathbb{Z}_q^n} \mathsf{Pr}\left[Y \ : \ Y \xleftarrow{\$} \mathscr{D}^n_{\mathscr{L}_q^\perp(A),V,\sigma}\right] \\
&\leq \frac{1}{2^{n-1}} \ ,
\end{aligned}$$

where the last inequality follows from [Peikert and Rosen, 2006, Lemma 2.10] (the hypothesis on the standard variation σ holds always in our applications, as it is needed to use the trapdoor in Lemma 2.17). From the last inequality, the inequality in Equation 3.4 follows trivially. □

Therefore, it follows from Theorem 3.12 that adding a OTS and using Fiat-Shamir heuristic we can obtain a NIZK proof. As OTS we choose the Lamport signature (see Section 2.2.4).

A simulator Σ_{NIZK} for this NIZK proof can be constructed exactly as in the case of the original protocol (cf. proof of Theorem 3.8). Let st be the internal state of the simulator containing the parameters of the proof and $\mathscr{Q}$ be a list of all the queries to the random oracle.

$\Sigma_{NIZK}(1, st, q)$**:** on input a query q to the random oracle H, it looks up q in the list $\mathscr{Q}$ of already queried values. If $\mathscr{Q}$ contains an entry $(q, \mathbf{c})$, it outputs $(\mathbf{c}, st')$. Otherwise, it sets $\mathbf{c} \xleftarrow{\$} \mathscr{C}$, updates the list of queries $\mathscr{Q} \leftarrow \mathscr{Q} \cup \{(q, \mathbf{c})\}$, and outputs $(\mathbf{c}, st')$, where st' is the updated internal state of the simulator.

$\Sigma_{NIZK}(2, st, x)$**:** on input $(\mathbf{A}, \mathbf{U})$, it samples $\mathbf{Z} \xleftarrow{\$} \mathscr{D}^m_{R,\sigma}$ (rejecting the samples that do not satisfy the norm bound; the probability that rejection is necessary is small though, thanks to Lemma 2.16), $\mathbf{c} \xleftarrow{\$} \mathscr{C}$, and sets $\mathbf{T} = \mathbf{AZ} - \mathbf{cU} \bmod q$. Then it generates keys for the OTS signature, $(otssk, otsvk) \leftarrow \mathsf{OTSGen}(1^\lambda)$. If there exists already an entry $((\mathbf{A}, \mathbf{U}, \mathbf{T}, otsvk), \mathbf{c}')$ in $\mathscr{Q}$ and $\mathbf{c}' \neq \mathbf{c}$, the simulator discards $\mathbf{Z}, \mathbf{c}, \mathbf{T}$ and starts over. Otherwise, it updates the list of queries setting $\mathscr{Q} \leftarrow \mathscr{Q} \cup \{((\mathbf{A}, \mathbf{U}, \mathbf{T}, otsvk), \mathbf{c})\}$ (if necessary), it sets the transcript to be $\pi = (\mathbf{T}, \mathbf{c}, \mathbf{Z})$, and it produces a one-time signature on the transcript $ots \leftarrow \mathsf{OTSSign}(otssk, \pi)$. Finally, the simulator outputs (π, ots, st'), where st' is the updated internal state of the simulator.

Simulators $\Sigma_{NIZK,1}$ and $\Sigma'_{NIZK,2}$ simply run $\Sigma_{NIZK}(1, \cdot, \cdot)$ and $\Sigma_{NIZK}(2, \cdot, \cdot)$ respectively. The simulator $\Sigma_{NIZK,2}$ first checks that the instance $(\mathbf{A}, \mathbf{U}, \mathbf{S})$ it has been queried on is in the language, then runs $\Sigma_{NIZK}(2, \mathbf{A}, \mathbf{U})$.

Remark that non-triviality holds for a standard deviation $\sigma \geq \omega(\sqrt{\log(nm)})$ as a direct consequence of a result by Peikert and Rosen [2006]. Indeed, they proved that with this assumption on the standard deviation it holds that $\mathscr{D}_{\mathbb{Z}^{nm},\sigma} \leq \frac{1+\epsilon}{1-\epsilon} 2^{-nm}$, hence the min-entropy function is greater than n (cf. [Peikert and Rosen, 2006], Lemma 2.11) and Theorem 3.12 applies. Our proof is better though, as it does not require a large standard deviation.

Remark 7. In this section we have described a relaxed NIZK proof for a very generic relation $\mathscr{R}$ over lattices where there are no assumptions on $\mathbf{A}$ and $\mathbf{U}$. However, when using this protocol in combination with other primitives, we have to impose the additional condition that $\mathbf{A}$ and $\mathbf{U}$ have a particular structure (e.g., they model the verification equation of a signature, or the encryption procedure of some encryption scheme). Obviously, this does not affect the protocol itself, as such relations are subsets of the more generic relation we have considered so far.

3.4.1 The Choice of the Challenge Set

The choice of the challenge set is very important, as it directly influences the length of a proof. Indeed, as we already mentioned, to make the soundness error negligible, a proof has to be repeated multiple times. Hence, the length of the proof does not only depend on the length of a single transcript, but on the number of repetitions needed as well. In the original paper by Lyubashevsky [2012], the challenges were chosen to be binary polynomials of small degree. This guarantees two things: first, that the product $\mathbf{cS}$ of a challenge and the witness has small norm, hence the standard deviation σ bounding the norm of the prover's response is small, and second that the soundness error is small enough not to require repetitions.

In our constructions, we wanted to maintain these advantages, while making the proof efficient enough to result in relatively short proofs when combined with other lattice-based building blocks. On top of this, we wanted to add to the challenge set a specific structure, that allows to prove that the witness satisfies some algebraic properties (cf. Section 3.4.2) Hence, we choose the challenge set to be a subset of a particular subset of a subring of R_q: $\mathcal{C} \subseteq R_3^{(2^{K_c})}$ for some $K_c > 0$ (that will be set when building the protocols), i.e., challenges are polynomials with coefficients in $\{0, \pm 1\}$ as in the original protocol, but that only have 2^{K_c} nonzero coefficients in specific positions (cf. Section 2.3.2 for the definition of the subring $R^{(K)}$). Then, the set of relaxed challenges is $\bar{\mathcal{C}} \subseteq R_5^{(2^{K_c})}$.

3.4.2 Proving Knowledge of Bounded-Degree Secrets in a Subring

In our construction of an anonymous attribute token scheme, we will use the previous protocol to let a prover prove knowledge of a $[\mathbf{m}; \mathbf{s}]$ where $\mathbf{m}$ is a small element in a subring $R_q^{(2^{K_m})}$ of R_q and with degree $\deg(\mathbf{m}) < d$ for some constant $d < n$. The fact that $\mathbf{m}$ is in the subring can be proved by exploiting the subring structure. Indeed, the challenge space $\mathcal{C} = R_3^{(2^{K_c})}$ is a subset of $R_q^{(2^{K_m})}$ when $K_m \geq K_c$. To have the largest possible set of challenges, we set $K_c = K_m$. By also sampling the first component $\mathbf{y}_m$ of the "masking" vector $\mathbf{Y} = [\mathbf{y}_m; \mathbf{y}_s]$ from the subring $R^{(2^{K_m})}$, the output vector $[\mathbf{z}_m; \mathbf{z}_s] = [\mathbf{y}_m; \mathbf{y}_s] + \mathbf{c}[\mathbf{m}; \mathbf{s}]$ will be such that $\mathbf{z}_m \in R_q^{(2^{K_m})}$. Sampling a discrete Gaussian distribution from the subring $R_q^{(2^{K_m})}$ can be done by sampling from $\mathcal{D}_{\mathbb{Z}^{2^{K_m}}, \sigma}$ and mapping the 2^{K_m} coordinates into the non-zero coefficients of the polynomials. The zero-knowledge property remains guaranteed by rejection sampling.

Proving that $\mathbf{m}$ is of degree strictly less than $d < n$ can be done by carefully choosing the challenge set and the domain of the masking vector. In particular,

if $\deg(\mathbf{m}) \leq d_m$ and challenges are chosen to be polynomials of degree d_c such that $d_c + d_m < d$, then $\deg(\mathbf{mc}) < d$. Letting the prover sample the masking vector $\mathbf{y}_m$ from the polynomials of degree less than d and applying rejection sampling as usual preserves the zero-knowledge property when computing $\mathbf{z}_m = \mathbf{mc} + \mathbf{y}_m$. By letting the verifier additionally check that $\deg(\mathbf{z}_m) < d$, the extractor is guaranteed to be able to extract a witness $\bar{\mathbf{m}} = \mathbf{z}_{m,1} - \mathbf{z}_{m,2}$ of degree strictly less than d.

Note that sampling a discrete Gaussian distributions of polynomials of degree at most $d-1$ from the subring $R^{(2^{K_m})}$ can be done by sampling from $\mathcal{D}_{\mathbb{Z}^m,\sigma}$ for $m = \lfloor (d-1)n/2^{K_m} \rfloor$ and mapping coordinates to coefficients. To have a clearer notation, we define $\mathcal{Y}_d$ to be the set of elements in the subring $R^{(2^{K_m})}$ with degree at most $d-1$, so that the distribution of the full masking vector $\mathbf{Y}$ can be written as $\mathcal{D}_{\mathcal{Y}\times R,\sigma}$.

A possible drawback of this technique, however, is that it shrinks the size of the challenge space, so that the proof may have to be repeated several times to obtain soundness.

3.5 Relaxed Commitment

Relaxed Σ-protocols can be used to prove knowledge of a message and a valid opening of a given commitment. For this composition to be meaningful, the commitment should allow for "relaxed opening", i.e., the opening algorithm should accept messages and opening information that would not be accepted as input, respectively produced as output, by the honest commitment algorithm. This ensures that the output of a successful extractor is meaningful in the context of the commitment.

We define a correspondingly relaxed binding property that divides messages into classes and only considers binding attacks for messages belonging to different classes. We also present an instantiation of such commitment scheme over lattices. The peculiar structure of our relaxed commitment scheme allows us to build privacy-preserving signatures combining only such commitment with the relaxed Σ-protocol previously defined and with the relaxed signature defined in Section 3.6.

3.5.1 Commitments

A commitment scheme C is a triple of PT algorithms $(\mathsf{ParGen_c}, \mathsf{Commit}, \mathsf{OpenVf})$ such that

Parameters Generation. On input the security parameter λ and the message space $\mathscr{U}$, $\mathsf{ParGen}_c(\mathscr{U}, 1^\lambda)$ outputs the parameters *cpar* of the scheme.

Commitment Generation. On input *cpar* and a message $\mu \in \mathscr{U}$, $\mathsf{Commit}(cpar, \mu)$ outputs a commitment c to μ with the corresponding opening information o.

Opening. On input a commitment c, opening information o and a message μ, $\mathsf{OpenVf}(cpar, c, \mu, o)$ outputs 1 if o is a valid opening information for c w.r.t. μ, and 0 otherwise.

A commitment scheme must satisfy three standard properties: correctness, the hiding property, and the binding property.

Definition 3.14 (Correctness)**.** A commitment scheme C is correct if for all $\mu \in \mathscr{U}$ and for all λ:

$$\Pr\left[\mathsf{OpenVf}(cpar, c, \mu, o) = 1 \ : \ \begin{array}{c} cpar \leftarrow \mathsf{ParGen}_c(\mathscr{U}, 1^\lambda), \\ (c, o) \leftarrow \mathsf{Commit}(cpar, \mu) \end{array}\right] \geq 1 - \nu(\lambda) \ .$$

The hiding property ensures that a commitment value does not reveal information about the committed message.

Definition 3.15 (Hiding)**.** A commitment scheme C is hiding if for all PPT algorithms $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$

$$\left|\Pr\left[b' = b \ : \ \begin{array}{c} cpar \leftarrow \mathsf{ParGen}_c(\mathscr{U}, 1^\lambda),\ (\mu_0, \mu_1, st) \leftarrow \mathsf{A}_1(cpar), \\ b \xleftarrow{\$} \{0,1\},\ (c, o) \leftarrow \mathsf{Commit}(cpar, \mu_b),\ b' \leftarrow \mathsf{A}_2(st, c) \end{array}\right] - \frac{1}{2}\right|$$

is negligible.

Finally, the commitment should be binding, meaning that an adversary cannot open a commitment to different values.

Definition 3.16 (Binding)**.** A commitment scheme C is binding if for all PPT algorithms A, for all security parameters λ,

$$\Pr\left[\begin{array}{c} \mathsf{OpenVf}(cpar, c, \mu_0, o_0) = 1 \\ \wedge\, \mathsf{OpenVf}(cpar, c, \mu_1, o_1) = 1 \end{array} \ : \ \begin{array}{c} cpar \leftarrow \mathsf{ParGen}_c(\mathscr{U}, 1^\lambda), \\ (c, \mu_0, o_0, \mu_1, o_1) \leftarrow \mathsf{A}(cpar) \end{array}\right] \leq \nu(\lambda).$$

3.5.2 Relaxed Commitments

A relaxed commitment scheme rC for message space $\mathscr{U}$ and relaxed message space $\bar{\mathscr{U}} \supseteq \mathscr{U}$ consists of a triple of algorithms $(\mathsf{ParGen_c}, \mathsf{Commit}, \mathsf{OpenVf})$ such that:

Parameters Generation. On input the security parameter λ and the message space $\mathscr{U}$, $\mathsf{ParGen_c}(\mathscr{U}, 1^\lambda)$ outputs the parameters *cpar* of the scheme.

Commitment Generation. On input *cpar* and a message $\mu \in \mathscr{U}$, $\mathsf{Commit}(cpar, \mu)$ outputs a commitment c to μ with the corresponding opening information o.

Opening. On input a commitment c, opening information $\bar{o}$ and a message $\bar{\mu}$, $\mathsf{OpenVf}(cpar, c, \bar{\mu}, \bar{o})$ outputs 1 if $\bar{o}$ is a valid opening information for c w.r.t. $\bar{\mu} \in \bar{\mathscr{M}}$, and 0 otherwise.

A commitment scheme must satisfy the standard correctness and hiding properties that are recalled in Section 3.5.1. The binding property though needs to be adapted to preserve a meaning in presence of relaxed opening. Therefore, we define a partition of the relaxed message space $\bar{\mathscr{U}}$ and deem an attack successful if the adversary can open a commitment to two messages coming from different components of the partition. The partition is defined by the message relaxation function $f : \mathscr{U} \to 2^{\bar{\mathscr{U}}}$ that maps a message $\mu \in \mathscr{U}$ to a partition component $f(\mu) \subseteq \bar{\mathscr{U}}$. We say that the commitment scheme is *f-relaxed binding* if no adversary can open a commitment to two messages from different components.

Definition 3.17 (Relaxed Binding)**.** A relaxed commitment scheme rC is f-binding for a function $f : \mathscr{U} \mapsto 2^{\bar{\mathscr{U}}}$ if for all polynomial-time A

$$\mathsf{Pr}\left[\begin{array}{c} \mathsf{OpenVf}(cpar, c, \bar{\mu}_0, \bar{o}_0) = 1 \\ \wedge\, \mathsf{OpenVf}(cpar, c, \bar{\mu}_1, \bar{o}_1) = 1 \\ \wedge \nexists\, \mu \in \mathscr{U} \,:\, \{\bar{\mu}_0, \bar{\mu}_1\} \subseteq f(\mu) \end{array} : \begin{array}{c} cpar \leftarrow \mathsf{ParGen_c}(\mathscr{U}, 1^\lambda), \\ (c, \bar{\mu}_0, \bar{o}_0, \bar{\mu}_1, \bar{o}_1) \leftarrow \mathsf{A}(cpar) \end{array}\right] \leq \nu(\lambda)\,.$$

3.5.3 The Relation between Message and Challenge Spaces

As we already mentioned, we aim to design a commitment scheme where the relaxed Σ-protocol from Section 3.4 can be used to prove knowledge of a committed message, where the message and opening information are part of the witness. The problem with relaxed Σ-protocols is that they cannot guarantee the extraction of a valid witness for the original relation $\mathscr{R}$ (in this case, the opening verification relation), but only for the relaxed relation $\bar{\mathscr{R}}$. The witnesses in the

latter have larger norms and explicitly admit "small multiples": if $(\mathbf{S}, \mathbf{1})$ is a valid witness in $\mathscr{R}$ so that $\mathbf{AS} = \mathbf{U} \bmod q$, then $(\bar{\mathbf{S}} = \bar{\mathbf{c}}\mathbf{S}, \bar{\mathbf{c}})$ is a valid witness in $\bar{\mathscr{R}}$ so that $\mathbf{A}\bar{\mathbf{S}} = \bar{\mathbf{c}}\mathbf{U} \bmod q$, where $\bar{\mathbf{c}} \in \bar{\mathscr{C}} = \{\mathbf{c} - \mathbf{c}' \ : \ \mathbf{c}, \mathbf{c}' \in \mathscr{C}\}$ and where $\mathscr{C}$ is the challenge space. Therefore, relaxing the opening verification of the commitment scheme to accept extracted messages and opening information entails allowing a commitment to be opened to a small multiple $\bar{\mathbf{c}}\mathbf{m}$ of the originally committed message $\mathbf{m} \in \mathscr{U}$. In order to preserve a meaningful notion of relaxed binding, we must choose the message and challenge spaces so that the sets of small multiples of different messages are disjoint, i.e., that there do not exist distinct $\mathbf{m}, \mathbf{m}' \in \mathscr{U}$ and $\mathbf{c}, \mathbf{c}' \in \bar{\mathscr{C}}$ such that $\mathbf{mc} = \mathbf{m}'\mathbf{c}' \bmod q$.

For efficiency reasons, we choose messages and challenges from the subring $R_3^{(2^{K_m})}$ so that they have at most 2^{K_m} nonzero coefficients. By choosing the message and challenge spaces as

$$\begin{aligned}
\mathscr{U} &= \{\mathbf{1}\} \cup \{\mathbf{m} \in R_3^{(2^{K_m})} \ : \ \deg(\mathbf{m}) = n/2 \ \wedge \ \mathbf{m} \text{ is irreducible in } \mathbb{Z}_q[\mathbf{x}]\} \\
\mathscr{C} &= \{\mathbf{c} \in R_3^{(2^{K_m})} \ : \ \deg(\mathbf{c}) < n/4\} \\
\bar{\mathscr{U}} &= \{\bar{\mathbf{m}} \in R_{2r+1}^{(2^{K_m})} \ : \ \deg(\bar{\mathbf{m}}) < 3n/4\} \ , \ r \geq 1 \\
\bar{\mathscr{C}} &= \{\mathbf{c} - \mathbf{c}' \ : \ \mathbf{c}, \mathbf{c}' \in \mathscr{C}\} \ ,
\end{aligned} \tag{3.5}$$

we have that each $\bar{\mathbf{m}} \in \bar{\mathscr{U}}$ can have at most one irreducible factor of degree $n/2$ in $\mathbb{Z}_q[\mathbf{x}]$. By defining the message relaxation function f as

$$\begin{aligned}
f(\mathbf{m}) &= \{\bar{\mathbf{m}} \in \bar{\mathscr{U}} \ : \ \mathbf{m} | \bar{\mathbf{m}} \text{ in } \mathbb{Z}_q[\mathbf{x}]\} \quad \text{for } \mathbf{m} \neq \mathbf{1} \\
f(\mathbf{1}) &= \{\bar{\mathbf{m}} \in \bar{\mathscr{U}} \ : \nexists \, \mathbf{m} \in \mathscr{U} \setminus \{\mathbf{1}\} \ : \ \mathbf{m} | \bar{\mathbf{m}} \text{ in } \mathbb{Z}_q[\mathbf{x}]\} \ ,
\end{aligned} \tag{3.6}$$

the unique factorization of polynomials in $\mathbb{Z}_q[\mathbf{x}]$ (cf. Section 3.2.4 in [Cohen, 2013]) guarantees that the partition components $f(\mathbf{m})$ and $f(\mathbf{m}')$ are disjoint for any distinct $\mathbf{m}, \mathbf{m}' \in \mathscr{U}$.

To generate elements of $\mathscr{U}$, we suggest to generate random monic polynomials of degree $n/2$ in $R_3^{(2^{K_m})}$ and test them for irreducibility, which can be done efficiently (e.g., using Proposition 3.4.4 in [Cohen, 2013]). By Gauss' formula (cf. [Gauss, 2006]), the number of monic polynomials of degree $n/2$ that are irreducible in $\mathbb{Z}_q[\mathbf{x}]$ is approximately $q^{n/2}/(n/2)$. Assuming that the irreducible polynomials are "spread evenly" across $\mathbb{Z}_q[\mathbf{x}]$, one expects to sample an average of $n/2$ polynomials until finding an irreducible one.

3.5.4 Relaxed Commitment Scheme from Lattices

Our relaxed commitment uses message space $\mathscr{U}$ and relaxed message space $\bar{\mathscr{U}}$ defined in Equation (3.5). The algorithms of our relaxed commitment scheme rC are as follows.

Parameters generation. ParGen_c selects a uniformly random commitment key $\mathbf{C} \xleftarrow{\$} R_q^{1\times m}$ and sets the parameters (cf. Section 3.5.3)

- K_m and r, that define the message set $\mathscr{U}$,
- $\bar{N}_c$ and $\bar{N}_{c,\infty}$, that define the set of valid pairs message-opening information $\mathscr{OV}$.

These parameters should be such that:

$$\log_q(2n\bar{N}_{c,\infty}^2) + log_q(2) < \log_q(\frac{q-1}{4rn\bar{N}_{c,\infty}^2}) \,, \quad 2n\bar{N}_{c,\infty}(r+\bar{N}_{c,\infty}) < \frac{q-1}{2} \quad .$$

It outputs $cpar = (\mathbf{C}, K_m, r, \bar{N}_c, \bar{N}_{c,\infty})$.

Commitment generation. On input $(cpar, \mathbf{m})$, the algorithm Commit checks that $\mathbf{m} \in R_3^{(2^{K_m})}$, that $\deg(\mathbf{m}) = n/2$, and that $\mathbf{m}$ is irreducible. It then selects uniformly random $\mathbf{E} \xleftarrow{\$} R_3^{1\times m}$ and $\mathbf{b} \xleftarrow{\$} R_3$, and constructs the commitment as $\mathbf{F} := (\mathbf{C} + \mathbf{mG} + \mathbf{E})\mathbf{b}^{-1} \bmod q$. It outputs $(\mathbf{F}, (\mathbf{1}, \mathbf{E}, \mathbf{b}))$.

Opening verification. On input a message $\bar{\mathbf{m}}$, a commitment $\mathbf{F}$, and opening values $(\bar{\mathbf{c}}, \bar{\mathbf{E}}, \bar{\mathbf{b}})$, OpenVf outputs 1 if $\mathbf{F} = (\bar{\mathbf{c}}\mathbf{C} + \bar{\mathbf{m}}\mathbf{G} + \bar{\mathbf{E}})\bar{\mathbf{b}}^{-1} \bmod q$, $\bar{\mathbf{m}} \in \bar{\mathscr{U}}$ and $(\bar{\mathbf{c}}, \bar{\mathbf{E}}, \bar{\mathbf{b}}) \in \mathscr{OV}$, where $\mathscr{OV}$ is the set of the valid openings:

$$\mathscr{OV} = \left\{ (\bar{\mathbf{c}}, \bar{\mathbf{E}}, \bar{\mathbf{b}}) \in \bar{\mathscr{C}} \times R_q^{1\times m} \times R_q : \; \|[\bar{\mathbf{E}}, \bar{\mathbf{b}}]\|_2 \le \bar{N}_c \;\wedge\; \|[\bar{\mathbf{E}}, \bar{\mathbf{b}}]\|_\infty \le \bar{N}_{c,\infty} \right\} \,,$$

and $\bar{\mathscr{C}}$ was defined in Section 3.5.3.

It is easy to see that our construction satisfies correctness. We prove the hiding property under a new assumption, defined as Assumption 1 below. To gain trust in this assumption, we also give a selective variant in Assumption 2 that we show to be equivalent to RLWE and that, through a complexity leveraging argument, implies Assumption 1.

Assumption 1. Consider the following game between an adversary A and a challenger for fixed $m \in \mathbb{N}$ and distribution $\mathscr{D}$:

1. The challenger outputs a uniformly random $\mathbf{C} \xleftarrow{\$} R_q^{1\times m}$ to A.

2. A sends back $\mathbf{m} \in \mathscr{U}$.

3. The challenger samples a uniformly random bit $b \xleftarrow{\$} \{0,1\}$. If $b = 1$, it samples an error vector $\mathbf{E} \xleftarrow{\$} \mathscr{D}^m$ and a uniform secret $\mathbf{s} \xleftarrow{\$} \mathscr{D}$, and sends $\mathbf{F} := (\mathbf{C} + \mathbf{mG} - \mathbf{E})\mathbf{s}^{-1} \bmod q$ to A. Otherwise, it sends a uniform $\mathbf{F} \xleftarrow{\$} R_q^{1\times m}$ to A.

4. A sends a bit b' to the challenger.

The advantage of A in winning the game is $\left|\mathsf{Pr}(b = b') - \frac{1}{2}\right|$. The assumption states that no PPT A can win the previous game with non-negligible advantage.

Assumption 2 (Selective variant of Assumption 1.)**.** Consider the game of Assumption 1, but with steps 1 and 2 switched, meaning, A outputs $\mathbf{m} \in \mathscr{U}$ before being given $\mathbf{C}$:

1. A sends back $\mathbf{m} \in \mathscr{U}$.

2. The challenger samples a uniformly random $\mathbf{C} \xleftarrow{\$} R_q^{1\times m}$ and a uniformly random bit $b \xleftarrow{\$} \{0,1\}$. If $b = 1$, it samples an error vector $\mathbf{E} \xleftarrow{\$} \mathscr{D}^m$ and a uniform secret $\mathbf{s} \xleftarrow{\$} \mathscr{D}$, and sends $\mathbf{C}$ and $\mathbf{F} := (\mathbf{C} + \mathbf{mG} - \mathbf{E})\mathbf{s}^{-1} \bmod q$ to A. Otherwise, it sends a uniform $\mathbf{F} \xleftarrow{\$} R_q^{1\times m}$ to A.

3. A sends a bit b' to the challenger.

The assumption states that no PPT adversary can win this game with non-negligible advantage.

Assumption 2 is at least as hard as RLWE with m samples and distribution $\mathscr{D}$. It is then possible to reduce Assumption 2 to 1 with a complexity leveraging argument by guessing the value of $\mathbf{m} \in \mathscr{U}$.

Theorem 3.18. *Assumption 2 holds for $m \in \mathbb{N}$ and distribution $\mathscr{D}$ if the $\mathrm{RLWE}_{m,\mathscr{D}}$ assumption holds.*

Proof. Let A be an attacker breaking Assumption 2. Then it is possible to design a PPT algorithm B that breaks $\mathrm{RLWE}_{m,\mathscr{D}}$ exploiting A in essentially the same time and with the same advantage as A. On input an instance $(\mathbf{a}_1, \mathbf{b}_1), \ldots, (\mathbf{a}_m, \mathbf{b}_m)$, algorithm B runs A to obtain $\mathbf{m}$. It then sets $\mathbf{A} = [\mathbf{a}_1; \ldots; \mathbf{a}_m]$, $\mathbf{B} = [\mathbf{b}_1; \ldots; \mathbf{b}_m]$, $\mathbf{C} := \mathbf{B} - \mathbf{mG} \bmod q$ and $\mathbf{F} := \mathbf{A}$, and feeds $(\mathbf{B}, \mathbf{F})$ back to A. When A outputs $b' = 1$, then B decides that its input came from $\mathrm{RLWE}_{m,\mathscr{D}}$, otherwise that it was uniform.

Note that if $(\mathbf{a}_1, \mathbf{b}_1), \ldots, (\mathbf{a}_m, \mathbf{b}_m)$ come from the uniform distribution over R_q^2, then also $\mathbf{C}$ and $\mathbf{F}$ are uniformly distributed in $R_q^{1\times m}$. If they come from the RLWE distribution, however, then there exist an $\mathbf{s}$ and $\mathbf{E}$ sampled from $\mathscr{D}$ and $\mathscr{D}^m$, respectively, such that $\mathbf{As} + \mathbf{E} \equiv_q \mathbf{B}$. Therefore, $\mathbf{A} \equiv_q \frac{\mathbf{B}-\mathbf{E}}{\mathbf{s}} \equiv_q \frac{\mathbf{C}+\mathbf{mG}-\mathbf{E}}{\mathbf{s}}$.

The runtime of B is linear in the runtime of A, and its success probability is $\epsilon_{\mathsf{A}} + 1/2$, where ϵ_{A} is the advantage of A in breaking Assumption 2. □

The proof of the following theorem follows from a straightforward complexity leveraging argument by guessing the value of $\mathbf{m} \in \mathscr{U}$.

Theorem 3.19. *Let A be a PPT algorithm that has advantage ϵ in breaking Assumption 1 in time t. Then there exists a PPT algorithm B with running time t and advantage $\frac{\epsilon}{|\mathscr{U}|}$ in breaking Assumption 2.*

Proof. Algorithm B simulates the interaction in Assumption 2 as it follows. First, it guesses a polynomial $\bar{\mathbf{m}} \in \mathscr{U}$ and sends it to the challenger. Upon receiving $\mathbf{C}, \mathbf{F}$ from the challenger, B forwards $\mathbf{C}$ to A and receives back $\mathbf{m}$. If $\mathbf{m} \neq \bar{\mathbf{m}}$, B aborts. Otherwise, it sends $\mathbf{F}$ to A. Finally, when A outputs a bit b, B forwards it to the challenger. The success probability of B is the success probability ϵ_{A} of A times the probability of correctly guessing $\mathbf{m}$, i.e., $\frac{\epsilon_A}{|\mathscr{U}|}$. The runtime of B is linear in the runtime of A. □

We are now ready to prove the hiding property of the commitment scheme under Assumption 1.

Theorem 3.20 (Hiding)**.** *The relaxed commitment scheme above is computationally hiding when Assumption 1 holds for m and the uniform distribution over R_3.*

Proof. Given A breaking the hiding property of the commitment scheme, consider the following adversary B in the game of Assumption 1. Upon receiving $\mathbf{C} \in R_q^{1\times m}$ from the challenger, B sends it to A as part of the public parameters *cpar*. When A sends challenge messages $\mathbf{m}_0$, $\mathbf{m}_1$, B samples a random bit $b \xleftarrow{\$} \{0,1\}$ and sends $\mathbf{m}_b$ to the challenger. Upon receiving $\mathbf{F}$, B sends it to A as the commitment. When A outputs a bit $b' = b$, B outputs $b'' = 1$ to the challenger, otherwise $b'' = 0$. It is clear that when B's input $\mathbf{F}$ is uniform, then A's view is independent of b, so that A has zero advantage guessing b, while if B's input is based on $\mathbf{m}_b$, it is distributed exactly as a commitment of $\mathbf{m}_b$. The advantage of B in breaking Assumption 1 is therefore half the advantage of A in

breaking the hiding property:

$$
\begin{aligned}
&\mathsf{Pr}[\mathsf{B}\text{ succeeds}] = \\
&= \mathsf{Pr}\left[b' = b \ : \ RLWE_{m,\mathscr{U}(R_3)}\right]\mathsf{Pr}\left[RLWE_{m,\mathscr{U}(R_3)}\right] + \mathsf{Pr}[b' = b \ : \ \mathscr{U}(R_3)]\mathsf{Pr}[\mathscr{U}(R_3)] \\
&= \left(\epsilon + \frac{1}{2}\right)\cdot\frac{1}{2} + \frac{1}{2}\cdot\frac{1}{2} \\
&= \frac{\epsilon}{2} + \frac{1}{2}
\end{aligned}
$$

where the first equality follows from the definition of advantage. The runtime of B is linear in the runtime of A, hence B runs in polynomial time. □

Remark that relying on the hardness of RLWE with the uniform distribution is enough. Indeed, that particular instantiation of RLWE is hard as long as it only outputs a constant number of samples, and this is enough for the reduction, as to generate a commitment only m samples from $A_{\mathbf{s},\mathscr{U}(R_3)}$ are necessary.

Finally, the relaxed binding property holds under the RSIS assumption.

Theorem 3.21 (Relaxed Binding)**.** *The relaxed commitment scheme above is f-relaxed binding for the function f in Equation (3.6) if the $RSIS_{m+1,q,\beta_c}$ assumption holds for $\beta_c = 2n\bar{N}_{c,\infty}(r + \bar{N}_{c,\infty})$.*

Proof. Assume that there exists a PPT algorithm A that breaks the f-relaxed binding property. We construct an algorithm B that solves the $\mathrm{RSIS}_{m+1,q,\beta_c}$ problem as follows.

On input $[\mathbf{A}\ \mathbf{1}] \in R_q^{1\times(m+1)}$, algorithm B runs A on input parameters *cpar* that include $\mathbf{C} = \mathbf{A}$. When A outputs a commitment $\mathbf{F}$, two distinct messages $\bar{\mathbf{m}}_0, \bar{\mathbf{m}}_1 \in \bar{\mathscr{U}}$, and two openings $(\bar{\mathbf{c}}_0, \bar{\mathbf{E}}_0, \bar{\mathbf{b}}_0)$ and $(\bar{\mathbf{c}}_1, \bar{\mathbf{E}}_1, \bar{\mathbf{b}}_1)$ such that $\mathsf{OpenVf}(cpar, \mathbf{F}, \bar{\mathbf{m}}_i, (\bar{\mathbf{c}}_i, \bar{\mathbf{E}}_i, \bar{\mathbf{b}}_i)) = 1$ for $i = 0, 1$. We have that

$$\mathbf{F} \equiv_q (\bar{\mathbf{c}}_0\mathbf{C} + \bar{\mathbf{m}}_0\mathbf{G} + \bar{\mathbf{E}}_0)\bar{\mathbf{b}}_0^{-1} \equiv_q (\bar{\mathbf{c}}_1\mathbf{C} + \bar{\mathbf{m}}_1\mathbf{G} + \bar{\mathbf{E}}_1)\bar{\mathbf{b}}_1^{-1}$$

that rearranging the terms yields

$$(\bar{\mathbf{b}}_1\bar{\mathbf{c}}_0 - \bar{\mathbf{b}}_0\bar{\mathbf{c}}_1)\mathbf{A} + (\bar{\mathbf{b}}_1\bar{\mathbf{m}}_0 - \bar{\mathbf{b}}_0\bar{\mathbf{m}}_1)\mathbf{G} + \bar{\mathbf{b}}_1\bar{\mathbf{E}}_0 - \bar{\mathbf{b}}_0\bar{\mathbf{E}}_1 \ = \ \mathbf{0} \bmod q\ . \tag{3.7}$$

Recalling that $\mathbf{G} = [1\ \lceil q^{1/m}\rceil \ldots \lceil q^{(m-1)/m}\rceil]$, the first component of the above vector is

$$(\bar{\mathbf{b}}_1\bar{\mathbf{c}}_0 - \bar{\mathbf{b}}_0\bar{\mathbf{c}}_1)\mathbf{a}_1 + \bar{\mathbf{b}}_1\bar{\mathbf{m}}_0 - \bar{\mathbf{b}}_0\bar{\mathbf{m}}_1 + \bar{\mathbf{b}}_1\bar{\mathbf{e}}_{0,1} - \bar{\mathbf{b}}_0\bar{\mathbf{e}}_{1,1} \ = \ 0 \bmod q\ .$$

where $\mathbf{a}_1$, $\bar{\mathbf{e}}_{0,1}$, and $\bar{\mathbf{e}}_{1,1}$ are the first components of $\mathbf{A}$, $\bar{\mathbf{E}}_0$, and $\bar{\mathbf{E}}_1$, respectively.

Setting $\mathbf{S} := [\bar{\mathbf{b}}_1\bar{\mathbf{c}}_0 - \bar{\mathbf{b}}_0\bar{\mathbf{c}}_1 \; ; \; 0 \; ; \; \ldots \; ; \; 0 \; ; \; \bar{\mathbf{b}}_1\bar{\mathbf{m}}_0 - \bar{\mathbf{b}}_0\bar{\mathbf{m}}_1 + \bar{\mathbf{b}}_1\bar{\mathbf{e}}_{0,1} - \bar{\mathbf{b}}_0\bar{\mathbf{e}}_{1,1}]$, we have that $[\mathbf{A}\ \mathbf{1}]\mathbf{S} = 0 \bmod q$, as required, so B outputs $\mathbf{S}$ as its $\mathrm{RSIS}_{m+1,q,\beta_c}$ solution.

To complete the proof we have to show that $\mathbf{S} \neq \mathbf{0}_m$ (we use the equality here, as $\mathbf{S}$ has infinity norm bounded by $\beta_c < q$) and that $\|\mathbf{S}\| \leq \beta_c$.

First, assume that $\mathbf{S} = \mathbf{0}_m$. This would mean in particular that $\bar{\mathbf{b}}_1\bar{\mathbf{c}}_0 - \bar{\mathbf{b}}_0\bar{\mathbf{c}}_1 = \mathbf{0}$. Therefore, from Equation (3.7) it follows that

$$(\bar{\mathbf{b}}_1\bar{\mathbf{m}}_0 - \bar{\mathbf{b}}_0\bar{\mathbf{m}}_1)\mathbf{G} + \bar{\mathbf{b}}_1\bar{\mathbf{E}}_0 - \bar{\mathbf{b}}_0\bar{\mathbf{E}}_1 \;=\; \mathbf{0}_{1\times m} \bmod q \;. \tag{3.8}$$

The proof that Equation (3.8) does not hold proceeds in 2 steps:

1. we show that $\bar{\mathbf{b}}_1\bar{\mathbf{m}}_0 - \bar{\mathbf{b}}_0\bar{\mathbf{m}}_1 \neq \mathbf{0}$.
2. we prove that $(\bar{\mathbf{b}}_1\bar{\mathbf{m}}_0 - \bar{\mathbf{b}}_0\bar{\mathbf{m}}_1)\mathbf{G} \neq -\bar{\mathbf{b}}_1\bar{\mathbf{E}}_0 + \bar{\mathbf{b}}_0\bar{\mathbf{E}}_1 \bmod q$.

Indeed, if $\bar{\mathbf{b}}_1\bar{\mathbf{m}}_0 - \bar{\mathbf{b}}_0\bar{\mathbf{m}}_1 = \mathbf{0}$, then multiplying both sides with $\bar{\mathbf{c}}_0$ and substituting $\bar{\mathbf{b}}_1\bar{\mathbf{c}}_0 = \bar{\mathbf{b}}_0\bar{\mathbf{c}}_1$ (that follows from $\bar{\mathbf{b}}_1\bar{\mathbf{c}}_0 - \bar{\mathbf{b}}_0\bar{\mathbf{c}}_1 = \mathbf{0}$) yields $\bar{\mathbf{b}}_0(\bar{\mathbf{c}}_1\bar{\mathbf{m}}_0 - \bar{\mathbf{c}}_0\bar{\mathbf{m}}_1) = \mathbf{0}$, implying that either $\bar{\mathbf{b}}_0 = \mathbf{0}$ or $\bar{\mathbf{c}}_1\bar{\mathbf{m}}_0 = \bar{\mathbf{c}}_0\bar{\mathbf{m}}_1$.

The former is not possible because $\bar{\mathbf{b}}_0^{-1}$ must exist in order to pass the opening verification algorithm. The latter is impossible as well, because $\bar{\mathbf{c}}_0, \bar{\mathbf{c}}_1$ are polynomials of degree less than $n/4$ (cf. the definition of $\bar{\mathscr{C}}$ in Section 3.5.3), while $\bar{\mathbf{m}}_0, \bar{\mathbf{m}}_1 \in \bar{\mathscr{U}}$ are of degree less than $3n/4$, so that their products $\bar{\mathbf{c}}_1\bar{\mathbf{m}}_0, \bar{\mathbf{c}}_0\bar{\mathbf{m}}_1$ are of degree less than n. Therefore, if $\bar{\mathbf{c}}_1\bar{\mathbf{m}}_0 = \bar{\mathbf{c}}_0\bar{\mathbf{m}}_1$ in $R_q = \mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$, then also $\bar{\mathbf{c}}_1\bar{\mathbf{m}}_0 = \bar{\mathbf{c}}_0\bar{\mathbf{m}}_1$ in $\mathbb{Z}_q[\mathbf{x}]$.

However, to be a valid f-binding attack, there cannot exist an $\mathbf{m} \in \mathscr{U}$ so that $\{\bar{\mathbf{m}}_0, \bar{\mathbf{m}}_1\} \in f(\mathbf{m})$. This implies that at least one message $\bar{\mathbf{m}}_b \in \{\bar{\mathbf{m}}_0, \bar{\mathbf{m}}_1\}$ has an irreducible divisor $\mathbf{m}$ of degree $n/2$ that doesn't divide $\bar{\mathbf{m}}_{1-b}$. Since $\bar{\mathbf{c}}_1\bar{\mathbf{m}}_0$ and $\bar{\mathbf{c}}_0\bar{\mathbf{m}}_1$ are polynomials of degree less than n and $\mathbb{Z}_q[x]$ is a unique factorization domain, it must hold that $\bar{\mathbf{c}}_1\bar{\mathbf{m}}_0 \neq \bar{\mathbf{c}}_0\bar{\mathbf{m}}_1$, and thereby that $\bar{\mathbf{b}}_1\bar{\mathbf{m}}_0 - \bar{\mathbf{b}}_0\bar{\mathbf{m}}_1 \neq 0$ in Equation (3.8).

Finally, we prove that $(\bar{\mathbf{b}}_1\bar{\mathbf{m}}_0 - \bar{\mathbf{b}}_0\bar{\mathbf{m}}_1)\mathbf{G} \neq -\bar{\mathbf{b}}_1\bar{\mathbf{E}}_0 + \bar{\mathbf{b}}_0\bar{\mathbf{E}}_1 \bmod q$ by proving that the there exists a component of the vector on the left-hand side that has larger coefficients than the corresponding component of the vector on the right-hand side.

Let $\mathbf{g}_i = \lceil q^{(i-1)/m} \rceil$, $i = 1, \ldots, m$. We can rearrange Equation (3.8) so that the components of the vectors $\mathbf{G}$ and $\mathbf{E}$ satisfy:

$$(\bar{\mathbf{b}}_1\bar{\mathbf{m}}_0 - \bar{\mathbf{b}}_0\bar{\mathbf{m}}_1)\mathbf{g}_j \;=\; \bar{\mathbf{b}}_0\bar{\mathbf{e}}_{1,j} - \bar{\mathbf{b}}_1\bar{\mathbf{e}}_{0,j} \bmod q \;\forall\; j = 1, \ldots, m. \tag{3.9}$$

Applying Lemma 2.12 for all $i = 1, \ldots, m$ it holds $\|\bar{\mathbf{b}}_0\bar{\mathbf{e}}_{1,i} - \bar{\mathbf{b}}_1\bar{\mathbf{e}}_{0,i}\|_\infty \leq 2\bar{N}_{c,\infty}^2 n$, as we know from the definition of $\mathscr{OV}$ that the infinity norm of $\mathbf{E}$ and $\mathbf{b}$ should be less than $\bar{N}_{c,\infty}$, and that $\bar{N}_{c,\infty}^2 n < \frac{q-1}{2}$.

Since $\bar{\mathbf{b}}_1\bar{\mathbf{m}}_0 - \bar{\mathbf{b}}_0\bar{\mathbf{m}}_1 \neq 0$, this polynomial must contain at least one non-zero monomial $a_i\mathbf{x}^i$. Let k be such that $q^{k/m} \leq a_i \leq q^{(k+1)/m}$, $k = 0,\ldots,m-1$. As Equation 3.9 holds for all j, it holds in particular for j such that

$$1 \leq j \leq \min\{m-1+k, m(1-\log_q(2rn\bar{N}_{c,\infty}))-1\} .$$

Such j exists as $k \in \{1,\ldots,m\}$ and $m(1-\log_q(2rn\bar{N}_{c,\infty}))-1 \geq 2$.

Then the coefficient of $\mathbf{x}^i$ in the left-hand side of Equation (3.9) is $a_i\mathbf{g}_j$ such that

1. $2n\bar{N}_{c,\infty}^2 < \|a_i\mathbf{g}_j\|_\infty$,
2. $\|a_i\mathbf{g}_j\|_\infty \leq \frac{q-1}{2}$,
3. $j \in \{1,\ldots,m\} \subset \mathbb{N}$ and $\mathbf{g}_j = \lceil q^{(j-1)/m} \rceil$.

We show in the following that such j exists.

Observe that $\|a_i\mathbf{g}_j\|_\infty \geq q^{k/m}q^{(j-1)/m}$. Hence, Condition 1 implies that

$$q^{(k+j-1)/m} > 2n\bar{N}_{c,\infty}^2 . \tag{3.10}$$

From $\|a_i\mathbf{g}_j\|_\infty \leq \|\mathbf{a}_i\|_\infty q^{j/m} \leq 2rn\bar{N}_{c,\infty}^2 q^{j/m}$ and Condition 2 it follows that

$$q^{j/m}2rn\bar{N}_{c,\infty}^2 \leq \frac{q-1}{2} . \tag{3.11}$$

Putting the inequalities (3.10) and (3.11) together yields

$$\frac{2n\bar{N}_{c,\infty}^2}{q^{(k-1)/m}} < q^{\frac{j}{m}} \leq \frac{q-1}{4rn\bar{N}_{c,\infty}^2} .$$

Given that $k = 1,\ldots,m-1$ and $j \in \mathbb{N}$ the previous inequality becomes:

$$\lceil m\log_q(2n\bar{N}_{c,\infty}^2)\rceil < j \leq m\log_q(\frac{q-1}{4rn\bar{N}_{c,\infty}^2}) ,$$

i.e., (remember that $m = \log_2 q$)

$$\log_q(2n\bar{N}_{c,\infty}^2) + log_q(2) < \log_q(\frac{q-1}{4rn\bar{N}_{c,\infty}^2}) .$$

The latter inequality is satisfied by the parameters by construction.

Finally, we have to bound the norm of $\mathbf{S}$. Applying Lemma 2.12 and the triangular inequality, and recalling that to be a valid opening value $\|[\mathbf{E}\ \mathbf{b}]\|_\infty \leq \bar{N}_{c,\infty}$ we obtain that:

$$\begin{aligned}\|\mathbf{S}\|_\infty &\leq \max\{\|\bar{\mathbf{b}}_1\bar{\mathbf{c}}_0 - \bar{\mathbf{b}}_0\bar{\mathbf{c}}_1\|_\infty,\ \|\bar{\mathbf{b}}_1\bar{\mathbf{m}}_0 - \bar{\mathbf{b}}_0\bar{\mathbf{m}}_1 + \bar{\mathbf{b}}_1\bar{\mathbf{e}}_{0,1} - \bar{\mathbf{b}}_0\bar{\mathbf{e}}_{1,1}\|_\infty\} \\ &\leq \max\{2(\bar{N}_{c,\infty}2n),\ 2(\bar{N}_{c,\infty}rn + \bar{N}_{c,\infty}^2 n)\} \\ &= 2n\bar{N}_{c,\infty}\cdot\max\{2,(r+\bar{N}_{c,\infty})\} \\ &= 2n\bar{N}_{c,\infty}(r+\bar{N}_{c,\infty}) =: \beta_c\ ,\end{aligned}$$

where the last equality holds because $r + \bar{N}_{c,\infty} > 2$. □

Finally, the following property is needed when using this commitment scheme to build relaxed ZK proofs.

Lemma 3.22. *Let cpar be defined as in Section 3.5.4. Assume that there exists a PPT algorithm* A *that chooses* $\mathbf{m} \in \mathscr{U}$ *and is then able to distinguish between the following two distributions:*

- ***distribution 1**: $\mathbf{C}$ and $\mathbf{F}$ uniformly sampled vectors in $R_q^{1\times m}$.*
- ***distribution 2**: $\mathbf{C} \xleftarrow{\$} R_q^{1\times m}$, $\mathbf{F} = \mathbf{b}^{-1}(\mathbf{C}+\mathbf{mG}+\mathbf{E})$ mod q, where $\mathbf{G}$ is the gadget matrix and $\mathbf{E} \xleftarrow{\$} R_3^{1\times m}$.*

Then there exists a distinguisher D *that is able to solve the $RLWE_{m,\mathscr{U}(R_q)}$ decisional problem exploiting* A *with the same success probability.*

Proof. The distinguisher D gets as input from the RLWE game a pair $(\mathbf{A},\mathbf{B}) \in R_q^{1\times m} \times R_q^{1\times m}$ and has to distinguish whether it was sampled according to the uniform or RLWE distribution. To do that, it feeds A as input the pair $(\mathbf{C},\mathbf{F})$, where $\mathbf{C} := \mathbf{B} - \mathbf{mG}$ mod q and $\mathbf{F} := \mathbf{A}$ mod q. Then, B returns *distribution 1* if A returns *uniform*, and *distribution 2* if A returns *RLWE*. Its success probability is the same as A. Indeed, if $(\mathbf{A},\mathbf{B})$ are uniformly distributed, then $\mathbf{C}$ (and obviously $\mathbf{F}$) are uniformly distributed as well. In the other case, there exists an $\mathbf{s} \in R_3$ such that $\mathbf{C} = \mathbf{B}-\mathbf{mG}$ mod $q = \mathbf{As}+\mathbf{E}-\mathbf{mG}$ mod q, i.e., $\mathbf{F} = \mathbf{A} = (\mathbf{C}+\mathbf{mG}-\mathbf{E})\mathbf{s}^{-1}$ mod q. Hence $(\mathbf{C},\mathbf{F})$ is distributed exactly as in *distribution 2*. □

3.6 Relaxed Signature

We now introduce a signature scheme for which the protocol rΣ from Section 3.4 can be used to prove knowledge of a signature on a committed message. Similarly to the relaxed commitments of the previous section, we also define *relaxed*

signature schemes to accommodate for the relaxed extraction of the $\mathsf{r}\Sigma$ protocol. More specifically, the verification algorithm is relaxed to accept messages and signatures that could never be signed, respectively produced, by the honest signing algorithm. At the same time, we also relax the unforgeability notion so that the adversary's forgery cannot be on a message that is within the span, through a function f, of its previous signing queries.

3.6.1 Signatures

A signature scheme S for a message set $\mathscr{M}$ is a 4-ple of PPT algorithms ($\mathsf{SParGen}$, $\mathsf{SKeyGen}, \mathsf{Sign}, \mathsf{SVerify}$) such that

Parameters Generation. On input the security parameter λ, $\mathsf{SParGen}$ outputs the parameters of the scheme *spar*.

Key Generation. On input *spar*, $\mathsf{SKeyGen}$ outputs a signing key *ssk* and a verification key *svk*.

Signing. On input a message $\mu \in \mathscr{M}$ and the signing key *ssk*, it outputs a signature σ.

Verification. On input a signature σ, a message μ, and a verification key *svk*, $\mathsf{SVerify}$ outputs 1 if σ is a valid signature on μ w.r.t. *svk*, 0 otherwise.

A signature scheme is correct if the verifier always accepts (i.e., outputs 1) honestly generated signatures.

Definition 3.23 (Correctness)**.** A signature scheme S is correct if for all security parameters $\lambda \in \mathbb{N}$ and for all messages $\mu \in \mathscr{M}$ it holds that:

$$\mathsf{Pr}\left[1 \leftarrow \mathsf{SVerify}(\mathit{svk}, \mu, \sigma) \;\middle|\; \begin{array}{r} \mathit{spar} \leftarrow \mathsf{SParGen}(1^\lambda), \\ (\mathit{ssk}, \mathit{svk}) \leftarrow \mathsf{SKeyGen}(\mathit{spar}), \\ \sigma \leftarrow \mathsf{Sign}(\mathit{ssk}, \mu) \end{array}\right] = 1\,.$$

Security requires it to be impossible to forge a valid signature w.r.t. *svk* without knowing the corresponding *ssk* (this notion was introduced by Goldwasser et al. [1988]).

Definition 3.24 (Unforgeability)**.** A signature scheme S is said to be existentially unforgeable against (adaptive) chosen-message attacks (euf-acma) if a PPT adversary A has negligible probability in winning the following game:

Experiment $\mathsf{Exp}_{\mathsf{A}}^{\mathsf{euf-acma}}(1^\lambda)$
$spar \leftarrow \mathsf{SParGen}(1^\lambda)$
$(ssk, svk) \leftarrow \mathsf{SKeyGen}(spar)$
$(\mu^*, \sigma^*) \leftarrow \mathsf{A}^{\mathcal{O}_S}(spar, svk)$
If $1 \leftarrow \mathsf{SVerify}(svk, \mu^*, \sigma^*)$
and $\nexists \sigma$ such that $(\mu^*, \sigma) \in \mathcal{Q}_S$,
then return 1 else return 0.

Oracle $\mathcal{O}_S(\mu)$
$\sigma \leftarrow \mathsf{Sign}(ssk, \mu)$
$\mathcal{Q}_S \leftarrow \mathcal{Q}_S \cup \{(\mu, \sigma)\}$
Return σ .

A stronger notion of unforgeability considers a valid forgery even the case in which the adversary outputs a fresh signature on an already signed message. This notion is called *strong unforgeability under chosen-message attacks* and was introduced by An et al. [2002]. However, this seems impossible to guarantee in presence of relaxed protocols from lattice-based hardness assumptions, hence we only consider (and "relax") the weaker notion.

3.6.2 Relaxed Signatures

A relaxed signature scheme associated with message space $\mathcal{M}$ and relaxed messages space $\bar{\mathcal{M}} \supseteq \mathcal{M}$ consists of four PPT algorithms ($\mathsf{SParGen}, \mathsf{SKeyGen}, \mathsf{Sign}, \mathsf{SVerify}$):

Parameters Generation. The parameter generation algorithm $\mathsf{SParGen}$ takes as input the security parameter 1^λ, and outputs the system parameters *spar*.

Key Generation. The key generation algorithm $\mathsf{SKeyGen}$ takes as input *spar*, and outputs a signing key *ssk* and a verification key *svk*.

Signing. The signing algorithm Sign takes as input *ssk* and a message $\mu \in \mathcal{M}$, and outputs a signature σ.

Verification. The verification algorithm $\mathsf{SVerify}$ takes as input *svk*, a message $\bar{\mu} \in \bar{\mathcal{M}}$ and a signature $\bar{\sigma}$, and returns 1 if the signature is valid or 0 if it is invalid.

As in the case of standard signatures, correctness requires that $\mathsf{SVerify}(svk, M, \sigma) = 1$ for all messages $\mu \in \mathcal{M}$, for all security parameters $\lambda \in \mathbb{N}$, for all (ssk, svk) generated by $\mathsf{SKeyGen}(spar)$, and for all σ generated honestly running $\mathsf{Sign}(ssk, M)$.

Relaxed unforgeability is parameterized by a message relaxation function $g : \mathcal{M} \to 2^{\bar{\mathcal{M}}}$.The adversary in the *g-relaxed unforgeability* game below wins the game if it can output a valid signature on a message $\bar{\mu} \in \bar{\mathcal{M}}$ that is not in the span through g of its signature queries.

Definition 3.25 (Relaxed Unforgeability)**.** A relaxed signature scheme $(\mathsf{SParGen}, \mathsf{SKeyGen}, \mathsf{Sign}, \mathsf{SVerify})$ is g-relaxed existentially unforgeable against (adaptive) chosen-message attacks (g-euf-acma) if all PPT A have negligible probability in winning the following game:

Experiment $\mathsf{Exp}_{\mathsf{A}}^{g-\mathsf{euf}-\mathsf{acma}}(1^\lambda)$
$spar \leftarrow \mathsf{SParGen}(1^\lambda)$
$\mathcal{Q}_S \leftarrow \emptyset$
$(ssk, svk) \leftarrow \mathsf{SKeyGen}(spar)$
$(\bar{\mu}, \bar{\sigma}) \leftarrow \mathsf{A}^{\mathcal{O}_S}(spar, svk)$
If $1 \leftarrow \mathsf{SVerify}(svk, \bar{\mu}, \bar{\sigma})$
and such that $\bar{\mu} \notin g(\mathcal{Q}_S)$,
then return 1 else return 0.

Oracle $\mathcal{O}_S(\mu)$
$\sigma \leftarrow \mathsf{Sign}(ssk, \mu)$
$\mathcal{Q}_S \leftarrow \mathcal{Q}_S \cup \{\mu\}$
Return σ .

The concept of relaxed signatures is somewhat reminiscent of a technique used for proofs of knowledge of a strong-RSA-based signature in groups of unknown order [Camenisch and Lysyanskaya, 2003]. Here, one has to prove that the message lies in a certain space, but the correctness of such a proof is only guaranteed when the actual message lies in a smaller interval. The approach was used in several privacy-preserving protocols, but was never formalized and did not require an adapted unforgeability notion.

3.6.3 Relaxed Signature Scheme from Lattices

We describe a relaxed signature scheme with message space $\mathcal{M} = \{(\mathbf{m}, \alpha) \in \mathcal{U} \times \{0,1\}^*\}$, where $\mathcal{U} \subseteq R_3$ is a (small) set of binary polynomials (e.g., $\mathcal{U}$ can be set as in Equation (3.5)). In a typical use case, $\mathbf{m}$ is a user identity and α an attribute value assigned to that user. Our scheme combines a weakly secure version of Boyen signatures [Boyen, 2010] to sign user identities and Gentry-Peikert-Vaikuntanathan signatures [Gentry et al., 2008] to sign attribute values.

To use the $\mathsf{r}\Sigma$ protocol from Section 3.4 to prove knowledge of a signature for a committed user identity $\mathbf{m}$, we relax the verification algorithm so that the (relaxed) witness that can be extracted from a valid $\mathsf{r}\Sigma$ protocol is still considered a valid signature for a message from the relaxed message space $\bar{\mathcal{M}} = \bar{\mathcal{U}} \times \{0,1\}^*$, where $\bar{\mathcal{U}}$ is as defined in Equation (3.5).

Our relaxed signature scheme rS is described as follows:

System Parameters. Parameters $spar = (n, q, m, \sigma_t, \sigma, r, N_s, \bar{N}_s, \bar{N}_{s,\infty}, \bar{C}, \mathbf{C})$ include a uniformly random matrix $\mathbf{C} \in R_q^{1\times m}$, a gadget vector $\mathbf{G}$ of length m as defined in Theorem 2.17, and a hash function $H : \{0,1\}^* \to R_q$. It also contains the parameters listed below.

- σ_t is the standard deviation of the (Gaussian) distribution of the trapdoor,
- $\sigma = \sqrt{n \log q}(\log n)^{3/2}$ is the standard deviation of the (Gaussian) distribution of the signature,
- r is a bound on the norm of user identities $\bar{\mathbf{m}}$ (as in Equation (3.5)),
- $N_s = 1.05\sigma\sqrt{n(2m+2)}$ is a bound on the norm of honestly created signatures,
- $\bar{N}_s$, $\bar{N}_{s,\infty}$, and $\bar{C}$ are bounds on the norm of components of signatures accepted by the relaxed verification algorithm, $\bar{N}_s \geq N_s$, $\bar{N}_{s,\infty} > 8\sigma$, $\bar{C} \geq 1$
- $\mathscr{C}$, and $\bar{\mathscr{C}}$ are challenge spaces defined in Equation (3.5).

When discussing the correctness of the signature, we give precise formulas for all the previous parameters but $\bar{N}_s$, $\bar{N}_{s,\infty}$, and $\bar{C}$. These last three will be discussed in Section 3.7. For correctness to hold, we only need to impose that $\bar{N}_s > N_s$ and $\bar{C} \geq 1$. From the unforgeability of the signature, we get the following bound

$$2(2^{K_m} - 1) < (q-1)/2\,, \quad 8n^2 m \bar{C} \sigma_t \bar{N}_{s,\infty} \leq \frac{q-1}{2}\,.$$

Key Generation. The signer chooses a uniform polynomial $\mathbf{a} \in R_q$ and sets $\mathbf{A} := [\mathbf{a}\ \mathbf{1}] \in R_q^{1\times 2}$. The secret signing key is sampled as $\mathbf{X} \xleftarrow{\$} \mathscr{D}_{R_q,\sigma_t}^{2\times m}$. Then the public verification key is the vector

$$\mathbf{V} := \begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{C} & \mathbf{1}\end{bmatrix} = \begin{bmatrix}\mathbf{A} & \mathbf{AX}+\mathbf{G} & \mathbf{C} & \mathbf{1}\end{bmatrix} \in R_q^{1\times(3+2m)}\,.$$

Signing. If $\mu = (\mathbf{m}, \alpha) \notin \mathscr{M}$ then abort. Otherwise, the signer calculates $\mathbf{S} \leftarrow \mathsf{SampleD}\left(\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{C}+\mathbf{mG}\end{bmatrix}, H(\alpha), \sigma\right)$ (where SampleD is defined in Lemma 2.18) and outputs a signature $\sigma = (\mathbf{1}, [\mathbf{S}; \mathbf{0}], \mathbf{1})$. The entry $(\mathbf{m}, \alpha, \sigma)$ is stored so that the same signature σ is returned next time that $(\mathbf{m}, \alpha)$ is queried.

Verification. Verification of a signature $\bar{\sigma} = (\bar{\mathbf{c}}_1, \bar{\mathbf{S}}, \bar{\mathbf{c}}_2)$ on message $\bar{\mu} = (\bar{\mathbf{m}}, \alpha)$ returns 1 if

- $\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{C}+\bar{\mathbf{m}}\mathbf{G} & \mathbf{1}\end{bmatrix}\bar{\mathbf{S}} = \bar{\mathbf{c}}_2 H(\alpha) \bmod q$,
- $\bar{\mu} \in \bar{\mathscr{M}}$,

- $\bar{\sigma} \in \{(\bar{\mathbf{c}}_1, \bar{\mathbf{S}}, \bar{\mathbf{c}}_2) \in \bar{\mathscr{C}} \times R_q^{3+2m} \times R_q : \|\bar{\mathbf{S}}\|_2 \leq \bar{N}_s \wedge \|\bar{\mathbf{S}}\|_\infty \leq \bar{N}_{s,\infty} \wedge \|\bar{\mathbf{c}}_2\|_\infty \leq \bar{C}\}$ [2].

Otherwise, it returns 0.

Remark 8. The peculiar structure of $\mathscr{U}$ is needed to compensate for the loose security definitions, in particular for the relaxed binding property of the commitment scheme (cf. the construction of Anonymous Attribute Tokens in Chapter 4).

In the case of group signatures (cf. Sections 5.3 and 5.5) this is not necessary anymore. Indeed, the commitment scheme is only used to hide part of the message (see Section 3.7), while a verifiable encryption scheme compensates for the relaxed special soundness of the Σ-protocol, allowing to recover a signed message in the original message space. Hence, $\mathscr{U}$ can be the whole subring, $\mathscr{U} = R_3^{(16)}$. Moreover, in the context of group signatures, an element $\mathbf{m} \in \mathscr{U}$ encodes a user's identity, but there is no need for the bit string. Therefore, we substitute the output of the hash of the bit-string with a constant polynomial $\mathbf{u}$ chosen uniformly at random in R_q during the key generation and sign only messages in $\mathscr{M} = \mathscr{U}$. The modified scheme is trivially still unforgeable under the same assumption in the Random Oracle Model, as it is simply the original signature scheme where messages are restricted to be in $\mathscr{U} \times \{0\}$.

3.6.4 Correctness

To prove the correctness of rS, we analyze the choices of the parameters.

Theorem 3.26. *The relaxed signature scheme* rS *is correct.*

Proof. The proof has two steps:

1. verify that it is possible to produce a signature, and
2. verify that a honestly generated signature satisfies the requirement of the verification algorithm.

The standard deviation $\sigma = \sqrt{n \log q}(\log n)^{3/2}$ satisfies the hypothesis of Theorem 2.17, hence we are able to sample from $D^{\perp}_{[\mathbf{A}\ \mathbf{AR}+\mathbf{G}],H(\alpha),\sigma}$.

By Lemma 2.18 we are also able to sample $\mathbf{S}$ from $D^{\perp}_{[\mathbf{A}\ \mathbf{AR}+\mathbf{G}\ \mathbf{U}+\mathbf{mG}],H(\alpha),\sigma}$.

[2] We define the set with respect to both norms because in different constructions we have switched between the two norms.

Regarding the norm of a signature, Lemma 2.16 guarantees that the norms of a honestly generated signature can be bounded as $\|\mathbf{S}\|_2 \leq 1.05\sigma\sqrt{n(2m+2)} = N_s < \bar{N}_s$ and $\|\mathbf{S}\|_\infty \leq 8\sigma < \bar{N}_{s,\infty}$ with high probability. Finally, any message in $\mathcal{M}_{K_m}$ is also in $\bar{\mathcal{M}}_{K_m}$ by construction. Observe that $\bar{C} \geq 1$, hence the verification algorithm accepts any signature generated by Sign. □

3.6.5 Unforgeability

We prove the g-unforgeability of our rS scheme under Assumption 3 described below. Assumption 3 is very similar to the g-unforgeability experiment itself, but, similarly to what we did for the hiding property of the rC scheme, we gain trust in the assumption by introducing a selective variant in Assumption 4 that we show to be implied by the RLWE and RSIS assumptions. A complexity leveraging argument can be used to show that Assumption 3 holds when Assumption 4 holds.

Essentially, Assumption 3 states that it should be hard to find a short vector in some coset $\mathscr{L}^\perp(\mathbf{M}) + \mathbf{c}_2 H(\alpha)$ where $\mathbf{M} := \begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{c}_1\mathbf{C} + \mathbf{m}\mathbf{G} & \mathbf{1}\end{bmatrix}$ (for some $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{m}$ chosen by the solver) without knowing a trapdoor for $\mathbf{M}$.

Assumption 3. Consider the following game between an adversary A and a challenger for fixed $m \in \mathbb{N}$ and distribution D:

1. The challenger chooses $\mathbf{a} \xleftarrow{\$} R_q$, $\mathbf{C} \xleftarrow{\$} R_q^{1\times m}$, and $\mathbf{X} \xleftarrow{\$} \mathscr{D}_{R_q,\sigma_t}^{2\times m}$. It sets $\mathbf{A} := \begin{bmatrix}\mathbf{a} & \mathbf{1}\end{bmatrix}$ and $\mathbf{B} := \mathbf{A}\mathbf{X} + \mathbf{G}$, where $\mathbf{G} = [1 \lceil q^{1/m} \rceil \ldots \lceil q^{(m-1)/m} \rceil]$.

2. The challenger runs A on input $\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{C} & \mathbf{1}\end{bmatrix}$, giving it access to a random oracle $H : \{0,1\}^* \to R_q$ and an oracle $\mathscr{O}_S$ that on input $\mathbf{m} \in \mathscr{U}$ and a string $\alpha \in \{0,1\}^*$ outputs a small vector $\binom{\mathbf{S}}{1}$ in $\mathscr{L}^\perp\left(\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{C} + \mathbf{m}\mathbf{G} & \mathbf{1}\end{bmatrix}\right) + H(\alpha)$ such that $\|\mathbf{S}\|_2 \leq N_S$.

3. Algorithm A outputs $\bar{\mathbf{m}} \in \bar{\mathscr{U}}$, $\bar{\mathbf{c}}_1, \bar{\mathbf{c}}_2 \in \bar{\mathscr{C}}$, and a vector $\bar{\mathbf{S}}$. Algorithm A wins the game if $(\bar{\mathbf{c}}_1, \bar{\mathbf{S}}, \bar{\mathbf{c}}_2) \in \bar{\Sigma}$, $\bar{\mathbf{m}} \in \bar{\mathscr{U}}$, such that $\bar{\mathbf{S}}$ is a short vector of the coset $\mathscr{L}^\perp\left(\begin{bmatrix}\mathbf{A} & \mathbf{B} & \bar{\mathbf{C}} & \mathbf{1}\end{bmatrix}\right) + \mathbf{c}_2 H(\alpha))$ where $\bar{\mathbf{C}} = \bar{\mathbf{c}}_1\mathbf{C} + \bar{\mathbf{m}}\mathbf{G}$, and $\bar{\mathbf{m}}\bar{\mathbf{c}}_1^{-1}$ was not queried to the $\mathscr{O}_S$ oracle.

The assumption states that no PPT algorithm A can win the game with non-negligible probability.

Assumption 4 (Selective variant of Assumption 3.)**.** Consider the following game between an adversary A and a challenger for fixed $m \in \mathbb{N}$ and distribution D:

1. On input only the security parameter λ, A outputs a message $\bar{\mathbf{m}} \in \bar{\mathscr{U}}$.

2. The challenger aborts if $\bar{\mathbf{m}} \notin \bar{\mathscr{U}}$. Otherwise, it chooses $\mathbf{a} \xleftarrow{\$} R_q$, $\mathbf{C} \xleftarrow{\$} R_q^{1\times m}$, and $\mathbf{X} \xleftarrow{\$} \mathscr{D}_{R_q,\sigma_t}^{2\times m}$. It sets $\mathbf{A} := \begin{bmatrix}\mathbf{a} & \mathbf{1}\end{bmatrix}$ and $\mathbf{B} := \mathbf{AX} + \mathbf{G}$, where $\mathbf{G} = [1 \lceil q^{1/m} \rceil \ldots \lceil q^{(m-1)/m} \rceil]$.

3. The challenger runs A on input $\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{C} & \mathbf{1}\end{bmatrix}$, giving it access to a random oracle $H : \{0,1\}^* \to R_q$ and an oracle $\mathscr{O}_S$ that on input $\mathbf{m} \in \mathscr{U}$ and a string $\alpha \in \{0,1\}^*$ outputs a small vector $\binom{\mathbf{S}}{1}$ in the coset $\mathscr{L}^\perp\left(\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{C}+\mathbf{mG} & \mathbf{1}\end{bmatrix}\right) + H(\alpha)$ such that $\|\mathbf{S}\|_2 \le N_S$.

4. Algorithm A outputs $\bar{\mathbf{c}}_1, \bar{\mathbf{c}}_2 \in \bar{\mathscr{C}}$, and a vector $\bar{\mathbf{S}}$. Algorithm A wins the game if $(\mathbf{c}_1, \mathbf{S}, \mathbf{c}_2) \in \bar{\Sigma}$, such that $\bar{\mathbf{S}}$ is a short vector in $\mathscr{L}^\perp\left(\begin{bmatrix}\mathbf{A} & \mathbf{B} & \bar{\mathbf{C}} & \mathbf{1}\end{bmatrix}\right) + \mathbf{c}_2 H(\alpha))$ where $\bar{\mathbf{C}} = \bar{\mathbf{c}}_1\mathbf{C} + \bar{\mathbf{m}}\mathbf{G}$, and $\bar{\mathbf{m}}\bar{\mathbf{c}}_1^{-1}$ was not queried to the $\mathscr{O}_S$ oracle.

The assumption states that no PPT algorithm A can win the game with non-negligible probability.

In the following theorem, we show that Assumption 4 is implied by the RSIS and RLWE assumptions.

Theorem 3.27 (Hardness of Assumption 4)**.** *Let* A *be a probabilistic algorithm that breaks Assumption 4 in time t with probability ϵ_A. Then there exists a probabilistic algorithm* B *that either breaks $RLWE_{m,\mathscr{D}_\sigma}$ in time t with probability ϵ_A or $RSIS_{3+m,q,\beta_s}$ where $\beta_s = \bar{N}_{s,\infty}(2+\bar{C}n) + 8n^2 m\sigma_t \bar{N}_{s,\infty}(2+\bar{C})$ in time t with probability $\epsilon_B \ge (\epsilon_A - \epsilon_{\mathrm{RLWE}})/(2\cdot|\bar{\mathscr{C}}|)$, where ϵ_{RLWE} is the probability of breaking $RLWE_{1,\chi}$ where either $\chi = \mathscr{D}_{\sigma_t}$ or $\chi = \mathscr{D}_\sigma$ problem over R_q in time t, in the Random Oracle Model.*

Proof. We construct the algorithm B as follows. Let $\bar{\mathbf{m}} \in \bar{\mathscr{U}}$ be the message output by A at the beginning of the game. Algorithm B is given a vector $\mathbf{A}' = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{m+1}] \in R_q^{1\times(2+m)}$ as RSIS challenge. To solve RSIS it should find a short vector $\mathbf{Y} \in R_q^{3+m}$ such that $\begin{bmatrix}\mathbf{A}' & \mathbf{1}\end{bmatrix}\mathbf{Y} = \mathbf{0} \bmod q$. First, B constructs $\mathbf{A} := \begin{bmatrix}\mathbf{a}_{m+1} & \mathbf{1}\end{bmatrix} \in R_q^{1\times 2}$ and $\mathbf{B} := [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_m] \in R_q^{1\times m}$, and samples $\mathbf{R}$ from $\mathscr{D}_{R_3,\sigma_t}^{2\times m}$. It guesses $\bar{\mathbf{c}}_1 \xleftarrow{\$} \bar{\mathscr{C}}$ as part of the solution of Assumption 4 that A will output in step 3. Then algorithm B constructs the public parameter as $\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{C} & \mathbf{1}\end{bmatrix} := \begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{AR} - \bar{\mathbf{m}}\bar{\mathbf{c}}_1^{-1}\mathbf{G} & \mathbf{1}\end{bmatrix}$. Finally, it sends $\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{C} & \mathbf{1}\end{bmatrix}$ to A and simulates the random oracle $\mathscr{O}_H$ and the signing oracle $\mathscr{O}_S$ as follows.

Random Oracle queries. When A makes a query $H(\alpha)$, B returns its previous response if α was already queried, otherwise it programs $H(\alpha)$ as follows. It samples $\mathbf{S} = [\mathbf{S}_1; \mathbf{S}_2; \mathbf{S}_3]$ from $D_{q,\sigma}^{2+2m}$, and programs $H(\alpha) := \begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{AR}\end{bmatrix}\mathbf{S} \bmod q$. It stores $(\alpha, \mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3)$ and returns $H(\alpha)$ to A.

Signing Oracle queries. When A makes a query to $\mathcal{O}_S$ with input $\mu = (\mathbf{m}, \alpha)$, B first checks that $\mu \in \mathcal{M}$. It then proceeds as follows:

- If $\mathbf{m} = \bar{\mathbf{c}}_1^{-1}\bar{\mathbf{m}} \bmod q$, B simulates a hash query $\mathcal{O}_H(\alpha)$ as described above and reads the corresponding $\mathbf{S}_1$, $\mathbf{S}_2$, $\mathbf{S}_3$ from the list. Then it returns $\sigma := (\mathbf{1}, [\mathbf{S}_1; \mathbf{S}_2; \mathbf{S}_3; \mathbf{0}], \mathbf{1})$. Remark that $(\mathbf{m}, \alpha) \in \mathcal{M}$ implies $\mathbf{m} = \bar{\mathbf{c}}_1^{-1}\bar{\mathbf{m}} \bmod q \in \mathcal{U}$.
- If $\mathbf{m} \neq \bar{\mathbf{c}}_1^{-1}\bar{\mathbf{m}} \bmod q$, B queries $H(\alpha)$ to $\mathcal{O}_H$. It samples $\mathbf{S}_2 \xleftarrow{\$} \mathcal{D}_\sigma^m$ and sets $\mathbf{v} = \mathbf{B}\mathbf{S}_2 \bmod q$. Finally, it samples $[\mathbf{S}_1;\ \mathbf{S}_3]$ from $D^{\perp}_{[\mathbf{A}\ \mathbf{AR}+(\mathbf{m}-\bar{\mathbf{c}}_1^{-1}\bar{\mathbf{m}})\mathbf{G}], H(\alpha)-\mathbf{v}, \sigma}$ using $\mathbf{R}$ as trapdoor, and it returns $\sigma := (\mathbf{1}, [\mathbf{S}_1; \mathbf{S}_2; \mathbf{S}_3; \mathbf{0}], \mathbf{1})$. Given that σ satisfies the hypothesis of Theorem 2.17, this sampling procedure is possible if $\mathbf{m}-\bar{\mathbf{c}}_1^{-1}\bar{\mathbf{m}} \equiv_q \bar{\mathbf{c}}_1^{-1}(\bar{\mathbf{c}}_1\mathbf{m}-\bar{\mathbf{m}})$ is invertible. This is true by Lemma 2.11 as the numerator has infinity norm bound by

$$\begin{aligned}\|\bar{\mathbf{c}}_1\mathbf{m}-\bar{\mathbf{m}}\|_\infty &\leq \|\bar{\mathbf{c}}_1\mathbf{m}\|_\infty + \|\bar{\mathbf{m}}\|_\infty \\ &\leq \|\bar{\mathbf{c}}_1\|_\infty\|\mathbf{m}\|_1 + \|\bar{\mathbf{m}}\|_\infty \\ &\leq 2\cdot 1\cdot(2^{K_m}-1) + r \\ &< \sqrt{q/2}\end{aligned}$$

where the second inequality holds by Lemma 2.12 as $2\cdot 1\cdot(2^{K_m}-1) < (q-1)/2$. As a side note, this is compatible with the bounds imposed by the commitment scheme (cf. Section 3.5.4). The oracle returns $\sigma := (\mathbf{1}, [\mathbf{S}_1;\ \mathbf{S}_2;\ \mathbf{S}_3;\ \mathbf{0}], \mathbf{1})$.

B is computationally indistinguishable from the challenger in Assumption 4 under RLWE. This result is summarized in Theorem 3.28.

Upon receiving a valid pair $\big((\bar{\mathbf{m}}, \alpha'), (\mathbf{c}_1', \mathbf{S}', \mathbf{c}_2')\big)$ from A, B aborts if $\mathbf{c}_1'$ is not the value that it guessed before. Otherwise, substituting $\mathbf{C} = \mathbf{AR}-{\mathbf{c}_1'}^{-1}\bar{\mathbf{m}}\mathbf{G} \bmod q$ in $\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{c}_1'\mathbf{C}+\bar{\mathbf{m}}\mathbf{G} & \mathbf{1}\end{bmatrix}\mathbf{S}' = \mathbf{c}_2' H(\alpha') \bmod q$ yields:

$$\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{c}_1'\mathbf{AR} & \mathbf{1}\end{bmatrix}\mathbf{S}' = \mathbf{c}_2' H(\alpha') \bmod q \tag{3.12}$$

as $\mathbf{c}_1'\mathbf{C}+\bar{\mathbf{m}}\mathbf{G} = \mathbf{c}_1'\mathbf{AR} - \mathbf{c}_1'\bar{\mathbf{m}}{\mathbf{c}_1'}^{-1}\mathbf{G} + \bar{\mathbf{m}}\mathbf{G} = \mathbf{c}_1'\mathbf{AR}$.
Now, algorithm B simulates a query $\mathcal{O}_H(\alpha')$ and recovers $(\alpha, \mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3)$ from the

list. For $\mathbf{T} = [\mathbf{T}_1; \mathbf{T}_2] = [\mathbf{S}_1 + \mathbf{R}\mathbf{S}_3; \mathbf{S}_2]$ we have that $\begin{bmatrix}\mathbf{A} & \mathbf{B}\end{bmatrix}\mathbf{T} = H(\alpha') \bmod q$. Combining this with Equation (3.12) and decomposing $\mathbf{S}'$ as $\mathbf{S}' = [\mathbf{S}'_1; \mathbf{S}'_2; \mathbf{S}'_3; \mathbf{s}'_4]$ yields:

$$\begin{bmatrix}\mathbf{B} & \mathbf{A}\end{bmatrix}\begin{pmatrix}\mathbf{S}'_1 + \mathbf{c}'_1\mathbf{R}\mathbf{S}'_3 - \mathbf{c}'_2\mathbf{T}_1 + \begin{pmatrix}\mathbf{0}\\ \mathbf{s}'_4\end{pmatrix}\\ \mathbf{S}'_2 - \mathbf{c}'_2\mathbf{T}_2\end{pmatrix} = \mathbf{0} \bmod q \ . \tag{3.13}$$

Denote the column vector on the left-hand side as $\mathbf{V} = [\mathbf{V}_1; \mathbf{V}_2]$.

If $\mathbf{V} \neq \mathbf{0}_{(m+3)\times 1} \bmod q$, then we have that $\begin{bmatrix}\mathbf{B} & \mathbf{A}\end{bmatrix}\mathbf{V} = \begin{bmatrix}\mathbf{A}' & \mathbf{1}\end{bmatrix}\mathbf{V} = \mathbf{0} \bmod q$, so that B obtained a solution for RSIS with norm bounded by β_s. By the triangular inequality, Lemma 2.12 the following bound holds:

$$\begin{aligned}
\|\mathbf{V}\|_\infty &\leq \max\left\{\|\mathbf{S}'_1 + \bar{\mathbf{c}}_1\mathbf{R}\mathbf{S}'_3 - \bar{\mathbf{c}}_2\mathbf{T}_1 + \begin{pmatrix}\mathbf{0}\\ \mathbf{s}'_4\end{pmatrix}\|_\infty, \|\mathbf{S}'_2 - \bar{\mathbf{c}}_2\mathbf{T}_2\|_\infty\right\}\\
&\leq \|\mathbf{S}'_1 + \mathbf{c}'_1\mathbf{R}\mathbf{S}'_3 - \mathbf{c}'_2\mathbf{T}_1 + \begin{pmatrix}\mathbf{0}\\ \mathbf{s}'_4\end{pmatrix}\|_\infty\\
&\leq \|\mathbf{S}'_1\|_\infty + \|\mathbf{c}'_1\mathbf{R}\bar{\mathbf{S}}_3\|_\infty + \|\mathbf{c}'_2\mathbf{S}_1\|_\infty + \|\mathbf{c}'_2\mathbf{R}\mathbf{S}_3\|_\infty + \|\mathbf{s}'_4\|_\infty\\
&\leq \bar{N}_{s,\infty} + n^2m \cdot 2 \cdot (8\sigma_t) \cdot \bar{N}_{s,\infty} + \bar{C}\bar{N}_{s,\infty}n + n^2m \cdot \bar{C} \cdot (8\sigma_t) \cdot \bar{N}_{s,\infty} + \bar{N}_{s,\infty}\\
&= \bar{N}_{s,\infty}(2 + \bar{C}n) + 8n^2m\sigma_t\bar{N}_{s,\infty}(2 + \bar{C})\\
&=: \beta_s
\end{aligned}$$

where we can use Lemma 2.12 as $8n^2m\bar{C}\sigma_t\bar{N}_{s,\infty} \leq \frac{q-1}{2}$. Hence, $\|\mathbf{V}\|_\infty \leq \beta$.

Finally, we need to check that the vector $\mathbf{V}$ is nonzero. If $\mathbf{V} = \mathbf{0}_{(m+3)\times 1} \bmod q$, then we have in particular that $\mathbf{V}_1 = \mathbf{0}_{2\times 1} \bmod q$ and $\mathbf{V}_2 = \mathbf{0}_{m\times 1} \bmod q$, from which it follows that $\mathbf{S}'_1 + \mathbf{c}'_1\mathbf{R}\mathbf{S}'_3 + \binom{\mathbf{0}}{\mathbf{s}'_4} = \mathbf{c}'_2\mathbf{T}_1 \bmod q$ and $\mathbf{S}'_2 = \mathbf{c}'_2\mathbf{T}_2 \bmod q$. Multiplying both sides of both equations with $\mathbf{c}_2'^{-1}$ yields $\mathbf{T}_1 = \mathbf{c}_2'^{-1}(\mathbf{S}'_1 + \mathbf{c}'_1\mathbf{R}\mathbf{S}'_3 + \binom{\mathbf{0}}{\mathbf{s}'_4}) \bmod q$ and $\mathbf{T}_2 = \mathbf{c}_2'^{-1}\mathbf{S}'_2 \bmod q$. Recall that $\|\mathbf{c}'_2\|_2 < q/2$ and q is a prime, $q \equiv 5 \bmod 8$, therefore $\mathbf{c}'$ is invertible by Lemma 2.11. Now, consider the deterministic function $h : R_q^{2+m} \to R_q$ where $h(\mathbf{Y}) := \begin{bmatrix}\mathbf{A} & \mathbf{B}\end{bmatrix}\mathbf{Y} \bmod q$. By Lemma 2.5, for a randomly chosen $\mathbf{Y}$ there exists with probability at least $1 - 2^{-\lambda}$ another vector $\mathbf{Y}' \neq \mathbf{Y}$ such that $h(\mathbf{Y}) = h(\mathbf{Y}')$. The parameter λ should be such that $|R_q^{m+2}| \geq 2^\lambda |R_q|$, thus $\lambda \leq \log q \cdot n(m+1)$. Moreover, A's view is (computationally) independent of B's choice for $\mathbf{T}$, because only its image $h(\mathbf{T}) = \begin{bmatrix}\mathbf{A} & \mathbf{B}\end{bmatrix}\mathbf{T} = H(\alpha')$ was output by the hash function and because $\mathbf{T}$ was never used to simulate a query to $\mathcal{O}_S$. Indeed, the only query that would involve $\mathbf{T}$ in the simulation is $\mathcal{O}_S((\bar{\mathbf{m}}\mathbf{c}_1'^{-1}, \alpha'))$:

- if $\bar{\mathbf{m}}\mathbf{c}_1'^{-1} \in \mathcal{U}$, A never queried for $(\bar{\mathbf{m}}\mathbf{c}_1'^{-1}, \alpha')$ (otherwise the output of A would not be a valid solution);

- if $\bar{\mathbf{m}}\mathbf{c}_1'^{-1} \notin \mathcal{U}$ this query never happened, as $(\bar{\mathbf{m}}\mathbf{c}_1'^{-1}, \alpha')$ would not be accepted as input by $\mathcal{O}_S$.

Hence, the probability that A outputs a $\mathbf{T}' = [\mathbf{c}_2'^{-1}(\mathbf{S}_1' + \mathbf{c}_1'\mathbf{R}\mathbf{S}_3') \; ; \; \mathbf{c}_2'^{-1}\mathbf{S}_2']$ such that $\mathbf{T}' = \mathbf{T}$ is at most $\frac{1}{2}$.

Therefore, B outputs a nonzero solution of RSIS with probability $\epsilon_B \geq \frac{\epsilon_A - \epsilon_{\text{RLWE}}}{2 \cdot |\hat{\mathscr{C}}|}$ in time t, where ϵ_{RLWE} is the probability of breaking the RLWE problem over R_q in time t. □

In the following we prove that an algorithm A against Assumption 4 cannot distinguish B from a challenger behaving as specified by Assumption 4. The result is stated in the following theorem.

Theorem 3.28. *Assume there exists an adversary* A *is able to distinguish the simulator* B *in the proof of Theorem 3.27 from a honest signer. Then there exists a distinguisher* D *that can solve RLWE*$_{1,\chi}$ *where either* $\chi = \mathscr{D}_{\sigma_t}$ *or* $\chi = \mathscr{D}_{\sigma}$ *exploiting* A *with advantage* $\epsilon_{\mathsf{D}} \geq m/4\epsilon_{\mathsf{A}}$

Indistinguishability relies on RLWE, that is a computational assumption. It would be also possible to have statistical indistinguishability by adapting the results from [Gentry et al., 2008] to rings, but this would require $\mathbf{A}$ to have a larger dimension (namely, $\mathbf{A}$ should be in $R_q^{1\times\ell}$ where $\ell \geq 2\log(q)$ with the given parameters settings).

To make the proof easier to understand, we split it in two different lemmas, Lemma 3.30 and Lemma 3.31. We start by proving a warm-up lemma, that shows that A cannot distinguish the public parameters generated by the simulator B from honestly generated ones. The lemma itself is not used in the proof of Theorem 3.28, but the techniques used to prove it are recycled when proving the theorem.

Lemma 3.29. *Let n be a power of 2, q a prime such that $q \equiv 5$ mod 8, r is defined as in Section 3.6.3, and $\sigma_t > 0$. If there exists a PPT algorithm* A *that can win the experiment in Figure 3.4 with advantage ϵ_{A}, then there exists a PPT algorithm* B *that can solve RLWE*$_{1,\chi}$ *where $\chi = \mathscr{D}_{\sigma_t}$ exploiting* A *with advantage $\epsilon_{\mathsf{B}} \geq \frac{m}{2}\epsilon_{\mathsf{A}}$.*

Proof. We prove the result with a sequence of indistinguishable game hops. Let A be a distinguisher with advantage $\epsilon_{\mathsf{A}} = |\mathsf{Pr}[1 \leftarrow \mathsf{Exp}_{\mathsf{A}}^1(1^\lambda)] - \mathsf{Pr}[1 \leftarrow \mathsf{Exp}_{\mathsf{A}}^0(1^\lambda)]|$,. Denote by $\mathbf{view}_i$ the probability distribution of the view of the adversary in Game i, i.e., of the triple $(\mathbf{A}, \mathbf{B}, \mathbf{C})$, and by $\mathbf{Game_i}$ the event that A returns 1 at the end of Game i.

Experiment $\mathsf{Exp}^{b}_{\mathsf{A}}(1^{\lambda})$
$spar \leftarrow \mathsf{SParGen}(1^{\lambda})$
$ct_b \leftarrow 0$
$b' \leftarrow \mathsf{A}^{\mathscr{D}_b}(spar)$
Return b'.

Oracle $\mathscr{D}_0(\bar{\mathbf{m}})$:
If $ct_0 \geq 1 \ \vee \ \bar{\mathbf{m}} \notin R_{2r+1}^{(2^{K_m})}$ abort.
$ct_0 \leftarrow ct_0 + 1$
$\mathbf{a} \xleftarrow{\$} \mathscr{U}(R_q)$
$\mathbf{A} \leftarrow \begin{bmatrix} \mathbf{a} & \mathbf{1} \end{bmatrix}$
$\mathbf{R} \xleftarrow{\$} \mathscr{D}^{2\times m}_{R_q,\sigma_t}$
$\mathbf{C} \xleftarrow{\$} \mathscr{U}(R_q^{1\times m})$
$\mathbf{B} \leftarrow \mathbf{AR} + \mathbf{G} \bmod q$
Return $(\mathbf{A}, \mathbf{B}, \mathbf{C})$.

Oracle $\mathscr{D}_1(\bar{\mathbf{m}})$
If $ct_1 \geq 1 \ \vee \ \bar{\mathbf{m}} \notin R_{2r+1}^{(2^{K_m})}$ abort.
$ct_1 \leftarrow ct_1 + 1$
$\mathbf{a} \xleftarrow{\$} \mathscr{U}(R_q)$
$\mathbf{A} \leftarrow \begin{bmatrix} \mathbf{a} & \mathbf{1} \end{bmatrix}$
$\mathbf{B} \xleftarrow{\$} \mathscr{U}(R_q^{1\times m})$
$\mathbf{R} \xleftarrow{\$} \mathscr{D}^{2\times m}_{R_q,\sigma_t}$
$\mathbf{c} \xleftarrow{\$} \mathscr{U}(\bar{\mathscr{C}})$
$\mathbf{C} \leftarrow \mathbf{AR} - \bar{\mathbf{m}}\mathbf{c}^{-1}\mathbf{G} \bmod q$
Return $(\mathbf{A}, \mathbf{B}, \mathbf{C})$.

Figure 3.4. Indistinguishability experiment from Lemma 3.29.

Game 0. Game 0 is exactly $\mathsf{Exp}^{0}_{\mathsf{A}}(1^{\lambda})$. Then $\mathsf{Pr}[1 \leftarrow \mathsf{Exp}^{0}_{\mathsf{A}}(1^{\lambda})] = \mathsf{Pr}[\mathbf{Game_0}]$.

Game 1. Game 1 is equal to Game 0 except for the generation of $\mathbf{C}$, that it is now set to $\mathbf{C} = \mathbf{U} + \mathbf{G} \bmod q$, where $\mathbf{U} \xleftarrow{\$} R_q^{1\times m}$. The vector $\mathbf{C}$ is still uniformly distributed, hence $\mathbf{view}_1 \approx_s \mathbf{view}_0$, i.e., $\mathsf{Pr}[\mathbf{Game_0}] = \mathsf{Pr}[\mathbf{Game_1}]$.

Game 2. Game 2 is equal to Game 1 except for the generation of $\mathbf{C}$, that is now set to be $\mathbf{C} := \mathbf{AR} - \bar{\mathbf{m}}\mathbf{c}^{-1}\mathbf{G}$ where $\mathbf{R} \xleftarrow{\$} \mathscr{D}^{2\times m}_{R_q,\sigma_t}$, $\bar{\mathbf{m}}$ is the element in $R_{2r+1}^{(2^{K_m})}$ chosen by A, and $\mathbf{c} \xleftarrow{\$} \bar{\mathscr{C}}$. Let A be an algorithm able to distinguish Game 2 from Game 1. We construct a PPT algorithm B_2 that can solve $\mathrm{RLWE}_{1,\chi}$ where $\chi = \mathscr{D}_{\sigma_t}$ exploiting A.

Consider the following hybrid distributions:

Step 0: $\mathbf{a} \xleftarrow{\$} R_q$,
$\mathbf{R} \leftarrow \mathscr{D}^{2\times m}_{\sigma_t}$,
$\mathbf{A} = \begin{bmatrix} \mathbf{a} & \mathbf{1} \end{bmatrix}$,
$\mathbf{B} = \mathbf{AR} \bmod q$,
$\mathbf{C} = \mathbf{U} - \bar{\mathbf{m}}\mathbf{c}^{-1}\mathbf{G} \bmod q$, $\mathbf{U} \xleftarrow{\$} R_q^{1\times m}$ (as in Game 1).

Step 1: $\mathbf{a} \xleftarrow{\$} R_q$,
$\mathbf{R} \leftarrow \mathscr{D}^{2\times m}_{\sigma_t}$,
$\mathbf{A} = \begin{bmatrix} \mathbf{a} & \mathbf{1} \end{bmatrix}$,
$\mathbf{B} = \mathbf{AR} \bmod q$,
$\mathbf{s}_1, \mathbf{e}_1 \leftarrow \mathscr{D}_{\sigma_t}$,
$\mathbf{c}_1 = \mathbf{a}\mathbf{s}_1 + \mathbf{e}_1 \bmod q$,

$\mathbf{c}_i \xleftarrow{\$} R_q$ for $i = 2, \ldots, m$,
$\mathbf{C} = \begin{bmatrix}\mathbf{c}_1 & \ldots & \mathbf{c}_m\end{bmatrix} - \bar{\mathbf{m}}\mathbf{c}^{-1}\mathbf{G} \bmod q$.

Step k: $\mathbf{a} \xleftarrow{\$} R_q$,
$\mathbf{R} \leftarrow \mathscr{D}_{\sigma_t}^{2\times m}$,
$\mathbf{A} = \begin{bmatrix}\mathbf{a} & \mathbf{1}\end{bmatrix}$,
$\mathbf{B} = \mathbf{AR} \bmod q$,
$\mathbf{s}_i, \mathbf{e}_i \leftarrow \mathscr{D}_{\sigma_t}$ for $i = 1, \ldots, k$,
$\mathbf{c}_i = \mathbf{a}\mathbf{s}_i + \mathbf{e}_i \bmod q$ for $i = 1, \ldots, k$,
$\mathbf{c}_i \xleftarrow{\$} R_q$ for $i = k+1, \ldots, m$,
$\mathbf{C} = \begin{bmatrix}\mathbf{c}_1 & \ldots & \mathbf{c}_m\end{bmatrix} - \bar{\mathbf{m}}\mathbf{c}^{-1}\mathbf{G} \bmod q$.

For $k = m$ $\mathbf{C}$ is generated exactly as in Game 2. Distinguishing **Step k** from **Step k+1** amounts to solve RLWE. Indeed, let $(\mathbf{a}, \mathbf{b})$ be the instance B_2 gets from the RLWE oracle. B_2 runs A generating $\mathbf{C}$ as follows:

$\mathbf{R} \leftarrow \mathscr{D}_{\sigma_t}^{2\times m}$,
$\mathbf{A} = \begin{bmatrix}\mathbf{a} & \mathbf{1}\end{bmatrix}$,
$\mathbf{B} = \mathbf{AR} \bmod q$,
$\mathbf{s}_i, \mathbf{e}_i \leftarrow \mathscr{D}_{\sigma_t}$ for $i = 1, \ldots, \bar{k}$,
$\mathbf{c}_i = \mathbf{a}\mathbf{s}_i + \mathbf{e}_i \bmod q$ for $i = 1, \ldots, \bar{k}$,
$\mathbf{c}_{\bar{k}+1} = \mathbf{b}$,
$\mathbf{c}_i \xleftarrow{\$} R_q$ for $i = \bar{k}+2, \ldots, m$,
$\mathbf{C} = \begin{bmatrix}\mathbf{c}_1 & \ldots & \mathbf{c}_m\end{bmatrix} - \bar{\mathbf{m}}\mathbf{c}^{-1}\mathbf{G} \bmod q$.

It is clear that if $(\mathbf{a}, \mathbf{b})$ is sampled from the RLWE distribution then this is **Step k+1**, otherwise this is **Step k**. B_2 outputs what A outputs. The advantage of B_2 is

$$\begin{aligned}\epsilon_2 &= \left|\mathsf{Pr}[1 \leftarrow \mathsf{B}_2^{RLWE,\,\mathsf{A}}] - \mathsf{Pr}[1 \leftarrow \mathsf{B}_2^{U,\,\mathsf{A}}]\right| \\ &= \left|\mathsf{Pr}[\mathbf{Game_1} : \textbf{Step k+1}] - \mathsf{Pr}[\mathbf{Game_1} : \textbf{Step k}]\right| .\end{aligned}$$

Hence, the advantage of A in distinguishing Game 2 from Game 1 is:

$$\begin{aligned}\left|\mathsf{Pr}[\mathbf{Game_2}] - \mathsf{Pr}[\mathbf{Game_1}]\right| &\leq \sum_{k=0}^{m+1} \left|\mathsf{Pr}[\mathbf{Game_1} : \textbf{Step k}] - \mathsf{Pr}[\mathbf{Game_1} : \textbf{Step k+1}]\right| \\ &= \tfrac{1}{m}\epsilon_2 .\end{aligned}$$

Game 3. In Game 3 everything is generated as in Game 2 except for $\mathbf{B}$, that it is now generated as $\mathbf{B} = \mathbf{U} + \mathbf{G} \bmod q$, where $\mathbf{U}$ is chosen uniformly at random in $R_q^{1\times m}$. Distinguishing Game 3 from Game 2 is exactly equivalent to distinguish the distributions $(\mathbf{A}, \mathbf{U})$ and $(\mathbf{A}, \mathbf{AR})$ where $\mathbf{R}$ is a matrix sampled from $\mathscr{D}_{R_q,\sigma_t}^{2\times m}$, hence through the same hybrid argument used before we get that:

$$|\mathsf{Pr}[\mathbf{Game_3}] - \mathsf{Pr}[\mathbf{Game_2}]| \;\leq\; \frac{1}{m}\epsilon_3 \,,$$

where ϵ_3 is the advantage of a distinguisher B_3 in solving RLWE exploiting A.

Game 4. In Game 4 everything is generated as in Game 3 except for $\mathbf{B}$, that it is now generated as $\mathbf{B} = \mathbf{U} \bmod q$, where $\mathbf{U}$ is chosen uniformly at random in $R_q^{1\times m}$. The vector $\mathbf{B}$ is still uniformly distributed, hence $\mathbf{view}_4 \approx_s \mathbf{view}_3$, i.e., $\mathsf{Pr}[\mathbf{Game_3}] = \mathsf{Pr}[\mathbf{Game_4}] = \mathsf{Pr}[1 \leftarrow \mathsf{Exp}_{\mathsf{A}}^1(1^\lambda)]$.

Putting it all together yields that the advantage of A is such that

$$\begin{aligned}
\epsilon_{\mathsf{A}} &= \left|\mathsf{Pr}[1 \leftarrow \mathsf{Exp}_{\mathsf{A}}^1(1^\lambda)] - \mathsf{Pr}[1 \leftarrow \mathsf{Exp}_{\mathsf{A}}^0(1^\lambda)]\right| \\
&= \left|\mathsf{Pr}[\mathbf{Game_4}] - \mathsf{Pr}[\mathbf{Game_0}]\right| \\
&\leq \sum_{k=0}^{3} |\mathsf{Pr}[\mathbf{Game_{k+1}}] - \mathsf{Pr}[\mathbf{Game_k}]| \\
&\leq \frac{2}{m}\epsilon_{RLWE} \,,
\end{aligned}$$

where ϵ_{RLWE} is the advantage in solving $\mathrm{RLWE}_{1,\chi}$. □

Theorem 3.28 is more general than Lemma 3.29: it says that even after the querying phase, A cannot distinguish D from the challenger. We split the result in two lemmas, depending on whether A queried $(\mathbf{m}, \alpha)$ where $\mathbf{m} = \bar{\mathbf{c}_1}^{-1}\bar{\mathbf{m}} \bmod q$. We prove the lemmas separately. The full proof of Theorem 3.28 can be found afterwards.

Lemma 3.30 ($\mathbf{m} = \bar{\mathbf{c}_1}^{-1}\bar{\mathbf{m}} \bmod q$)**.** *Let n be a power of 2, q a prime such that $q \equiv 5$ mod 8, r and σ as in Section 3.6.3, and $\sigma_t > 0$. Let $\mathbf{G}$ be the gadget vector, $H : \{0,1\}^* \longmapsto R_q$ be a random element of the family of hash functions defined on $\{0,1\}$ with values in R_q. If there exists a PPT algorithm A that can win the experiment in Figure 3.5 with advantage ϵ_{A}, then there exists a PPT algorithm B that can solve $RLWE_{1,\chi}$ where either $\chi = \mathscr{D}_{\sigma_t}$ or $\chi = \mathscr{D}_{\sigma}$ exploiting A with advantage $\epsilon_{\mathsf{B}} \geq m/5\epsilon_{\mathsf{A}}$.*

Experiment $\mathsf{Exp}^b_{\mathsf{A}}(1^\lambda)$
$spar \leftarrow \mathsf{SParGen}(1^\lambda)$
$ct_b \leftarrow 0$
$b' \leftarrow \mathsf{A}^{\mathscr{D}_b,\ \mathscr{O}_{H,b},\ \mathscr{O}_{S,b}}(spar)$
Return b'.

Oracle $\mathscr{D}_0(\bar{\mathbf{m}})$:
If $ct_0 \geq 1 \ \vee\ \bar{\mathbf{m}} \notin R_{2r+1}^{(2^{\kappa_m})}$ abort.
$ct_0 \leftarrow ct_0 + 1$
$\mathbf{a} \xleftarrow{\$} \mathscr{U}(R_q)$
$\mathbf{A} \leftarrow \begin{bmatrix}\mathbf{a} & \mathbf{1}\end{bmatrix}$
$\bar{\mathbf{c}} \xleftarrow{\$} \bar{\mathscr{C}}$
$\mathbf{R} \xleftarrow{\$} \mathscr{D}^{2\times m}_{R_q,\sigma_t}$
$\mathbf{C} \xleftarrow{\$} \mathscr{U}(R_q^{1\times m})$
$\mathbf{B} \leftarrow \mathbf{AR} + \mathbf{G} \bmod q$
Return $(\mathbf{A}, \mathbf{B}, \mathbf{C})$.

Oracle $\mathscr{O}_{H,0}(\alpha)$:
If $ct_0 = 0$ abort.
Return $H(\alpha)$.

Oracle $\mathscr{O}_{S,0}(\mathbf{m}, \alpha)$:
If $ct_0 \neq 1 \ \vee\ \mathbf{m} \neq \bar{\mathbf{c}}^{-1}\bar{\mathbf{m}} \bmod q$ abort.
$ct_0 \leftarrow ct_0 + 1$
$\mathbf{S} \xleftarrow{\$} \mathscr{D}^{\perp}_{[\mathbf{A}\ \mathbf{B}\ \mathbf{C}+\mathbf{mG}],\mathscr{O}_{H,0}(\alpha),\sigma}$
Return $\sigma := (\mathbf{1}, [\mathbf{S}; \mathbf{0}], \mathbf{1})$.

Oracle $\mathscr{D}_1(\bar{\mathbf{m}})$
If $ct_1 \geq 1 \ \vee\ \bar{\mathbf{m}} \notin R_{2r+1}^{(2^{\kappa_m})}$ abort.
$ct_1 \leftarrow ct_1 + 1$
$\mathbf{a} \xleftarrow{\$} \mathscr{U}(R_q)$
$\mathbf{A} \leftarrow \begin{bmatrix}\mathbf{a} & \mathbf{1}\end{bmatrix}$
$\bar{\mathbf{c}} \xleftarrow{\$} \bar{\mathscr{C}}$
$\mathbf{B} \xleftarrow{\$} \mathscr{U}(R_q^{1\times m})$
$\mathbf{R} \xleftarrow{\$} \mathscr{D}^{2\times m}_{R_q,\sigma_t}$
$\mathbf{C} \leftarrow \mathbf{AR} - \bar{\mathbf{m}}\bar{\mathbf{c}}^{-1}\mathbf{G} \bmod q$
Return $(\mathbf{A}, \mathbf{B}, \mathbf{C})$.

Oracle $\mathscr{O}_{H,1}(\alpha)$:
If $ct_1 = 0$ abort.
$\mathbf{S} \xleftarrow{\$} \mathscr{D}^{\perp}_{R_q^{2+2m},\mathbf{0}_{2+2m},\sigma}$
$L \leftarrow L \cup \{(\alpha, \mathbf{S})\}$
Return $\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{AR}\end{bmatrix}\mathbf{S} \bmod q$.

Oracle $\mathscr{O}_{S,1}(\mathbf{m}, \alpha)$:
If $ct_1 \neq 1 \ \vee\ \mathbf{m} \neq \bar{\mathbf{c}}^{-1}\bar{\mathbf{m}} \bmod q$ abort.
$ct_1 \leftarrow ct_1 + 1$
If $\nexists \mathbf{S} \ :\ (\alpha, \mathbf{S}) \in L$ query $\mathscr{O}_{H,1}(\alpha)$.
$\mathbf{S} \ :\ (\alpha, \mathbf{S}) \in L$
Return $\sigma := (\mathbf{1}, [\mathbf{S}; \mathbf{0}], \mathbf{1})$

Figure 3.5. Indistinguishability experiment from Lemma 3.30.

Proof. We again prove the claim with a sequence of indistinguishable games hops. Let A be a distinguisher for the experiment in Figure 3.5 with advantage $\epsilon_{\mathsf{A}} = |\mathsf{Pr}[1 \leftarrow \mathsf{Exp}^1_{\mathsf{A}}(1^\lambda)] - \mathsf{Pr}[1 \leftarrow \mathsf{Exp}^0_{\mathsf{A}}(1^\lambda)]|$,. Denote by $\mathbf{view}_i$ the probability distribution of the view of the adversary in Game i, i.e., of the triple $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ and of the answers of the random oracles $\mathscr{O}_{H,b}$ and $\mathscr{O}_{S,b}$, and by $\mathbf{Game_i}$ the event that A returns 1 at the end of Game i. The first games hops are similar to the game hops in the proof of Lemma 3.29.

Game 0. In Game 0 everything is constructed as in $\mathsf{Exp}^0_{\mathsf{A}}(1^\lambda)$. Then $\mathsf{Pr}[1 \leftarrow \mathsf{Exp}^0_{\mathsf{A}}(1^\lambda)] = \mathsf{Pr}[\mathbf{Game_0}]$.

Game 1. Game 1 is equal to Game 0 except for the generation of $\mathbf{C}$, that it is now set to $\mathbf{C} = \mathbf{U} + \bar{\mathbf{c}}^{-1}\bar{\mathbf{m}}\mathbf{G} \bmod q$, where $\mathbf{U} \xleftarrow{\$} R_q^{1\times m}$. The vector $\mathbf{C}$ is still uniformly distributed, hence $\mathbf{view}_1 \approx_s \mathbf{view}_0$, i.e., $\mathsf{Pr}[\mathbf{Game_0}] = \mathsf{Pr}[\mathbf{Game_1}]$.

Game 2. In Game 2, everything is constructed as in Game 1 except for $\mathbf{C}$, which is set as $\mathbf{C} \leftarrow \mathbf{AR}' + \bar{\mathbf{c}}^{-1}\bar{\mathbf{m}}\mathbf{G} \bmod q$ for some $\mathbf{R}' \xleftarrow{\$} \mathscr{D}^{2\times m}_{R_q,\sigma_t}$. The algorithm B can still

samples from $\mathscr{D}^{\perp}_{[\mathbf{A}\ \mathbf{B}\ \mathbf{AR}'],H(\alpha),\sigma}$ using $\mathbf{R}$, which is possible thanks to Theorem 2.17 and Lemma 2.18. Game 2 is computationally indistinguishable from Game 1 if the distribution of $(\mathbf{A},\mathbf{AR})$ is computationally indistinguishable from $(\mathbf{A},\mathbf{U})$. Hence, using a hybrid argument as in the proof of Lemma 3.29, it holds that there exists an algorithm B_2 that can solve $\mathrm{RLWE}_{1,\chi}$ exploiting A with advantage ϵ_2 such that

$$|\mathsf{Pr}[\mathbf{Game_2}]-\mathsf{Pr}[\mathbf{Game_1}]| \ \leq\ \frac{1}{m}\epsilon_2\ .$$

Game 3. Game 3 is equal to Game 2 except for the generation of $\mathbf{B}$, that is set to be $\mathbf{B} \leftarrow \mathbf{U}+\mathbf{G} \bmod q$ $\mathbf{U} \xleftarrow{\$} R_q^{1\times m}$. The sampling procedure of a signature is now different: B first samples $\mathbf{S}_2 \xleftarrow{\$} \mathscr{D}_\sigma^m$, then uses $\mathbf{R}'$ to sample $[\mathbf{S}_1;\mathbf{S}_3]$ from $\mathscr{D}_{[\mathbf{A}\ \mathbf{AR}'+\mathbf{G}],\mathbf{u},\sigma}$, where $\mathbf{u}=H(\alpha)-\mathbf{B}\mathbf{S}_2 \bmod q$, and sets $\mathbf{S}=[\mathbf{S}_1;\mathbf{S}_2;\mathbf{S}_3]$. The distribution of the $\mathbf{S}$ sampled in Game 2 is the same as in Game 3: in both cases $\mathbf{S}$ is distributed as a Gaussian with standard deviation σ, conditioned on the fact that $\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{AR}'+\mathbf{G}\end{bmatrix}\mathbf{S} = H(\alpha) \bmod q$. Hence, Game 3 is computationally indistinguishable from Game 2 as long as $(\mathbf{A},\mathbf{AR})$ is indistinguishable from $(\mathbf{A},\mathbf{U})$. The same hybrid argument as before yields that there exists an algorithm B_3 that can solve $\mathrm{RLWE}_{1,\chi}$ exploiting A with advantage ϵ_3 such that

$$|\mathsf{Pr}[\mathbf{Game_3}]-\mathsf{Pr}[\mathbf{Game_2}]| \ \leq\ \frac{1}{m}\epsilon_3\ .$$

Game 4. In Game 4, everything is constructed as in Game 3 except for $\mathbf{B}$, which is now simply a randomly chosen vector, i.e., $\mathbf{B} \xleftarrow{\$} R_q^{1\times m}$. The sampling procedure of a signature can be still done as in Game 3. Game 4 is statistically indistinguishable from Game 3, as $\mathbf{B}$ and $\mathbf{B}+\mathbf{G}$ are both distributed as a uniform if $\mathbf{B}$ is, hence $\mathbf{view}_4 \approx_s \mathbf{view}_3$, i.e., $\mathsf{Pr}[\mathbf{Game_4}] = \mathsf{Pr}[\mathbf{Game_3}]$.

Game 5. In Game 5, everything is constructed as in Game 4 except for the hash. On input α, a vector $\mathbf{S} \xleftarrow{\$} \mathscr{D}_{q,\sigma}^{2+2m}$ is sampled and the output of $\mathscr{O}_H(\alpha)$ is $\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{AR}'\end{bmatrix}\mathbf{S} \bmod q$. Signing is still done using the trapdoor hidden in the public key. Again, if an adversary could distinguish this from a uniformly sampled vector, then a simulator can exploit it to distinguish $(\mathbf{A},\mathbf{AR})$ from $(\mathbf{A},\mathbf{U})$, where in this case $\mathbf{R}$ comes from a Gaussian with standard deviation σ. Thus, through a similar hybrid argument as before we get that there exists an algorithm B_5 that can solve $\mathrm{RLWE}_{1,\mathscr{D}_\sigma}$ exploiting A with advantage ϵ_5 such that

$$\left|\mathsf{Pr}[\mathbf{Game_5}]-\mathsf{Pr}[\mathbf{Game_4}]\right| \ \leq\ \frac{2}{m}\epsilon_5\ .$$

Game 6. In Game 6, $\mathbf{C}$ is constructed as $\mathbf{C} = \mathbf{AR} \bmod q$ and the oracle $\mathcal{O}_S$ is programmed so that for each input α outputs the $\mathbf{S} \xleftarrow{\$} \mathcal{D}_{q,\sigma}^{2+2m}$ sampled to calculate $H(\alpha) = \begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{AR}\end{bmatrix}\mathbf{S} \bmod q$. The vector $\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{AR}\end{bmatrix}$ in Game 6 is indistinguishable from the vector in Game 5 if $(\mathbf{A}, \mathbf{AR})$ is indistinguishable from $(\mathbf{A}, \mathbf{U})$. The distribution of the outputs of $\mathcal{O}_S$ is indistinguishable from the distribution of $\mathbf{S}$ sampled from $\mathcal{D}^{\perp}_{[\mathbf{A}\ \mathbf{B}\ \mathbf{AR}'+\mathbf{G}],H(\alpha),\sigma}$. Indeed Lemma 5.2 in [Gentry et al., 2008] adapted to rings guarantees that in both cases the distribution of the output $\mathbf{S}$ is $D_{R_q^{2+2m},\mathbf{0}_{2+2m},\sigma}$. Hence the hybrid argument used in Game 2 yields that there exists an algorithm B_6 that can solve $\mathrm{RLWE}_{1,\chi}$ exploiting A with advantage ϵ_6 such that it yields,

$$|\mathsf{Pr}[\mathbf{Game_6}] - \mathsf{Pr}[\mathbf{Game_5}]| \leq \frac{1}{m}\epsilon_6 .$$

Moreover,

$$\mathsf{Pr}[\mathbf{Game_6}] = \mathsf{Pr}[1 \leftarrow \mathsf{Exp}^1_{\mathsf{A}}(1^\lambda)] .$$

Putting it all together yields that the advantage of A is such that

$$\begin{aligned}
\epsilon_{\mathsf{A}} &= \left|\mathsf{Pr}[1 \leftarrow \mathsf{Exp}^1_{\mathsf{A}}(1^\lambda)] - \mathsf{Pr}[1 \leftarrow \mathsf{Exp}^0_{\mathsf{A}}(1^\lambda)]\right| \\
&= |\mathsf{Pr}[\mathbf{Game_6}] - \mathsf{Pr}[\mathbf{Game_0}]| \\
&\leq \sum_{k=0}^{5} |\mathsf{Pr}[\mathbf{Game_{k+1}}] - \mathsf{Pr}[\mathbf{Game_k}]| \\
&\leq \frac{5}{m}\epsilon_{RLWE} ,
\end{aligned}$$

where ϵ_{RLWE} is the advantage in solving $\mathrm{RLWE}_{1,\chi}$. □

Lemma 3.31 ($\mathbf{m} \neq \bar{\mathbf{c}}_1^{-1}\bar{\mathbf{m}} \bmod q$)**.** *Let n be a power of 2, q a prime such that $q \equiv 5$ mod 8, r and σ as in Section 3.6.3, and $\sigma_t > 0$. Let $\mathbf{G}$ be the gadget vector, $H : \{0,1\}^* \longmapsto R_q$ be a random element of the family of hash functions defined on $\{0,1\}$ with values in R_q.*

If there exists a PPT algorithm A that can win the experiment in Figure 3.6 with advantage ϵ_{A}, then there exists a PPT algorithm B that can solve $RLWE_{1,\chi}$ where either $\chi = \mathcal{D}_{\sigma_t}$ or $\chi = \mathcal{D}_{\sigma}$ exploiting A with advantage $\epsilon_{\mathsf{B}} \geq m/4\epsilon_{\mathsf{A}}$.

Proof. We again prove the claim with a sequence of indistinguishable games hops. Let A be a distinguisher for the experiment in Figure 3.6 with advantage $\epsilon_{\mathsf{A}} = |\mathsf{Pr}[1 \leftarrow \mathsf{Exp}^1_{\mathsf{A}}(1^\lambda)] - \mathsf{Pr}[1 \leftarrow \mathsf{Exp}^0_{\mathsf{A}}(1^\lambda)]|$,. Denote by $\mathbf{view}_i$ the probability distribution of the view of the adversary in Game i, i.e., of the triple $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ and of the answers of the random oracles $\mathcal{O}_{H,b}$ and $\mathcal{O}_{S,b}$, and by $\mathbf{Game_i}$ the event that A returns 1 at the end of Game i. The first games hops are similar to the game hops in the proof of Lemma 3.29.

Experiment $\mathsf{Exp}^{b}_{\mathsf{A}}(1^\lambda)$
$spar \leftarrow \mathsf{SParGen}(1^\lambda)$
$ct_b \leftarrow 0$
$\bar{\mathbf{m}} \xleftarrow{\$} R^{(2^{Km})}_{2r+1}$
$\bar{\mathbf{c}}_1 \xleftarrow{\$} \bar{\mathscr{C}}$
$b' \leftarrow \mathsf{A}^{\mathscr{D}_b,\ \mathscr{O}_{H,b},\ \mathscr{O}_{S,b}}(spar)$
Return b'.

Oracle $\mathscr{D}_{0,\bar{\mathbf{m}},\bar{\mathbf{c}}_1}$:
If $ct_0 \geq 1$ abort.
$ct_0 \leftarrow ct_0 + 1$
$\mathbf{a} \xleftarrow{\$} \mathscr{U}(R_q)$
$\mathbf{A} \leftarrow \begin{bmatrix}\mathbf{a} & \mathbf{1}\end{bmatrix}$
$\mathbf{R} \xleftarrow{\$} \mathscr{D}^{2\times m}_{R_q,\sigma_t}$
$\mathbf{C} \xleftarrow{\$} \mathscr{U}(R_q^{1\times m})$
$\mathbf{B} \leftarrow \mathbf{AR} + \mathbf{G} \bmod q$
Return $(\mathbf{A}, \mathbf{B}, \mathbf{C})$.

Oracle $\mathscr{O}_{H,0}(\alpha)$:
If $ct_0 = 0$ abort.
Return $H(\alpha)$.

Oracle $\mathscr{O}_{S,0}(\mathbf{m}, \alpha)$:
If $ct_0 \neq 1 \ \vee\ \mathbf{m} = \bar{\mathbf{c}}_1^{-1}\bar{\mathbf{m}} \bmod q$ abort.
$\mathbf{S} \xleftarrow{\$} \mathscr{D}^{\perp}_{[\mathbf{A}\ \mathbf{B}\ \mathbf{C}+\mathbf{mG}],\mathscr{O}_{H,0}(\alpha),\sigma}$
Return $\sigma := (\mathbf{1}, [\mathbf{S}; \mathbf{0}], \mathbf{1})$.

Oracle $\mathscr{D}_{1,,\bar{\mathbf{m}},\bar{\mathbf{c}}_1}$
If $ct_1 \geq 1$ abort.
$ct_1 \leftarrow ct_1 + 1$
$\mathbf{a} \xleftarrow{\$} \mathscr{U}(R_q)$
$\mathbf{A} \leftarrow \begin{bmatrix}\mathbf{a} & \mathbf{1}\end{bmatrix}$
$\mathbf{B} \xleftarrow{\$} \mathscr{U}(R_q^{1\times m})$
$\mathbf{R} \xleftarrow{\$} \mathscr{D}^{2\times m}_{R_q,\sigma_t}$
$\mathbf{C} \leftarrow \mathbf{AR} - \bar{\mathbf{m}}\bar{\mathbf{c}}^{-1}\mathbf{G} \bmod q$
Return $(\mathbf{A}, \mathbf{B}, \mathbf{C})$.

Oracle $\mathscr{O}_{H,1}(\alpha)$:
If $ct_1 = 0$ abort.
$\mathbf{S} \xleftarrow{\$} \mathscr{D}^{\perp}_{R_q^{2+2m},\mathbf{0}_{2+2m},\sigma}$
$L \leftarrow L \cup \{(\alpha, \mathbf{S})\}$
Return $\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{AR}\end{bmatrix}\mathbf{S} \bmod q$.

Oracle $\mathscr{O}_{S,1}(\mathbf{m}, \alpha)$:
If $ct_1 \neq 1 \ \vee\ \mathbf{m} = \bar{\mathbf{c}}_1^{-1}\bar{\mathbf{m}} \bmod q$ abort.
$ct_1 \leftarrow ct_1 + 1$
If $\nexists \mathbf{S} \ :\ (\alpha, \mathbf{S}) \in L$ query $\mathscr{O}_{H,1}(\alpha)$.
$\mathbf{S}_2 \xleftarrow{\$} \mathscr{D}^{m}_{\sigma}$
$\mathbf{v} \leftarrow \mathbf{BS}_2 \bmod q$
$[\mathbf{S}_1;\ \mathbf{S}_3] \xleftarrow{\$} D^{\perp}_{[\mathbf{A}\ \mathbf{AR}+(\mathbf{m}-\bar{\mathbf{c}}_1^{-1}\bar{\mathbf{m}})\mathbf{G}],H(\alpha)-\mathbf{v},\sigma}$
Return $\sigma := (\mathbf{1}, [\mathbf{S}_1; \mathbf{S}_2; \mathbf{S}_3; \mathbf{0}], \mathbf{1})$.

Figure 3.6. Indistinguishability experiment from Lemma 3.31.

Game 0. In Game 0 everything is constructed as in $\mathsf{Exp}^0_{\mathsf{A}}(1^\lambda)$. Then $\mathsf{Pr}[1 \leftarrow \mathsf{Exp}^0_{\mathsf{A}}(1^\lambda)] = \mathsf{Pr}[\mathbf{Game_0}]$.

Game 1. In Game 1 B generates $\mathbf{C}$ as $\mathbf{U} - \mathbf{m}^*\mathbf{c}^{*-1}\mathbf{G} \bmod q$, where $\mathbf{U} \xleftarrow{\$} R_q^{1\times m}$. Since $\mathbf{U}$ is uniformly random, so is $\mathbf{U} - \mathbf{m}^*\mathbf{c}^{*-1}\mathbf{G}$, hence $\mathbf{view}_1 \approx_s \mathbf{view}_0$, i.e., $\mathsf{Pr}[\mathbf{Game_0}] = \mathsf{Pr}[\mathbf{Game_1}]$.

Game 2. In Game 2 B generates $\mathbf{C}$ as $\mathbf{C} \leftarrow \mathbf{AR}' - \mathbf{m}^*\mathbf{c}^{*-1}\mathbf{G} \bmod q$), where $\mathbf{R}'$ has the same distribution as $\mathbf{R}$. If there is a polynomial-time algorithm that can distinguish Game 2 from Game 1, then, the hybrid argument used in Lemma 3.30 yields that an algorithm B_2 that can solve $\mathrm{RLWE}_{1,\chi}$ exploiting A has advantage ϵ_2 such that

$$|\mathsf{Pr}[\mathbf{Game_2}] - \mathsf{Pr}[\mathbf{Game_1}]| \ \leq\ \frac{1}{m}\epsilon_2 \ .$$

Game 3. Game 3 is exactly like Game 2 except $\mathscr{O}_S$ is now implemented by using $\mathbf{R}'$ as the "trapdoor". Since $\mathbf{m} \neq \mathbf{m}^*\mathbf{c}^{*-1}$ B can again use Theorem 2.17 to

generate an element from the distribution $\mathscr{D}^{\perp}_{[\mathbf{A}\ \mathbf{AR}+\mathbf{G}\ \mathbf{AR}'-(\mathbf{m}-\mathbf{m}^*\mathbf{c}^{*-1})\mathbf{G}],H\alpha,\sigma}$ using $\mathbf{R}'$ as a trapdoor as long as $\mathbf{m}-\mathbf{m}^*\mathbf{c}^{*-1}$ is invertible. By Lemma 2.18 we know that using $\mathbf{R}$ or $\mathbf{R}'$ as a trapdoor produces the same distribution. Thus the distributions of the outputs of Games 2 and 3 are the same, i.e., $\mathbf{view}_3 \approx_s \mathbf{view}_2$ and $\mathsf{Pr}[\mathbf{Game_3}] = \mathsf{Pr}[\mathbf{Game_2}]$.

Game 4. In Game 4 B generates $\mathbf{B}$ as $\mathbf{B} \xleftarrow{\$} R_q^{1\times m}$. If there is algorithm that can distinguish between Game 3 and 4, then the hybrid argument yields that an algorithm B_4 that can solve $\mathrm{RLWE}_{1,\chi}$ exploiting A has advantage ϵ_4 such that

$$\left|\mathsf{Pr}[\mathbf{Game_4}] - \mathsf{Pr}[\mathbf{Game_3}]\right| \ \leq \ \frac{1}{m}\epsilon_4 \ .$$

Game 5. Finally, in Game 5 B implements everything as in Game 4 except for the hash oracle. On input α, a vector $\mathbf{S} \xleftarrow{\$} \mathscr{D}^{2+2m}_{q,\sigma}$ is sampled and the output of $\mathscr{O}_H(\alpha)$ is $\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{AR}'\end{bmatrix}\mathbf{S} \bmod q$. Again, if an adversary could distinguish this from a uniformly sampled vector, then a simulator can exploit it to distinguish $(\mathbf{A}, \mathbf{AR})$ from $(\mathbf{A}, \mathbf{U})$ where now $\mathbf{R}$ comes from a Gaussian with standard deviation σ. Thus, through a similar hybrid argument as before we get that there exists an algorithm B_5 that can solve $\mathrm{RLWE}_{1,\mathscr{D}_\sigma}$ exploiting A with advantage ϵ_5 such that

$$\left|\mathsf{Pr}[\mathbf{Game_5}] - \mathsf{Pr}[\mathbf{Game_4}]\right| \ \leq \ \frac{2}{m}\epsilon_5 \ .$$

Observe that Game 5 is exactly $\mathsf{Exp}^1_{\mathsf{A}}(1^\lambda)$, hence

$$\mathsf{Pr}[\mathbf{Game_6}] = \mathsf{Pr}[1 \leftarrow \mathsf{Exp}^1_{\mathsf{A}}(1^\lambda)] \ .$$

Putting it all together yields that the advantage of A is such that

$$\begin{aligned}
\epsilon_{\mathsf{A}} &= \left|\mathsf{Pr}[1 \leftarrow \mathsf{Exp}^1_{\mathsf{A}}(1^\lambda)] - \mathsf{Pr}[1 \leftarrow \mathsf{Exp}^0_{\mathsf{A}}(1^\lambda)]\right| \\
&= \left|\mathsf{Pr}[\mathbf{Game_6}] - \mathsf{Pr}[\mathbf{Game_0}]\right| \\
&\leq \sum_{k=0}^{5} \left|\mathsf{Pr}[\mathbf{Game_{k+1}}] - \mathsf{Pr}[\mathbf{Game_k}]\right| \\
&\leq \frac{4}{m}\epsilon_{RLWE} \ ,
\end{aligned}$$

where ϵ_{RLWE} is the advantage in solving $\mathrm{RLWE}_{1,\chi}$. □

Finally, Theorem 3.32 states that breaking Assumption 4 implies breaking Assumption 3. It follows from a straightforward complexity leveraging argument by guessing the polynomial $\bar{\mathbf{m}} \in \bar{\mathscr{U}}$.

Theorem 3.32. *Let* A *be a probabilistic algorithm that breaks Assumption 3 in time* t *with probability* ϵ_A*. Then, there exists a probabilistic algorithm* B *that breaks Assumption 4 in time* t *with probability* $\epsilon_B \leq \epsilon_A/|\bar{\mathscr{U}}|$ *in the Random Oracle Model.*

We define the g-euf-acma security of the relaxed signature scheme with respect to the message relaxation function

$$g(\mathbf{m}, \alpha) = \{(\bar{\mathbf{m}}, \alpha) : \bar{\mathbf{m}} \in f(\mathbf{m})\},$$

where the function f is as defined in Equation (3.6). A valid forgery is a signature $(\bar{\mathbf{c}}_1, \bar{\mathbf{S}}, \bar{\mathbf{c}}_2)$ on some message $(\bar{\mathbf{m}}, \alpha')$ such that the adversary never saw a signature on $(\mathbf{m}, \alpha')$ for any $\mathbf{m}$ such that $\bar{\mathbf{m}} \in f(\mathbf{m})$. The unforgeability of the signature scheme follows directly from Assumption 3.

Theorem 3.33. *An algorithm* A *that breaks the g-euf-acma unforgeability of the relaxed signature scheme in time* t *and probability* ϵ_A *can break the Assumption 3 in time* t *with probability* ϵ_A *in the Random Oracle Model.*

A valid forgery can be used to break Assumption 3 because unforgeability is defined w.r.t. a function g. This guarantees that $\bar{\mathbf{m}}$ and $\bar{\mathbf{c}}_1$ output by A are such that $\bar{\mathbf{m}}\bar{\mathbf{c}}_1^{-1}$ was not queried to $\mathscr{O}_S$ as specified by the assumption.

3.7 Putting it all together: Relaxed Proofs of Signatures on Committed Messages

The previous three relaxed primitives can be composed together to build a relaxed NIZK $(\mathsf{P}_{pt}, \mathsf{V}_{pt})$ to prove knowledge of a signature $\mathbf{S}$ on a message $(\mathbf{m}, \alpha)$, where $\mathbf{m}$ is kept secret while α is public. To hide both $\mathbf{S}$ and $\mathbf{m}$, the protocol exploits the relaxed commitment scheme defined in Section 3.5.4. The commitment is needed both for technical and practical reasons, as it allows to prove knowledge of the signature and the secret part of the message in two separate equations. On one hand, this gives a better bound on the extracted message, as rejection sampling can be performed separately on the two equations. On the other hand, this allows to prove knowledge of a set of signatures $\{\mathbf{S}_i\}_{i=1,\ldots,\ell}$ on messages $(\mathbf{m}, \alpha_i)$ for $i = 1\ldots,\ell$, i.e. on message pairs composed by the same secret $\mathbf{m}$ and by different public bit-strings α_i.

We start presenting the proof for a single pair message-signature. An intuition of how to generalize the scheme to the multiple-signatures case can be found at the end of this section.

Essentially, the idea is to define a relaxed NIZK proof for a relation $\mathscr{R}'$ derived from the combination of the signature and commitment, and then prove that this is in fact a relaxed NIZK proof for the relation $\mathscr{R}$ obtained from the verification equation of the signature scheme.

Let $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ be the public vectors in the verification key $\mathbf{V}$ of the signature scheme, H be its hash function. Given α and the public parameters of the signature, the goal is to prove knowledge of some "small" $(\mathbf{m}, (\mathbf{c}_1, \mathbf{S}, \mathbf{c}_2))$ such that $[\mathbf{A}\ \mathbf{B}\ \mathbf{c}_1\mathbf{C}+\mathbf{mG}\ \mathbf{1}]\mathbf{S} = \mathbf{c}_2 H(\alpha) \bmod q$. To construct a relaxed NIZK proof (cf. Section 3.4), we rewrite the characterizing equation as it follows. Let $(\mathbf{1}, \mathbf{S}, \mathbf{1})$ be a honestly-generated signature on $(\mathbf{m}, \alpha)$, i.e.

$$\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{1C}+\mathbf{mG} & \mathbf{1}\end{bmatrix}\mathbf{S} = \mathbf{1}H(\alpha) \bmod q\ . \tag{3.14}$$

Let $\mathbf{F}$ be a commitment to $\mathbf{m}$, $\mathbf{F} = \mathbf{b}^{-1}(\mathbf{C}+\mathbf{mG}+\mathbf{E}) \bmod q$. Substituting $\mathbf{C}+\mathbf{mG} = \mathbf{Fb}-\mathbf{E} \bmod q$ in Equation (3.14) and rearranging the terms yields that knowing the vectors $((\mathbf{1}, \mathbf{S}, \mathbf{1}), \mathbf{m}, \mathbf{E}, \mathbf{b})$ implies knowing a solution to the following equations:

$$\text{(I)} \underbrace{\begin{bmatrix}-\mathbf{G}^T & \mathbf{F}^T & -\mathbb{I}_m\end{bmatrix}}_{=\mathbf{A}_c} \underbrace{\begin{pmatrix}\bar{\mathbf{m}}\\ \bar{\mathbf{b}}\\ \bar{\mathbf{E}}^T\end{pmatrix}}_{=\bar{\mathbf{S}}_c} = \bar{\mathbf{c}}_1\mathbf{C}^T \bmod q\ , \tag{3.15}$$

$$\text{(II)} \underbrace{\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{F} & \mathbf{1}\end{bmatrix}}_{=\mathbf{A}_s} \underbrace{\begin{pmatrix}\bar{\mathbf{S}}_1\\ \bar{\mathbf{S}}_2\\ \bar{\mathbf{S}}_3\\ \bar{\mathbf{s}}_4\end{pmatrix}}_{=\bar{\mathbf{S}}_s} = \bar{\mathbf{c}}_2 H(\alpha) \bmod q\ ,$$

where in this case

$$\begin{array}{ll} & \bar{\mathbf{S}}_1 = \mathbf{S}_1 \\ \bar{\mathbf{m}} = \mathbf{m} & \bar{\mathbf{S}}_2 = \mathbf{S}_2 \\ \bar{\mathbf{b}} = \mathbf{b} & \bar{\mathbf{S}}_3 = \mathbf{bS}_3 \\ \bar{\mathbf{E}} = \mathbf{E} & \bar{\mathbf{s}}_4 = \mathbf{s}_4 - \mathbf{ES}_3 \\ & \bar{\mathbf{c}}_1 = \bar{\mathbf{c}}_2 = \mathbf{1} \end{array}\ .$$

Observe that the vectors $\bar{\mathbf{S}}_c$ and $\bar{\mathbf{S}}_s$ still have small norm. As the matrices $\mathbf{A}_c$ and $\mathbf{A}_s$, and the elements on the right side of the equations are now independent of the secrets, it is possible to build a relaxed Σ-protocol to prove knowledge of a solution to these equations using a generalized version of the protocol from Section 3.4. The relaxed Σ-protocol $(\mathsf{P}_{pt}, \mathsf{V}_{pt})$ is shown in Figure 3.7. Essentially, the

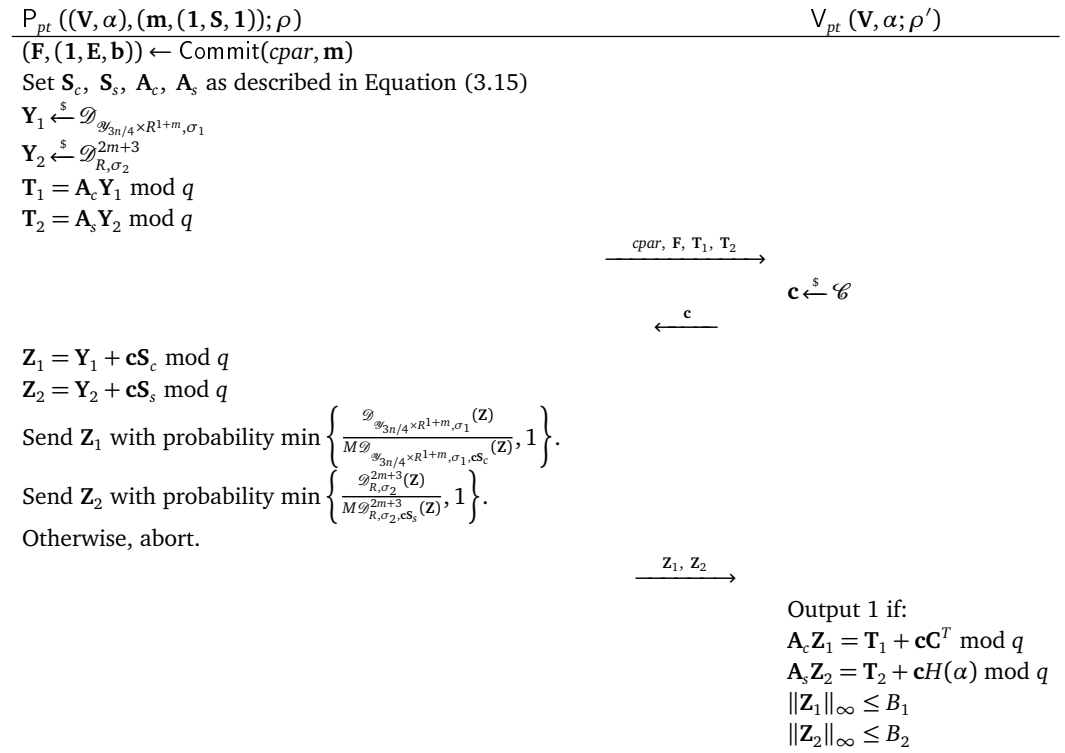

Figure 3.7. Relaxed Σ-protocol to prove knowledge of a signature on a partially hidden message.

only difference from the relaxed Σ-protocol described in Section 3.4 is that rejection sampling is done w.r.t. different probability distributions. This is for both efficiency reasons, and to be able to prove that the hidden part of the message $\mathbf{m}$ is in a particular subring of R_q, as it is shown in the proof of Theorem 3.34.

Finally, remark that to combine the commitment and the signature scheme it is necessary that their parameters are compatible. Hence, we assume that such parameters are public parameters of the scheme and that they are implicitly given as input to P_{pt} and V_{pt}.

We now prove that this protocol is in fact a valid relaxed Σ-protocol of a signature on a partially hidden message.

Theorem 3.34. *Given the following parameters*

- $N_{s,\infty}$ *as in Section 3.6.4,*
- $\bar{N}_s = 256n(m+1)\sigma_1\sigma_2\sqrt{n(2m+3)}$,

- $\bar{N}_{s,\infty} = 256n(m+1)\sigma_1\sigma_2$,
- $\bar{C} = 32n\sigma_1$,
- $r = 16\sigma_1$,
- $\mathscr{C} = R_3^{(2^{K_c})}$ *for some* $K_c > 0$,
- $\bar{\mathscr{C}} = R_5^{(2^{K_c})}$,
- $\sigma_1 := 12 \cdot 2^{K_c}$,
- $\sigma_2 := 12 \cdot 2^{K_c}(nm+1)N_{s,\infty}$,
- $B_i := 8\sigma_i$ *for* $i = 1, 2$.

satisfying the following bounds:

$$(mn+1)N_{s,\infty} \leq \frac{q-1}{2} \quad \wedge \quad 64n\sigma_1\sigma_2 \leq \frac{q-1}{2}$$

the protocol $(\mathsf{P}_{pt}, \mathsf{V}_{pt})$ *is a non-trivial relaxed* Σ*-protocol for the following relations:*

$$\begin{aligned}
\mathscr{R}_s = \{ \quad & ((\mathbf{V}, \alpha), (\mathbf{m}, (\mathbf{1}, \mathbf{S}, \mathbf{1}))) \ : \ \mathbf{m} \in \mathscr{U}, \\
& \begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{1C} + \mathbf{mG} & \mathbf{1} \end{bmatrix} \mathbf{S} = \mathbf{1}H(\alpha) \text{ and } \|\mathbf{S}\|_\infty \leq N_{s,\infty} \} \\
\bar{\mathscr{R}}_s = \{ \quad & ((\mathbf{V}, \alpha), (\bar{\mathbf{m}}, (\bar{\mathbf{c}}_1, \bar{\mathbf{S}}, \bar{\mathbf{c}}_2))) \ : \ \bar{\mathbf{m}} \in \bar{\mathscr{U}}, \ \bar{\mathbf{c}}_1 \in \bar{\mathscr{C}}, \ \|\bar{\mathbf{c}}_2\|_\infty \leq \bar{C} \\
& \begin{bmatrix} \mathbf{A} & \mathbf{B} & \bar{\mathbf{c}}_1\mathbf{C} + \bar{\mathbf{m}}\mathbf{G} & \mathbf{1} \end{bmatrix} \bar{\mathbf{S}} = \bar{\mathbf{c}}_2 H(\alpha) \text{ and } \|\bar{\mathbf{S}}\|_2 \leq \bar{N}_s, \ \|\bar{\mathbf{S}}\|_\infty \leq \bar{N}_{s,\infty} \}
\end{aligned}$$

under decisional $RLWE_{m,\mathscr{U}(R_q)}$.

Proof. Correctness is trivial. Indeed, if $\mathbf{Z}_1$ and $\mathbf{Z}_2$ are honestly generated, it holds that:

$$\begin{aligned}
\mathbf{A}_c\mathbf{Z}_1 &= \mathbf{A}_c\mathbf{Y}_1 + \mathbf{c}(-\mathbf{G}^T\mathbf{m} + \mathbf{F}^T\mathbf{b} - \mathbf{E}) \bmod q = \mathbf{T}_1 + \mathbf{cC} \bmod q \\
\mathbf{A}_s\mathbf{Z}_2 &= \mathbf{A}_s\mathbf{Y}_2 + \mathbf{c}(\mathbf{AS}_1 + \mathbf{BS}_2 + \mathbf{FbS}_3 + \mathbf{s}_4 - \mathbf{ES}_3) \bmod q \\
&= \mathbf{T}_2 + \mathbf{c}(\mathbf{AS}_1 + \mathbf{BS}_2 + (\mathbf{C} + \mathbf{mG} + \mathbf{E})\mathbf{b}^{-1}\mathbf{bS}_3 + \mathbf{s}_4 - \mathbf{ES}_3) \bmod q \\
&= \mathbf{T}_2 + \mathbf{c}(\mathbf{AS}_1 + \mathbf{BS}_2 + [\mathbf{C} + \mathbf{mG}]\mathbf{S}_3 + \mathbf{s}_4) \bmod q = \mathbf{T}_2 + \mathbf{c}H(\alpha) \bmod q \ .
\end{aligned}$$

The verifier can construct $\mathbf{A}_s$ and $\mathbf{A}_c$, as the prover sends $\mathbf{F}$ at the beginning of the interaction. Moreover, thanks to rejection sampling (cf. Lemma 3.7), the vectors $\mathbf{Z}_1$ and $\mathbf{Z}_2$ are distributed like Gaussians with standard deviations σ_1 and σ_2 respectively. Hence, Lemma 2.16 ensures that $\|\mathbf{Z}_1\|_\infty \leq 8\sigma_1 = B_1$ and $\|\mathbf{Z}_2\|_\infty \leq 8\sigma_2 = B_2$. Remark that rejection sampling requires the standard deviations σ_i to

$\Sigma(\mathbf{V}, \alpha, 1^\lambda)$:
$\mathbf{c} \xleftarrow{\$} \mathscr{C}$.
$\mathbf{F} \xleftarrow{\$} R_q^{1\times m}$
$\mathbf{Z}_1 \leftarrow \mathscr{D}_{\mathscr{Y}_{3n/4}\times R^{1+m},\sigma_1}$
$\mathbf{Z}_2 \leftarrow \mathscr{D}^{2m+3}_{R,\sigma_2}$
$\mathbf{T}_1 := \mathbf{A}_c\mathbf{Z}_1 - \mathbf{c}\mathbf{C}^T \bmod q$
$\mathbf{T}_2 := \mathbf{A}_s\mathbf{Z}_2 - \mathbf{c}H(\alpha) \bmod q$
Output $(\mathbf{F}, \mathbf{T}_1, \mathbf{T}_2, \mathbf{c}, \mathbf{Z}_1, \mathbf{Z}_2)$.

Figure 3.8. Simulator for a relaxed Σ-protocol of a signature on a partially hidden message. Recall that the set $\mathscr{Y}_d$ is defined in Section 3.4.2.

be large enough to hide the vectors $\mathbf{cS}_c$ and $\mathbf{cS}_s$. In particular, $\sigma_i = 12T_i$, where T_1 and T_2 are bounds on the infinity norm of $\mathbf{cS}_c$ and $\mathbf{cS}_s$ respectively. The values of the T_i's can be obtained as follows:

$$\begin{aligned}
\|\mathbf{cS}_c\|_\infty &\le \|\mathbf{c}\|_1\|\mathbf{S}_c\|_\infty \le 2^{K_c}\cdot 1 = 2^{K_c} = T_1\\
\|\mathbf{cS}_s\|_\infty &\le \|\mathbf{c}(\mathbf{s}_4 - \mathbf{ES}_3)\|\\
&\le \|\mathbf{c}\|_1\|\mathbf{s}_4 - \mathbf{ES}_3\|_\infty\\
&\le \|\mathbf{c}\|_1(\|\mathbf{s}_4\|_\infty + \|\mathbf{ES}_3\|_\infty)\\
&\le \|\mathbf{c}\|_1(\|\mathbf{s}_4\|_\infty + \|\mathbf{E}\|_1\|\mathbf{S}_3\|_\infty)\\
&\le 2^{K_c}(N_{s,\infty} + mnN_{s,\infty}) = 2^{K_c}(mn+1)N_{s,\infty} = T_2\ ,
\end{aligned}$$

where we have used the norm triangular inequality and Lemma 2.12 (as $(mn+1)N_{s,\infty} \le \frac{q-1}{2}$). Hence the probability that the verifier accepts is

$$\begin{aligned}
&1 - \mathsf{Pr}[\mathsf{P}_{pt}\text{ aborts}]\mathsf{Pr}[\text{honest }\mathbf{Z}_i\text{ do not satisfy the inequalities}] =\\
&1 - \left(1 - \frac{1-2^{-100}}{M}\right)^2 \cdot (2+m)2^{-275}\cdot(2m+3)2^{-275}\\
=&1 - \left(\frac{\exp(289/288) - 1 + 2^{-100}}{\exp(289/288)}\right)^2 \cdot (2+m)\cdot(2m+3)2^{-550}\\
\ge&1 - 2^{-550}(2+m)(2m+3)\ ,
\end{aligned}$$

where the probabilities are computed using Lemma 2.16 and Lemma 3.7 (where we recall that $M = \exp(12/\alpha + 1/(2\alpha^2))$ and $\alpha = 12$, as we set $\sigma_i = 12T_i$).

The HVZK of this protocol can be proved by constructing the simulator Σ shown in Figure 3.8. The distribution of $\mathbf{Z}_1$, $\mathbf{Z}_2$ is statistically indistinguishable from honestly generated $\mathbf{Z}_1$, $\mathbf{Z}_2$, because rejection sampling (i.e., Lemma 3.7) guarantees that the distribution of honestly generated $\mathbf{Z}_1$ is at statistical distance

$2^{-100}e^{-289/288}$ from $\mathscr{D}_{\mathscr{Y}_{3n/4}\times R^{1+m},\sigma_1}$ (an analogous argument can be made for $\mathbf{Z}_2$). As the distribution of $\mathbf{C}$ and $\mathbf{c}$ are the same both in the honestly-generated and in the simulated proof, the distributions of $\mathbf{T}_1$ and $\mathbf{T}_2$ are statistically indistinguishable as well. The distribution of $\mathbf{C}, \mathbf{F}$ is indistinguishable under decisional $\mathrm{RLWE}_{m,\mathscr{U}(R_q)}$ as shown in Lemma 3.22.

From this simulator is also easy to see that a cheating prover is successful (i.e., can output a valid proof without knowing a witness $\mathbf{S}$) if it can correctly guess the challenge. Hence, the soundness error of this proof is at least $\frac{1}{|\mathscr{C}|}$.

Finally, we prove special soundness. A knowledge extractor $(\mathsf{E}_1^s, \mathsf{E}_2^s)$ rewinds Π to obtain the vectors $[\bar{\mathbf{b}}; \bar{\mathbf{m}}; \bar{\mathbf{E}}]$, $[\bar{\mathbf{S}}_1; \bar{\mathbf{S}}_2; \bar{\mathbf{S}}_3; \bar{\mathbf{s}}_4]$ and the polynomial $\bar{\mathbf{c}}$ that satisfy equations (I) and (II) in (3.15). Multiplying equation (II) times $\bar{\mathbf{b}}$ and plugging in $\bar{\mathbf{b}}\mathbf{F} = \bar{\mathbf{c}}\mathbf{C} + \bar{\mathbf{m}}\mathbf{G} + \bar{\mathbf{E}}$ yields:

$$\mathbf{A}(\bar{\mathbf{b}}\bar{\mathbf{S}}_1) + \mathbf{B}(\bar{\mathbf{b}}\bar{\mathbf{S}}_2) + [\bar{\mathbf{c}}\mathbf{C} - \bar{\mathbf{m}}\mathbf{G}](\bar{\mathbf{S}}_3) + (\bar{\mathbf{b}}\bar{\mathbf{s}}_4 - \bar{\mathbf{E}}\bar{\mathbf{S}}_3) = \bar{\mathbf{c}}\bar{\mathbf{b}}H(\alpha) \tag{3.16}$$

The vector $\bar{\mathbf{S}} = [\bar{\mathbf{b}}\bar{\mathbf{S}}_1; \bar{\mathbf{b}}\bar{\mathbf{S}}_2; \bar{\mathbf{S}}_3; \bar{\mathbf{b}}\bar{\mathbf{s}}_4 - \bar{\mathbf{E}}\bar{\mathbf{S}}_3]$ has norm bounded by the norm of $\bar{\mathbf{b}}\bar{\mathbf{s}}_4 - \bar{\mathbf{E}}\bar{\mathbf{S}}_3$:

$$\begin{aligned}
\|\bar{\mathbf{S}}\|_\infty &\leq \|\bar{\mathbf{b}}\bar{\mathbf{s}}_4 - \bar{\mathbf{E}}\bar{\mathbf{S}}_3\|_\infty \\
&\leq \|\bar{\mathbf{b}}\bar{\mathbf{s}}_4\|_\infty + \|\bar{\mathbf{E}}\bar{\mathbf{S}}_3\|_\infty \\
&\leq n\|\bar{\mathbf{b}}\|_\infty\|\bar{\mathbf{s}}_4\|_\infty + \sum_{i=1}^{m}\|\bar{\mathbf{e}}_i\bar{\mathbf{s}}_{3,i}\|_\infty \\
&\leq n\|\bar{\mathbf{b}}\|_\infty\|\bar{\mathbf{s}}_4\|_\infty + mn\|\bar{\mathbf{E}}\|_\infty\|\bar{\mathbf{S}}_3\|_\infty \\
&\leq n(16\sigma_1 \cdot 16\sigma_2 + m \cdot 16\sigma_1 \cdot 16\sigma_2) \\
&= 256n(m+1)\sigma_1\sigma_2 = \bar{N}_{s,\infty} \ ,
\end{aligned}$$

where $\bar{\mathbf{e}}_i$ and $\bar{\mathbf{s}}_{3,i}$ are the components of $\bar{\mathbf{E}}$ and $\bar{\mathbf{S}}_3$ respectively, and where the inequalities follow from Lemma 2.16 and from Lemma 2.12 twice. The use of the latter yields the following constraints on the norms of the extracted vectors:

$$\begin{aligned}
n\|\bar{\mathbf{b}}\|_\infty\|\bar{\mathbf{s}}_4\|_\infty &\leq \frac{q-1}{2} \\
n\|\bar{\mathbf{E}}\|_\infty\|\bar{\mathbf{S}}_3\|_\infty &\leq \frac{q-1}{2} \\
\Rightarrow 256n\sigma_1\sigma_2 &\leq \frac{q-1}{2} \ .
\end{aligned}$$

It is possible to bound the Euclidean norm of $\bar{\mathbf{S}}$ using the bound on the infinity norm as it follows:

$$\|\bar{\mathbf{S}}\|_2 \leq \|\bar{\mathbf{S}}\|_\infty\sqrt{n(2m+3)} \leq 256n(m+1)\sigma_1\sigma_2\sqrt{n(2m+3)} = \bar{N}_s \ .$$

$\mathsf{P}^{H_c}_{pt}((\mathbf{V},\alpha),(\mathbf{m},(\mathbf{1},\mathbf{S},\mathbf{1}));\rho)$:
- $(otssk, otsvk) \leftarrow \mathsf{OTSGen}(1^\lambda)$
- $(\mathbf{F},(\mathbf{1},\mathbf{E},\mathbf{b})) \leftarrow \mathsf{Commit}(cpar,\mathbf{m})$
- Set $\mathbf{S}_c$, $\mathbf{S}_s$, $\mathbf{A}_c$, $\mathbf{A}_s$ as described in Equation (3.15)
- $\mathbf{Y}_1 \xleftarrow{\$} \mathscr{D}_{\mathscr{U}_{3n/4}\times R^{1+m},\sigma_1}$
- $\mathbf{Y}_2 \xleftarrow{\$} \mathscr{D}^{2m+3}_{R,\sigma_2}$
- $\mathbf{T}_1 = \mathbf{A}_c\mathbf{Y}_1 \bmod q$
- $\mathbf{T}_2 = \mathbf{A}_s\mathbf{Y}_2 \bmod q$
- $\mathbf{c} = H_c(cpar,\mathbf{F},\mathbf{T}_1,\mathbf{T}_2,\mathbf{A},\alpha,otsvk)$
- $\mathbf{Z}_1 = \mathbf{Y}_1 + \mathbf{c}\mathbf{S}_c \bmod q$
- $\mathbf{Z}_2 = \mathbf{Y}_2 + \mathbf{c}\mathbf{S}_s \bmod q$
- Send $\mathbf{Z}_1$ with probability $\min\left\{\frac{\mathscr{D}_{\mathscr{U}_{3n/4}\times R_q^{1+m},\sigma_1}(\mathbf{Z})}{M\mathscr{D}_{\mathscr{U}_{3n/4}\times R_q^{1+m},\sigma_1,\mathbf{c}\mathbf{S}_c}(\mathbf{Z})},1\right\}$.
- Send $\mathbf{Z}_2$ with probability $\min\left\{\frac{\mathscr{D}^{2m+3}_{R_q,\sigma_2}(\mathbf{Z})}{M\mathscr{D}^{2m+3}_{R_q,\sigma_2,\mathbf{c}\mathbf{s}_s}(\mathbf{Z})},1\right\}$.
- Otherwise, abort.
- $ots \leftarrow \mathsf{OTSSign}(otssk,((\mathbf{A},\alpha),cpar,\mathbf{F},\mathbf{T}_1,\mathbf{T}_2,\mathbf{c},\mathbf{Z}_1,\mathbf{Z}_2))$
- Return $\pi = (\mathbf{F},\mathbf{c},\mathbf{Z}_1,\mathbf{Z}_2,otsvk,ots)$.

$\mathsf{V}^{H_c}_{pt}(\mathbf{V},\alpha,\pi)$:
- $\mathbf{T}'_1 := \mathbf{A}_c\mathbf{Z}_1 - \mathbf{c}\mathbf{C}^T \bmod q$
- $\mathbf{T}'_2 := \mathbf{A}_s\mathbf{Z}_2 - \mathbf{c}H(\alpha) \bmod q$
- $msg := (\mathbf{A},\alpha,cpar,\mathbf{F},\mathbf{T}'_1,\mathbf{T}'_2,\mathbf{c},\mathbf{Z}_1,\mathbf{Z}_2)$
- Set $b = 1$ if the following conditions hold:
  - $\mathbf{c} = H_c(cpar,\mathbf{F},\mathbf{T}'_1,\mathbf{T}'_2,\mathbf{A},\alpha,otsvk)$
  - $\|\mathbf{Z}_1\|_\infty \le B_1$
  - $\|\mathbf{Z}_2\|_\infty \le B_2$
  - $1 \leftarrow \mathsf{OTSVf}(otsvk,msg,ots)$
- Return b .

Figure 3.9. Relaxed NIZK proof of a signature on a partially hidden message.

Observe that $\bar{N}_{s,\infty} > 8\sigma_2 > 8\sigma$, hence the correctness of the signature scheme is guaranteed (see Section 3.6.4). Moreover, $\bar{\mathbf{c}} \in \bar{\mathscr{C}}$, and, applying again Lemma 2.12, $\|\bar{\mathbf{c}}\bar{\mathbf{b}}\|_\infty \le \|\bar{\mathbf{c}}\|_\infty\|\bar{\mathbf{b}}\|_\infty n \le 2\cdot 16\sigma_1 n = 32n\sigma_1 =: \bar{C}$. Hence $(\bar{\mathbf{c}},\bar{\mathbf{S}},\bar{\mathbf{c}}\bar{\mathbf{b}}) \in \bar{\Sigma}$, i.e. the extractor outputs a valid signature. From point 2 in Lemma 2.16 the infinity norm of the extracted user's secret key $\bar{\mathbf{m}}$ is less than $16\sigma_1 =: r$.

Non-triviality can be proved with a bound similar to the one computed in the proof of Theorem 3.13. □

Let H_c be a hash function as in Section 3.4. It is possible to construct a non interactive protocol $(\mathsf{P}^{H_c}_{pt},\mathsf{V}^{H_c}_{pt})$ via the construction presented in Theorem 3.12 with the Lamport signature as OTS (Section 2.2.4) and shown in Figure 3.9.

In the following we make some observations about possible generalizations, adaptations, and efficiency improvements of the protocol.

Remark 9 (Aggregatable Proofs.)**.** A proof of knowledge of ℓ signatures $\mathbf{S}_i$ generated by signer i on ℓ messages $(\mathbf{m},\alpha_i)$ is constructed by combining ℓ of the previous proofs in parallel. Assume that the parameters of the rC and rS schemes are shared among all signers. This means that the verification key of signer j is $[\mathbf{A}_j\ \mathbf{B}_j\ \mathbf{C}]$ for the same $\mathbf{C}$. Hence, the prover can generate a commitment $\mathbf{F}$ to $\mathbf{m}$ using $\mathbf{C}$ as public matrix, and generate a proof Π_i that it knows a secret $\bar{\mathbf{S}}_c$ that satisfies relation (I) in (3.15) and $\bar{\mathbf{S}}_{s,i}$, $\bar{\mathbf{c}}$ that satisfy $[\mathbf{A}_i\ \mathbf{B}_i\ \mathbf{F}\ \mathbf{1}]\bar{\mathbf{S}}_{s,i} = \bar{\mathbf{c}}H(\alpha_i)$ for

$i = 1, \ldots, \ell$. The relaxed binding property of the commitment guarantees that the hidden part of the message $\mathbf{m}$ is the same in all proofs.

Remark 10. The parameters of the commitment scheme can also be included in the parameters of the scheme, and assumed to be given as input to both P_{pt} and V_{pt} (cf. Section 3.7). This does not affect the properties of the protocol, as these are always generated honestly (even by the simulator).

Remark 11. It is possible to further separate the rejection sampling procedure by isolating $\mathbf{m}$, and having three rejection sampling steps (one for $\mathbf{m}$, one for $\mathbf{S}_s$ and one for the rest of $\mathbf{S}_c$). This allows to tailor the Gaussian distribution of the element masking $\mathbf{m}$ so that the blow up in the norm of the message (i.e., the parameter r) is smaller. However, this would make the proof more complex and add one more rejection sampling step, increasing the time required to generate a proof. Therefore, we have decided to use the simpler protocol described in this section. More details on this construction can be found in [Boschini et al., 2017].

3.8 Relaxed Verifiable Encryption Scheme

We conclude this chapter by presenting two flavors of relaxed verifiable encryption scheme based on RLWE. The original scheme is due to Lyubashevsky and Neven [2017]. Their scheme is a relaxed verifiable encryption,i.e., a scheme to encrypt a witness w of $x \in L$ such that decryption of a valid ciphertext is guaranteed to yield a witness $\bar{w}$ in the relaxed language such that $(x, \bar{w}) \in \bar{\mathscr{R}}$.

The straightforward combination with the relaxed signature and commitment scheme in Section 3.6.3 and 3.5.4 does not yield a particularly efficient group signature scheme, however, because the Lyubashevsky-Neven verifiable encryption scheme encrypts and recovers the *full* witness. A group signature typically consists of a verifiable encryption of the user's identity together with a proof that the user knows a valid signature on the encrypted identity by the group manager. The verifiable encryption as defined by Lyubashevsky and Neven would therefore encrypt both the user's identity and the signature on it, which unnecessarily blows up the size of the verifiable ciphertext. Even when using a commitment to the user's identity to separate the proof of knowledge of the signature from the verifiable encryption, the ciphertext will encrypt the user's identity as well as the opening information to the commitment.

We therefore introduce a variant of the Lyubashevsky-Neven relaxed verifiable encryption scheme called relaxed *partial* verifiable encryption that, rather than decrypting the full witness $\bar{w}$, recovers only a function of that witness $g(\bar{w})$

while proving knowledge of the full witness $\bar{w}$ [Boschini et al., 2018a]. When constructing a group signature case, we will use a function g that outputs just the user's identity. Our scheme can be found in Section 3.8.4. We start by giving the generic definitions of relaxed partially and fully verifiable encryption schemes (cf. Section 3.8.1). Then we briefly describe the original verifiable encryption scheme by Lyubashevsky and Neven (cf. Section 3.8.3), as in Section 5.3.1 we motivate the introduction of our partial verifiable encryption by showing that the constructions built from the verifiable encryption by Lyubashevsky and Neven are too inefficient. Finally, in Section 3.8.4 we show how to modify this scheme to obtain a partially verifiable encryption scheme.

3.8.1 Definition of Relaxed Partial Verifiable Encryption

The definitions in this chapter are a combination of the definition of relaxed verifiable encryption by Lyubashevsky and Neven [2017] and of the definition of partially verifiable encryption by Boschini et al. [2018a].

Let L be a language with witness relation $\mathscr{R}$ and let $\bar{L} \supseteq L$ be a relaxed language with relaxed relation $\bar{\mathscr{R}} \supseteq \mathscr{R}$. Let $\bar{\mathscr{R}} \subseteq \bar{L} \times \bar{W}$ and let $g : \bar{W} \to D$ be a function.

Given relations $\mathscr{R}$, $\bar{\mathscr{R}}$ and function g, a *relaxed verifiable encryption scheme* is composed by four algorithms $(\mathsf{EKeyGen}, \mathsf{Enc}, \mathsf{EVerify}, \mathsf{Dec})$:

Key Generation. The key generation algorithm $\mathsf{EKeyGen}(1^\lambda)$ outputs a pair of keys (epk, esk).

Encryption The encryption algorithm $\mathsf{Enc}(epk, x, w, \ell)$ takes as input a pair $(x, w) \in \mathscr{R}$ and an encryption label $\ell \in \{0,1\}^*$. It returns a ciphertext t and a proof $\pi = (\alpha, \beta, \gamma)$.

Verification Verification $\mathsf{EVerify}(epk, x, t, \pi, \ell)$ returns 1 if π shows that t is a valid ciphertext w.r.t. x and epk with label ℓ, and returns 0 otherwise.

Decryption. Finally, the decryption algorithm $\mathsf{Dec}(esk, x, t, \pi, \ell)$ returns a value μ or a failure symbol $\perp$.

According to the original definition of verifiable encryption [Camenisch and Shoup, 2003], this scheme has to satisfy the following security properties. The first two impose essentially that both the encryption functionality and the ZK proof work correctly.

Definition 3.35 (Correctness)**.** The scheme is correct if for all $(x,w) \in \mathscr{R}$, and all $\ell \in \{0,1\}^*$,

$$\mathsf{Pr}\left[\mathsf{Dec}(esk, x, \mathsf{Enc}(epk, x, w, \ell)) = g(w) \ : \ (epk, esk) \leftarrow \mathsf{EKeyGen}(1^\lambda)\right] = 1 \ .$$

Definition 3.36 (Completeness)**.** The scheme satisfies completeness if for all $(x,w) \in \mathscr{R}$, and all $\ell \in \{0,1\}^*$,

$$\mathsf{Pr}[\mathsf{EVerify}(epk, \mathsf{Enc}(epk, x, w, \ell), \ell) = 1 \ : \ (epk, esk) \leftarrow \mathsf{EKeyGen}(1^\lambda)] = 1 \ .$$

Special soundness implies that a valid proof π is a proof of knowledge of a valid witness $\bar{w}$ for the relation $\bar{\mathscr{R}}$ and that decryption of the ciphertext t returns $g(\bar{w})$.

Definition 3.37 (Special soundness)**.** For all PPT adversaries A there exists a PPT extractor E such that:

$$\mathsf{Pr}\left[\begin{array}{c} b = b' = 1 \ \wedge \ \beta \neq \beta' \wedge \\ \left(\mathsf{Dec}(esk, x, t, \ell) \neq g(\bar{w}) \right. \\ \left. \vee \ (x, \bar{w}) \notin \bar{\mathscr{R}} \right) \end{array} \ : \ \begin{array}{c} (epk, esk) \leftarrow \mathsf{EKeyGen}(1^\lambda), \\ (x, t, (\alpha, \beta, \gamma, \beta', \gamma'), \ell) \leftarrow \mathsf{A}(epk, esk), \\ b \leftarrow \mathsf{EVerify}(epk, x, t, (\alpha, \beta, \gamma), \ell), \\ b' \leftarrow \mathsf{EVerify}(epk, x, t, (\alpha, \beta', \gamma'), \ell)), \\ \bar{w} \leftarrow \mathsf{E}(epk, esk, x, t, (\alpha, \beta, \gamma, \beta', \gamma'), \ell) \end{array}\right] \leq \nu(\lambda) \ .$$

Definition 3.38 (Chosen-ciphertext simulatability)**.** There exists a simulator Σ that outputs ciphertexts indistinguishable from honestly generated ones, i.e., the adversary has negligible advantage in the experiment $\mathsf{Exp}_{\mathsf{A}}^{\mathsf{ccas}-b}(1^\lambda)$ in Figure 3.10:

$$\left| \mathsf{Pr}\left[b' = b \ : \ b \xleftarrow{\$} \{0,1\}, \ b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{\mathsf{ccas}-b}(1^\lambda) \right] - \frac{1}{2} \right| \leq \nu(\lambda) \ .$$

The function g was introduced in the definition to model the fact that a partial verifiable encryption might only encrypts a part of the original witness. Hence, the difference between a relaxed verifiable encryption and a relaxed *partially* verifiable encryption is that in the former case g is the identity, i.e., $g(w) = w$.

Remark 12. The addition of the label ℓ is for technical reasons, and was not included in the work of Lyubashevsky and Neven. It is possible to obtain the original definitions by simply setting ℓ to be a fixed string. In fact, we will present both the relaxed verifiable encryption and the relaxed partially verifiable encryption without labels for the sake of clarity.

Experiment $\mathsf{Exp}^{\mathsf{ccas}-b}_{\mathsf{A}}(1^\lambda)$
$(epk, esk) \leftarrow \mathsf{EKeyGen}(1^\lambda)$
$(st, x^*, w^*, \ell^*) \leftarrow \mathsf{A}^{\mathsf{Dec}(esk,\cdot,\cdot,\cdot,\cdot)}(epk)$
$(t_0, \pi_0) \leftarrow \mathsf{Enc}(epk, x^*, w^*, \ell^*)$
$(t_1, \pi_1) \leftarrow \Sigma(epk, x^*, \ell^*)$.
$b' \leftarrow \mathsf{A}^{\mathsf{Dec}'(esk,\cdot,\cdot,\cdot,\cdot)}(st, t_b, \pi_b)$
Return b'.

Figure 3.10. Chosen-ciphertext simulatability experiment. The decryption oracle Dec' behaves exactly as Dec but it rejects queries on $(x^*, t_b, \pi_b, \ell^*)$.

3.8.2 RLWE Encryption scheme

We briefly introduce the RLWE encryption scheme by Lyubashevsky et al. [2013]. The security experiment for IND-CPA security can be found in Figure 3.12.

Parameters Generation. Let p, q be prime numbers, $p \ll q$. Let $\mathscr{M} \subseteq R_p$ be the message space, and χ be a probability distribution over R_q (bounds on parameters to ensure correctness can be found in Theorem 3.39). The algorithm $\mathsf{EParGen}_{RLWE}$ outputs all of them as a bundle *epar*.

Key Generation. The key generator $\mathsf{EKeyGen}_{RLWE}$ generates a RLWE key pair by sampling $\mathbf{a} \xleftarrow{\$} R_q$, $\mathbf{s} \xleftarrow{\$} \chi$ and $\mathbf{d} \leftarrow \chi$, and sets $\mathbf{b} = \mathbf{as} + \mathbf{d} \bmod q$. The encryption key is $epk = (\mathbf{a}, \mathbf{b})$, the decryption key is $esk = \mathbf{s}$.

Encryption. On input $\mathbf{m} \in \mathscr{M}$, the encryption algorithm Enc_{RLWE} generates the ciphertext $(\mathbf{v}, \mathbf{w})$ as:

$$\begin{aligned} \mathbf{v} &= p(\mathbf{ar} + \mathbf{e}) \bmod q \\ \mathbf{w} &= p(\mathbf{br} + \mathbf{f}) + \mathbf{m} \bmod q \ , \end{aligned} \tag{3.17}$$

where $\mathbf{e}, \mathbf{f}, \mathbf{r} \xleftarrow{\$} \chi$. The algorithm outputs $(\mathbf{v}, \mathbf{w})$.

Decryption. The decryption algorithm $\mathsf{Dec}_{RLWE}(\mathbf{v}, \mathbf{w}, \mathbf{s})$ computes $\mathbf{m}' = (\mathbf{w} - \mathbf{vs}) \bmod q \bmod p$; it returns $\mathbf{m}'$.

Theorem 3.39 (Lemma 8.3 and 8.4 in [Lyubashevsky et al., 2013])**.** *The above scheme is IND-CPA secure under $RLWE_{2,\chi}$.*

Moreover, if χ outputs elements with norm bounded by N with probability $1 - \nu(n)$, $p \cdot \chi$ is δ-subgaussian with parameter s for some $\delta = O(1)$, and $q \geq s\sqrt{2(N)^2 + n} \cdot \omega(\sqrt{\log n})$, then the decryption is correct with probability $1 - \nu(n)$.

Observe that we adapted the parameters bounds to work with the particular polynomial ring we have chosen.

3.8.3 Relaxed Verifiable Encryption

The relaxed verifiable encryption scheme in [Lyubashevsky and Neven, 2017] is essentially a combination of RLWE encryption [Lyubashevsky et al., 2010] and a relaxed Σ-protocol for linear relations over short vectors, i.e. relations of the form

$$\mathscr{R} = \left\{ ((\mathbf{B},\mathbf{u}),(\mathbf{M},\mathbf{1})) \in (R_p^{\ell\times k} \times R_p^{\ell}) \times (R_p^{k} \times R_p) \ : \ \begin{array}{c} \mathbf{BM} = \mathbf{u} \bmod p \\ \wedge \ \|\mathbf{M}\|_\infty \leq \gamma \end{array} \right\} \tag{3.18}$$

and relaxed language $\bar{L}$ with relation

$$\bar{\mathscr{R}} = \left\{ ((\mathbf{B},\mathbf{u}),(\bar{\mathbf{M}},\bar{\mathbf{c}})) \in (R_p^{\ell\times k} \times R_p^{\ell}) \times (R_p^{k} \times R_p) : \ \begin{array}{c} \mathbf{B}\bar{\mathbf{M}} = \bar{\mathbf{c}}\mathbf{u} \bmod p \\ \wedge \ \|\bar{\mathbf{M}}\|_\infty < 6\sigma \\ \wedge \ \bar{\mathbf{c}} \in \bar{\mathscr{C}} \end{array} \right\}, \tag{3.19}$$

where $\bar{\mathscr{C}} = \{\mathbf{c}-\mathbf{c}' \ : \ \mathbf{c},\mathbf{c}' \in \mathscr{C}\}$ for $\mathscr{C} = \{\mathbf{c} \in R : \|\mathbf{c}\|_\infty = 1, \|\mathbf{c}\|_1 \leq 36\}$.

For the sake of clarity, we present the CPA version of the verifiable encryption scheme, i.e., a scheme that satisfies the chosen-plaintext simulatability property, in whose security experiment the adversary has *no decryption oracle*. The scheme $(\mathsf{EKeyGen}_{LN}, \mathsf{Enc}_{LN}, \mathsf{EVerify}_{LN}, \mathsf{Dec}_{LN})$ encrypts messages $\mathbf{M} \in \mathscr{S}_1$ as follows.

Parameters. Let n be a power of 2, $p > 2$, and q be two primes such that $q \gg p$, σ_e a standard variation. Let $\mathscr{S}_i = \{\mathbf{a} \in R \ : \ \|\mathbf{a}\|_\infty \leq i\}$ and the message space be $\mathscr{S}_1$. Let the challenge spaces of the rNIZK be $\mathscr{C} = \{\mathbf{c} \in R : \|\mathbf{c}\|_\infty = 1, \|\mathbf{c}\|_1 \leq 36\}$ and $\bar{\mathscr{C}} = \{\mathbf{c}-\mathbf{c}' \ : \ \mathbf{c},\mathbf{c}' \in \mathscr{C}\}$.

Key Generation. The key generator $\mathsf{EKeyGen}_{LN}$ generates a RLWE key pair by sampling $\mathbf{a} \xleftarrow{\$} R_q$, $\mathbf{s} \xleftarrow{\$} \mathscr{S}_1$ and $\mathbf{d} \leftarrow \mathscr{S}_1$, and sets $\mathbf{b} = \mathbf{as} + \mathbf{d} \bmod q$. The encryption key is $epk = (\mathbf{a},\mathbf{b})$, the decryption key is $esk = \mathbf{s}$.

Encryption. On input $((\mathbf{B},\mathbf{u}),(\mathbf{M},\mathbf{1}),epk)$, the encryption algorithm Enc_{LN} generates the ciphertext $(\mathbf{V},\mathbf{W})$ by encrypting the witness with standard RLWE encryption:

$$\begin{aligned} \mathbf{V} &= p(\mathbf{aR}+\mathbf{E}) \bmod q \\ \mathbf{W} &= p(\mathbf{bR}+\mathbf{F}) + \mathbf{M} \bmod q \ , \end{aligned} \tag{3.20}$$

where $\mathbf{E}, \mathbf{F}, \mathbf{R} \xleftarrow{\$} \mathscr{S}_1^k$. Then it generates a relaxed proof Π as described in Section 3.4 for an instance $(\mathbf{B}, \mathbf{u}, \mathbf{V}, \mathbf{W})$ and witness $(\mathbf{M}, \mathbf{1}, \mathbf{R}, \mathbf{E}, \mathbf{F})$ that satisfies both Equation (3.20) and that contains a witness w.r.t. relation $\mathscr{R}$. The algorithm outputs $(\mathbf{V}, \mathbf{W}, \Pi)$.

Verification. The verification algorithm $\mathsf{EVerify}_{LN}$ is the standard verification algorithm of relaxed NIZKs (cf. Section 3.4).

Decryption. The decryption algorithm $\mathsf{Dec}_{LN}(\mathbf{s}, (\mathbf{B}, \mathbf{u}), (\mathbf{V}, \mathbf{W}), \Pi)$ first checks the validity of the proofs using the verification algorithm above, returning $\perp$ if it is not valid. It then decrypts the ciphertext by, for $i = 1, \ldots, l$, going over all challenges $\mathbf{c}' \in \bar{\mathscr{C}}$ to try to decrypt $(\bar{\mathbf{c}}\mathbf{V}, \bar{\mathbf{c}}\mathbf{W})$ as a RLWE ciphertext, where $\bar{\mathbf{c}} = \mathbf{c}^{(i)} - \mathbf{c}'$. It does so by computing $\bar{\mathbf{m}}' = (\mathbf{W} - \mathbf{V}\mathbf{s})\bar{\mathbf{c}} \bmod q$, checking that $\|\bar{\mathbf{m}}'\|_\infty < q/2C_1'$ where $C_1' = \max_{\mathbf{c} \in \bar{\mathscr{C}}} \|\mathbf{c}\|_1$, and if so, compute $\bar{\mathbf{m}} = \bar{\mathbf{m}}' \bmod p$ and return $\bar{\mathbf{m}}/\bar{\mathbf{c}} \bmod q$; otherwise, it returns $\perp$.

Decryption is successful as long as

$$(36r + 12)\sigma_e < q/2C_1' \tag{3.21}$$

(cf. Lemma 3.1 [Lyubashevsky and Neven, 2017]).

The relaxed verifiable encryption scheme is sound and chosen-plaintext simulatable under the RLWE assumption, and can be made chosen-ciphertext simulatable through the Naor-Yung approach [Naor and Yung, 1990]. We do not write the full CCA2 construction of the Lyubashevsky-Neven verifiable encryption scheme, as this can be built in the same way as the partially verifiable encryption in its chosen-ciphertext simulatable form, which is presented in the next section.

Theorem 3.40 (Lemma 3.2 and Theorem 5.2 in [Lyubashevsky and Neven, 2017])**.** *If the RLWE encryption scheme is IND-CPA secure and the rNIZK proof system is non-interactive zero-knowledge and simulation-sound, then the CCA version of the above construction is chosen-ciphertext simulatable.*

The runtime T of the decryption of a ciphertext generated by an adversary using q_H random oracle queries is such that

$$\mathsf{Pr}_{\hat{H},\hat{D}}[T \geq \alpha q_H] \leq \frac{1}{\alpha} + 2 \cdot \sqrt{\frac{q_H}{\alpha \cdot |\mathscr{C}|}} + \frac{q_H}{|\mathscr{C}|} ,$$

where $\hat{H}$ and $\hat{D}$ are the random coins of the ransom oracle and of the decryption respectively.

Remark 13. This encryption scheme (and its partially verifiable version) encrypts plaintexts that are polynomials of degree n with binary coefficients. In case it would be necessary to encrypt a bit string $\vec{b} = (b_1, \ldots, b_k)$, we assume the encryption algorithm first converts it to an element of $\mathcal{S}_1$ (or more than one, if $k > n$) by setting $b_i = 0$ for $k < i \leq n$ and constructing the polynomial $\mathbf{b} = \sum_{i=1}^{n} b_i x^{i-1}$ (the case $k > n$ is analogous).

3.8.4 Relaxed Partial Verifiable Encryption from Lattices

Let L and $\bar{L}$ be a language and its relaxed version defined w.r.t. the following relations

$$\begin{aligned}
\mathscr{R}_{\text{ve}} &= \left\{ \begin{array}{c} ((\mathbf{A},\mathbf{U}),(\mathbf{m},\mathbf{S},\mathbf{1})) \in \\ (R_q^{\ell_1\times(\ell_2+1)} \times R_q^{\ell_1}) \times (R_3^{(16)} \times R_3^{\ell_2} \times \{\mathbf{1}\}) \end{array} \;:\; \mathbf{A}\begin{bmatrix}\mathbf{m}\\ \mathbf{S}\end{bmatrix} = \mathbf{U} \bmod q \wedge \|\mathbf{S}\| \leq N \right\} \\
\bar{\mathscr{R}}_{\text{ve}} &= \left\{ \begin{array}{c} ((\mathbf{A},\mathbf{U}),(\bar{\mathbf{m}},\bar{\mathbf{S}},\bar{\mathbf{c}})) \in \\ (R_q^{\ell_1\times(\ell_2+1)} \times R_q^{\ell_1}) \times (\bar{\mathscr{U}} \times R_q^{\ell_2} \times \bar{\mathscr{C}}) \end{array} \;:\; \mathbf{A}\begin{bmatrix}\bar{\mathbf{m}}\\ \bar{\mathbf{S}}\end{bmatrix} = \bar{\mathbf{c}}\mathbf{U} \bmod q \wedge \|\bar{\mathbf{S}}\| \leq \bar{N} \right\}
\end{aligned} \tag{3.22}$$

for some sets $\bar{\mathscr{U}}, \bar{\mathscr{C}} \subseteq R_q$ and some integers $\ell_1, \ell_2, N, \bar{N} > 0$ (defined in the parameter generation algorithm). This is a pair of (relaxed) hard lattice relations as defined in Section 3.4 under RSIS.

We construct a relaxed partial verifiable encryption scheme for relations $\mathscr{R}_{\text{ve}}$ and $\bar{\mathscr{R}}_{\text{ve}}$ and function $g((\bar{\mathbf{m}},\bar{\mathbf{S}},\bar{\mathbf{c}})) = \bar{\mathbf{m}}/\bar{\mathbf{c}} \bmod q$. The scheme is a modified version of the "multi-shot" chosen-ciphertext secure verifiable encryption scheme of Lyubashevsky-Neven. The multi-shot scheme involves multiple parallel repetitions of the proof with sub-exponential challenge set sizes, but therefore decryption is strict polynomial time (as opposed to expected polynomial time for the one-shot scheme).

Rather than producing one big proof of knowledge of the terms in relation $\mathscr{R}_{\text{ve}}$, we split it into two proofs, one for each term. The first proof only contains the ciphertext equations and is repeated multiple times with a sub-exponential challenge set to enable efficient decryption. The second includes the relation equation as well as the ciphertext, proving that the encrypted plaintext is derived from a valid witness. The latter proof uses an exponential-size challenge set, so that it doesn't need to be repeated.

Parameters. Let p and q be two public primes with $p > 2$, and σ_i, $\mathscr{C}_i$, $\bar{\mathscr{C}}_i$ be the standard deviation, challenge set, and relaxed challenge set of two relaxed NIZK proof schemes $(\mathsf{P}_i^{H_c}, \mathsf{V}_i^{H_c})$. We set $\mathscr{C}_1 = \{\mathbf{c} \in R_3 \mid \|\mathbf{c}\|_1 \leq 32\}$ and $\mathscr{C}_2 = R_3^{(16)}$. Remark that the bound N on the norm of the witness depends on the choice of the norm: if we are considering the infinity norm, such bound

is $N = 1$, if we are considering the euclidean norm, $N = \sqrt{\ell_2}$. The other parameters of the rNIZK can be computed from it as shown in Section 3.4. This results in the relaxed challenge sets being $\bar{\mathscr{C}_i} = \{\mathbf{c}-\mathbf{c}' \in R_q \mid \mathbf{c}, \mathbf{c}' \in \mathscr{C}_i\}$, $\bar{\mathscr{C}_2} \subseteq R_5^{(16)}$. We assume that

$$q > \max\left\{ (32\sigma_i)^2,\ 16\sigma_i p + 26\sigma_i,\ 32C(2np + p + 1) \right\} , \tag{3.23}$$

$$p > \max\{ 32C\sigma_2,\ 16\sigma_i \} , \tag{3.24}$$

where C is defined in Lemma 3.41.

Key Generation. The algorithm generates two key pairs for RLWE encryption [Lyubashevsky et al., 2010], but discards the secret key of the second pair. It samples $\mathbf{s}_1, \mathbf{d}_1, \mathbf{s}_2, \mathbf{d}_2 \xleftarrow{\$} R_3$ and $\mathbf{a} \xleftarrow{\$} R_q$, and computes $\mathbf{t}_1 = \mathbf{a}\mathbf{s}_1 + \mathbf{d}_1 \bmod q$ and $\mathbf{t}_2 = \mathbf{a}\mathbf{s}_2 + \mathbf{d}_2 \bmod q$. The public key is $epk = (p, q, \mathbf{a}, \mathbf{t}_1, \mathbf{t}_2)$, the secret key is $esk = \mathbf{s}_1$.

Encryption. Given a witness $(\mathbf{m}, \mathbf{S}, \mathbf{1})$ for language member $(\mathbf{A}, \mathbf{U})$ in the relation $\mathscr{R}_{\text{ve}}$, the algorithm Enc uses the Naor-Yung technique [Naor and Yung, 1990] by encrypting $\mathbf{m}$ twice using standard RLWE encryption under public keys $\mathbf{t}_1$ and $\mathbf{t}_2$. More precisely, it samples $\mathbf{r}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{f}_1, \mathbf{f}_2 \xleftarrow{\$} R_3$ and sets $\mathbf{v}_1 = p(\mathbf{a}\mathbf{r} + \mathbf{e}_1) \bmod q$, $\mathbf{w}_1 = p(\mathbf{t}_1\mathbf{r} + \mathbf{f}_1) + \mathbf{m} \bmod q$, $\mathbf{v}_2 = p(\mathbf{a}\mathbf{r} + \mathbf{e}_2) \bmod q$, and $\mathbf{w}_2 = p(\mathbf{t}_2\mathbf{r} + \mathbf{f}_2) + \mathbf{m} \bmod q$.

Then, letting $\mathbf{A}_1$ be the first column of the matrix $\mathbf{A} = \begin{bmatrix}\mathbf{A}_1 & \mathbf{A}_2\end{bmatrix}$ in relation $\mathscr{R}_{\text{ve}}$, it constructs a NIZK proof Π_1 running $\mathsf{P}_1^{H_c}$ (cf. Section 3.4) for the relation

$$\begin{bmatrix} \mathbf{0} & p\mathbf{a} & p & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}^{1\times\ell_2} \\ \mathbf{1} & p\mathbf{t}_1 & \mathbf{0} & p & \mathbf{0} & \mathbf{0} & \mathbf{0}^{1\times\ell_2} \\ \mathbf{0} & p\mathbf{a} & \mathbf{0} & \mathbf{0} & p & \mathbf{0} & \mathbf{0}^{1\times\ell_2} \\ \mathbf{1} & p\mathbf{t}_2 & \mathbf{0} & \mathbf{0} & \mathbf{0} & p & \mathbf{0}^{1\times\ell_2} \\ \mathbf{A}_1 & \mathbf{0}^{\ell_1\times 1} & \mathbf{0}^{\ell_1\times 1} & \mathbf{0}^{\ell_1\times 1} & \mathbf{0}^{\ell_1\times 1} & \mathbf{0}^{\ell_1\times 1} & \mathbf{A}_2 \end{bmatrix} \begin{bmatrix} \mathbf{m} \\ \mathbf{r} \\ \mathbf{e}_1 \\ \mathbf{f}_1 \\ \mathbf{e}_2 \\ \mathbf{f}_2 \\ \mathbf{S} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{w}_1 \\ \mathbf{v}_2 \\ \mathbf{w}_2 \\ \mathbf{U} \end{bmatrix} \bmod q , \tag{3.25}$$

whereby it uses the challenge set $\mathscr{C}_1 = \{\mathbf{c} \in R_3 \mid \|\mathbf{c}\|_1 \leq 32\}$.

To enable Lyubashevsky-Neven's multi-shot decryption technique without having to repeat the above proof multiple times, the encryptor runs $\mathsf{P}_2^{H_c}$ to

construct a separate proof Π_2 for the relation

$$\begin{bmatrix} \mathbf{0} & p\mathbf{a} & p & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & p\mathbf{t}_1 & \mathbf{0} & p & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & p\mathbf{a} & \mathbf{1} & \mathbf{0} & p & \mathbf{0} \\ \mathbf{1} & p\mathbf{t}_2 & \mathbf{0} & \mathbf{0} & \mathbf{0} & p \end{bmatrix} \begin{bmatrix} \mathbf{m} \\ \mathbf{r} \\ \mathbf{e}_1 \\ \mathbf{f}_1 \\ \mathbf{e}_2 \\ \mathbf{f}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{w}_1 \\ \mathbf{v}_2 \\ \mathbf{w}_2 \end{bmatrix} \bmod q \ , \tag{3.26}$$

whereby it includes $epk, (\mathbf{A}, \mathbf{U}), (\mathbf{v}_1, \mathbf{w}_1, \mathbf{v}_2, \mathbf{w}_2), \Pi_1$ (and the label ℓ in case it is present) in the Fiat-Shamir hash. To obtain efficient decryption but keep the soundness error negligible, this proof is repeated $l = 11$ times with challenge set $\mathscr{C}_2 = R_3^{(16)}$. The algorithm outputs ciphertext $(\mathbf{v}_1, \mathbf{w}_1, \mathbf{v}_2, \mathbf{w}_2)$ and proof (Π_1, Π_2).

Verification. The verification algorithm $\mathsf{EVerify}((p, q, \mathbf{a}, \mathbf{t}_1, \mathbf{t}_2), (\mathbf{A}, \mathbf{U}), (\mathbf{v}_1, \mathbf{w}_1, \mathbf{v}_2, \mathbf{w}_2, \Pi_1, \Pi_2))$ checks that Π_1 and Π_2 are valid relaxed NIZK proofs for the relations of Equations (3.25) and (3.26) running $\mathsf{V}_1^{H_c}$ and $\mathsf{V}_2^{H_c}$, including the correct arguments $epk, (\mathbf{A}, \mathbf{U}), (\mathbf{v}_1, \mathbf{w}_1, \mathbf{v}_2, \mathbf{w}_2), \Pi_1$ (and the label ℓ in case it is present) in the Fiat-Shamir hash of Π_2.

Decryption. The decryption algorithm $\mathsf{Dec}(\mathbf{s}_1, (\mathbf{A}, \mathbf{U}), (\mathbf{v}_1, \mathbf{w}_1, \mathbf{v}_2, \mathbf{w}_2), (\Pi_1, \Pi_2),)$ first checks that the proofs are valid using the verification algorithm above, returning $\perp$ if it is not valid. It then decrypts the ciphertext by applying the Lyubashevsky-Neven multi-shot decryption on proof $\Pi_2 = (\mathbf{Y}^{(1)}, \mathbf{c}^{(1)}, \mathbf{Z}^{(1)}, \ldots, \mathbf{Y}^{(l)}, \mathbf{c}^{(l)}, \mathbf{Z}^{(l)})$ by, for $i = 1, \ldots, l$, going over all challenges $\mathbf{c}' \in \mathscr{C}_2$ to try to decrypt $(\bar{\mathbf{c}}\mathbf{v}, \bar{\mathbf{c}}\mathbf{w}_1)$ as a Ring-LWE ciphertext, where $\bar{\mathbf{c}} = \mathbf{c}^{(i)} - \mathbf{c}'$. It does so by computing $\bar{\mathbf{m}}' = (\mathbf{w}_1 - \mathbf{v}_1\mathbf{s}_1)\bar{\mathbf{c}} \bmod q$, checking that $\|\bar{\mathbf{m}}'\|_\infty < q/2C$ where C is as defined in Lemma 3.41, and if so, compute $\bar{\mathbf{m}} = \bar{\mathbf{m}}' \bmod p$ and return $\bar{\mathbf{m}}/\bar{\mathbf{c}} \bmod q$; otherwise, it returns $\perp$.

Decryption Runtime. Decryption terminates in time at most 2^{26}. Indeed, if the ciphertext is honestly generated the algorithm needs to guess the challenge only once. On the other hand, for a dishonestly generated ciphertext the probability that decryption fails is negligible. Indeed, if the adversary could answer only one challenge $\mathbf{c}$, when making the random oracle queries the probability of hitting always $\mathbf{c}$ would be $1/(\ell \cdot |\mathscr{C}_2|)$. Hence, a second challenge exists and decryption requires to guess a challenge $\mathbf{c}'$ at most $|\mathscr{C}_2| = 2^{26}$ times.

Observe that in our case the relation $\mathbf{A}\begin{bmatrix} \mathbf{m} \\ \mathbf{S} \end{bmatrix} = \mathbf{U}$ holds modulo q, while in the original scheme it has to hold modulo p. We show the correctness of the scheme

using Lemma 3.41, which is a variant of a result by Lyubashevsky and Neven [Lyubashevsky and Neven, 2017, Lemma 3.1].

Lemma 3.41. *Let* $\mathbf{a} \xleftarrow{\$} R_q$*, and* $\mathbf{t} = \mathbf{as} + \mathbf{d}$ *where* $\mathbf{s}, \mathbf{d} \xleftarrow{\$} R_3$*. If there exist* $\bar{\mathbf{r}}$*,* $\bar{\mathbf{e}}$*,* $\bar{\mathbf{f}}$*,* $\bar{\mathbf{m}}$*,* $\bar{\mathbf{c}}$ *such that*

$$p(\mathbf{a}\bar{\mathbf{r}} + \bar{\mathbf{e}}) = \bar{\mathbf{c}}\mathbf{v} \bmod q \quad \text{and} \quad p(\mathbf{t}\bar{\mathbf{r}} + \bar{\mathbf{f}}) + \bar{\mathbf{m}} = \bar{\mathbf{c}}\mathbf{w} \bmod q \tag{3.27}$$

and $\|p(\bar{\mathbf{r}}\mathbf{d} + \bar{\mathbf{f}} - \bar{\mathbf{e}}\mathbf{s}) + \bar{\mathbf{m}}\|_\infty < q/2C$ *and* $\|\bar{\mathbf{m}}\|_\infty < p/2C$*, where* $C = \max_{\bar{\mathbf{c}} \in \bar{\mathscr{C}_2}} \|\bar{\mathbf{c}}\|_1 = \max_{\bar{\mathbf{c}}, \bar{\mathbf{c}}' \in \mathscr{C}_2} \|\bar{\mathbf{c}} - \bar{\mathbf{c}}'\|_1$*, then*

1. $\|(\mathbf{w} - \mathbf{vs})\mathbf{c}' \bmod q\|_\infty < q/2C$ *and* $\|(\mathbf{w} - \mathbf{vs})\mathbf{c}' \bmod q \bmod p\|_\infty < p/2C$
2. *for any* $\bar{\mathbf{c}}' \in \bar{\mathscr{C}}$ *such that* $\|(\mathbf{w} - \mathbf{vs})\mathbf{c}' \bmod q\|_\infty < q/2C$ *and* $\|(\mathbf{w} - \mathbf{vs})\mathbf{c}' \bmod q \bmod p\|_\infty < p/2C$ *we have* $(\mathbf{w} - \mathbf{vs})\bar{\mathbf{c}}' \bmod q \bmod p / \bar{\mathbf{c}}' = \bar{\mathbf{m}}/\bar{\mathbf{c}}$.

Proof. The proof is a simple verification of the claims and it is very similar to the proof of Lemma 3.1 in [Lyubashevsky and Neven, 2017].

The first part follows easily from the hypotheses.
To prove the first part, we note that

$$(\mathbf{w} - \mathbf{vs})\bar{\mathbf{c}} \bmod q = p(\bar{\mathbf{r}}\mathbf{d} + \bar{\mathbf{f}} - \bar{\mathbf{e}}\mathbf{s}) + \bar{\mathbf{m}} \,,$$

which has ℓ_∞ length less than $\frac{q}{2C}$ by the hypothesis of the lemma, and therefore

$$(\mathbf{w} - \mathbf{vs})\bar{\mathbf{c}} \bmod q \bmod p = \bar{\mathbf{m}}$$

which has ℓ_∞ length less than $p/2C$, also by hypothesis.
To prove the second part, first note that

$$\begin{aligned}(\mathbf{w} - \mathbf{vs})\bar{\mathbf{c}}\,\bar{\mathbf{c}}' \bmod q \bmod p &= (p(\bar{\mathbf{r}}\mathbf{d} + \bar{\mathbf{f}} - \bar{\mathbf{e}}\mathbf{s}) + \bar{\mathbf{m}})\bar{\mathbf{c}}' \bmod q \bmod p \\ &= \bar{\mathbf{m}}\bar{\mathbf{c}}' \bmod p = \bar{\mathbf{m}}\bar{\mathbf{c}}'. \end{aligned} \tag{3.28}$$

We can then write

$$\begin{aligned}(\mathbf{w} - \mathbf{vs})\bar{\mathbf{c}}' \bmod q \bmod p/\bar{\mathbf{c}}' &= (\mathbf{w} - \mathbf{vs})\bar{\mathbf{c}}' \bmod q \bmod p\,\bar{\mathbf{c}}/(\bar{\mathbf{c}}\bar{\mathbf{c}}') \\ &= (\mathbf{w} - \mathbf{vs})\bar{\mathbf{c}}' \bmod q\,\bar{\mathbf{c}} \bmod p/(\bar{\mathbf{c}}\bar{\mathbf{c}}') \\ &= (\mathbf{w} - \mathbf{vs})\bar{\mathbf{c}}\bar{\mathbf{c}}' \bmod q \bmod p/(\bar{\mathbf{c}}\bar{\mathbf{c}}') \\ &= \bar{\mathbf{m}}\bar{\mathbf{c}}'/(\bar{\mathbf{c}}\bar{\mathbf{c}}') = \bar{\mathbf{m}}/\bar{\mathbf{c}}\end{aligned}$$

The first equality is an identity. The second equality holds because $\|(\mathbf{w} - \mathbf{vs})\bar{\mathbf{c}} \bmod q \bmod p\|_\infty < p/2C$ and so multiplication by $\bar{\mathbf{c}}$ does not cause a reduction modulo p. The third equality is true because $\|(\mathbf{w} - \mathbf{vs})\bar{\mathbf{c}} \bmod q\|_\infty < \frac{q}{2C}$ and so multiplication by $\bar{\mathbf{c}}$ does not cause a reduction modulo q. The last equality is due to (3.28). □

Hence, for decryption to be correct, we must choose parameters that guarantee that the values decrypted from Π_2 using $\mathbf{s}_i$ for $i = 1,2$ satisfy $\|p(\bar{\mathbf{r}}_i\mathbf{d}_i + \bar{\mathbf{f}}_i - \bar{\mathbf{e}}_i\mathbf{s}_i) + \bar{\mathbf{m}}\|_\infty < q/2C$ and $\|\bar{\mathbf{m}}_i\|_\infty < p/2C$, i.e., p, q and n should be such that $16\sigma_2(2np + p + 1) < q/2C$ and $16\sigma_2 < p/2C$, where $C \leq 64$ as challenges come from $R_3^{(16)}$. We enforce this condition on both ciphertexts to guarantee decryption to work using either $\mathbf{s}_1$ or $\mathbf{s}_2$. This allows to prove CCA simulatability following the Naor-Yung paradigm [Naor and Yung, 1990].

In the next lemma, we prove that with high probability the $\bar{\mathbf{m}}/\bar{\mathbf{c}}$ returned by decryption is equal to the polynomial $\bar{\mathbf{m}}'/\bar{\mathbf{c}}'$ returned from an extractor for Π_2.

Lemma 3.42. *Let $\bar{\mathbf{m}}$ and $\bar{\mathbf{c}}$ be the values output by decryption and $\bar{\mathbf{m}}', \bar{\mathbf{c}}'$ be the values extracted from Π_1. Then with probability $1 - \nu(\lambda)$, over the choice of the opening key $\mathbf{t}$, $\bar{\mathbf{m}}/\bar{\mathbf{c}} = \bar{\mathbf{m}}'/\bar{\mathbf{c}}'$.*

Proof. First, observe that because the $\mathbf{a}$ part of the public key is chosen uniformly at random, the $\mathbf{t}$ part is also distributed uniformly random in R_q (when seen apart from $\mathbf{a}$). The decryption algorithm gives that $p(\mathbf{t}\bar{\mathbf{r}} + \bar{\mathbf{f}}) + \bar{\mathbf{m}} = \bar{\mathbf{c}}\mathbf{w}$ and the correctness of extractor E gives also that $p(\mathbf{t}\bar{\mathbf{r}}' + \bar{\mathbf{f}}') + \bar{\mathbf{m}}' = \bar{\mathbf{c}}'\mathbf{w}$. Combining the above two equations we get

$$p(\mathbf{t}\tilde{\mathbf{r}} + \tilde{\mathbf{f}}) + \tilde{\mathbf{m}} = \mathbf{0} \tag{3.29}$$

where $\tilde{\mathbf{r}} = \bar{\mathbf{r}}/\bar{\mathbf{c}} - \bar{\mathbf{r}}'/\bar{\mathbf{c}}'$, $\tilde{\mathbf{f}} = \bar{\mathbf{f}}/\bar{\mathbf{c}} - \bar{\mathbf{f}}'/\bar{\mathbf{c}}'$, $\tilde{\mathbf{m}} = \bar{\mathbf{m}}/\bar{\mathbf{c}} - \bar{\mathbf{m}}'/\bar{\mathbf{c}}'$. We now want to compute the probability over the random choice of $\mathbf{t}$ that for a fixed $\tilde{\mathbf{r}}, \tilde{\mathbf{f}}$, and $\tilde{\mathbf{m}}$, that (3.29) holds. We first show that if $\tilde{\mathbf{r}} = \mathbf{0}$, then (3.29) cannot hold unless $\tilde{\mathbf{m}} = \mathbf{0}$ (and thus $\bar{\mathbf{m}}/\bar{\mathbf{c}} = \bar{\mathbf{m}}'/\bar{\mathbf{c}}'$). This is because $p\tilde{\mathbf{f}} + \tilde{\mathbf{m}} = \mathbf{0}$ implies that

$$p(\bar{\mathbf{f}}\bar{\mathbf{c}}' - \bar{\mathbf{f}}'\bar{\mathbf{c}}) + (\bar{\mathbf{m}}\bar{\mathbf{c}}' - \bar{\mathbf{m}}'\bar{\mathbf{c}}) = \mathbf{0} \bmod q, \tag{3.30}$$

and since $\|p(\bar{\mathbf{f}}\bar{\mathbf{c}}' - \bar{\mathbf{f}}'\bar{\mathbf{c}}) + (\bar{\mathbf{m}}\bar{\mathbf{c}}' - \bar{\mathbf{m}}'\bar{\mathbf{c}})\|_\infty < 32np(\sigma_2 + \sigma_1) + 32(16\sigma_2 + \sigma_1 n) < q/2$ (we used the triangle inequality and Lemma 2.12 to obtain the bound), no reduction modulo q takes place. And because $\|\bar{\mathbf{m}}\bar{\mathbf{c}}' - \bar{\mathbf{m}}'\bar{\mathbf{c}}\|_\infty < 32(16\sigma_2 + \sigma_1 n) < p$, (again using Lemma 2.12) the only way that (3.30) can be satisfied is if $\bar{\mathbf{m}}\bar{\mathbf{c}}' - \bar{\mathbf{m}}'\bar{\mathbf{c}} = \mathbf{0}$.

So now suppose that $\tilde{\mathbf{r}} \neq \mathbf{0}$. Then

$$\Pr_{\mathbf{t} \xleftarrow{\$} R_q}[(3.29)] = \Pr_{\mathbf{t} \xleftarrow{\$} R_q}[\mathbf{t} = -(\tilde{\mathbf{m}} + \tilde{\mathbf{f}})/(p\tilde{\mathbf{r}})] = 1/|R_q| = q^{-n}.$$

We will now compute the total number of possibilities for $\tilde{\mathbf{r}}$, $\tilde{\mathbf{f}}$, and $\tilde{\mathbf{m}}$ and apply the union bound. Since these three values are completely determined by $\bar{\mathbf{r}}$, $\bar{\mathbf{r}}'$, $\bar{\mathbf{f}}$,

$\bar{\mathbf{f}}'$, $\bar{\mathbf{m}}$, $\bar{\mathbf{m}}'$, $\bar{\mathbf{c}}$, $\bar{\mathbf{c}}'$, it is enough to bound the domain size of these elements. From the bounds on the infinity norm of Gaussian samples (Lemma 2.16) and rejection sampling, we know that there are $n_1 = (2 \cdot 8\sigma_2 + 1)^{2n}$ choices for $\bar{\mathbf{r}}$, $\bar{\mathbf{f}}$, $n_2 = (2 \cdot 8\sigma_1 + 1)^{3n}$ choices for $\bar{\mathbf{r}}'$, $\bar{\mathbf{f}}'$, $\bar{\mathbf{m}}'$ and $n_3 = (2 \cdot 8\sigma_1 + 1)^{16}$ choices for $\bar{\mathbf{m}}$ (since $\bar{\mathbf{m}} \in R^{(16)}$). The domains of $\bar{\mathbf{c}}$ and $\bar{\mathbf{c}}'$ are much less than 2^{512} each. Thus the number of possibilities for $\tilde{\mathbf{r}}, \tilde{\mathbf{f}}, \tilde{\mathbf{m}}$ is less than $n_1 \cdot n_2 \cdot n_3 \cdot 2^{512} \cdot 2^{512}$.Therefore, the probability over the choices of $\mathbf{t}$ that there exists some $\tilde{\mathbf{r}}, \tilde{\mathbf{e}}, \tilde{\mathbf{m}}$ that satisfies (3.29) is less than $(2 \cdot 8\sigma_2 + 1)^{2n} \cdot (2 \cdot 8\sigma_1 + 1)^{3n} \cdot (2 \cdot 8\sigma_1 + 1)^{16} \cdot 2^{512} \cdot 2^{512} \cdot q^{-n}$, that is negligible as long as $q \gg \sigma_i$. □

Remark that we have chosen a small set of ciphertexts in this case, as we intend to use this scheme in our privacy-preserving signatures only to encrypt users' identities.

Finally, for the CCA simulatability the proofs that we use in the scheme need to be unbounded non-interactive zero-knowledge and unbounded simulation sound. As Lyubashevsky and Neven observe, the result by Faust et al. [2012] (adapted to relaxed protocols) implies that the quasi unique response and HVZK properties of a Σ-protocol are enough to guarantee that the NI proof obtained through the FS heuristic has unbounded non-interactive zero-knowledge and unbounded simulation soundness in the ROM. Hence, following their reasoning, we prove that Π_2 has quasi-unique responses (HVZK follows from using the Σ-protocol from Section 3.4). Indeed, breaking quasi-uniqueness means finding $\mathbf{z} \neq \mathbf{z}'$ with ℓ_∞ norm less than $8\sigma_2$ such that $\mathbf{Mz} = \mathbf{Mz}' \bmod q$, where with $\mathbf{M}$ we mean the matrix in 3.26. Thus, either there is a non-zero tuple $(\mathbf{y}_1, \mathbf{y}_2) \in R_q$ with ℓ_∞ norm less than $16\sigma_2$ such that $p(\mathbf{ay}_1 + \mathbf{y}_2) = 0 \bmod q$ or $p\mathbf{y}_1 + \mathbf{y}_2 = 0 \bmod q$. Imposing $p > 16\sigma_2$ and $16\sigma_2 p + 16\sigma_2 < q$ implies that the second equality is not possible. Also, setting $(32\sigma_2)^2 < q$, we can use a standard probabilistic argument to show that for all $\mathbf{y}_1$, $\mathbf{y}_2$ of ℓ_∞ norm less than $16\sigma_2$,

$$\mathsf{Pr}_{\mathbf{a} \in R_q}[\mathbf{ay}_1 + p\mathbf{y}_2 = 0 \bmod q] = 2^{-\Omega(n)} .$$

Therefore for almost all $\mathbf{a}$, there will not be a short solution $(\mathbf{y}_1, \mathbf{y}_2)$ that satisfies $\mathbf{ay}_1 + p\mathbf{y}_2 = 0$. Observe that the same argument works for Π_1. Hence imposing the same inequalities on σ_1 yields simulation soundness also for Π_1, thus for the protocol (Π_1, Π_2).

Theorem 3.43. *If $RLWE_{\mathscr{U}(R_q)}$ is hard and the relaxed NIZK proof system is unbounded non-interactive zero-knowledge and unbounded simulation soundness, the scheme* (EKeyGen, Enc, EVerify, Dec) *is a relaxed partial verifiable encryption scheme w.r.t. the function g in the Random Oracle Model (where the adversary can make a polynomial number of queries to the oracle).*

Proof. Correctness is fast to check: if a ciphertext $(\mathbf{v}, \mathbf{w}_1, \mathbf{w}_2)$ is a honest encryption of some $\mathbf{m}$, then for every $\mathbf{c}' \in \mathscr{C}_1$, setting $\bar{\mathbf{c}} = \mathbf{c} - \mathbf{c}'$, we have

$$\begin{aligned}\|(\mathbf{w}_1 - \mathbf{v}\mathbf{s}_1)\bar{\mathbf{c}}\|_\infty &= \|p(\mathbf{d}\bar{\mathbf{r}} + \bar{\mathbf{f}} - \bar{\mathbf{e}}\mathbf{s}_1) + \bar{\mathbf{m}}\|_\infty \\ &\leq p(16\sigma_1 \cdot 1 \cdot n + 16\sigma_1 + 16\sigma_1 \cdot 1 \cdot n) + 16\sigma_1 < q/2C \ ,\end{aligned}$$

where in the first inequality we used the triangular inequality and Lemma 2.12, while the second is guaranteed by the condition $16\sigma_1(2np + p + 1) < q/2C$. Finally, the condition $16\sigma_1 < p/2C$ guarantees that $\|\bar{\mathbf{m}}\|_\infty < p/2C$, hence all the hypotheses of Lemma 3.41 are satisfied and decryption yields $\bar{\mathbf{m}}/\bar{\mathbf{c}} \bmod q = \mathbf{m}$.

Completeness is implied by the completeness of the relaxed NIZK proof in Section 3.4.

Soundness follows from Lemma 3.42 and from the soundness of the NIZK proof. Indeed, assume there exists an adversary A that can output a ciphertext $(\mathbf{v}, \mathbf{w}, \Pi_1, \Pi_2)$ such that decrypting $(\mathbf{v}, \mathbf{w}, \Pi_1)$ and extracting Π_2 it is possible to obtain $\bar{\mathbf{m}}, \bar{\mathbf{c}}$ and $\bar{\mathbf{m}}', \bar{\mathbf{c}}', \bar{\mathbf{S}}'$ such that either $\bar{\mathbf{m}}/\bar{\mathbf{c}} \neq \bar{\mathbf{m}}'/\bar{\mathbf{c}}'$ or $(\bar{\mathbf{m}}, \bar{\mathbf{S}}, \bar{\mathbf{c}}) \notin \bar{\mathscr{R}}_{\text{ve}}$. Lemma 3.42 guarantees that the first case happens with negligible probability, while the second case implies breaking the relaxed special soundness of the relaxed NIZK proof.

Finally, the proof of CCA simulatability is done with game hops. The proof follows closely the structure of the original proof by Lyubashevsky and Neven (cf. proof of Theorem 5.2 in the full version of [Lyubashevsky and Neven, 2017]).

The idea is to prove that, if there exists a PPT adversary A that breaks the chosen-ciphertext simulatability of our scheme with advantage ϵ, then it is possible to construct PPT algorithms B_{ZK} that breaks the zero-knowledge property of the underlying NIZK proof with advantage ϵ_{ZK}, B_{S} that breaks the simulation soundness of the underlying NIZK proof with advantage ϵ_S, and B_{RLWE} that breaks the IND-CPA security of the RLWE encryption scheme with advantage ϵ_{RLWE}, such that

$$\epsilon \leq \epsilon_{ZK} + 2\epsilon_{RLWE} + 16\epsilon_S \ .$$

The proof is articulated in 2 steps:

1. Define a simulator Σ_{pve},

2. prove with a sequence of game hops that if the adversary can distinguish the simulated ciphertext from an honestly generates one, it is possible to build B_{ZK}, B_S and B_{RLWE}.

Simulator $\Sigma_{\text{pve}}(\mathbf{a}, \mathbf{t}_1, \mathbf{t}_2, \mathbf{A}, \mathbf{U})$
$\mathbf{r}, \mathbf{e}_1, \mathbf{e}_2, \mathbf{f}_1, \mathbf{f}_2 \xleftarrow{\$} R_3$
$\mathbf{v}_1 := p(\mathbf{ar} + \mathbf{e}_1) \bmod q$
$\mathbf{v}_2 := p(\mathbf{ar} + \mathbf{e}_2) \bmod q$
$\mathbf{w}_1 := p(\mathbf{t}_1\mathbf{r} + \mathbf{f}_1) + \mathbf{1} \bmod q$
$\mathbf{w}_2 := p(\mathbf{t}_2\mathbf{r} + \mathbf{f}_2) + \mathbf{1} \bmod q$
$(\Pi_1, \Pi_2) \leftarrow \Sigma_{NIZK}(\mathbf{A}, \mathbf{U}, \mathbf{v}_1, \mathbf{w}_1, \mathbf{v}_2, \mathbf{w}_2, \mathbf{a}, \mathbf{t}_1, \mathbf{t}_2)$
Return $(\mathbf{v}_1, \mathbf{w}_1, \mathbf{v}_2, \mathbf{w}_2, \Pi_1, \Pi_2)$.

Figure 3.11. Simulator for the chosen-message simulatability experiment.

The simulator Σ_{pve} is shown in Figure 3.11. On input a language member $(\mathbf{A}, \mathbf{U})$, it simulates the outputs of a legit round of Enc by generating two RLWE encryptions of 1 and getting the NIZK proofs from the simulators Σ_{NIZK}[3] of the relaxed NIZK proofs(cf. Section 3.4). The queries of the adversary to the random oracle are handled through the simulator $\Sigma_{NIZK,1}$. Through a sequence of game hops, we prove that if an adversary can distinguish simulated ciphertexts from honestly generated ones, then there exists an algorithm B that either breaks Ring-LWE$_{\mathscr{U}(R_q)}$, or the unbounded non-interactive zero-knowledge or the unbounded simulation-soundness of the NIZK proof. Given the security experiment $\mathsf{Exp}_{\mathsf{A}}^{ccas-b}(1^\lambda)$ from Definition 3.38, the advantage of a PPT adversary in breaking CCA-simulatability is:

$$\begin{aligned}
\epsilon &= \left|\mathsf{Pr}\left[b' = b \ : \ b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{ccas-b}(1^\lambda)\right] - \tfrac{1}{2}\right| \\
&= \tfrac{1}{2}\left|\mathsf{Pr}\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{ccas-0}(1^\lambda)\right] + \mathsf{Pr}\left[b' = 1 \ : \ b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{ccas-1}(1^\lambda)\right] - 1\right| \\
&= \tfrac{1}{2}\left|\mathsf{Pr}\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{ccas-0}(1^\lambda)\right] - \mathsf{Pr}\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{ccas-1}(1^\lambda)\right]\right| .
\end{aligned}$$

Let $\mathbf{Game}_\mathsf{i}$ be the probability that the adversary outputs 0 when executed in the environment of Game i.

Game 0. This is the experiment $\mathsf{Exp}_{\mathsf{A}}^{ccas-0}(1^\lambda)$ where the bit is set to $b = 0$. Then, the probability that A outputs 0 in this game is:

$$\mathsf{Pr}[\mathbf{Game}_\mathbf{0}] = \mathsf{Pr}[b' = 0 \ : \ \ b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{ccas-0}(1^\lambda)] .$$

[3]We denote by Σ_{NIZK} the combination of the simulators of the two NIZK proofs. The two provers sharing the same random oracle does not generate problems due to collision, because the queries coming from the two provers have always different structures as the relations are different.

Experiment $\mathsf{Exp}^{indcpa-b}_{\mathsf{B}_2}(1^\lambda)$
$(\mathbf{a},\mathbf{b}) \leftarrow \mathsf{EKeyGen}_{RLWE}(1^\lambda)$
$(\mathbf{m}^*, st) \leftarrow \mathsf{B}_2(\mathbf{a},\mathbf{b})$
$(\mathbf{v}_0,\mathbf{w}_0) \leftarrow \mathsf{Enc}_{RLWE}(\mathbf{m}^*,\mathbf{a},\mathbf{b})$
$(\mathbf{v}_1,\mathbf{w}_1) \leftarrow \mathsf{Enc}_{RLWE}(\mathbf{1},\mathbf{a},\mathbf{b})$
$b' \leftarrow \mathsf{B}_2(\mathbf{v}_b,\mathbf{w}_b,st)$
Return b'.

Figure 3.12. IND-CPA security game for RLWE encryption scheme.

Game 1. In this game the NIZK proofs in the challenge ciphertext $(\mathbf{v}_1^*, \mathbf{v}_2^*, \mathbf{w}_1^*, \mathbf{w}_2^*, \Pi_1^*, \Pi_2^*)$ are generated using the simulator Σ_{NIZK}. If the adversary can distinguish Game 1 from Game 0 with non-negligible probability, then it is possible to build an algorithm B_1 that break the zero knowledge property of the NIZK proof as follows. The algorithm B_1 has access to an oracle $\mathcal{O}$, and to win the ZK experiment it has to output 1 if $\mathcal{O} = (H_c, \mathsf{P}^{H_c})$, or 0 if $\mathcal{O} = (\Sigma_1, \Sigma_2)$ (cf. Definition 3.4). It runs the protocol as in Game 0, except for the generation of the proofs Π_1^* and Π_2^*. These are obtained from the prover oracle $\mathcal{O}$ that is either $\mathsf{P}_i^{H_c}$, or $\Sigma_{NIZK,2}$. The queries by A to the random oracle H_c are handled by B_1 through $\mathcal{O}$ as well. The oracle either honestly evaluates H_c, or uses $\Sigma_{NIZK,1}$. At the end of the game, B_1 outputs the bit $1-b'$, where b' is the bit output by A. The probability that B_1 breaks the ZK property is

$$\begin{aligned}\epsilon_1 &= \left|\Pr\left[0 \leftarrow \mathsf{B}_1^{H_c(\cdot),\mathsf{P}^{H_c}(\cdot,\cdot)}(1^\lambda)\right] - \Pr\left[0 \leftarrow \mathsf{B}_1^{\Sigma_1(\cdot),\Sigma_2(\cdot,\cdot)}(1^\lambda)\right]\right| \\ &= \frac{1}{2}\left|\Pr[\mathbf{Game_0}] - \Pr[\mathbf{Game_1}]\right| .\end{aligned}$$

Game 2. This game is equal to Game 1, except for the redundant RLWE ciphertext $(\mathbf{v}_2^*, \mathbf{w}_2^*)$, which is generated as the encryption of $\mathbf{1}$ instead of being the encryption of the challenge message $\mathbf{m}^*$. This does not affect the NIZK proofs Π_1^* and Π_2^*, as they are simulated anyway. If a PPT algorithm A has non-negligible advantage in distinguishing Game 2 from Game 1, then we can build a PPT algorithm B_2 that exploits A to win the IND-CPA experiment $\mathsf{Exp}^{indcpa-b}_{\mathsf{B}_2}(1^\lambda)$ (cf. Figure 3.12) of the RLWE encryption.

Upon receiving the public RLWE encryption key $(\mathbf{a},\mathbf{t})$, B_2 samples $\mathbf{s}_1, \mathbf{d}_1 \xleftarrow{\$} R_3$ and sets $\mathbf{t}_1 := \mathbf{a}\mathbf{s}_1 + \mathbf{d}_1 \bmod q$, $\mathbf{t}_2 := \mathbf{t}$ and $epk = (\mathbf{a}, \mathbf{t}_1, \mathbf{t}_2)$. Whenever it received a decryption query, B_2 can decrypt the ciphertext using $\mathbf{s}_1$. When A

returns the challenge $(\mathbf{A}^*, \mathbf{U}^*, \mathbf{m}^*, \mathbf{S}^*, \mathbf{1}^*)$, B_2 uses $\mathbf{m}^*$ as the challenge message in the IND-CPA game. Upon receiving the challenge RLWE ciphertext $(\mathbf{v}^*, \mathbf{w}^*)$, B_2 generates the challenge ciphertext in the CCA-simulatability experiment as $(\mathbf{v}_1^*, \mathbf{w}_1^*) \leftarrow \mathsf{Enc}_{RLWE}(\mathbf{a}, \mathbf{t}_1, \mathbf{m}^*)$, $(\mathbf{v}_2^*, \mathbf{w}_2^*) := (\mathbf{v}^*, \mathbf{w}^*)$ and it simulates the proofs Π_1^* and Π_2^*. B_2 returns to the challenger in the IND-CPA experiment the bit b' output by A.
It is clear that if $(\mathbf{v}^*, \mathbf{w}^*)$ is an encryption of $\mathbf{1}$, the view of A is exactly the same as if it was playing Game 2, while if $(\mathbf{v}^*, \mathbf{w}^*)$ is the encryption of $\mathbf{m}^*$ it is exactly game 1. Therefore, the advantage of B_2 in breaking the IND-CPA security of the RLWE encryption is

$$\begin{aligned}
\epsilon_2 &= \left|\mathsf{Pr}\left[b' = b \ : \ b \xleftarrow{\$} \{0,1\},\ b' \leftarrow \mathsf{Exp}_{\mathsf{B}_2}^{indcpa-b}(1^\lambda)\right] - \tfrac{1}{2}\right| \\
&= \tfrac{1}{2}\left|\mathsf{Pr}\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}_{\mathsf{B}_2}^{indcpa-0}(1^\lambda)\right] + \mathsf{Pr}\left[b' = 1 \ : \ b' \leftarrow \mathsf{Exp}_{\mathsf{B}_2}^{indcpa-1}(1^\lambda)\right] - 1\right| \\
&= \tfrac{1}{2}\left|\mathsf{Pr}\left[b' = 0 \wedge b' \leftarrow \mathsf{Exp}_{\mathsf{B}_2}^{indcpa-0}(1^\lambda)\right] - \mathsf{Pr}\left[b' = 0 \wedge b' \leftarrow \mathsf{Exp}_{\mathsf{B}_2}^{indcpa-1}(1^\lambda)\right]\right| \\
&= \tfrac{1}{2}\left|\mathsf{Pr}\left[\mathbf{Game_1}\right] - \mathsf{Pr}\left[\mathbf{Game_2}\right]\right| .
\end{aligned}$$

Game 3. In this game the decryption is performed using the secret key $\mathbf{s}_2$ instead of $\mathbf{s}_1$. As the decryption procedure does not change when switching from using $\mathbf{s}_1$ to $\mathbf{s}_2$, the output $\mathbf{m}_1/\mathbf{c}_1 \bmod q$ of the decryption of a ciphertext $(\mathbf{v}_1, \mathbf{w}_1, \mathbf{V}_2, \mathbf{W}_2, \Pi_1, \Pi_2)$ w.r.t. $\mathbf{s}_1$ has the same distribution as the output $\mathbf{m}_2/\mathbf{c}_2 \bmod q$ of the decryption of the same ciphertext w.r.t. $\mathbf{s}_1$. Hence, the only way A can distinguish Game 2 from Game 3 is by encrypting different messages in $(\mathbf{v}_1, \mathbf{w}_1)$ and $(\mathbf{v}_2, \mathbf{w}_2)$. In this case though it is possible to construct an algorithm B_3 that breaks the simulation soundness of the NIZK proof exploiting A. Such algorithm handles the decryption queries of A by decrypting w.r.t. both $\mathbf{s}_1$ and $\mathbf{s}_2$, and checking that the plaintexts obtained are the same. Then, B_3 rewinds A on the random oracle query for which the plaintexts resulted different to extract from Π_1 a complete witness w.r.t. the relation $\mathscr{R}_{\text{ve}}$ for the relaxed simulation soundness game (cf. Definition 3.10). Lemma 3.42 guarantees that the plaintexts extracted from Π_1 are equal to the ones obtained from decryption with all but negligible probability. According to the Generalized Forking Lemma (cf. Lemma 2.6), this degrades the success probability of B_3 by $\epsilon_{\mathsf{A}}/8$, where ϵ_{A} is the advantage of A. The success probability of B_3 is then the advantage of A in distinguishing Game 3 from Game 2

$$\epsilon_3 \geq \frac{1}{16}\left|\mathsf{Pr}\left[\mathbf{Game_2}\right] - \mathsf{Pr}\left[\mathbf{Game_3}\right]\right| .$$

Game 4. In this game, the challenge ciphertext $(\mathbf{v}_1^*, \mathbf{w}_1^*)$ is generated as an encryption of $\mathbf{1}$ as well. Analogously to Game 2, if there exists a PPT adversary A that distinguishes Game 4 from Game 3 with non-negligible advantage, then we can construct a PPT algorithm B_4 that breaks the IND-CPA security of the RLWE encryption with advantage

$$\epsilon_4 = \frac{1}{2}\left|\mathsf{Pr}[\mathbf{Game_3}] - \mathsf{Pr}\left[\mathbf{Game_4}\right]\right| .$$

Game 5. In Game 5 everything is the same as in Game 4, except that now decryption queries are answered using the secret key $\mathbf{s}_1$ instead of $\mathbf{s}_2$. Similarly to Game 3, if there exists a PPT adversary A that distinguishes Game 5 from Game 4 with non-negligible advantage, then we can construct a PPT algorithm B_5 that breaks the simulation-soundness property of the NIZK proof with advantage

$$\epsilon_5 \geq \frac{1}{16}\left|\mathsf{Pr}\left[\mathbf{Game_4}\right] - \mathsf{Pr}[\mathbf{Game_5}]\right| .$$

Remark that in Game 5 A is now interacting with a challenger that behaves as in $\mathsf{Exp}_{\mathsf{A}}^{ccas-1}(1^\lambda)$, where the simulator is the one in Figure 3.11. Hence,

$$\mathsf{Pr}[\mathbf{Game_5}] = \mathsf{Pr}[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{ccas-1}(1^\lambda)] .$$

To conclude the proof, we need to construct B_{ZK}, B_S and B_{RLWE}. The algorithm B_S (resp., B_{RLWE}) runs either B_3 or B_5 (resp., either B_2 or B_4) with probability $\frac{1}{2}$. Hence, its advantage in winning the CCA-simulatability experiment (resp., the IND-CPA experiment) is $\epsilon_S = 1/2(\epsilon_3 + \epsilon_5)$ (resp., $\epsilon_{RLWE} = 1/2(\epsilon_2 + \epsilon_4)$). Putting it all together we get that the advantage of a PPT adversary in breaking CCA-simulatability is:

$$\begin{aligned}
\epsilon &= \frac{1}{2}\left|\mathsf{Pr}\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{ccas-0}(1^\lambda)\right] - \mathsf{Pr}\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{ccas-1}(1^\lambda)\right]\right| \\
&= \frac{1}{2}\left|\mathsf{Pr}[\mathbf{Game_0}] - \mathsf{Pr}[\mathbf{Game_5}]\right| \\
&= \frac{1}{2}\left|\sum_{i=0}^{4} \mathsf{Pr}[\mathbf{Game_i}] - \mathsf{Pr}[\mathbf{Game_{i+1}}]\right| \\
&\leq \sum_{i=0}^{4} \frac{1}{2}\left|\mathsf{Pr}[\mathbf{Game_i}] - \mathsf{Pr}[\mathbf{Game_{i+1}}]\right| \\
&= \epsilon_1 + \epsilon_2 + 8\epsilon_3 + \epsilon_4 + 8\epsilon_5 = \epsilon_{ZK} + 2\epsilon_{RLWE} + 16\epsilon_S .
\end{aligned}$$

□

Chapter 4

Anonymous Attribute Tokens from Relaxed Building Protocols

The primitives defined in the previous chapter are now put to use to build Anonymous Attribute Tokens (AAT). This protocol was introduced by Camenisch et al. [2012] and can be seen as simplified anonymous credentials, allowing users to obtain from an issuer a credential that contains a list of attributes. Users can selectively disclose subsets of these attributes to verifiers in such a way that not even the verifier and the issuer together can link different presentations by the same user. Depending on the anonymity guarantees there can be two "flavors" of AATs. If it is not possible to extract the user identity from the token, not even by the credentials' issuer, then the AAT is said to be *without opening* (AAT-O). If the AAT allows for an opener to revoke the anonymity of a user (e.g., in case of suspected misbehavior), the AAT scheme is said to be an AAT *with opening* (AAT+O). In this chapter we build both types of AAT from the relaxed building blocks of Chapter 3. The content is based on the work published in [Boschini et al., 2018b] (full version: [Boschini et al., 2017]).

4.1 Anonymous Attribute Tokens without Opening

Anonymous attribute tokens allow users to obtain from an issuer a credential containing a list of attributes. They can then selectively disclose subsets of these attributes to verifiers in such a way that not even the verifier and the issuer together can link different presentations by the same user.

In this section, we focus on AAT schemes without anonymity revocation (AAT-O), i.e., without a trusted opener who can de-anonymize presentation tokens. AAT+R are defined in Section 4.2.1.

4.1.1 Definition of AAT-O Schemes

Consider for simplicity the scenario in which each user identity corresponds to exactly one credential. That credential contains all the attributes that the issuer will ever issue to that user identity. This still models the general case, as one can see the user identity as a credential identity that binds together the attributes of that credential, and assume that issuers can hand multiple such identity credentials to the same user. Remark that for the sake of simplicity we model the protocol as interactions between multiple users and a single issuer. Considering multiple issuers would make the model and the proofs unnecessarily convoluted.

An AAT-O is composed by the following algorithms:

System Parameters Generation. The public parameters of the scheme are generated from the security parameter as $apar \leftarrow \mathsf{SPGen}(1^\lambda)$. They are common to all the parties.

Issuer Key Generation. An issuer generates a public key ipk and corresponding secret key isk by running $\mathsf{IKGen}(apar)$.

Credential issuance. To issue a credential for attributes $(\alpha_i)_{i=1}^{\ell}$ to a user, the issuer samples a user identity id from the set $\mathscr{U}$ of allowed user identities, checks that id is not in the list $\mathscr{S}$ of issued user identities (otherwise, it aborts) and runs $\mathsf{Issue}(isk, id, (\alpha_i)_{i=1}^{\ell})$. It hands the resulting user identity id and credential $cred$ to the user.

Presentation. A user creates a presentation token pt revealing a subset of attributes $(\alpha_i)_{i\in R}$, $R \subseteq \{1,\ldots,\ell\}$, from a credential while authenticating a message μ by running $\mathsf{Present}(ipk, cred, R, \mu)$.

Verification. The verifier checks the validity of a presentation token by running $\mathsf{Verify}(ipk, R, (\alpha_i)_{i\in R}, \mu, pt)$ which returns *accept* or *reject*.

Correctness requires that if the above algorithms are executed honestly, then Verify returns 1 with probability one.

Definition 4.1 (Correctness)**.** An AAT-O scheme $(\mathsf{SPGen}, \mathsf{IKGen}, \mathsf{Issue}, \mathsf{Present}, \mathsf{Verify})$ satisfies correctness if for all user identities id, all $\ell \in \mathbb{N}$, all $\alpha_1,\ldots,\alpha_\ell \in \{0,1\}^*$, and all sets $R \subseteq \{1,\ldots,\ell\}$ the following holds

$$\Pr\left[\, 1 \leftarrow \mathsf{Verify}(ipk, R, (\alpha_i)_{i\in R}, \mu, pt) \;\middle|\; \begin{array}{c} apar \leftarrow \mathsf{SPGen}(1^\lambda), \\ (ipk, isk) \leftarrow \mathsf{IKGen}(apar), \\ (id, cred) \leftarrow \mathsf{Issue}(isk, id, (\alpha_i)_{i=1}^{\ell}), \\ pt \leftarrow \mathsf{Present}(ipk, cred, R, \mu) \end{array} \right] \geq 1 - \nu(\lambda)\,.$$

In the unforgeability experiment, the adversary is given access to the issuance oracles $\mathcal{O}_{IC}$ and $\mathcal{O}_{IH}$ (that issue credentials to corrupt and honest users respectively) and to an oracle $\mathcal{O}_{pres}$ that generates presentation tokens by honest users. An AAT-O is said to be unforgeable if the adversary cannot create a presentation token revealing a set of attributes that was never issued in one credential by the issuance oracles, nor presented in that combination and for the given message by the presentation oracle.

The experiment requires the instantiation of four lists: $\mathcal{S}$ that contains all the issued *id*, $\mathbf{L}_c$ to keep track of the attributes issued to dishonest users, $\mathbf{L}_h$ where the credentials and identities issued to honest users are stored, and $\mathbf{L}_p$ that keeps track of the presentation tokens produced by honest users. The adversary uses unique credential identifiers *cid* to refer to particular pairs of user identity and credential issued to honest users; the credential identifier can be obtained for example as the hash of its corresponding pair $(id, cred)$, i.e., $cid \leftarrow H(id, cred)$, where $H : \{0,1\}^* \to \{0,1\}^\lambda$ is a collision-resistant hash.

Definition 4.2 (Unforgeability)**.** An AAT-O scheme $(\mathsf{SPGen}, \mathsf{IKGen}, \mathsf{Issue}, \mathsf{Present}, \mathsf{Verify})$ satisfies unforgeability if for all PPT adversaries A the advantage of A in winning the unforgeability experiment in Figure 4.1 is negligible:

$$\mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}_{\mathsf{A}}^{\mathsf{unf}}(1^\lambda)\right] \leq \nu(\lambda)\,.$$

Finally, anonymity requires that no PPT adversary can distinguish between two presentation tokens for the same attributes and message, but derived from different credentials provided by the adversary.

Definition 4.3 (Anonymity)**.** An AAT-O scheme $(\mathsf{SPGen}, \mathsf{IKGen}, \mathsf{Issue}, \mathsf{Present}, \mathsf{Verify})$ satisfies anonymity if for all PPT adversaries A the advantage of A in winning the anonymity experiment in Figure 4.2 is negligible:

$$\left|\mathsf{Pr}\left[b' = b \;:\; b \xleftarrow{\$} \{0,1\},\; b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{\mathsf{anon}-b}(1^\lambda)\right] - \frac{1}{2}\right| \leq \nu(\lambda)\,.$$

4.1.2 Compact AAT-O from Lattices

From the relaxed primitives that we introduced in Chapter 3, it is possible to construct an AAT-O scheme with compact presentation tokens. In such scheme, a user identity is a small polynomial $\mathbf{m}$ in the set $\mathcal{U}$ defined in Equation (3.5). A credential for ℓ attributes $\{\alpha_i\}_i$ is composed by ℓ relaxed signatures (cf. Section 3.6.3) on the pairs user identity-attribute $(\mathbf{m}, \alpha_i)$. A user can then prove in zero-knowledge using the relaxed NIZK $(\mathsf{P}_{pt}^{H_c}, \mathsf{V}_{pt}^{H_c})$ in Section 3.4. Efficiency follows from the aggregation property shown at the end of Section 3.7.

Experiment $\mathsf{Exp}^{\mathsf{unf}}_{\mathsf{A}}(1^\lambda)$
$apar \leftarrow \mathsf{SPGen}(1^\lambda)$
$\mathscr{S} \leftarrow \emptyset$, $\mathbf{L}_c \leftarrow \emptyset$, $\mathbf{L}_h \leftarrow \emptyset$, $\mathbf{L}_p \leftarrow \emptyset$
$(ipk^*, isk^*) \leftarrow \mathsf{IKGen}(apar)$
$(pt^*, R^*, (\alpha_i^*)_{i\in R^*}, \mu^*) \leftarrow \mathsf{A}^{\mathscr{O}_{IC},\mathscr{O}_{IH},\mathscr{O}_{pres}}(apar, ipk^*)$
If $1 \leftarrow \mathsf{Verify}(ipk^*, R^*, (\alpha_i^*)_{i\in R^*}, \mu^*, pt^*)$
and $\nexists$ *cid* such that $(cid, (\alpha_i^*)_{i\in R^*}) \in \mathbf{L}_c$
and $\nexists$ *cid* such that $(cid, R^*, \mu^*) \in \mathbf{L}_p$
then return 1 else return 0.

Oracle $\mathscr{O}_{IC}((\alpha_i)_{i=1}^{\ell})$
$id \leftarrow \mathscr{U} \setminus \mathscr{S}$
$cred \leftarrow \mathsf{Issue}(isk^*, id, (\alpha_i)_{i=1}^{\ell})$
$\mathscr{S} \leftarrow \mathscr{S} \cup \{id\}$
$cid \leftarrow H(id, cred)$
$\mathbf{L}_c \leftarrow \mathbf{L}_c \cup \{(cid, (\alpha_i^*)_{i\in R^*})\}$
Return $(id, cred)$.

Oracle $\mathscr{O}_{IH}((\alpha_i)_{i=1}^{\ell})$
$id \leftarrow \mathscr{U} \setminus \mathscr{S}$
$cred \leftarrow \mathsf{Issue}(isk^*, id, (\alpha_i)_{i=1}^{\ell})$
$cid \leftarrow H(id, cred)$
$\mathbf{L}_h \leftarrow \mathbf{L}_h \cup \{(cid, id, cred)\}$
$\mathscr{S} \leftarrow \mathscr{S} \cup \{id\}$
Return cid .

Oracle $\mathscr{O}_{pres}(cid, R, \mu)$
Get $cred$ such that $(cid, id, cred) \in \mathbf{L}_h$
(abort if it does not exist).
$pt \leftarrow \mathsf{Present}(ipk^*, cred, R, \mu)$
$\mathbf{L}_p \leftarrow \mathbf{L}_p \cup \{(cid, R, \mu)\}$
Return pt .

Figure 4.1. Unforgeability experiment for AAT-O.

Remark 14. To combine the commitment and the signature scheme in a way that preserve security it is necessary that their parameters are compatible (e.g., that the matrix $\mathbf{C}$ of both scheme is the same). Hence, we define an additional algorithm $\mathsf{DerivePar}_s$ that derives the parameters for the signature scheme (which in the AAT-O are generated by the issuer) from the parameters of the commitment (which in the AAT-O case are general parameters of the protocol and are generated at the beginning by a trusted third party).

Signature Parameters Derivation. On input the parameters *cpar*, $\mathsf{DerivePar}_s$ sets the public vector $\mathbf{C}$ to be the same vector contained in *cpar* and computes the rest of the signature parameters as in Section 3.6.3.

User identities are sampled from the set $\mathscr{U}$ defined in Section 3.5.3. In practice, this allows to issue around $3^{(2^{K_m-1}-1)}/(n/2)$ identities. To keep track of the issued *id*, the issuer stores them in a list $\mathscr{S}$. We now describe our lattice-based AAT-O.

System Parameter Generation. The algorithm SPGen first generates the commitment parameters $cpar \leftarrow \mathsf{ParGen}_c(1^\lambda)$. Then it derives the parame-

Experiment $\mathsf{Exp}^{\mathsf{anon-b}}_{\mathsf{A}}(1^\lambda)$
$apar \leftarrow \mathsf{SPGen}(1^\lambda)$
$(ipk^*, isk^*) \leftarrow \mathsf{IKGen}(apar)$
$count \leftarrow 0$
$b' \leftarrow \mathsf{A}^{\mathcal{O}^b_C}(apar, ipk^*, isk^*)$
Return b'.

Oracle $\mathcal{O}^b_C(cred^*_0, cred^*_1, R^*, (\alpha^*_i)_{i\in R^*}, \mu^*)$
$count \leftarrow count + 1$
If $count \geq 2$ abort.
For $i = 0, 1$:
$pt^*_i \xleftarrow{\$} \mathsf{Present}(ipk^*, cred^*_i, R^*, \mu^*)$
If $\mathsf{Verify}(ipk^*, R^*, (\alpha^*_i)_{i\in R^*}, \mu^*, pt^*_0) = 1$
and $\mathsf{Verify}(ipk^*, R^*, (\alpha^*_i)_{i\in R^*}, \mu^*, pt^*_1) = 1$
return pt^*_b, else abort.

Figure 4.2. Anonymity experiment for AAT-O.

ters of the relaxed signature $spar \leftarrow \mathsf{DerivePar}_s(cpar)$ and outputs $apar = (cpar, spar)$.

Issuer Key Generation. The issuer runs the signing key generation $\mathsf{SKeyGen}$ to obtain $isk = \mathbf{X}$ and the public matrix $ipk = \begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{C} & \mathbf{1}\end{bmatrix}$.

Issuance. To issue a credential to a user for attributes $(\alpha_i)_{i=1}^{\ell}$, the issuer chooses an $id = \mathbf{m} \in \mathscr{U}$, checks that $\mathbf{m} \notin \mathscr{S}$ and computes signatures on $(\mathbf{m}, i\|\alpha_i)$ using the Sign algorithm. The credential consists of $\mathbf{m}$, $(\alpha_i)_{i=1}^{\ell}$ together with the resulting signatures $(\mathbf{1}, [\mathbf{S}_i; \mathbf{1}], \mathbf{1})$. The issuer adds $\mathbf{m}$ to $\mathscr{S}$.

Presentation. To create a presentation token for attributes $(\alpha_i)_{i\in R}$ and message μ, the user runs $\mathsf{P}^{H_c}_{pt}$ to generate NIZK proofs Π_i that it knows signatures on (the hidden) $\mathbf{m}$ and $i\|\alpha_i$ for $i \in R$, whereby it includes the message μ in the Fiat-Shamir hash. The presentation token pt consists of the transcripts $(\Pi)_{i\in R}$.

Verification. The verifier checks the validity of $(\Pi)_{i\in R}$ w.r.t. the message μ running $\mathsf{V}^{H_c}_{pt}$. If the tests pass, it outputs *accept*, otherwise *reject*.

The correctness of the scheme follows trivially from the correctness of the building blocks.

Unforgeability relies on the relaxed unforgeability of the rS scheme, on the relaxed binding property of the rC scheme and on the relaxed simulation soundness and zero-knowledge of the $\mathsf{r}\Sigma$ scheme. The proof strategy is to run the adversary and extract from the forged presentation token using the Generalized Forking Lemma (Lemma 2.6 in Section 2.2.3).

Theorem 4.4 (Unforgeability). *Let ϵ_{BIN} be the probability of breaking the binding property of the commitment scheme rC underlying $\mathsf{r}\Sigma$, ϵ_{SS} be the probability of*

breaking the relaxed simulation soundness of $\mathsf{r}\Sigma$*, and* ϵ_{ZK} *be the probability of breaking the breaks zero-knowledge of* $\mathsf{r}\Sigma$*.*

Assume A *is an adversary that runs in time* t_A*, makes* q_C *random oracle queries for credentials issued to corrupt users (if* A *queries for a credential on* $(id, (\alpha_i)_{i=1,\dots,\ell})$*, we count it as* ℓ *queries) and* q_H *queries for credentials issued to honest users and* q_P *presentation tokens of honest users and breaks the unforgeability of the AAT-O scheme with probability* ϵ_A*.*

There exists an algorithm B *that breaks the unforgeability of the signature in time* $t_B = 8\ell^2 t_A(q_C + q_H + q_P)/\epsilon_A \cdot \ln(8\ell/\epsilon_A) + q_C \cdot t_S + q_P \cdot t_\Sigma + t_\mathsf{E} + \mathrm{poly}(\lambda)$ *(where* t_Σ *and* t_E *are the runtime of the simulator and of the extractor of the rNIZK respectively,* t_S *is the runtime of the signing oracle, and* ℓ *is the number of attributes contained in the forged pt) with probability* $\epsilon_B = (1 - \epsilon_{ZK}) \cdot \epsilon_A/8 - (\epsilon_{BIN} + \epsilon_{SS})$ *after asking* q_C *queries to the signing oracle in the Random Oracle Model.*

Proof. To prove the unforgeability of the tokens, we define a simulator B that simulates the unforgeability experiment in Figure 4.1 and the oracles to exploit A to forge a signature. The simulator is described in details in Figure 4.3.

In the following we give a high level description of it. The simulator B has access to an oracle $\mathcal{O}_\mathrm{s}$ that, when prompted the first time, outputs the parameters of the signature scheme *spar* and the verification key *svk*. Then, whenever it receives in input $(\mathbf{m}, \alpha)$, it outputs a signature on it.

B derives the parameters of the commitment from the parameters of the signature using an algorithm $\mathsf{DerivePar}_\mathrm{c}$. This algorithm derives $\mathbf{C}$ from the verification key $\mathbf{V}$ of the signature, and computes the rest of the parameters using the m, n, and q in *spar*. To win the signature unforgeability game, B runs A simulating the oracle as it follows:

Issuance to corrupt user: on input attributes $(\alpha_i)_i$, it chooses $\mathbf{m} \in \mathcal{U} \setminus \mathcal{S}$ and queries $\mathcal{O}_\mathrm{s}$ with $(\mathbf{m}, i \| \alpha_i)$. It returns $\mathbf{m}$ and the outputs of the signing oracle and stores the issued identity $\mathbf{m}$ and the issued attributes with a credential identifier. Queries to this oracle are counted in q_C.

Issuance to honest users: on input attributes $(\alpha_i)_i$, it selects a random *cid* and stores $((\alpha_i)_i, cid)$. Queries to this oracle are counted in q_H.

Presentation by honest users: on input attributes $(\alpha_i)_i$, it outputs a simulated NIZK using the simulator Σ described in Figure 3.8 generalized to the aggregatable case (cf. Remark 9). Queries to this oracle are counted in q_P.

The simulated oracles are indistinguishable from honestly implemented ones, as the first two have exactly the same distribution and the third is indistinguishable

Simulator $\mathsf{B}^{\mathsf{A},\mathscr{O}_s}(1^\lambda)$
$q_C = q_H = q_D = 0$
$\mathscr{S} \leftarrow \emptyset$, $\mathbf{L}_c \leftarrow \emptyset$, $\mathbf{L}_h \leftarrow \emptyset$, $\mathbf{L}_p \leftarrow \emptyset$
$(spar, ipk^*) \leftarrow \mathscr{O}_s(1^\lambda)$
$cpar \leftarrow \mathsf{DerivePar}_c(spar)$
$apar \leftarrow (cpar, spar)$
$(pt^*, pt', R^*, (\alpha_i^*)_{i\in R^*}, \mu^*), \mu^*) \leftarrow \mathsf{GF}_{\mathsf{A}}^{\mathscr{O}_{IC},\mathscr{O}_{IH},\mathscr{O}_{pres}}(apar, ipk^*)$
If $\exists cid$ such that $(cid, R^*, \mu^*) \in \mathbf{L}_p$ abort.
$(\bar{\mathbf{m}}_i, \bar{\sigma}_i)_{i\in R^*} \leftarrow \mathsf{E}((pt^*, R^*, (\alpha_i^*)_{i\in R^*}), (pt', R^*, (\alpha_i^*)_{i\in R^*}))$
If $\exists\ j$ such that $\nexists\ (\alpha_i)_{i\in R}$, cid, $\mathbf{m}$ such that $\alpha_j^* \in \{\alpha_i\}_{i\in R} \wedge (cid, \mathbf{m}, (\alpha_i)_{i\in R}) \in \mathbf{L}_c$ return $((\bar{\mathbf{m}}_j, j\|\alpha_j^*), \bar{\sigma}_j)$.
ElseIf $\exists i,\ j,\ (\alpha_i)_{i\in R}$, $(\alpha_i)_{i\in R'}$, cid_i, cid_j, $\mathbf{m}_i$, $\mathbf{m}_j$ such that
$\alpha_i^* \in \{\alpha_i\}_{i\in R} \wedge \alpha_j^* \in \{\alpha_i\}_{i\in R'} \wedge \{(cid_i, \mathbf{m}_i, (\alpha_i)_{i\in R}), (cid_j, \mathbf{m}_j, (\alpha_i)_{i\in R'})\} \subseteq \mathbf{L}_c$
If $\forall k f(\mathbf{m}_i) \cap f(\bar{\mathbf{m}}_k) = \emptyset$ return $((\mathbf{m}_i, i\|\alpha_i^*), \bar{\sigma}_i^*)$.
If $\forall k f(\mathbf{m}_j) \cap f(\bar{\mathbf{m}}_k) = \emptyset$ return $((\mathbf{m}_j, j\|\alpha_j^*), \bar{\sigma}_j^*)$.
Else abort.

Oracle $\mathscr{O}_{IC}((\alpha_i)_{i=1}^\ell)$
$id \leftarrow \mathscr{U} \setminus \mathscr{S}$
$cred \leftarrow (\mathscr{O}_s(id, 1\|\alpha_1), \ \ldots, \mathscr{O}_s(id, \ell\|\alpha_\ell))$
$\mathscr{S} \leftarrow \mathscr{S} \cup \{id\}$
$cid \leftarrow H(id, cred)$
$\mathbf{L}_c \leftarrow \mathbf{L}_c \cup \{(cid, id, (\alpha_i^*)_{i\in R^*})\}$
$q_C \leftarrow q_C + 1$
Return $(id, cred)$.

Oracle $\mathscr{O}_{IH}((\alpha_i)_{i=1}^\ell)$
$cid \xleftarrow{\$} \{0,1\}^\lambda$
$\mathbf{L}_h \leftarrow \mathbf{L}_h \cup \{(cid, (\alpha_i)_{i=1}^\ell)\}$
$q_H \leftarrow q_H + 1$
Return cid .

Oracle $\mathscr{O}_{pres}(cid, R, \mu)$
Get $(\alpha_i)_{i=1}^\ell$ such that $(cid, (\alpha_i)_{i=1}^\ell) \in \mathbf{L}_h$
(abort if it does not exist).
$pt \leftarrow \Sigma(ipk, (\alpha_i)_{i\in R}, \mu, 1^\lambda)$
$\mathbf{L}_p \leftarrow \mathbf{L}_p \cup \{(cid, R, \mu)\}$
$q_P \leftarrow q_P + 1$
Return pt .

Figure 4.3. Simulator of the unforgeability experiment for AAT-O. The token pt' results from rewinding A, hence it differ from pt^* only in the values of challenges and responses.

thanks to the zero-knowledge property of the rNIZK. Let $(pt^*, R^*, (\alpha_i^*)_{i\in R^*}, M^*)$ be the forgery output by A. The simulator rewinds A using the Generalized Forking Lemma (this corresponds to applying the algorithm GF to A, cf. Lemma 2.6) and extracts an identities $\bar{\mathbf{m}}_i$ and signatures $\bar{\sigma}_i$ on $(\bar{\mathbf{m}}_i, i\|\alpha_i^*) \in \mathscr{M}$ using the extractor E of the rNIZK. By the relaxed binding property of the commitment scheme, the identities $\bar{\mathbf{m}}_i$ are such that there exists $\mathbf{m} \in \mathscr{U}$ such that $\{\bar{\mathbf{m}}_i\}_i \subseteq f(\mathbf{m})$. Moreover, for it to be a valid forgery, there should exist at least one α_j^* that either was not part of any issued credential, or a pair of attributes α_i^*, α_j^* that were issued within different credentials. In the first case, $(\bar{\mathbf{m}}_j, j\|\alpha_j^*)$ with signature $\bar{\sigma}_j$ is a valid forgery. In the second case, this means that the signing algorithm signed messages $(\mathbf{m}_i, i\|\alpha_i^*)$, $(\mathbf{m}_j, j\|\alpha_j^*)$ in $\mathscr{M}$ for some distinct $\mathbf{m}_i$, $\mathbf{m}_j$

such that $f(\mathbf{m}_i) \cap f(\mathbf{m}_j) = \emptyset$ (by construction of $\mathscr{U}$). Then either $(\bar{\mathbf{m}}_i, i \| \alpha_i^*), \bar{\sigma}_i$ or $(\bar{\mathbf{m}}_j, j \| \alpha_j^*), \bar{\sigma}_j$ is a valid forgery.

We now compute the success probability ϵ_B and the runtime t_B of B. Let ϵ_{BIN} be the probability of breaking the binding property of rC, ϵ_{SS} be the probability of breaking the relaxed simulation soundness, and ϵ_{ZK} be the probability that A breaks zero-knowledge. Then, from the Generalized Forking Lemma it follows that the success probability of B is $\epsilon_B = (1-\epsilon_{ZK}) \cdot \epsilon_A/8 - (\epsilon_{BIN} + \epsilon_{SS})$. By the Generalized Forking Lemma, the running time of B is $t_B = 8\ell^2 t_A(q_C+q_H+q_P)/\epsilon_A \cdot \ln(8\ell/\epsilon_A) + q_C \cdot t_S + q_P \cdot t_\Sigma + t_\mathsf{E} + \text{poly}(\lambda)$, where t_Σ and t_E are the runtime of the simulator and of the extractor of the rNIZK respectively, t_S is the runtime of the signing oracle, and $\ell = |R^*|$. □

Anonymity is guaranteed by the zero-knowledge property of $\mathsf{r}\Sigma$ (hence, implicitly, on the hiding property of the rC scheme).

Theorem 4.5 (Anonymity)**.** *If an adversary* A *running in time* t *breaks the anonymity of the AAT-O with probability at most* ϵ_A*, then there exists a PPT adversary that breaks the zero-knowledge property of* $\mathsf{r}\Sigma$ *with advantage at most* $\epsilon_B \geq \epsilon_A/2$ *in the Random Oracle Model.*

Proof. Let $\mathsf{Exp}_\mathsf{A}^{\mathsf{anon-b}}(1^\lambda)$ be the experiment in Figure 4.2. The success probability of A is:

$$\begin{aligned}
\epsilon_A &= \left| \Pr\left[b = b' \ : \ b \xleftarrow{\$} \{0,1\}, b' \leftarrow \mathsf{Exp}_\mathsf{A}^{\mathsf{anon-b}}(1^\lambda) \right] - \frac{1}{2} \right| \\
&= \frac{1}{2} \left| \Pr\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}_\mathsf{A}^{\mathsf{anon-0}}(1^\lambda) \right] + \Pr\left[b' = 1 \ : \ b' \leftarrow \mathsf{Exp}_\mathsf{A}^{\mathsf{anon-1}}(1^\lambda) \right] - 1 \right| \\
&= \frac{1}{2} \left| \Pr\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}_\mathsf{A}^{\mathsf{anon-0}}(1^\lambda) \right] + \Pr\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}_\mathsf{A}^{\mathsf{anon-1}}(1^\lambda) \right] \right| .
\end{aligned}$$

We prove that such probability is negligible if the rNIZK is zero-knowledge through a standard sequence of game hops. Let $\mathbf{Game_i}$ the probability that A outputs 0 at the end of the i-th game.

Game 0. Game 0 executes $\mathsf{Exp}_\mathsf{A}^{\mathsf{anon-0}}(1^\lambda)$. Hence,

$$\Pr[\mathbf{Game_0}] = \Pr\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}_\mathsf{A}^{\mathsf{anon-0}}(1^\lambda) \right] .$$

Game 1. In the first game, everything is the same as in Game 0 but the challenge oracle $\mathscr{O}_C^0$, which now simulates the proofs (but still outputs pt_0^*). If the adversary can distinguish Game 0 from Game 1, then it is possible to build a distinguisher

B_1 that breaks the zero-knowledge proof of $\mathsf{r}\Sigma$ with advantage

$$\epsilon_1 = \frac{1}{2}\left|\mathsf{Pr}\left[\mathbf{Game_0}\right] - \mathsf{Pr}\left[\mathbf{Game_1}\right]\right| .$$

The construction of B_1 is analogous to the construction of B_1 in Game 1 in the proof of Theorem 3.43, hence we omit it.

Game 2. In the second game, everything is the same as in Game 0 but the challenge oracle $\mathcal{O}_C^0$, which now outputs pt_1^*. As both pt_0^* and pt_1^* are generated by the simulator, which does not take as input $cred_b^*$, the two games are perfectly indistinguishable:

$$\mathsf{Pr}\left[\mathbf{Game_2}\right] = \mathsf{Pr}\left[\mathbf{Game_1}\right] .$$

Game 3. In the last game the challenge oracle is switched back to generate the proofs honestly. This is exactly $\mathsf{Exp}_{\mathsf{A}}^{\mathsf{anon}-1}(1^\lambda)$, hence it holds

$$\mathsf{Pr}[\mathbf{Game_3}] = \mathsf{Pr}\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{\mathsf{anon}-1}(1^\lambda)\right] .$$

Moreover, if A is able to distinguish Game 3 from Game 2 it is possible to construct a distinguisher B_3 that exploits A to break the zero knowledge property of $\mathsf{r}\Sigma$ analogously to what we did in Game 1 with advantage

$$\epsilon_3 = \frac{1}{2}\left|\mathsf{Pr}\left[\mathbf{Game_2}\right] - \mathsf{Pr}\left[\mathbf{Game_3}\right]\right| .$$

Therefore, the success probability of A is

$$\begin{aligned}
\epsilon_A &= \frac{1}{2}\left|\mathsf{Pr}\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{\mathsf{anon}-0}(1^\lambda)\right] + \mathsf{Pr}\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{\mathsf{anon}-1}(1^\lambda)\right]\right| \\
&= \frac{1}{2}\left|\mathsf{Pr}\left[\mathbf{Game_0}\right] - \mathsf{Pr}\left[\mathbf{Game_3}\right]\right| \\
&= \frac{1}{2}\left|\sum_{i=0}^{2} \mathsf{Pr}\left[\mathbf{Game_i}\right] - \mathsf{Pr}\left[\mathbf{Game_{i+1}}\right]\right| \\
&\leq \frac{1}{2}(2\epsilon_1 + 0 + 2\epsilon_3) \\
&= 2\epsilon_{ZK} .
\end{aligned}$$

□

4.1.3 Parameters and Storage Requirements

We present six different sets of parameters, depending on the level of security that is required. To compute them, we follow the general methodology from [Alkim

et al., 2016]. This will give us a wide choice of parameters, from an optimistic choice that only guarantees classical security to a very pessimistic choice for quantum security. RSIS and RLWE are analyzed in their corresponding forms of SIS and LWE[1]. The best algorithm to find short vectors in a lattice is the BKZ algorithm, whose latest version was published by Chen and Nguyen [2011]. This algorithm reduces the lattice basis into blocks of size b and then calls an SVP (Shortest Vector Problem) oracle on such blocks. When computing the runtime of BKZ, we will ignore the number of calls to the oracle, as it is known to be polynomial Hanrot et al. [2011] and it is rather complex to compute. This makes all parameters choices significantly more conservative than needed in reality. Now, considering the SVP oracle, Alkim et al. estimated the heuristic complexity as it follows: for classical algorithms (e.g., lattice sieve algorithms) it is around $\approx 2^{0.292b}$, for quantum algorithms (e.g., sieving plus Grover's algorithm) around $\approx 2^{0.265b}$ and overall they would not go below a heuristic complexity of $\approx 2^{0.2075b}$ excluding major theory breakthroughs. To estimate the optimal block size b, we use the Hermite root factor δ: we first compute δ for the SVP instance, then we obtain b from the (optimistic estimate) Hermite root factor of the solution output by BKZ $\delta = ((\pi b)^{1/b} \cdot b/2\pi e)^{1/2(b-1)}$. We do not take into account other types of attack (e.g., [Arora and Ge, 2011; Kirchner and Fouque, 2015]), as for our choice of parameters they would be not effective.

For each of the 3 possible scenarios, we give two sets of parameters, distinguishing whether security is based on complexity leveraging. Recall that such a technique is used in the reductions in Theorem 3.32 and in Theorem 3.19. Basing the security of the scheme on these reductions means that parameters have to compensate for the loss in tightness. Not relying on complexity leveraging means to assume that the hardness of Assumption 4 (resp. Assumption 2) implies the hardness of Assumption 3 (resp. Assumption 1) *without any tightness loss*.

Parameters that guarantee 128 bits of security are shown in Table 4.1. The third column contains the maximum value of the Hermite root factor that guarantees 128 bits of security in the different cases. As message space, we have chosen $K_m = 6$, hence $\mathcal{U} \subseteq R_3^{(64)}$, and the same for the challenge space, $\mathcal{C} \subseteq R_3^{(64)}$ (thus the proofs have to be repeated 6 times). In case complexity leveraging is used, the values of δ are computed taking into account the necessary compensation for the loss in tightness in the proofs. As observed in Section 4.1.2, the scheme supports an estimated number of users around $3^{(2^{K_m-1}-1)}/(n/2)$. In practice, for $K_m = 6$ the number of supported users is 2^{40} for $n = 2^{10}$ and 2^{39} for $n = 2^{11}$. The

[1]This analysis was accurate at the time the work was published, and might not be up to date anymore.

			Parameters				Sizes		
Compl. Lev.	Security	δ	n	q	m	σ_t	ipk(KB)	usk(KB)	$token$(MB)
NO	classical	1.003735	2^{10}	$\sim 2^{88}$	13	4	304.128	98.24	1.5
NO	quantum	1.003488	2^{10}	$\sim 2^{88}$	14	4	323.656	103.36	1.58
NO	worst-case	1.002926	2^{10}	$\sim 2^{88}$	17	4	394.24	113.984	1.86
YES	classical	1.0005036	2^{11}	$\sim 2^{92}$	52	4	2472.96	462.272	11.15
YES	quantum	1.0004646	2^{11}	$\sim 2^{92}$	57	4	2708.48	503.232	12.19
YES	worst-case	1.0003788	2^{11}	$\sim 2^{92}$	70	4	3320.832	609.728	14.7

Table 4.1. Table of parameters for the AAT scheme without opening. All the values are rounded up.

set $\bar{\mathcal{U}}$ results to have cardinality 2^{1805} (resp. 2^{1829}) for $n = 2^{11}$ (resp. $n = 2^{12}$). Hence to compensate for complexity leveraging we consider SVP instances that offer $\sim 2^{1850}$ bits of security (as in the proof of Theorem 3.27 both $\bar{\mathbf{m}}$ and $\bar{\mathbf{c}}$ have to be guessed).

In the following, we give an example of how we computed the storage requirements in Table 4.1. Consider the parameters in the first line of Table 4.1 (classical security, no complexity leveraging). With those values, a polynomial $\mathbf{a} \in R_q$ can be stored in at most $n \log_2 q/8 = 11.264$ KB. The issuer public key contains by $\mathbf{a} \in R_q$ and $\mathbf{B}, \mathbf{C} \in R_q^{1\times m}$. Hence it is composed by 27 polynomials in R_q, and it requires 304.128 KB of storage. The issuer secret key is composed by the trapdoor $\mathbf{X} \in R_q^{2\times m}$ sampled from a Gaussian with standard deviation $\sigma_t = 4$, thus its components have infinity norm less than $8 \cdot 4 = 32$ with high probability. Therefore, storing it requires at most $2mn \cdot \log_2 32/8 = 16.64$ KB. The user secret key is composed by the identity $\mathbf{m}$ and the signature $\mathbf{S}$, thus it can be stored in 98.24 KB. A signature is composed by a commitment, i.e. a vector in $R_q^{1\times m}$, and by the transcript of the NIZK proof (a challenge in $\mathcal{C}$ and two responses, i.e. vectors in R_q of length $m+2$ and $2m+3$ respectively). The length of the commitment is $mn \log_2 q$ bits that is 146.432 KB. The length of each proof is at most 0.225 MB plus the one-time signature. Hence the length of the presentation token is less than 1.5 MB plus the one-time signature used in the relaxed Σ-protocol.

4.1.4 Simple Optimization: Multiple Rejection Sampling Steps

We sketch an optimization of the scheme where rejection sampling is done separately for the user identity and the rest of the vector $\mathbf{S}_c$. This optimization was included in the original work (cf. [Boschini et al., 2017]), where it had quite the impact on the parameters of the scheme. However, in this work we have improved the bounds in Section 3.7 using the infinity norm, and this optimization

does not improve the parameters of scheme that much anymore. Therefore, we have decided to mention it (for completeness), but not to work out the details, as the resulting protocol would not be of interest.

When using complexity leverage, parameters get considerably larger because the set $\bar{\mathscr{U}}$ can contain up to 2^{1538} elements. Its dimension is determined by the norm of the vector $\mathbf{Z}_1$ in Section 3.7. This length in turn is a function of the length of $\mathbf{m}$, $\mathbf{E}$ and $\mathbf{b}$, where the norms of $\mathbf{b}$ and $\mathbf{E}$ are considerably larger than the norm of $\mathbf{m}$, as $\mathbf{m} \in R_3^{(2^{K_m})}$. Hence, it can happen that the norm of an extracted message $\bar{\mathbf{m}}$ is heavily dependent on the norm of $\mathbf{b}$ and $\mathbf{E}$. To avoid this, we can modify the relaxed protocol in Section 3.7 to do rejection sampling separately on $\mathbf{m}$. Indeed, the prover can sample the error polynomial $\mathbf{y}_m$ from a Gaussian with standard variation σ_m proportional to the norm of $\mathbf{m}$, and another masking vector $\mathbf{Y}_1$ from a Gaussian with standard deviation σ proportional to the norm of $[\mathbf{b};\mathbf{E}]$. Then, one would do everything as in the original algorithm except for the rejection sampling part. This part would have to be done separately for $\mathbf{z}_m = \mathbf{y}_m + \mathbf{cm} \bmod q$ and $\mathbf{Z}_1 = \mathbf{Y}_1 + \mathbf{c}[\mathbf{b};\mathbf{E}] \bmod q$. The advantage is that now we have a tighter bound on the norm of $\mathbf{z}_m$. The disadvantage is that we have to do three rejection sampling steps (one on $[\mathbf{S}_1;\mathbf{S}_2;\mathbf{S}_3;-\mathbf{E}\mathbf{S}_3]$, one on $[\mathbf{b};\mathbf{E}]$ and one on $\mathbf{m}$) instead of one, thus reducing the acceptance probability. To prevent that, we have to increase the standard deviations.

4.2 Anonymous Attribute Tokens with Opening

There are scenarios where it can be necessary to revoke the anonymity of a misbehaving user to establish which user has created a particular token. This cannot be done with an AAT-O, but requires an *opener*, i.e., an entity that can extract *id* from the presentation token thanks to a secret key. This type of AATs with opening are denoted by AAT+O. An AAT+O scheme immediately gives rise to a group signature scheme with chosen-ciphertext anonymity (i.e., where the adversary has access to an opening oracle) by handing each user a credential of a fixed attribute as a signing key. To sign a message μ, the user creates a presentation token for μ. We will deal with this in Section 5.2.

4.2.1 Definition of AAT+O Schemes

We adapt the syntax and security notions of AAT+O schemes [Camenisch et al., 2012] to a setting where the issuer and opener are separate entities, rather than having a central group manager that performs both roles. The syntax of an

AAT+O scheme largely follows that of an AAT-O scheme. The system parameters generation, issuer key generation, and credential issuance are as defined for AAT-O schemes in Section 4.1.1. In addition to those algorithms, there are two algorithms that are run by the opener and a different presentation and verification algorithms (as it takes as input the opener public key as well):

Opener Key Generation. The opener key generation algorithm $\mathsf{OKGen}(apar)$ generates the opener's key pair (opk, osk).

Presentation. A user creates a presentation token pt revealing a subset of attributes $(\alpha_i)_{i\in R}$, $R \subseteq \{1,\ldots,\ell\}$, from a credential while authenticating a message μ by running $\mathsf{Present}(ipk, opk, cred, R, \mu)$.

Verification. The verifier checks the validity of a presentation token by running $\mathsf{Verify}(ipk, opk, R, (\alpha_i)_{i\in R}, \mu, pt)$ which returns *accept* or *reject*.

Opening. The opening algorithm $id \leftarrow \mathsf{Open}(ipk, osk, R, (\alpha_i)_{i\in R}, \mu, pt)$ recovers the user's identity.

Correctness requires that verification and opening of a honestly generated token are coherent.

Definition 4.6 (Correctness)**.** An AAT+O scheme $(\mathsf{SPGen}, \mathsf{IKGen}, \mathsf{OKGen}, \mathsf{Issue}, \mathsf{Present}, \mathsf{Verify}, \mathsf{Open})$ satisfies correctness if for all user identities id, all $\ell \in \mathbb{N}$, all $\alpha_1,\ldots,\alpha_\ell \in \{0,1\}^*$, and all sets $R \subseteq \{1,\ldots,\ell\}$ the following holds

$$\mathsf{Pr}\left[\begin{array}{c} 1 \leftarrow \mathsf{Verify}(ipk, opk, R, (\alpha_i)_{i\in R}, \mu, pt) \\ \wedge\ id' = id \end{array} : \begin{array}{c} apar \leftarrow \mathsf{SPGen}(1^\lambda),\ (ipk, isk) \leftarrow \mathsf{IKGen}(apar), \\ (id, cred) \leftarrow \mathsf{Issue}(isk, id, (\alpha_i)_{i=1}^{\ell}), \\ pt \leftarrow \mathsf{Present}(ipk, cred, R, \mu), \\ id' \leftarrow \mathsf{Open}(ipk, osk, R, (\alpha_i)_{i\in R}, \mu, pt) \end{array}\right] \leq \nu(\lambda)\,.$$

In terms of security, an AAT+O scheme must satisfy traceability (which corresponds to the unforgeability notion of the AAT-O) and anonymity.

Definition 4.7 (Traceability)**.** An AAT+O scheme $(\mathsf{SPGen}, \mathsf{IKGen}, \mathsf{OKGen}, \mathsf{Issue}, \mathsf{Present}, \mathsf{Verify}, \mathsf{Open})$ satisfies traceability if for all PPT adversaries A the advantage of A in winning the unforgeability experiment in Figure 4.4 is negligible:

$$\mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{\mathsf{trac}}_{\mathsf{A}}(1^\lambda)\right] \leq \nu(\lambda)\,.$$

We describe a strong notion of full anonymity, often referred to as CCA2 anonymity, where the adversary is given access to an opening oracle. This oracle keeps track of the tokens it has opened through the list $\mathbf{L}_o$.

Experiment $\mathsf{Exp}^{\mathsf{trac}}_{\mathsf{A}}(1^\lambda)$
$apar \leftarrow \mathsf{SPGen}(1^\lambda)$
$(ipk^*, isk^*) \leftarrow \mathsf{IKGen}(apar)$
$(opk^*, osk^*) \xleftarrow{\$} \mathsf{OKGen}(apar)$
$(ssk, svk) \leftarrow \mathsf{SKeyGen}(spar)$
$(pt^*, R^*, (\alpha_i^*)_{i\in R^*}, \mu^*) \leftarrow \mathsf{A}^{\mathcal{O}_{IC}, \mathcal{O}_{IH}, \mathcal{O}_{pres}, \mathcal{O}_O}(apar, ipk^*, opk^*)$
$id^* \leftarrow \mathsf{Open}(ipk^*, osk^*, R, (\alpha_i)_{i\in R}, \mu, pt)$
If $1 \leftarrow \mathsf{Verify}(ipk^*, opk^*, R^*, (\alpha_i^*)_{i\in R^*}, \mu^*, pt^*)$
and $\nexists$ *cid* such that $(cid, (\alpha_i^*)_{i\in R^*}) \in \mathbf{L}_c$
and $(id^*, (\alpha_i^*)_{i\in R^*}, \mu^*) \notin \mathbf{L}_p$
then return 1 else return 0.

Oracle $\mathcal{O}_{pres}(opk, cid, R, \mu)$
Get *cred* such that $(cid, id, cred) \in \mathbf{L}_h$
(abort if it does not exist).
$pt \leftarrow \mathsf{Present}(ipk^*, opk, cred, R, \mu)$
$\mathbf{L}_p \leftarrow \mathbf{L}_p \cup \{(id, R, \mu)\}$
Return pt .

Oracle $\mathcal{O}_O(pt, ipk^*, R, (\alpha_i)_{i\in R}, \mu)$
$id \leftarrow \mathsf{Open}(ipk^*, osk^*, R, (\alpha_i)_{i\in R}, \mu, pt)$
Return pt .

Figure 4.4. Traceability experiment for AAT+O. The oracles $\mathcal{O}_{IC}$ and $\mathcal{O}_{IH}$ can be found in Figure 4.1.

Experiment $\mathsf{Exp}^{\mathsf{anon}-b}_{\mathsf{A}}(1^\lambda)$
$apar \leftarrow \mathsf{SPGen}(1^\lambda)$
$(opk^*, osk^*) \xleftarrow{\$} \mathsf{OKGen}(apar)$
$(ipk^*, isk^*) \leftarrow \mathsf{IKGen}(apar)$
$count \leftarrow 0$
$b' \leftarrow \mathsf{A}^{\mathcal{O}^b_C, \mathcal{O}_O}(apar, opk^*, ipk^*, isk^*)$
Return b'.

Oracle $\mathcal{O}^b_C(cred^*_0, cred^*_1, R^*, (\alpha_i^*)_{i\in R^*}, \mu^*)$
$count \leftarrow count + 1$
If $count \geq 2$ abort.
For $i = 0, 1$:
$pt^*_i \xleftarrow{\$} \mathsf{Present}(ipk^*, opk^*, cred^*_i, R^*, \mu^*)$
If $\mathsf{Verify}(ipk^*, opk^*, R^*, (\alpha_i^*)_{i\in R^*}, \mu^*, pt^*_0) = 1$
and $\mathsf{Verify}(ipk^*, opk^*, R^*, (\alpha_i^*)_{i\in R^*}, \mu^*, pt^*_1) = 1$
then return pt^*_b, else abort.

Oracle $\mathcal{O}_O(R, (\alpha_i)_{i\in R}, \mu, pt)$
If $count = 1 \ \wedge \ pt = pt^*$ abort.
$id \leftarrow \mathsf{Open}(osk^*, R, (\alpha_i)_{i\in R}, \mu, pt)$
Return id.

Figure 4.5. Anonymity experiment for AAT+O.

Definition 4.8 (Anonymity)**.** An AAT+O scheme $(\mathsf{SPGen}, \mathsf{IKGen}, \mathsf{OKGen}, \mathsf{Issue}, \mathsf{Present}, \mathsf{Verify}, \mathsf{Open})$ satisfies anonymity if for all PPT adversaries A the advantage of A in winning the anonymity experiment in Figure 4.5 is negligible:

$$\left| \Pr\left[b' = b \ \wedge \ pt^* \notin \mathbf{L}_o \ : \ b \xleftarrow{\$} \{0,1\}, \ b' \leftarrow \mathsf{Exp}^{\mathsf{anon}-b}_{\mathsf{A}}(1^\lambda) \right] - \frac{1}{2} \right| \leq \nu(\lambda) \ .$$

4.2.2 AAT+O from Lattices

We present a construction of an AAT+O scheme obtained by combining our relaxed signature and commitment scheme with the relaxed verifiable encryption

scheme by Lyubashevsky and Neven [2017]. We see our AAT+O scheme mainly as a proof of concept that demonstrates that our framework of relaxed cryptographic primitives, glued together with relaxed Σ-protocols, can be composed generically to build efficient privacy-enhancing protocols. At the same time, we expect that a direct construction that builds on the same principles, but that is optimized for the specific use case, can easily outperform our generic construction. We therefore refrain from suggesting concrete parameter sizes and giving efficiency estimates, but rather leave such numbers to future work.

As described in Section 3.7, the opening verification equation of the commitment scheme can be rewritten as

$$\underbrace{\begin{bmatrix} -\mathbf{G}^T & \mathbf{F}^T & -\mathbb{I}_m \end{bmatrix}}_{=\mathbf{A}_c} \underbrace{\begin{pmatrix} \mathbf{m} \\ \mathbf{b} \\ \mathbf{E}^T \end{pmatrix}}_{=\mathbf{s}_c} = \mathbf{C}^T \bmod q \ . \tag{4.1}$$

One can observe that this is a linear relation that can be used for the verifiable encryption scheme $(\mathsf{EKeyGen}_{LN}, \mathsf{Enc}_{LN}, \mathsf{EVerify}_{LN}, \mathsf{Dec}_{LN})$ of [Lyubashevsky and Neven, 2017] described in Section 3.8.3. We can therefore add opening to our AAT-O scheme from Section 4.1.2 by including in the presentation token a verifiable encryption of the user identity $\mathbf{m}$ that is committed to in $\mathbf{F}$, so that $\mathbf{m}$ can be recovered by decrypting the ciphertext. Remark that in that case we need to encrypt modulo a q' larger than q^2.

We now present our AAT+O scheme $(\mathsf{SPGen}, \mathsf{IKGen}, \mathsf{OKGen}, \mathsf{Issue}, \mathsf{Present}, \mathsf{Verify}, \mathsf{Open})$ that we obtained combining the relaxed signature rS with the relaxed Σ-protocol $\mathsf{r}\Sigma$ and the Lyubashevsky-Neven relaxed verifiable encryption. To allow for composability we redefine the prover algorithm from Remark 9 as an algorithm $\mathsf{P}_{pt}^{H_c}$ that outputs both the transcript of the proof and the randomness $(\mathbf{E}, \mathbf{b})$ used to generate the commitment $\mathbf{F}$ to the user identity. The verifier remains unchanged. We also introduce an algorithm $\mathsf{DerivePar}_{\mathrm{ve}}$ that on input the parameters of the signature chooses the two primes needed by the encryption to be q and $q' \gg q$ that satisfy the bound in Equation (3.21).

Issuer key generation, and issuance are performed exactly as in the AAT-O scheme from Section 4.1.2. The other algorithms are described as follows.

System Parameter Generation. The algorithm SPGen first generates the commitment parameters $cpar \leftarrow \mathsf{ParGen}_{\mathrm{c}}(1^\lambda)$. Then it derives the parameters

[2]The notation is misleading here, as in Section 3.8.3 we described the protocol as follows: the relation was defined modulo p, while encryption would be done modulo q, where $q \gg p$. In the case of our AAT+O scheme, the relation holds modulo q and the encryption is done modulo a q' greater than q.

of the relaxed signature $spar \leftarrow \mathsf{DerivePar}_s(cpar)$ and the verifiable encryption parameters $epar \leftarrow \mathsf{DerivePar}_{ve}(cpar)$ and outputs $apar = (cpar, spar, epar)$.

Opening Key Generation. The opener generates the keys for the verifiable encryption scheme running $(epk, esk) \leftarrow \mathsf{EKeyGen}_{LN}(1^\lambda)$ and sets $opk = epk$ and $osk = esk$.

Presentation. To create a presentation token for a opener key opk, attributes $(\alpha_i)_{i\in R}$ and a message μ, the user first proceeds as in the AAT-O scheme, i.e., it runs the non-interactive (aggregated) prover $\mathsf{P}_{pt}^{H_c}((\mathbf{V}, (\alpha_i)_{i\in R}), (\mathbf{m}, (\mathbf{1}, (\mathbf{S}_i)_{i\in R}, \mathbf{1}))), \mu)$ from Section 3.7, whereby it includes the message μ in the Fiat-Shamir hash. Remark that all the proofs have the same $\mathbf{F}$, as the protocol aggregates the signature as shown in Remark 9. Then, it generates a one-time signature key pair $(otsvk, otssk) \leftarrow \mathsf{OTSGen}(1^\lambda)$, constructs $\mathbf{S}_c$ (cf. Equation (3.15)), and generates a verifiable ciphertext $c \xleftarrow{\$} \mathsf{Enc}_{LN}(opk, \mathbf{A}_c, \mathbf{S}_c, otsvk)$ as per Equation (4.1), where $otsvk$ is only used as input of the Fiat-Shamir hash of the verifiable encryption. Finally, it computes a one-time signature $ots \leftarrow \mathsf{OTSSign}(otssk, ((\Pi_i)_{i\in R}, c))$ and outputs the token $pt = ((\Pi_i)_{i\in R}, c, otsvk, ots)$.

Verification. The verifier checks the validity of $(\Pi_i)_{i\in R}$ w.r.t. $\mathbf{F}$ and the message μ running $\mathsf{V}_{pt}^{H_c}(\mathbf{V}, \alpha_i, \Pi_i, \mu)$, checks that $\mathsf{EVerify}_{LN}(opk, \mathbf{A}_c, otsvk, c) = 1$ with $otsvk$ in the Fiat-Shamir hash, and that $\mathsf{OTSVf}(otsvk, ((\Pi_i)_{i\in R}, c)) = 1$. If all tests pass, it outputs *accept*, otherwise *reject*.

Opening. To open a presentation token pt, the opener first runs the verification algorithm Verify, returning $\perp$ if it rejects. The opener decrypts $(\bar{\mathbf{S}}_c, \bar{\mathbf{c}}) \leftarrow \mathsf{Dec}_{LN}(osk, c)$ and recovers $\bar{\mathbf{m}}$ as the first coordinate from $\bar{\mathbf{S}}_c$. It then recovers an irreducible factor $\mathbf{m}$ of degree $n/2$ of $\bar{\mathbf{m}}$ (through dividing by $\bar{\mathbf{c}}$) and returns $\mathbf{m}$, or returns $\perp$ if such a factor does not exist.

Again, correctness is trivial, as it follows from the correctness of the building blocks. The runtime of the opening algorithm is exactly the runtime of the decryption, and can be bound using Theorem 3.40. As the rNIZK proof guarantees that $\deg(\bar{\mathbf{m}}) = 3/4n$, a honestly generated $\bar{\mathbf{m}}$ can contain only one irreducible factor of degree exactly $n/2$, that is the original identity $\mathbf{m}$ used to produce the token.

The traceability of the AAT+O scheme can be proved essentially in the same way as the unforgeability of the AAT-O scheme in Section 4.1.2.

Theorem 4.9 (Traceability)**.** *Let ϵ_{BIN} be the probability of breaking the binding property of the commitment scheme* rC *underlying* $\mathsf{r}\Sigma$*, ϵ_{SS} be the probability of breaking the relaxed simulation soundness of* $\mathsf{r}\Sigma$*, and ϵ_S be the probability of breaking the soundness of the Lyubashevsky-Neven relaxed verifiable encryption scheme.*

Assume A *is an adversary that runs in time t_A, makes q_C queries for credentials issued to corrupt users (if* A *queries for a credential on $(id, (\alpha_i)_{i=1,\dots,\ell})$, we count it as ℓ queries) and q_H queries for credentials issued to honest users, asks for q_P presentation tokens of honest users, and breaks the traceability of the AAT+O scheme with probability ϵ_A.*

There exists an algorithm B *that breaks the unforgeability of the signature in time $t_B = 8\ell^2 t_A(q_C + q_H + q_P)/\epsilon_A \cdot \ln(8\ell/\epsilon_A) + (q_C + q_H) \cdot t_S + t_{\mathsf{E}} + \mathrm{poly}(\lambda)$ (where t_{E} is the runtime of the extractor of the rNIZK, t_S is the runtime of the signing oracle, and ℓ is the number of attributes contained in the forged pt) with probability $\epsilon_B = \epsilon_A/8 - (\epsilon_{BIN} + \epsilon_{SS} + \epsilon_S)$ after asking $q_C + q_H$ queries to the signing oracle in the Random Oracle Model.*

Proof sketch. We give here a sketch of the proof, as it is very similar to the proof of unforgeability of our AAT-O (Theorem 4.4).

To prove the theorem statement we show that a successful adversary A against traceability implies a successful adversary B against the unforgeability of the signature which can simulate the traceability experiment without being detected thanks to the zero-knowledge property of $\mathsf{r}\Sigma$ and to the soundness and chosen-ciphertext simulatability of the relaxed verified encryption scheme. The algorithm B is described in Figure 4.6. It has access to a signing oracle $\mathcal{O}_s$ that, when prompted the first time, outputs the parameters of the signature scheme *spar* and the verification key *svk*. Then, on input a message $(\mathbf{m}, \alpha)$, the oracle outputs a valid signature on it w.r.t. *svk*. The parameters of the commitment are derived from *spar* through an algorithm $\mathsf{DerivePar}_c$.

The simulator B simulates the oracles as follows

Issuance to corrupt user: on input attributes $(\alpha_i)_i$, it chooses $\mathbf{m} \in \mathcal{U} \setminus \mathcal{S}$ and queries $\mathcal{O}_s$ with $(\mathbf{m}, i \| \alpha_i)$. It returns $\mathbf{m}$ and the outputs of the signing oracle and stores the issued identity $\mathbf{m}$ and the issued attributes with a credential identifier. Queries to this oracle are counted in q_C. This algorithm is the same as in the proof of Theorem 4.4.

Issuance to honest users: on input attributes $(\alpha_i)_i$, it chooses $\mathbf{m} \in \mathcal{U} \setminus \mathcal{S}$ and queries $\mathcal{O}_s$ with $(\mathbf{m}, i \| \alpha_i)$. Then it computes the credential identifier *cid* and returns it as output. Queries to this oracle are counted in q_H. The credential is stored in the list $\mathbf{L}_h$ to be used to produce presentation tokens for honest users.

Presentation by honest users: on input attributes $(\alpha_i)_i$, it recover the entry $(cid, cred, (\alpha_i)_{i=1}^{\ell})$ from $\mathbf{L}_h$ (it aborts if it does not exist). Then it generates the presentation token honestly. Queries to this oracle are counted in q_P.

To break the traceability of the AAT+O, A has to output a presentation token that has been produced without knowing the *usk* corresponding to the *id* hidden in it. Hence, B uses the rewinding algorithm GF from the Generalized Forking Lemma (Lemma 2.6) and extracts valid relaxed signatures of $(\bar{\mathbf{m}}, i \| \alpha_i)$ for all revealed attributes, as well as a relaxed opening $(\bar{\mathbf{S}}_{c,i}, \bar{\mathbf{c}}_i)$ for the commitment $\mathbf{F}$ (in Figure 4.6 we have denoted by E the extractor obtained combining the extractor of the rNIZK and the extractor of the verifiable encryption scheme). By the soundness of the relaxed verifiable encryption scheme, the decryption of c also recovers a possibly different relaxed opening $(\bar{\mathbf{S}'}_c, \bar{\mathbf{c}'})$ for $\mathbf{F}$. Let $\bar{\mathbf{m}}_i$ be the message contained in $\bar{\mathbf{S}}_{c,i}$ and let $\bar{\mathbf{m}}'$ be the message contained in $\bar{\mathbf{S}'}_c$. By the relaxed binding property of rC, for all $i \in R$ the message $\bar{\mathbf{m}}_i$ shares an $n/2$-degree irreducible factor with $\bar{\mathbf{m}}'$. This is the user identity under which the credential was created. Hence, A can be used to obtain a forgery against the signature scheme using a similar reduction as in the unforgeability proof.

The runtime of B is $t_B = 8\ell^2 t_A (q_C + q_H + q_P)/\epsilon_A \cdot \ln(8\ell/\epsilon_A) + (q_C + q_H) \cdot t_S + t_{\mathsf{E}} + \mathrm{poly}(\lambda)$ (where t_{E} is the runtime of the extractor of the rNIZK, t_S is the runtime of the signing oracle, ℓ is the number of attributes contained in the forged *pt*).

Let ϵ_{BIN} be the probability of breaking the binding property of rC, ϵ_{SS} be the probability of breaking the relaxed simulation soundness, and ϵ_S be the probability that A breaks the soundness of the relaxed verifiable encryption scheme. Then B succeeds with probability $\epsilon_B = \epsilon_A/8 - (\epsilon_{BIN} + \epsilon_{SS} + \epsilon_S)$ after asking $q_C + q_H$ queries to the signing oracle in the Random Oracle Model. □

Theorem 4.10 (Anonymity)**.** *If the relaxed Σ-protocol $\mathsf{r}\Sigma$ is zero-knowledge, the one-time signature scheme is strongly unforgeable, and the Lyubashevsky-Neven verifiable encryption scheme is chosen-ciphertext simulatable, then the AAT+O scheme is anonymous in the Random Oracle Model.*

Proof sketch. We give an outline of the proof, as it is very similar to the proof of Theorem 4.5.

Let $\mathsf{Exp}_{\mathsf{A}}^{\mathsf{anon}-\mathsf{b}}(1^{\lambda})$ be the experiment in Figure 4.5. The success probability of A is:

$$\begin{aligned}\epsilon_A &= \left|\mathsf{Pr}\left[b = b' \ : \ b \xleftarrow{\$} \{0,1\}, b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{\mathsf{anon-b}}(1^\lambda)\right] - \frac{1}{2}\right| \\ &= \frac{1}{2}\left|\mathsf{Pr}\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{\mathsf{anon-0}}(1^\lambda)\right] + \mathsf{Pr}\left[b' = 1 \ : \ b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{\mathsf{anon-1}}(1^\lambda)\right] - 1\right| \\ &= \frac{1}{2}\left|\mathsf{Pr}\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{\mathsf{anon-0}}(1^\lambda)\right] + \mathsf{Pr}\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{\mathsf{anon-1}}(1^\lambda)\right]\right| .\end{aligned}$$

We prove that such probability is negligible if the rNIZK is zero-knowledge through a standard sequence of game hops. Let $\mathbf{Game_i}$ the probability that A outputs 0 at the end of the i-th game.

Game 0. Game 0 executes $\mathsf{Exp}_{\mathsf{A}}^{\mathsf{anon-0}}(1^\lambda)$. Hence,

$$\mathsf{Pr}[\mathbf{Game_0}] = \mathsf{Pr}\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{\mathsf{anon-0}}(1^\lambda)\right] .$$

Game 1. In Game 1, B rejects all opening queries for presentation tokens $pt = ((\Pi_i)_{i\in R}, c, otsvk, ots)$ where $c = c_0^*$ included in pt_0^*. Because $otsvk^*$ is included in the Fiat-Shamir hash of c_0^*, the fact that $c = c_0^*$ and that c is valid according to $\mathsf{EVerify}$ means that $otsvk = otsvk^*$. The only way for the adversary to submit a presentation token $pt \neq pt_0^*$ that is rejected in this game but not in the previous game is therefore by using $((\Pi_i)_{i\in R}, ots) \neq ((\Pi_i^*)_{i\in R}, ots^*)$, but any adversary doing so can be used to break the strong one-time unforgeability of the OTS scheme. Therefore,

$$\epsilon_1 = |\mathsf{Pr}[\mathbf{Game_1}] - \mathsf{Pr}[\mathbf{Game_0}]| .$$

Game 2. In Game 2, B generates the ciphertext c_i^* in the target presentation tokens pt_i^* running the simulator of the relaxed verifiable encryption. Note that in the reduction, we never have to answer opening queries for presentation tokens containing c_i^*, as these were already rejected in Game 2. If an adversary A_1 can distinguish Game 1 from Game 0 then it is possible to build a distinguisher B_1 that breaks the chosen-ciphertext simulatability property of the verifiable encryption with probability

$$\begin{aligned}\epsilon_2 &= \left|\mathsf{Pr}\left[b = b' \ : \ b \xleftarrow{\$} \{0,1\}, b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{\mathsf{ccas-b}}(1^\lambda)\right] - \frac{1}{2}\right| \\ &= \frac{1}{2}\left|\mathsf{Pr}\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{\mathsf{ccas-0}}(1^\lambda)\right] + \mathsf{Pr}\left[b' = 1 \ : \ b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{\mathsf{ccas-1}}(1^\lambda)\right] - 1\right| \\ &= \frac{1}{2}\left|\mathsf{Pr}\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{\mathsf{ccas-0}}(1^\lambda)\right] + \mathsf{Pr}\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{\mathsf{ccas-1}}(1^\lambda)\right]\right| \\ &= \frac{1}{2}|\mathsf{Pr}[\mathbf{Game_2}] - \mathsf{Pr}[\mathbf{Game_1}]| .\end{aligned}$$

Game 3. In Game 3, B simulates the rest of the proofs in the target presentation tokens using the simulator of the rNIZK. Distinguishing Game 3 from Game 2 implies the existence of a distinguisher B_3 that breaks the zero-knowledge property of the rNIZK with advantage,

$$\epsilon_3 \;=\; \frac{1}{2}\left|\mathsf{Pr}[\mathbf{Game_3}]-\mathsf{Pr}[\mathbf{Game_2}]\right| \;.$$

Game 4. In Game 4, B sends pt_1^* instead of pt_0^*. As they are both generated in the same way using the simulators (i.e., independently of b), Game 4 is perfectly indistinguishable from Game 3:

$$\mathsf{Pr}[\mathbf{Game_3}] = \mathsf{Pr}[\mathbf{Game_4}] \;.$$

Game 5. In Game 5, B generates the rNIZKs in pt_i^* honestly. Again, the probability of distinguishing this game from the previous one is equal to the advantage of an algorithm B_5 that breaks the zero-knowledge property of $\mathsf{r}\Sigma$ exploiting the game distinguisher:

$$\epsilon_5 \;=\; \frac{1}{2}\Big|\mathsf{Pr}[\mathbf{Game_5}]-\mathsf{Pr}[\mathbf{Game_4}]\Big| \;.$$

Game 6. In Game 6, B generates the ciphertexts c_i^* in pt_i^* honestly. The probability of distinguishing this game from the previous one is equal to the advantage of an algorithm B_6 that breaks the CCS property of the relaxed verifiable encryption scheme exploiting the game distinguisher:

$$\epsilon_6 \;=\; \frac{1}{2}\left|\mathsf{Pr}[\mathbf{Game_6}]-\mathsf{Pr}[\mathbf{Game_5}]\right| \;.$$

Game 7. Finally, in Game 7 B reverse back to accepting all opening queries for presentation tokens $pt = ((\Pi_i)_{i\in R}, c, otsvk, ots)$ where $c = c_1^*$ included in pt_1^*. As for Game 1, an adversary able to distinguish this game from the last one can be exploited to break the strong unforgeability of the OTS with advantage

$$\epsilon_7 = \Big|\mathsf{Pr}[\mathbf{Game_7}]-\mathsf{Pr}[\mathbf{Game_6}]\Big| \;.$$

Remark that Game 7 is in fact $\mathsf{Exp}_{\mathsf{A}}^{\mathsf{anon}-1}(1^\lambda)$,i.e.,

$$\mathsf{Pr}[\mathbf{Game_7}] = \mathsf{Pr}\left[b'=0 \;:\; b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{\mathsf{anon}-1}(1^\lambda)\right] \;.$$

Therefore,

$$\begin{aligned}
\epsilon_A &= \frac{1}{2}\left|\Pr\left[b'=0 \;:\; b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{\mathsf{anon}-0}(1^\lambda)\right] + \Pr\left[b'=0 \;:\; b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{\mathsf{anon}-1}(1^\lambda)\right]\right| \\
&= \frac{1}{2}\left|\Pr[\mathbf{Game_0}] - \Pr[\mathbf{Game_7}]\right| \\
&= \frac{1}{2}\left|\sum_{i=0}^{6} \Pr[\mathbf{Game_i}] - \Pr[\mathbf{Game_{i+1}}]\right| \\
&\leq \frac{1}{2}(2\epsilon_1 + 2\epsilon_2 + 2\epsilon_3 + 0 + 2\epsilon_5 + 2\epsilon_6 + 2\epsilon_7) \\
&= \epsilon_{OTS} + 2\epsilon_{CCS} + 2\epsilon_{ZK}
\end{aligned}$$

where ϵ_{OTS}, ϵ_{CCS}, and ϵ_{ZK} are the advantage in breaking the strong unforgeability of the OTS, the chosen-ciphertext simulatability of the verifiable encryption scheme and the zero-knowledge property of the rNIZK proof system.

□

Simulator $\mathsf{B}_{\mathsf{A},\mathscr{O}_s}(1^\lambda)$
$q_C = q_H = q_D = 0$
$\mathscr{S} \leftarrow \emptyset$, $\mathbf{L}_c \leftarrow \emptyset$, $\mathbf{L}_h \leftarrow \emptyset$, $\mathbf{L}_p \leftarrow \emptyset$
$(spar, ipk^*) \leftarrow \mathscr{O}_s(1^\lambda)$
$cpar \leftarrow \mathsf{DerivePar}_c(spar)$
$epar \leftarrow \mathsf{DerivePar}_{ve}(cpar)$
$apar \leftarrow (cpar, spar, epar)$
$(pt^*, pt', R^*, (\alpha_i^*)_{i\in R^*}, \mu^*) \leftarrow \mathsf{GF}_{\mathsf{A}}^{\mathscr{O}_{IC},\mathscr{O}_{IH},\mathscr{O}_{pres},\mathscr{O}_O}(apar, ipk^*)$
$((\bar{\mathbf{S}}_{c,i}, \bar{\mathbf{c}}_i, \bar{\mathbf{m}}_i, \bar{\sigma}_i)_{i\in R^*}, (\bar{\mathbf{S}}'_c, \bar{\mathbf{c}}')) \leftarrow \mathsf{E}((pt^*, R^*, (\alpha_i^*)_{i\in R^*}), (pt', R^*, (\alpha_i^*)_{i\in R^*}))$
If $\exists\ j$ such that $\nexists\ (\alpha_i)_{i\in R},\ cid,\ \mathbf{m} \mid \alpha_j^* \in \{\alpha_i\}_{i\in R} \wedge (cid, \mathbf{m}, (\alpha_i)_{i\in R}) \in \mathbf{L}_c$ return $((\bar{\mathbf{m}}_j, j \| \alpha_j^*), \bar{\sigma}_j)$.
ElseIf $\exists i,\ j,\ (\alpha_i)_{i\in R},\ (\alpha_i)_{i\in R'},\ cid_i,\ cid_j,\ \mathbf{m}_i,\ \mathbf{m}_j$ such that
$\alpha_i^* \in \{\alpha_i\}_{i\in R} \wedge \alpha_j^* \in \{\alpha_i\}_{i\in R'} \wedge \{(cid_i, \mathbf{m}_i, (\alpha_i)_{i\in R}), (cid_j, \mathbf{m}_j, (\alpha_i)_{i\in R'})\} \subseteq \mathbf{L}_c$
If $\forall k f(\mathbf{m}_i) \cap f(\bar{\mathbf{m}}_k) = \emptyset$ return $((\mathbf{m}_i, i \| \alpha_i^*), \bar{\sigma}_i^*)$.
If $\forall k f(\mathbf{m}_j) \cap f(\bar{\mathbf{m}}_k) = \emptyset$ return $((\mathbf{m}_j, j \| \alpha_j^*), \bar{\sigma}_j^*)$.
Else abort.

Oracle $\mathscr{O}_{IH}((\alpha_i)_{i=1}^\ell)$
$id \leftarrow \mathscr{U} \setminus \mathscr{S}$
$cred \leftarrow (\mathscr{O}_s(id, 1 \| \alpha_1),\ \ldots, \mathscr{O}_s(id, \ell \| \alpha_\ell))$
$\mathscr{S} \leftarrow \mathscr{S} \cup \{id\}$
$cid \leftarrow H(id, cred)$
$\mathbf{L}_h \leftarrow \mathbf{L}_h \cup \{(cid, cred, (\alpha_i)_{i=1}^\ell)\}$
$q_H \leftarrow q_H + 1$
Return cid .

Oracle $\mathscr{O}_O(R, (\alpha_i)_{i\in R}, \mu, pt)$
If $0 \leftarrow \mathsf{Verify}(ipk, opk, R, (\alpha_i)_{i\in R}, \mu, pt)$ abort.
Return $id \leftarrow \mathsf{Open}(ipk, osk, R, (\alpha_i)_{i\in R}, \mu, pt)$.

Oracle $\mathscr{O}_{pres}(cid, R, \mu)$
Recover the entry $(cid, cred, (\alpha_i)_{i=1}^\ell)$ from $\mathbf{L}_h$
(abort if it does not exist).
$(otsvk, otssk) \xleftarrow{\$} \mathsf{OTSGen}(1^\lambda)$
$(\mathbf{E}, \mathbf{b}, (\Pi_i)_{i\in R}) \leftarrow \mathsf{P}_{pt}^{H_c}(ipk, (\alpha_i)_{i\in R}, \mu, 1^\lambda)$
$\mathbf{S}_c \leftarrow [\mathbf{m};\ \mathbf{b};\ \mathbf{E}^T]$
$c \leftarrow \mathsf{Enc}_{LN}((\mathbf{A}_c, \mathbf{C}^T), (\mathbf{S}_c, \mathbf{1}), opk)$
$ots \xleftarrow{\$} \mathsf{OTSSign}(otssk, ((\Pi_i)_{i\in R}, c))$
$pt \leftarrow ((\Pi_i)_{i\in R}, c, otsvk, ots)$
$\mathbf{L}_p \leftarrow \mathbf{L}_p \cup \{(R, \mu)\}$
$q_P \leftarrow q_P + 1$
Return pt .

Figure 4.6. Simulator of the traceability experiment for AAT+O. The oracle $\mathscr{O}_{IC}$ is the same as in Figure 4.3. The token pt' results from rewinding A, hence it differ from pt^* only in the values of challenges and responses.

Chapter 5

Group Signatures from Relaxed Protocols

In this final chapter we present the group signatures that are the final result of our journey into lattice-based privacy-preserving signatures. We include three schemes ([Boschini et al., 2018b], [Boschini et al., 2018a], [Boschini et al., 2020]), presented from the oldest (and less efficient) one, to the most recent and more powerful scheme. Our group signatures follow the common design pattern [Ateniese et al., 2000; Chase and Lysyanskaya, 2006; Benhamouda et al., 2014] whereby a user's signing key is a signature of its identity by the group manager, and where a group signature consists of a ciphertext encrypting the user's identity, together with a signature of knowledge of a signature by the group manager on the encrypted identity.

Formal definitions of group signatures are given in the following.

5.1 Formal Definitions of Group Signatures

Group signatures [Chaum, 1991] allow a user to output signatures on behalf of a group without revealing information about its identity (besides its group membership). Differently from ring signatures [Rivest et al., 2001], group signatures are centralized, meaning that the secret signing keys of the users are generated through an interaction with a group manager. Moreover, the anonymity of the signer can be lifted by an opener if there is a suspected misconduct. In the following we present two flavors of group signatures: standard group signatures, where the issuer is assumed to be always honest, and fully anonymous group signatures, where the issuer can be corrupted. The models are adaptations of the original models by [Bellare et al., 2003, 2005].

5.1.1 Group Signatures with Trusted Issuer

A group signature is a set of algorithms $(\mathsf{GPg}, \mathsf{GKg}, \mathsf{UKg}, \mathsf{OKg}, \mathsf{GSign}, \mathsf{GVerify}, \mathsf{GOpen})$ that regulate the interactions between by a group manager, an opener and users.

Parameters Generation. The group signature parameters *par* are generated via $\mathsf{GPg}(1^\lambda)$ (where λ is the security parameter).

Group Manager Key Generation. The group manager generates its keys running $(gpk, gsk) \leftarrow \mathsf{GKg}(par)$.

Opener Key Generation. The opener generates its keys running $(opk, osk) \leftarrow \mathsf{OKg}(gpk)$.

Joining. If a user wants to join, it sends its identity to the group manager and obtains back its user secret key $usk \leftarrow \mathsf{UKg}(gsk, id)$.

Signing. The user can sign a message μ on behalf of the group using its secret key with the algorithm $\mathsf{GSign}(usk, gpk, opk, \mu)$.

Verification. A signature *sig* on a message μ can be verified with the algorithm $\{1, 0\} \leftarrow \mathsf{GVerify}(\mu, sig, gpk, opk)$.

Opening. The opener can recover the identity of the group member that signed a message μ running $id \leftarrow \mathsf{GOpen}(\mu, sig, osk)$.

We require the scheme to be correct (honestly generated signatures satisfies verification and can be opened to the identity of the signer), traceable (the group manager should be able to link every signature to the user who produced it) and anonymous (signatures produced by different users should be indistinguishable).

Definition 5.1 (Correctness)**.** A group signature $(\mathsf{GPg}, \mathsf{GKg}, \mathsf{OKg}, \mathsf{UKg}, \mathsf{GSign}, \mathsf{GVerify}, \mathsf{GOpen})$ satisfy *correctness* if for all user's identities *id* and messages μ it holds:

$$\mathsf{Pr}\left[\begin{array}{c} 1 \leftarrow \mathsf{GVerify}(\mu, sig, gpk, opk), \\ id \leftarrow \mathsf{GOpen}(\mu, \sigma, osk) \end{array} : \begin{array}{c} par \leftarrow \mathsf{GPg}(1^\lambda), \\ (gpk, gsk) \leftarrow \mathsf{GKg}(par), \\ (opk, osk) \leftarrow \mathsf{OKg}(gpk), \\ usk \leftarrow \mathsf{UKg}(gsk, id), \\ sig \leftarrow \mathsf{GSign}(usk, gpk, opk, \mu) \end{array}\right] \geq 1 - \nu(\lambda)\,.$$

Traceability requires that a corrupted user should not output a signature that cannot be opened, or that can be opened to the identity of a honest user.

Experiment $\mathsf{Exp}^{\mathsf{trac}}_{\mathsf{A}}(1^\lambda)$
$\mathcal{Q}_{\mathsf{UKg}} \leftarrow \emptyset$, $\mathcal{Q}_{\mathsf{GSign}} \leftarrow \emptyset$
$par \leftarrow \mathsf{GPg}(1^\lambda)$
$(gpk, gsk) \leftarrow \mathsf{GKg}(par)$
$(opk, osk) \leftarrow \mathsf{OKg}(gpk)$
$(\mu^*, sig^*) \leftarrow \mathsf{A}^{\mathcal{O}_{\mathsf{UKg}}, \mathcal{O}_{\mathsf{GSign}}}(gpk, opk, osk)$
$id \leftarrow \mathsf{GOpen}(\mu^*, sig^*, osk)$
If $1 \leftarrow \mathsf{GVerify}(\mu^*, sig^*, gpk, opk)$
and $id \notin \mathcal{Q}_{\mathsf{UKg}} \wedge \nexists\, usk : (id, usk, \mu^*) \in \mathcal{Q}_{\mathsf{GSign}}$
then return 1 else return 0.

Oracle $\mathcal{O}_{\mathsf{UKg}}(id)$
$usk \leftarrow \mathsf{UKg}(gsk, id)$
$\mathcal{Q}_{\mathsf{UKg}} \leftarrow \mathcal{Q}_{\mathsf{UKg}} \cup \{id\}$
Return usk .

Oracle $\mathcal{O}_{\mathsf{GSign}}(id, \mu)$
If $id \in \mathcal{Q}_{\mathsf{UKg}}$ abort.
If $\nexists\, usk, \mu' : (id, usk, \mu') \in \mathcal{Q}_{\mathsf{GSign}}$
then $usk \leftarrow \mathsf{UKg}(gsk, id)$
$\mathcal{Q}_{\mathsf{GSign}} \leftarrow \mathcal{Q}_{\mathsf{GSign}} \cup \{(id, usk, \mu)\}$
Else recover the entry (id, usk, μ').
$sig \leftarrow \mathsf{GSign}(usk, gpk, opk, \mu)$
Return sig .

Figure 5.1. Traceability experiment for group signatures with trusted issuance.

Definition 5.2 (Traceability)**.** A group signature satisfies the traceability property if for all PPT adversaries A it holds:

$$\mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{\mathsf{trac}}_{\mathsf{A}}(1^\lambda)\right] \leq \nu(\lambda) \,.$$

where the experiment is shown in Figure 5.1.

Finally, *anonymity* guarantees that no adversary should be able to tell whether two signatures were generated by the same user.

Definition 5.3 (Anonymity)**.** A group signature satisfies *anonymity* if for all PPT adversaries A the following probability is negligible,

$$\left|\mathsf{Pr}\left[b = b' \,:\, b \xleftarrow{\$} \{0,1\},\ b' \leftarrow \mathsf{Exp}^{\mathsf{anon}-b}_{\mathsf{A}}(1^\lambda)\right] - \frac{1}{2}\right|,$$

where the experiment is shown in Figure 5.2.

5.1.2 Group Signatures with Blind Issuance

We present and discuss the Bellare-Shi-Zhang (BSZ) model for dynamic group signature [Bellare et al., 2005]. All the results and observations presented in the section are taken from their work, (unless otherwise stated). The main difference with the previous model is that the issuance is now blind, meaning that, at the end of the interaction, the issuer does not get the signing key of the user.

Experiment $\mathsf{Exp}^{\mathsf{anon}-b}_{\mathsf{A}}(1^\lambda)$
$count \leftarrow 0$
$par \leftarrow \mathsf{GPg}(1^\lambda)$
$(gpk, gsk) \leftarrow \mathsf{GKg}(par)$
$(opk, osk) \leftarrow \mathsf{OKg}(gpk)$
$b' \leftarrow \mathsf{A}^{\mathcal{O}_{\mathsf{GOpen}}, \mathcal{O}^b_C}(gpk, gsk, opk)$
Return b'.

Oracle $\mathcal{O}^b_C(gpk, opk, \mu^*, id_0, id_1)$
$count \leftarrow count + 1$
If $count \geq 2$ abort.
For $i = 0, 1$:
$sig^*_b \leftarrow \mathsf{GSign}(\mathsf{UKg}(gsk, id_b), gpk, opk, \mu^*)$
If $1 \leftarrow \mathsf{GVerify}(\mu^*, sig^*_b, gpk, opk)$ for $i = 0, 1$
then return sig^*_b, else abort.

Oracle $\mathcal{O}_{\mathsf{GOpen}}(\mu, sig, osk)$
If $count \geq 2 \ \wedge \ (\mu, sig) = (\mu^*, sig^*_b)$ abort.
$id \leftarrow \mathsf{GOpen}(\mu, sig, osk)$
Return id.

Figure 5.2. Anonymity experiment for group signature with trusted issuer.

A (dynamic) group signature is a 8-uple of algorithms $(\mathsf{GKg}, \mathsf{UKg}, \mathsf{Join}, \mathsf{Iss}, \mathsf{GSign}, \mathsf{GVerify}, \mathsf{GOpen}, \mathsf{GJudge})$ between a group manager, an opener and users. It is called *dynamic* because users can join at any time during the lifespan of the group (while in static group signatures where joining can only happen during the setup phase at the beginning).

To exclude man-in-the-middle attacks, the model implicitly assumes the existence of a PKI that allows to obtain certified copies of the public key of the entities, represented by a public list **upk** whose i-th entry contains the public key of user i. The position i in the list corresponds to the user identity. Finally, the group manager keeps another list called **reg**, whose i-th entry contains the public key of the i-th user and the registration information (the first message from the user to the issuer).

Key Generation. A trusted third party generates the keys of the group manager and opener running $(gpk, gsk, opk, osk) \leftarrow \mathsf{GKg}(1^\ell)$ (where ℓ is the security parameter). For the sake of clarity, we distinguish the group public key gpk from the opener's public key opk. A user generates its user secret key as $(usk, upk) \leftarrow \mathsf{UKg}(gpk)$.

Joining Phase. The joining phase is an interactive protocol between a user i (running algorithm Join) and the group manager (running Iss). Each takes as input an incoming message (ϵ if it is the first step of the interaction) and the current state, and outputs an outgoing message, an updated state, and a decision ($accept$, $reject$, $continue$). At the end of the interaction, the user obtains a signing key gsk_i that includes its keys usk, upk and its identity i as group member. If at the end of the interaction the user accepts, the

group manager creates a new entry $\mathbf{reg}[i]$ in the list **reg** containing the identity of user i and the registration information (the first message from the user). Remark that this does not contain the full gsk_i, as the issuer does not knowusk_i. The list is needed to allow the opener to prove that the identity it has extract from the signature corresponds to an existing group member.

Signing. A user can sign a message μ on behalf of the group using its signing key with the algorithm $\mathsf{GSign}(gsk_i, gpk, opk, \mu)$.

Verification. A signature σ on a message μ can be verified with the algorithm $\{1,0\} \leftarrow \mathsf{GVerify}(\mu, \sigma, gpk, opk)$.

Opening. The opener can recover the identity i' of the group member that signed a message μ running $(i', \tau) \leftarrow \mathsf{GOpen}(\mu, \sigma, gpk, osk, \mathbf{reg})$. The algorithm outputs the identity of the signer with a proof that the opening procedure was performed honestly (contained in τ). The list is needed to prove that the identity i' that the opener outputs corresponds to an existing group member. If opening fails, it outputs $(i', \tau) = (\epsilon, 0)$

Judge. A third party can check whether the opener honestly opened a signature by running the algorithm $0,1 \leftarrow \mathsf{GJudge}(gpk, opk, upk_{i'}, \mu, \sigma, i', \tau)$[1].

The security requirements of the group signature are correctness, anonymity, traceability, and non frameability.

The formal definitions require the definition of some lists. In the following, we clarify what are the lists for the BSZ model and how are they used.

HU To keep track of honest users, a list **HU** is created. Such list contains as i-th entry the public and secret key of the user with identity i. Hence, by "adding i to the list **HU**" we mean $\mathbf{HU}[i] \leftarrow (upk_i, usk_i)$. Empty entries are set to $\perp$.

HK The signing keys of the users in **HU** are stored in another list **HK** (this list was implicit in the original model). The list **HK** is updated setting $\mathbf{HK}[i] \leftarrow gsk_i$ only if $i \in \mathbf{HU}$.

[1] In an earlier construction we considered giving the Judge algorithm the list **reg** as input. This seemed to strengthen the role of the judge, but it would not allow for it to be invoked by the users, as they have no access to the list **reg**.

CU The list **CU** contains all the users that were corrupted *before* joining the group (i.e., corrupted by A through CrptU). The adversary is allowed to corrupt users after they become group members (through USK), but such users are not into this list. In fact, such list contains the users that might have colluded with the group manager (in case M is corrupted) to tamper with the list **reg** (impacting honest opening and consequently J). To detect when the adversary forges a signature by simply querying a user's signing key, the non-frameability experiment checks A's queries to USK before accepting a forgery.

CH In **CH** are stored the challenge message-signature pairs generated during the anonymity game.

GSig Finally, we need a list **GSig** of the signatures output by the signing oracle.

The adversary has also access to the following oracles (cf. Figure 2 in [Bellare et al., 2005], we only give an informal description here):

$\mathcal{O}_{\mathsf{Iss}}$**:** this oracle (denoted SndToI in [Bellare et al., 2005]) performs honestly the issuer's side of the joining phase. During the interaction, the adversary impersonates a user i willing to join the group. The oracle first checks that the user i is a corrupted user; if $i \notin$ **CU** the algorithm aborts. At the end of the interaction it updates the list of users l with the new user credential.

$\mathcal{O}_{\mathsf{GOpen}}$**:** on input a pair (μ, σ), if $(\mu, \sigma) \notin$ **CH** this oracle (denoted Open in [Bellare et al., 2005]) outputs the honest opening of the signature, otherwise it outputs $\perp$.

$\mathcal{O}_{\mathsf{GSign}}$**:** on input i and a message μ, it aborts if $i \notin$ **HU**. Otherwise, it recovers the corresponding signing key gsk_i from **HK** (it aborts if such key does not exist), then it outputs a signature $\sigma \leftarrow \mathsf{GSign}(gsk_i, gpk, opk, \mu)$. It stores (i, μ) in a list **GSig**.

Chall_b**:** on input a message μ and two identities i_0, i_1, it recovers gsk_{i_b} from **HK** and outputs a signature σ on μ using the signing key of user i_b if $i_0, i_1 \in$ **HU** (otherwise it aborts). The algorithm adds the entry (μ, σ) to **CH**.

RReg**:** on input i, outputs the i-th entry of the list $\mathbf{reg}[i]$ to the adversary[2].

WReg**:** on input i and a string B (that is a valid list entry), sets $\mathbf{reg}[i]$ to B.

[2]Remark that there is no need to add i to the list of corrupted users **CU**, as **reg** does not contain the full gsk_i but only the registration information.

	Group Manager	User	Opener	Judge
Correctness		✓(the adversary can add users)		
Anonymity	✓	✓(all *usk* can be leaked)		
Traceability		✓(all *usk* can be leaked)	✓ (partially)	
Non frameability	✓	✓(there should exist at least one honest user)	✓	

Table 5.1. Corrupted entities depending on the security property.

SndToU: this algorithm allows the adversary to choose a user i and to run with i the joining protocol impersonating the issuer. The result of this interaction is that there is a new honest group member, hence the algorithm adds i to **HU** and the resulting gsk_i to **HK**.

CrptU: the adversary uses this algorithm whenever it wants to corrupt a user *before* it has joined the group. On input i, upk', it first checks whether i is in **HU** or **CU**. If $i \in \mathbf{CU} \cup \mathbf{HU}$, it returns ϵ. Then it looks up the user with keys (upk_i, usk_i), and sets its key to be (upk', usk_i). Finally, if i is in **HU**, it removes it and adds (upk', usk_i) to **CU** it is in **CU**, it updates it to be (upk', usk_i).

USK: on input the user's public key upk_i, outputs the corresponding secret key usk_i from **HU** and secret signing key gsk_i from **HK**.

AddU: on input a user identity i, it creates a new honest user with identity i. If $i \in \mathbf{CU} \cup \mathbf{HU}$ it aborts. Otherwise, it generates usk_i, upk_i, adds i to **HU**, and executes honestly the group joining protocol by running Join and Iss on behalf of user i and of the issuer respectively, where both algorithms are initialized with the necessary keys. If the protocol ends successfully, it returns upk_i and stores gsk_i in **HK**.

We slightly modified the BSZ model to clarify the definitions. Mainly, we introduced a list **GSig**, to make it easy to verify whether a message was signed with a particular *usk* by the honest signing algorithm $\mathcal{O}_{\mathsf{GSign}}$, and we do not assume that access to WReg implies that A is able to read the list too. The result is that, whenever the adversary can corrupt the issuer, it is given access to both RReg and WReg, otherwise, if A can corrupt the O it is only given access to RReg.

To understand why we need these oracles, we analyze the different degrees of corruption of the entities depending on the security property we want to guarantee. We briefly explain this in the following; our observations are summarized in Table 5.1. Note that the Judge is always honest by construction.

The algorithms $\mathcal{O}_{\mathsf{Iss}}$, $\mathcal{O}_{\mathsf{GOpen}}$ and $\mathcal{O}_{\mathsf{GSign}}$ model the honest behavior of the issuer, the opener and a user respectively. The other algorithms model the various

hostile scenarios. There are different degrees of corruption, but the distinction Bellare et al. are concerned with is between *partial* and *full* corruption. In both cases the secret key is leaked to the adversary. However, a partially corrupt entity still performs its task honestly, while a fully corrupt one will give the adversary full control, allowing deviations from the protocols.

Note that this distinction is not relevant in the case of the issuer. In fact, knowing the issuer secret key is not useful for the adversary if it cannot interact with users. Hence in the definitions a *corrupted* issuer is always assumed to be *fully* corrupted (i.e., when the adversary gets *gsk* it is also given access to the oracle SndToU).

The case of the opener is different. In fact, in this case knowing the secret opening key is useful to the adversary, as it is enough to de-anonymize signatures produced by group members.

Finally, regarding users, the adversary has multiple options. It can either create users (querying $\mathcal{O}_{\mathsf{Iss}}$ or exploiting knowledge of *gsk*), ask users to sign messages of its choice (interacting with $\mathcal{O}_{\mathsf{GSign}}$), or corrupt a user to either change its public key (CrptU), or reveal its secret key (USK).

The algorithm Chall_b is used in the anonymity experiment to model the generation of the challenge signature. In this way, it is not necessary to distinguish in the experiment the adversary's capabilities before and after receiving the challenge.

Correctness requires that honestly generated signatures satisfy the verification checks, can be opened to the correct identity of the signer, and that the proof generated by a honest opener is always accepted by the GJudge algorithm.

Definition 5.4 (Correctness)**.** A group signature (GKg, UKg, Join, Iss, GSign, $\mathsf{GVerify}$, GOpen, GJudge) satisfies *correctness* if every adversary A has no advantage in winning the experiment $\mathsf{Exp}^{corr}_{\mathsf{A}}(1^\lambda)$ in Figure 5.3, i.e.,

$$\mathsf{Adv}^{corr}_{\mathsf{A}}(\lambda) := \mathsf{Pr}[1 \leftarrow \mathsf{Exp}^{corr}_{\mathsf{A}}(1^\lambda)] = \nu(\lambda)\ .$$

Anonymity informally requires the adversary not to be able to distinguish which user has signed a message of its choice. For the property to be meaningful, the adversary should not be allowed to corrupt the opener. On the other hand, the adversary is allowed to *fully corrupt* (i.e., to know the secret key of) both the users and the issuer. In particular, the adversary should not be able to recognize the signatures generated by a user even if it recovers the secret key of the user.

Definition 5.5 (Anonymity)**.** A group signature (GKg, UKg, Join, Iss, GSign,

Experiment $\mathsf{Exp}^{corr}_{\mathsf{A}}(1^\lambda)$
$\mathbf{CU} \leftarrow \emptyset$, $\mathbf{HU} \leftarrow \emptyset$
$(gpk, gsk, opk, osk) \leftarrow \mathsf{GKg}(1^\lambda)$
$(i, \mu) \leftarrow \mathsf{A}^{\mathsf{AddU},\ \mathsf{RReg}}(gpk, opk)$
If $(\mathbf{HU}[i] = \perp \ \vee\ \mathbf{HK}[i] = \perp)$ return 0.
$\sigma \leftarrow \mathsf{GSign}(gsk_i, gpk, opk, \mu)$
If $0 \leftarrow \mathsf{GVerify}(\mu, \sigma, gpk, opk)$ return 1.
$(i', \tau) \leftarrow \mathsf{GOpen}(\mu, \sigma, gpk, osk)$
If $(i' \neq i \ \vee\ 0 \leftarrow \mathsf{GJudge}(gpk, opk, upk_{i'}, \mu, \sigma, i', \tau))$ return 1, else return 0.

Figure 5.3. Correctness experiment for group signatures with blind issuance.

Experiment $\mathsf{Exp}^{an}_{\mathsf{A},b}(1^\lambda)$
$(gpk, gsk, opk, osk) \leftarrow \mathsf{GKg}(1^\lambda)$
$\mathbf{CU} \leftarrow \emptyset$, $\mathbf{HU} \leftarrow \emptyset$, $\mathbf{CH} \leftarrow \emptyset$
$b' \leftarrow \mathsf{A}^{\mathcal{O}_{\mathsf{GOpen}},\ \mathsf{CrptU},\ \mathsf{USK},\ \mathsf{RReg},\ \mathsf{WReg},\ \mathsf{SndToU},\ \mathsf{Chall}_b}(gpk, gsk, opk)$
Return b'.

Figure 5.4. Anonymity experiment for group signature with blind issuance.

$\mathsf{GVerify}, \mathsf{GOpen}, \mathsf{GJudge}$) satisfies anonymity if

$$\mathsf{Adv}^{anon}_{\mathsf{A}}(\lambda) := \left| \Pr\left[b = b' \ :\ b \xleftarrow{\$} \{0,1\},\ b' \leftarrow \mathsf{Exp}^{\mathsf{an}}_{\mathsf{A},b}(1^\lambda) \right] - \frac{1}{2} \right|,$$

is negligible in the security parameter for all PPT adversaries A, where the experiment is shown in Figure 5.4.

Coherently with Bellare et al., in $\mathsf{Exp}^{an}_{\mathsf{A},b}$ the adversary is not given access to a signing oracle, as it would be redundant.

The notion of traceability for group signature essentially means that the opener can always link a signature back to the signer (cf. [Bellare et al., 2003]). When the group signature has blinded joining protocol, such notion is split in two. This is because when the joining protocol is blinded both the issuer and the opener can be corrupted. When the joining protocol is not blinded (i.e., the issuer gets to see the user secret key), no security can be guaranteed in such scenario.

This is modeled with two different security notions. First, an adversary A should not be able to produce a signature that cannot be opened by an honest opener, or such that a honest opener is unable to prove that the opening was correctly executed, even if A corrupts the opener (*traceability*). If A could corrupt

Experiment $\mathsf{Exp}^{tr}_{\mathsf{A}}(1^\lambda)$
$(gpk, gsk, opk, osk) \leftarrow \mathsf{GKg}(1^\lambda)$
$\mathbf{CU} \leftarrow \emptyset$, $\mathbf{HU} \leftarrow \emptyset$, $\mathbf{GSig} \leftarrow \emptyset$
$(\mu^*, \sigma^*) \leftarrow \mathsf{A}^{\mathcal{O}_{\mathsf{Iss}},\ \mathsf{AddU},\ \mathsf{USK},\ \mathsf{CrptU},\ \mathsf{RReg}}(gpk, opk, osk)$
If $0 \leftarrow \mathsf{GVerify}(\mu^*, \sigma^*, gpk, opk)$, return 0
Else $(i^*, \tau^*) \leftarrow \mathsf{GOpen}(\mu^*, \sigma^*, gpk, osk)$.
If $(i^* = \epsilon) \ \vee \ 0 \leftarrow \mathsf{GJudge}(gpk, opk, upk_{i^*}, \mu^*, \sigma^*, i^*, \tau^*)$ return 0.
Else return 1.

Figure 5.5. Traceability experiment for group signature with blind issuance.

the issuer this would be trivial to achieve, as the issuer could issue a signing key gsk_i to a user i, without adding the user keys to the list **reg**. In this way, A can produce valid signatures using gsk_i, but the opener cannot recover the user's public key, as $\mathbf{reg}[i]$ does not exist. Analogously, if A could fully corrupt the opener the definition would be meaningless, as the adversary would win it always by simply forcing the opener to declare itself unable to open the forged signature A outputs.

Definition 5.6 (Traceability)**.** A group signature satisfies traceability if all PPT adversaries A have negligible advantage in the experiment $\mathsf{Exp}^{tr}_{\mathsf{A}}$ in Figure 5.5, i.e.,

$$\mathsf{Adv}^{trac}_{\mathsf{A}}(\lambda) := \mathsf{Pr}[1 \leftarrow \mathsf{Exp}^{tr}_{\mathsf{A}}] = \nu(\lambda)\ .$$

Remark that again there is no need to give to the adversary explicit access to an oracle that produces signatures of honest users, as the adversary could just query the user's secret keys through USK.

On the other hand, it should be impossible for A to generated a signature that links back to a honest user who did not produce it, even if it corrupted both the issuer and the opener (*non-frameability*).

Definition 5.7 (Non-frameability)**.** A group signature satisfies non-framea- bility if all PPT adversaries A have negligible advantage in the experiment $\mathsf{Exp}^{nf}_{\mathsf{A}}$ in Figure 5.6, i.e., $\mathsf{Adv}^{non-fr}_{\mathsf{A}}(\lambda) := \mathsf{Pr}[1 \leftarrow \mathsf{Exp}^{nf}_{\mathsf{A}}] = \nu(\lambda)$.

5.2 Warm-up: Group Signatures from AAT+O

From the AAT+O presented in Section 4.2.2 it is possible to construct a group signature with trusted issuer in a quite simple way. In fact, the scheme is exactly

Experiment $\mathsf{Exp}^{nf}_{\mathsf{A}}(1^\lambda)$
$(gpk, gsk, opk, osk) \leftarrow \mathsf{GKg}(1^\lambda)$
$\mathbf{CU} \leftarrow \emptyset$, $\mathbf{HU} \leftarrow \emptyset$, $\mathbf{GSig} \leftarrow \emptyset$
$(\mu^*, \sigma^*) \leftarrow \mathsf{A}^{\mathcal{O}_{\mathsf{GSign}}, \mathsf{AddU}, \mathsf{USK}, \mathsf{CrptU}, \mathsf{RReg}, \mathsf{WReg}, \mathsf{SndToU}}(gpk, gsk, opk, osk)$
If $0 \leftarrow \mathsf{GVerify}(\mu^*, \sigma^*, gpk, opk)$, return 0
Else $(i^*, \tau^*) \leftarrow \mathsf{GOpen}(\mu^*, \sigma^*, gpk, osk)$.
If $\mathbf{HU}[i^*] \neq \perp \wedge 1 \leftarrow \mathsf{GJudge}(gpk, opk, upk_{i^*}, \mu^*, \sigma^*, i^*, \tau^*) \wedge (i^*, \mu^*) \notin \mathbf{GSig}$
return 1.
Else return 0.

Figure 5.6. Non-frameability experiment for group signature with blind issuance.

the AAT+O scheme without the attributes. The part of the verification key that was generated from the attributes can be set to be a uniformly random element $\mathbf{u} \xleftarrow{\$} R_q$. Remark that this does not affect the unforgeability of the relaxed signature, as this change is equivalent to simply restrict the message set of the scheme to $\mathscr{U} \times \{\bar{\alpha}\}$ for a fixed, randomly chosen attribute $\bar{\alpha}$.

As we already noted when presenting the AAT+O, the straightforward combination of the verifiable encryption with the AAT-O from Section 4.1.2 does not result in an efficient scheme. That been said, we decided to include this scheme in the dissertation as well, to have a starting point from which we would build the other two schemes, and to have a basis for comparison. To clarify how the scheme works, instead of presenting it as a combination of the algorithms of $\mathsf{r}\Sigma$, rC, rS, and of the Lyubashevsky-Neven verifiable encryption, we show the actual computations that group manager, opener and users have to do when running the different group signature algorithms. We assume as usual that the parameters of the scheme are generated by a trusted third party.

Let the challenge space be $\mathscr{C} = R_5^{(2^K)}$, and the user identities be $\mathscr{U} \times \{\bar{\alpha}\} \subseteq R_3^{(2^K)} \times \{\bar{\alpha}\}$ (for a random $\bar{\alpha}$ chosen in the parameter generation step and $\mathscr{U}$ as in Section 3.5.3), i.e., $K_c = K_m = K$. To have negligible soundness error, it is enough to repeat the NIZK proof 2 times.

Parameters Generation. The parameter generation $\mathsf{GPg}(1^\lambda)$ outputs a collection of parameters *par* that includes the signature parameters *spar* from Section 3.6.3, the parameters for the prover in Section 3.7, and the parameters p and σ_e of the LN verifiable encryption in Section 3.8.3, plus a random attribute $\bar{\alpha} \in \{0,1\}^*$.

Group Manager Key Generation. The group manager runs the key generation

algorithm $\mathsf{SKeyGen}$ of the signature to obtain $gsk = \mathbf{X}$ and the public matrix $\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{C} & \mathbf{1}\end{bmatrix}$.

Opener Key Generation. The opener chooses $\mathbf{s}_1, \mathbf{s}_2 \leftarrow \mathscr{U}(R_3)$, and computes the public key of the encryption scheme $\mathbf{d} \leftarrow R_q$ and $\mathbf{t} = \mathbf{d}\mathbf{s}_1 + \mathbf{s}_2 \bmod q$.

Joining. The group manager chooses[3] a random element $\mathbf{m} \in \mathscr{M}$ and computes a signature on $(\mathbf{m}, \bar{\alpha})$ using the Sign algorithm. The signing key of the user *id* is $\mathbf{m}$ together with the resulting signature $(\mathbf{1}, [\mathbf{S}; \mathbf{1}], \mathbf{1})$. The group manager stores $\mathbf{reg}[id] \leftarrow \mathbf{m}$ for later reference.

Signing. Generating a signature by the user $id = \mathbf{m}$ on a message μ is exactly equal to producing a presentation token for the credential $(\mathbf{m}, \bar{\alpha})$ in the AAT+O. The user runs the non-interactive (aggregated) prover $\mathsf{P}'(\mathbf{V}, \mathbf{S}, (\mathbf{m}, \bar{\alpha}), \mu)$ from Section 3.7, whereby it includes the message μ in the Fiat-Shamir hash. Then, it generates a one-time signature key pair $(otsvk, otssk) \leftarrow \mathsf{OTSGen}(1^\lambda)$, constructs $\mathbf{S}_c$ (cf. Equation (3.15)), and generates a verifiable ciphertext $c \xleftarrow{\$} \mathsf{Enc}_{LN}(opk, \mathbf{A}_c, \mathbf{S}_c, otsvk)$ as per Equation (4.1), where *otsvk* is only used as input of the Fiat-Shamir hash of the verifiable encryption. Finally, it computes a one-time signature $ots \leftarrow \mathsf{OTSSign}(otssk, (\Pi, c))$ and outputs the token $pt = (\Pi, c, otsvk, ots)$.

Verification. The verifier checks the validity of Π w.r.t. $\mathbf{F}$ and the message μ running $\mathsf{V}(\mathbf{V}, \bar{\alpha}, \Pi, \mu)$, checks that $\mathsf{EVerify}_{LN}(opk, \mathbf{A}_c, otsvk, c) = 1$ with *otsvk* in the Fiat-Shamir hash, and that $\mathsf{OTSVf}(otsvk, (\Pi, c)) = 1$. If all tests pass, it outputs *accept*, otherwise *reject*.

Opening. The opener first verifies the signature, returning $\perp$ if it rejects. Then, it decrypts $(\bar{\mathbf{S}}_c, \bar{\mathbf{c}}) \leftarrow \mathsf{Dec}_{LN}(osk, c)$ and recovers $\bar{\mathbf{m}}$ as the first coordinate from $\bar{\mathbf{S}}_c$. He then recovers an irreducible factor $\mathbf{m}$ of degree $n/2$ of $\bar{\mathbf{m}}$ and returns $\mathbf{m}$, or returns $\perp$ if such a factor does not exist.

[3] This seem to contradict the formal definition of group signature with trusted issuance (cf. Section 5.1.1), as in that case the user identity is given as input to UKg. However, this can be solved by adding two assumptions. First, we assume that each user has obtained a certified public key from a trusted PKI (that could be put in place by the same trusted third party that generated the parameters of the scheme). Second, to get the user secret key, each user has first to authenticate with the issuer using the certified public key. Then, user identities are in a bijective correspondence with the certified public keys (and the pairs can be stored by the issuer in the list **reg** shared with the opener). Therefore, when the opener recovers *id*, it can recover the public key of the user as well.

Observe that the opening time is at most linear in the number of possible challenges.

It is trivial to see that proving the correctness and security of this group signature is exactly the same as proving correctness and security of the AAT+O in Section 4.2.2, as the only difference is that the set of possible attributes is restricted to $\{\bar{a}\}$. Indeed, the definition of anonymity of AAT+O (Definition 4.8) is exactly equivalent to the definition of anonymity for group signatures with trusted issuer (Definition 5.3). The same holds for the correctness of the schemes. The only difference is in the traceability: the traceability definition for group signatures with trusted issuer assumes that the opener can be partially corrupt, meaning that the opener secret key is leaked to the adversary, but the opener still answers honestly to opening queries. Conversely, in the definition of traceability of the AAT+O, the opener is honest, i.e., the adversary does not get to see the opener secret key. Therefore, this scheme achieves a weaker security notion than the other two group signatures presented in this chapter.

We include the theorem statements for completeness.

Theorem 5.8 (Traceability)**.** *Let ϵ_{BIN} be the probability of breaking the binding property of the commitment scheme rC underlying $\mathsf{r}\Sigma$, ϵ_{SS} be the probability of breaking the relaxed simulation soundness of $\mathsf{r}\Sigma$, and ϵ_S be the probability of breaking the soundness of the Lyubashevsky-Neven relaxed verifiable encryption scheme.*

Assume A is an adversary that runs in time t_A, makes q_C queries to the issuer to add corrupt users and q_H queries to the issuer to add honest users, asks for q_P signatures by honest users, and breaks the traceability of the group signature scheme with probability ϵ_A without corrupting the opener.

There exists an algorithm B that breaks the unforgeability of the signature in time $t_B = 8\ell^2 t_A(q_C+q_H+q_P)/\epsilon_A \cdot \ln(8\ell/\epsilon_A) + (q_C+q_H)\cdot t_S + t_{\mathsf{E}} + \mathrm{poly}(\lambda)$ (where t_{E} is the runtime of the extractor of the rNIZK, t_S is the runtime of the signing oracle, and ℓ is the number of attributes contained in the forged pt) with probability $\epsilon_B = \epsilon_A/8 - (\epsilon_{BIN}+\epsilon_{SS}+\epsilon_S)$ after asking q_C+q_H queries to the signing oracle in the Random Oracle Model.

Theorem 5.9 (Anonymity)**.** *If the relaxed Σ-protocol $\mathsf{r}\Sigma$ is zero-knowledge, the one-time signature scheme is strongly unforgeable, and the Lyubashevsky-Neven verifiable encryption scheme is chosen-ciphertext simulatable, then the group signature scheme is anonymous in the Random Oracle Model.*

5.3 Group Signature with Trusted Issuance from Lattices

The combination of Boschini et al.'s relaxed signature scheme [Boschini et al., 2018b] with our relaxed partial verifiable encryption scheme (cf. Section 3.8.4) yields an efficient group signature with practical parameters (see Section 5.3.4). It also allows not to rely anymore on the relaxed binding property of the commitment scheme, thus allowing us to drop the requirement on the space of user identities $\mathscr{U}$: in our construction $\mathscr{U}$ is the entire subspace $R_3^{(2^K)}$ (not the subset of its irreducible elements). This construction was presented at ACNS 2018 [Boschini et al., 2018a].

5.3.1 Improvements over the Trivial Construction

The differences with the construction in Section 5.2 all lie in the choice of verifiable encryption scheme. In fact, with the construction we also introduced the variant of the Lyubashevsky-Neven scheme presented in Section 3.8.4.

This partially verifiable encryption scheme allows us to split the proof of knowledge of a witness for the relations $(\mathscr{R}, \bar{\mathscr{R}})$ in Equation (3.18) and (3.19) in two separate rNIZKs, one which is also used to decrypt and that only proves that the ciphertext was correctly generated, and the other that only guarantees that the plaintext was part of a valid witness for the public instance of the relation. In this way, we can do rejection sampling separately on the identity and on the commitment opening values, and we do not need to stick to a small challenge set in both profs. Splitting the rejection sampling step is a technicality that has quite the impact on the size of the transcript. Indeed, recall that the dimension of the transcript depends on the standard deviation of the Gaussian distribution used to hide the distribution of the witness. Such standard deviation has to be larger than the norm of the witness to actually hide it. Hence, the larger the norm of the witness, the larger the standard deviation (and, consequently, the longer the transcript) has to be. This would not be a problem if we were to use the infinity norm, but in the original construction we used the Euclidean norm, which obviously increases when the witness vector gets longer. In practice this trick allowed us to save a factor of $O(\sqrt{m})$ in the length of the signature. The fact that we do not use both proofs for decryption allows to generate them using two different challenge sets: a small one ($\mathscr{C}_2 = R_3^{(16)}$) for the proof of knowledge of the plaintext, that requires to repeat the proof to have negligible soundness error but allows the guessing step in the decryption, and a

larger one ($\mathscr{C}_1 = \{\mathbf{c} \in R_3 \mid \|\mathbf{c}\|_1 \leq 32\}$) in the other rNIZK, so that there is no need to repeat it to have a negligible soundness error. These two changes have a big impact on how a signature is generated: it is no more necessary to run the prover from Section 3.7, but we define a simpler rNIZK (P, V) that only prove knowledge of Relation (II) in Equation (3.15). The different choice of message and challenge space impacted also the upper bound β_s on the norm of the RSIS solution in the proof of the hardness of Assumption 4 in Section 3.6.5. The new statement of the theorem follows.

Theorem 5.10 (Hardness of Assumption 4)**.** *Let* A *be a probabilistic algorithm that breaks Assumption 4 in time t with probability* ϵ_A*. Then there exists a probabilistic algorithm* B *that either breaks* $RLWE_{m,\mathscr{D}_\sigma}$ *in time t with probability* ϵ_A *or* $RSIS_{3+m,q,\beta_s}$ *in time t with probability* $\epsilon_B \geq (\epsilon_A - \epsilon_{\text{LWE}})/(2 \cdot |\mathscr{C}|)$*, where* $\beta_s = N'^2 + \frac{\sigma_t^2}{\pi} n^2 (\sqrt{2} + \sqrt{m} + \log n)^2 (2\sqrt{2^{K_c}})^2 N'^2 + \frac{\sigma^2}{\pi} n (1 + \sqrt{2} + \log n)^2 (C'^2 + (1.05\sigma_t \sqrt{n})^2)$*,* ϵ_{LWE} *is the probability of breaking the Ring-LWE problem over* R_q *in time t, in the Random Oracle Model.*

Moreover, our partially verifiable encryption allows for a parameters setup that allows to decrypt exactly (cf. Lemma 3.42). Therefore there is no need for the binding property of the relaxed commitment scheme, hence for the restriction of the message space to irreducible polynomials. Indeed, instead of extracting the identity of the signer we can now simply get it decrypting the forged signature and Lemma 3.42 guarantees that the result of the decryption corresponds with high probability to (part of) the output of the extractor.

Finally, as in the partial verifiable encryption the relation is defined modulo q there is no need to encrypt modulo a larger prime (which blows up the dimension of the ciphertext, as q could be as big as 2^{92}, which was our most conservative choice fr the AAT-O) as in the original scheme.

5.3.2 The Group Signature Scheme

In the following we present the group signature. The protocol makes use of a relaxed Σ-protocol (P, V) as defined in Section 3.4, a relaxed signature scheme $(\mathsf{SParGen}, \mathsf{SKeyGen}, \mathsf{Sign}, \mathsf{SVerify})$ as defined in Section 3.6.3 and of a partially verifiable encryption scheme $(\mathsf{EKeyGen}, \mathsf{Enc}, \mathsf{EVerify}, \mathsf{Dec})$ as defined in Section 3.8.4. To compose these schemes, we define an algorithm $\mathsf{DerivePar}_{\text{pve}}$ that derives the parameters of the partially verifiable encryption scheme from the parameters of the signature scheme.

Parameter Derivation for pvenc. The algorithm $\mathsf{DerivePar}_{\text{pve}}$ chooses a p that satisfies the bounds in Equation (3.23) with the q and n already set by

$\mathsf{SParGen}$. Then it sets the rest of the parameters (the challenge sets $\mathscr{C}_1$ and $\mathscr{C}_2$, the standard deviations, the norm bounds, ...) according to the specifications in Section 3.8.4.

Remember that now we have restricted the message set of the relaxed signature to be $\mathscr{M} = \mathscr{U} \times \{\bar{\alpha}\}$ for a randomly selected binary string $\bar{\alpha}$.

Parameters Generation. On input the security parameter λ, the algorithm GPg runs the parameter generator of the signature scheme $spar \leftarrow \mathsf{SParGen}(1^\lambda)$ and derives the parameters $epar$ of the partially verifiable encryption using $\mathsf{DerivePar}_{\mathrm{pve}}$. Finally, it sets the parameters $ppar = (\sigma_0, N_0, \bar{N}_0, \mathscr{C}_0)$ of the relaxed Σ-protocol according to the specifications in Section 3.4. In particular, the challenge space is set to $\mathscr{C}_0 = \{\mathbf{c} \in R_3 \ : \ \|\mathbf{c}\|_1 \leq 32\}$ so that the proof only needs to be repeated once, as indeed $|\mathscr{C}_0| > 2^{256}$. More details about the choice of the parameters can be found in the proof of Theorem 5.11. It outputs $par := (ppar, spar, epar)$.

Group Manager Key Generation. The group manager generates the keys $gsk = \mathbf{X}$ and $gpk = \left(\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{C} & \mathbf{1}\end{bmatrix}, \bar{\alpha}\right)$ by running $\mathsf{SKeyGen}$ and choosing a random string $\bar{\alpha} \xleftarrow{\$} \{0,1\}^\lambda$.

Opener Key Generation. The opener runs the key generation algorithm of the partially verifiable encryption scheme $\mathsf{EKeyGen}(1^\lambda)$ and returns the resulting key pair $(opk = epk, osk = esk)$.

User Key Generation. The group manager generates a signing key user identity $id = \mathbf{m} \in \mathscr{U} = R_3^{(16)}$ by running $\mathsf{Sign}(gsk, \mathbf{m})$ to yield $(\mathbf{1}, \begin{bmatrix}\mathbf{S} & ; & \mathbf{0}\end{bmatrix}, \mathbf{1})$ as described in Section 3.6.3. Recall that $\mathbf{S}$ is a short vector that satisfies $\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{C}+\mathbf{mG}\end{bmatrix}\mathbf{S} = \mathbf{u} \bmod q$, where we denote by $\mathbf{u}$ the output of $H(\bar{\alpha})$ for the sake of succintness. It then returns $usk := \mathbf{S}$.

Signing Algorithm. This is an informal description of the algorithm. The formal construction can be found in Figure 5.7.

The user first generates a key one-time signature key pair $(otssk, otsvk) \leftarrow \mathsf{OTSGen}(1^\lambda)$. The user then blinds its identity $\mathbf{m}$ using the technique from Section 3.6.3 by choosing random $\mathbf{E} \xleftarrow{\$} R_3^{1\times m}$ and $\mathbf{b} \xleftarrow{\$} R_3$, and computing $\mathbf{F} = \mathbf{b}^{-1}(\mathbf{C} + \mathbf{mG} + \mathbf{E}) \bmod q$. If $\mathbf{S} = \begin{bmatrix}\mathbf{S}_1 & ; & \mathbf{S}_2 & ; & \mathbf{S}_3\end{bmatrix}$ with $\mathbf{S}_1 \in R_q^{2\times 1}$ and $\mathbf{S}_2,\ \mathbf{S}_3 \in R_q^{m\times 1}$, then it holds $\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{F} & \mathbf{1}\end{bmatrix}\begin{bmatrix}\mathbf{S}_1 & ; & \mathbf{S}_2 & ; & \mathbf{bS}_3 & ; & -\mathbf{ES}_3\end{bmatrix} = \mathbf{u} \bmod q$. Denoting by $\mathbf{S}_0 := \begin{bmatrix}\mathbf{S}_1 & ; & \mathbf{S}_2 & ; & \mathbf{bS}_3 & ; & -\mathbf{ES}_3\end{bmatrix}$, the user can

$\mathsf{GSign}(usk, gpk, opk, \mu)$
$(otssk, otsvk) \leftarrow \mathsf{OTSGen}(1^\lambda)$
$\mathbf{E} \xleftarrow{\$} R_3^{1\times m}$
$\mathbf{b} \xleftarrow{\$} R_3$
$\mathbf{F} \leftarrow \mathbf{b}^{-1}(\mathbf{C} + \mathbf{mG} + \mathbf{E}) \bmod q$
$(\mathbf{T}_0;\ \mathbf{Y}_0) \leftarrow \mathsf{P}_0(ppar, \begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{F} & \mathbf{1}\end{bmatrix}, \mathbf{u}, \begin{bmatrix}\mathbf{S}_1 & ; & \mathbf{S}_2 & ; & \mathbf{bS}_3 & ; & -\mathbf{ES}_3\end{bmatrix})$
$\mathbf{c}_0 \leftarrow H_0(\mathbf{F}, \mathbf{T}_0, \mathbf{A}, \bar{\alpha}, otsvk)$
$\mathbf{Z}_0 \leftarrow \mathsf{P}_1(ppar, \begin{bmatrix}\mathbf{S}_1 & ; & \mathbf{S}_2 & ; & \mathbf{bS}_3 & ; & -\mathbf{ES}_3\end{bmatrix}, \mathbf{c}_0, \mathbf{Y}_0)$
$\Pi_0 \leftarrow (\mathbf{T}_0, \mathbf{c}_0, \mathbf{Z}_0)$
$(t, \pi) \leftarrow \mathsf{Enc}(opk, (\begin{bmatrix}\mathbf{G}^\mathsf{T} & \mathbf{F}^\mathsf{T} & \mathbb{I}_m\end{bmatrix}, -\mathbf{C}^\mathsf{T}), (\mathbf{m}, \begin{bmatrix}-\mathbf{b} & ; & \mathbf{E}^\mathsf{T}\end{bmatrix}, \mathbf{1}), otsvk)$
$ots \leftarrow \mathsf{OTSSign}(otssk, (\mathbf{A}, \mathbf{B}, \mathbf{F}, \mathbf{u}, \Pi_0, t, \pi, \mu))$
Return $sig = (\mathbf{F}, \Pi_0, t, \pi, otsvk, ots)$.

Figure 5.7. Signing algorithm

therefore create a relaxed NIZK proof Π_0 for the relation

$$\begin{aligned}\mathscr{R}_0 &= \left\{((\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{F} & \mathbf{1}\end{bmatrix}, \mathbf{u}), (\mathbf{S}_0, \mathbf{1})) \ : \ \begin{matrix}\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{F} & \mathbf{1}\end{bmatrix}\mathbf{S}_0 = \mathbf{u} \bmod q \\ \wedge\ \|\mathbf{S}_0\| \le N_0\end{matrix}\right\} \\ \bar{\mathscr{R}}_0 &= \left\{((\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{F} & \mathbf{1}\end{bmatrix}, \mathbf{u}), (\bar{\mathbf{S}}_0, \bar{\mathbf{c}})) \ : \ \begin{matrix}\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{F} & \mathbf{1}\end{bmatrix}\bar{\mathbf{S}}_0 = \bar{\mathbf{c}}\mathbf{u} \bmod q \\ \wedge\ \bar{\mathbf{c}} \in \bar{\mathscr{C}}_0\ \wedge\ \|\bar{\mathbf{S}}_0\| \le \bar{N}_0\end{matrix}\right\}\end{aligned} \tag{5.1}$$

running the prover $(\mathsf{P}_0, \mathsf{P}_1)$ from Section 3.4, where it feeds P_1 a challenge $\mathbf{c}_0$ sampled using the hash function $H_0 : \{0,1\}^* \to \mathscr{C}_0$ including *otsvk* among the inputs of the Fiat-Shamir hash H_0.

Next, recall that $\begin{bmatrix}\mathbf{G}^\mathsf{T} & \mathbf{F}^\mathsf{T} & \mathbb{I}_m\end{bmatrix}\begin{bmatrix}\mathbf{m} & ; & -\mathbf{b} & ; & \mathbf{E}^\mathsf{T}\end{bmatrix} = -\mathbf{C}^\mathsf{T} \bmod q$. Setting $\mathbf{T}_{\mathrm{ve}} := \begin{bmatrix}-\mathbf{b} & ; & \mathbf{E}^\mathsf{T}\end{bmatrix}$ the prover can therefore use the verifiable encryption scheme to encrypt a witness of the languages with relations

$$\begin{aligned}\mathscr{R}_{\mathrm{ve}} &= \left\{\begin{matrix}((\begin{bmatrix}\mathbf{G}^\mathsf{T} & \mathbf{F}^\mathsf{T} & \mathbb{I}_m\end{bmatrix}, -\mathbf{C}^\mathsf{T}), (\mathbf{m}, \mathbf{T}_{\mathrm{ve}}, \mathbf{1})) \in (R_q^{m\times(m+2)} \times R_q^m) \times (\mathscr{U} \times R_q^{m+1} \times \{\mathbf{1}\}) \\ : \quad \begin{bmatrix}\mathbf{G}^\mathsf{T} & \mathbf{F}^\mathsf{T} & \mathbb{I}_m\end{bmatrix}\begin{bmatrix}\mathbf{m} \\ \mathbf{T}_{\mathrm{ve}}\end{bmatrix} = -\mathbf{C}^\mathsf{T} \bmod q\ \wedge\ \|\mathbf{T}_{\mathrm{ve}}\| \le N_{\mathrm{ve}}\end{matrix}\right\} \\ \bar{\mathscr{R}}_{\mathrm{ve}} &= \left\{\begin{matrix}((\begin{bmatrix}\mathbf{G}^\mathsf{T} & \mathbf{F}^\mathsf{T} & \mathbb{I}_m\end{bmatrix}, -\mathbf{C}^\mathsf{T}), (\bar{\mathbf{m}}, \bar{\mathbf{T}}_{\mathrm{ve}}, \bar{\mathbf{c}})) \in (R_q^{m\times(m+2)} \times R_q^m) \times (\bar{\mathscr{U}} \times R_q^{m+1} \times \bar{\mathscr{C}}_{\mathrm{ve}}) \\ : \quad \begin{bmatrix}\mathbf{G}^\mathsf{T} & \mathbf{F}^\mathsf{T} & \mathbb{I}_m\end{bmatrix}\begin{bmatrix}\bar{\mathbf{m}} \\ \bar{\mathbf{T}}_{\mathrm{ve}}\end{bmatrix} = -\bar{\mathbf{c}}\mathbf{C}^\mathsf{T} \bmod q\ \wedge\ \|\bar{\mathbf{T}}_{\mathrm{ve}}\| \le \bar{N}_{\mathrm{ve}}\end{matrix}\right\}\end{aligned}$$

The user runs the encryption algorithm $\mathsf{Enc}(opk, x, w, otsvk)$ with language member $x = (\begin{bmatrix}\mathbf{G}^\mathsf{T} & \mathbf{F}^\mathsf{T} & \mathbb{I}_m\end{bmatrix}, -\mathbf{C}^\mathsf{T})$, witness $w = (\mathbf{m}, \begin{bmatrix}-\mathbf{b} & ; & \mathbf{E}^\mathsf{T}\end{bmatrix}, \mathbf{1})$, and the verification key *otsvk* as the encryption label, to generate a ciphertext $t = (\mathbf{v}_1, \mathbf{w}_1, \mathbf{v}_2, \mathbf{w}_2)$ and proof $\pi = (\Pi_1, \Pi_2)$. The user then computes

the one-time signature $ots \leftarrow \mathsf{OTSSign}(otssk, (\mathbf{A}, \mathbf{B}, \mathbf{F}, \mathbf{u}, \Pi_0, t, \pi, \mu))$ and returns the group signature $sig = (\mathbf{F}, \Pi_0, t, \pi, otsvk, ots)$.

Verification Algorithm. The verifier checks the one-time signature by running $\mathsf{OTSVf}(otsvk, (\mathbf{A}, \mathbf{B}, \mathbf{F}, \mathbf{u}, \Pi_0, t, \pi, \mu), ots)$, checks the NIZK proof Π_0 running the verifier V_1 in the group signature $sig = (\mathbf{F}, \Pi_0, t, \pi)$, making sure that *otsvk* is included in the Fiat-Shamir hash, and checks the encryption proof by running $\mathsf{EVerify}(opk, x, t, \pi, otsvk)$ with $x = (\begin{bmatrix}\mathbf{G}^\intercal & \mathbf{F}^\intercal & \mathbb{I}_m\end{bmatrix}, -\mathbf{C}^\intercal)$ and with *otsvk* as the encryption label. If all tests succeed then it outputs 1, else it outputs 0.

Opening Algorithm. The opener first verifies the group signature by running the $\mathsf{GVerify}$ algorithm above. If it is invalid, then the opener returns $\perp$, else it decrypts $\mathbf{m} \leftarrow \mathsf{Dec}(esk, x, t, \pi, otsvk)$ with x as above and returns $id = \mathbf{m}$.

We now proceed to prove that the scheme is correct.

Theorem 5.11 (Correctness)**.** *If the parameters are generated as in the parameters generation, the group signature described above is correct.*

Sketch of the proof of Theorem 5.11. The proof consists of 4 steps

- proving that it is possible for the group manager to produce a signature on a user identity
- proving that a user can produce a signature through the relaxed Σ-protocol and the partially verifiable encryption
- proving that verification accepts honestly generated signatures
- proving that the decryption of the ciphertext contained in a honestly generated signature outputs the identity of the signer.

The first point follows from the fact that the group manager always signs a message contained in a subset of $R_3^{(2^{16})} \times \{0,1\}^*$. The parameters of the signature are correct and satisfy the bounds in Section 3.6.3 because they are generated through the parameter generation algorithm $\mathsf{SParGen}$ of the relaxed signature.

Producing a signature is always possible. The less straightforward step is to make sure that such signature verifies. To guarantee that the verification of Π_0 succeeds, it should hold $N_0 \geq \bar{N}_0$ (the linear relation trivially holds). To set the parameter N_0 we have to bound the norm of the witness. A honest $\mathbf{S}_0$ is generated as $\mathbf{S}_0 = \begin{bmatrix}\mathbf{S}_1 & \mathbf{S}_2 & \mathbf{b}\mathbf{S}_3 & -\mathbf{E}\mathbf{S}_3\end{bmatrix}$, where the vector $\mathbf{S} = \begin{bmatrix}\mathbf{S}_1 & \mathbf{S}_2 & \mathbf{S}_3\end{bmatrix} \in$

$R_q^{1\times(2+2m)}$ is sampled from a Gaussian with standard deviation σ. Hence it each of its components has norm bounded by $1.05\sigma\sqrt{n}$. Moreover, using the bounds in Lemma 2.12, it holds $\|\mathbf{bS}_3\| \leq 8\sigma n\sqrt{m}$ and $\|-\mathbf{ES}_3\| \leq \sqrt{\sum_{i=1}^{m}\|\mathbf{E}_i\mathbf{S}_{3,i}\|_2^2} \leq 8\sigma n\sqrt{m}$. Hence we can set the bound N_0 to be:

$$N_0 = \sqrt{(2+m)(1.05\sigma\sqrt{n})^2 + m(8\sigma n)^2 + m(8\sigma n)^2}.$$

Then, from rejection sampling we know that the noise vector is sampled from a Gaussian with standard deviation $\sigma_0 = 12T_0$, where $T_0 \geq N_0$ as it is obtained from N_0 as a bound on the norm of $\mathbf{cS}_0$ for $\mathbf{c} \in \mathscr{C}_0$, and $\bar{N}_0 = 2.1\sigma_0\sqrt{n(3+2m)}$. Therefore, $\bar{N}_0 \geq N_0$, and the verification of Π_0 outputs 1.

Finally, verification and decryption of the ciphertext succeed with high probability as the parameters are set according to the specifications in Section 3.8.4. We only compute the bound N_{ve}, as this is not set in Section 3.8.4 because it depends on the choice of the norm. The value N_{ve} in $\mathscr{R}_{\text{ve}}$ bounds the norm of a vector of polynomials with coefficients in $\{0,1\}$ one of which is in $R_3^{(16)}$, hence $N_{\text{ve}} := \sqrt{256 + n(m^2+1)}$. □

5.3.3 Security of the Group Signature

We now prove anonymity and traceability of the scheme according to the definitions in Section 5.1.1.

Theorem 5.12 (Traceability)**.** *Let ϵ_{ZK} be the probability of breaking the zero-knowledge property of $\mathsf{r}\Sigma$ and ϵ_{OTS} be the probability of breaking the strong unforgeability of the OTS.*

Assume A is an adversary that runs in time t_A, makes q_H queries to the random oracle and queries the oracle $\mathcal{O}_{\mathsf{UKg}}$ at most q_U times, and breaks the traceability of the group signature scheme with probability ϵ_A.

There exists an algorithm B that breaks the unforgeability of the relaxed signature in time $t_{\mathsf{B}} = t_{\mathsf{A}} \cdot 72q_H/\epsilon_{\mathsf{A}} \cdot \ln(24/\epsilon_{\mathsf{A}}) + poly(\lambda)$ with probability $\epsilon_{\mathsf{B}} \geq (1-\epsilon_{ZK}-2^{-\omega(\log(\lambda))})\epsilon_{\mathsf{A}}/8 - \epsilon_{OTS}/8$ after asking q_U queries to the signing oracle in the Random Oracle Model.

Proof. Given a traceability adversary A, we construct a simulator B that simulates the traceability experiment in Figure 5.1 and the oracles to exploit A to forge a signature. The simulator is described in details in Figure 5.8.

In the following we give a high level description of it. The simulator B has access to an oracle $\mathcal{O}_{\text{s}}$ that, when prompted the first time, outputs the parameters

of the signature scheme *spar* and the verification key *svk*. Then, whenever it receives in input $(\mathbf{m}, \alpha)$, it outputs a signature on it.

On input $\left[\mathbf{A} \quad \mathbf{B} \quad \mathbf{C} \quad \mathbf{1}\right]$, B samples a random $\bar{\alpha}$ and honestly generates an opening key pair (opk, osk) according to the OKg algorithm. Remark that the distribution of the public keys generated by B is exactly the same of the honestly generated keys. It then runs A, simulating its oracle queries as follows:

- $\mathcal{O}_{\mathsf{UKg}}(id)$: B queries its oracle $\mathcal{O}_{\mathrm{s}}(id, \bar{\alpha})$ to obtain a small vector $\left[\mathbf{S}\right]$ such that $\left[\mathbf{A} \quad \mathbf{B} \quad \mathbf{C}+\mathbf{mG}\right] = \mathbf{u} \bmod q$ (where $\mathbf{m} = id$ and $\mathbf{u} = H(\bar{\alpha})$). It then register the query in a list $\mathcal{Q}_{\mathsf{UKg}}$ and returns $usk = \mathbf{S}$ to A.

- $\mathcal{O}_{\mathsf{GSign}}(id, \mu)$: B generates a OTS key pair $(otssk, otsvk) \leftarrow \mathsf{OTSGen}(1^{\lambda})$, chooses random $\mathbf{E} \xleftarrow{\$} R_3^{1\times m}$ and $\mathbf{b} \xleftarrow{\$} R_3$, and computes $\mathbf{F} = \mathbf{b}^{-1}(\mathbf{C}+\mathbf{mG}+\mathbf{E})$ for $\mathbf{m} = id$. It then simulates the zero-knowledge proof of Π_0 by programming the random oracle, including *otsvk* in the Fiat-Shamir hash. It then honestly computes the verifiable encryption and the one-time signature and returns the resulting group signature to A.

- Random oracle queries are responded through lazy sampling (responding coherently to the same query).

The simulated oracles can be distinguished from the honestly executed ones with probability $\epsilon_{ZK} + 2^{-\omega(\log(\lambda))}$, that is, either if A can distinguish the simulator from the honest prover (i.e., A break the zero-knowledge property) or if $\mathcal{O}_{\mathsf{GSign}}$ aborts (which happens with probability $2^{-\omega(\log(\lambda))}$ thanks to non-triviality).

Eventually, A outputs its forgery (μ^*, sig^*). Let $id^* \leftarrow \mathsf{GOpen}(\mu^*, sig^*, osk)$ and let $otsvk^*$ be the OTS verification key in sig^*. If the forgery is valid and $otsvk^*$ was returned as part of a $\mathcal{O}_{\mathsf{GSign}}$ response by B before, then A must have forged the one-time signature scheme, because sig^* or μ^* must be different from that specific oracle query and response.

If $otsvk^*$ is different, then the hash queries involved in Π_0^* and Π_1^* in sig^* cannot have been involved in previous $\mathcal{O}_{\mathsf{GSign}}$ responses, because both hash queries contain $otsvk^*$. By the Generalized Forking Lemma (Lemma 2.6), B can then rewind A on the hash queries involved in Π_0^* and Π_1^* in sig^*, respectively, to obtain valid responses to two different challenges. From these, by the special soundness of the relaxed Σ-protocol and the special soundness of the verifiable encryption scheme, B can extract[4] from these witnesses $(\bar{\mathbf{T}} = (\bar{\mathbf{T}}_1, \bar{\mathbf{T}}_2, \bar{\mathbf{T}}_3, \bar{\mathbf{t}}_4), \bar{\mathbf{c}}_0)$

[4]The extraction of the signature is represented in Figure 5.8 by the extractor E, which is obtained combining the extractor of the relaxed Σ-protocol and the extractor of the partially verifiable encryption.

and $(\bar{\mathbf{m}}, \begin{bmatrix}\bar{\mathbf{b}} & ; & \bar{\mathbf{E}}\end{bmatrix}, \bar{\mathbf{c}}_1)$ such that

$$\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{F} & \mathbf{1}\end{bmatrix}\begin{bmatrix}\bar{\mathbf{T}}_1\\ \bar{\mathbf{T}}_2\\ \bar{\mathbf{T}}_3\end{bmatrix} = \bar{\mathbf{c}}_0\mathbf{u} \bmod q \quad \text{and} \quad \begin{bmatrix}\mathbf{G}^\top & \mathbf{F}^\top & \mathbb{I}_m\end{bmatrix}\begin{bmatrix}\bar{\mathbf{m}}\\ \bar{\mathbf{b}}\\ \bar{\mathbf{E}}\end{bmatrix} = -\bar{\mathbf{c}}_1\mathbf{C}^\top \bmod q\ .$$

and such that $id^* = \bar{\mathbf{m}}/\bar{\mathbf{c}}_1 \bmod q$. From these, one can verify that

$$\begin{bmatrix}\mathbf{A} & \mathbf{B} & \bar{\mathbf{c}}_1\mathbf{C} - \bar{\mathbf{m}}\mathbf{G} & \mathbf{1}\end{bmatrix}\begin{bmatrix}\bar{\mathbf{b}}\bar{\mathbf{T}}_1\\ \bar{\mathbf{b}}\bar{\mathbf{T}}_2\\ \bar{\mathbf{T}}_3\\ \bar{\mathbf{t}}_4 - \bar{\mathbf{E}}\bar{\mathbf{T}}_3\end{bmatrix} = \bar{\mathbf{b}}\bar{\mathbf{c}}_0\mathbf{u} \bmod q\ .$$

Also, we know that B never queried $id^* = \bar{\mathbf{m}}/\bar{\mathbf{c}}_1$ to its $\mathcal{O}_s$ oracle, because it would only have done so if A would have submitted id^* to its $\mathcal{O}_{\mathsf{UKg}}$ oracle, which would invalidate the forgery. Let $\bar{\mathbf{S}} := \begin{bmatrix}\bar{\mathbf{b}}\bar{\mathbf{T}}_1 & ; & \bar{\mathbf{b}}\bar{\mathbf{T}}_2 & ; & \bar{\mathbf{T}}_3 & ; & \bar{\mathbf{t}}_4 - \bar{\mathbf{E}}\bar{\mathbf{T}}_3\end{bmatrix}$ and $\bar{\mathbf{c}}_2 := \bar{\mathbf{b}}\bar{\mathbf{c}}_0$. The vector $\bar{\mathbf{S}}$ has norm in the order of the norm of $\bar{\mathbf{E}}^T\bar{\mathbf{T}}_3$ and applying Lemma 2.12 we have $\|\bar{\mathbf{E}}^T\bar{\mathbf{T}}_3\| \leq 2.1\sigma_h\sqrt{n} \cdot 2.1\sigma_s\sqrt{n} \cdot \sqrt{nm}$, (where the inequality holds as by Lemma 2.16 we can bound the product of the infinity norms as $\|\bar{\mathbf{E}}^T\|_\infty\|\bar{\mathbf{T}}_3\|_\infty n < 16\sigma_h \cdot 16\sigma_s n$ that is less than $q/2$ for our choice of parameters), and, applying again Lemma 2.12, $\|\bar{\mathbf{c}}_2\| = \|\bar{\mathbf{b}}\bar{\mathbf{c}}_0\| \leq \|\bar{\mathbf{b}}\|\|\bar{\mathbf{c}}_0\|\sqrt{n} \leq 12 \cdot 2.1\sigma_h n =: C'$. Hence, we have that by outputting $\bar{\mathbf{m}}, \bar{\mathbf{c}}_1, \bar{\mathbf{c}}_2, \bar{\mathbf{S}}$, algorithm A breaks the unforgeability of the relaxed signature scheme with $N' = 2^{\lceil \log_2[2(2.1\sigma_d\sqrt{n} \cdot 2.1\sigma_h\sqrt{n} \cdot \sqrt{nm})]\rceil}$, C' and $\mathscr{C} = \mathscr{C}_1$.

Finally, we compute the runtime and success probability of B.

The runtime t_B of B is essentially the runtime of the algorithm GF and can be obtained using Lemma 2.6: $t_B = t_A \cdot 72q_H/\epsilon_A \cdot \ln(24/\epsilon_A) + poly(\lambda)$, where t_A and ϵ_A are the runtime and success probability of A.

The success probability ϵ_B of B is

$$\begin{aligned}
\epsilon_B &= \Pr[\mathsf{GF}\text{ succeeds } \wedge \text{ A has not forged the OTS}]\\
&= \Pr[\mathsf{GF}\text{ succeeds } \wedge \text{ A does not abort}] - \Pr[\mathsf{GF}\text{ succeeds } \wedge \text{ A has forged the OTS}]\\
&\geq \epsilon_A/8(1 - \Pr(\text{A aborts})) - \epsilon_{OTS}/8\\
&= (1 - \epsilon_{ZK} - 2^{-\omega(\log(\lambda))})\epsilon_A/8 - \epsilon_{OTS}/8\ ,
\end{aligned}$$

where ϵ_{ZK} is the probability that A breaks the zero-knowledge property of the relaxed Σ-protocol and ϵ_{OTS} is the probability that A breaks the strong unforgeability of the OTS. □

Theorem 5.13 (Anonymity)**.** *Our group signature scheme is anonymous in the Random Oracle Model if the NIZK proof is zero-knowledge and if the relaxed partially verifiable encryption scheme of Section 3.8.4 is chosen-ciphertext simulatable.*

Proof. We prove the theorem through a sequence of games. Let ϵ_{A} be the advantage of A in winning the anonymity experiment (cf. Figure 5.2). Let B an algorithm that plays as the challenger in the experiment. Let $\mathbf{Game_i}$ the probability that A wins the (possibly simulated) experiment at the end of the i-th game.

Game 0. In Game 0 B runs the real anonymity experiment. Then it holds,

$$\mathsf{Pr}[\mathbf{Game_0}] = \mathsf{Pr}\left[b = b' \,:\, b \xleftarrow{\$} \{0,1\},\ b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{\mathsf{anon}-b}(1^\lambda)\right] .$$

Game 1. In Game 1 B simulates the oracle $\mathcal{O}_C^b$ as follows. It does everything honestly but the generation of the ciphertext, which is done using the simulator Σ_{pve} (cf. Figure 3.11) of the chosen-ciphertext simulatability property of the partial verifiable encryption scheme instead of a real encryption as part of the challenge group signature sig^*. Note that thereby the ciphertext is independent of the encrypted witness. Any adversary distinguishing this game from the previous one can be used to win the chosen-ciphertext simulatability experiment (Figure 3.10). To see why the reduction works, observe that all group signatures that A submits to its opening oracle must use a different OTS verification key than $otsvk^*$ in sig^*, lest A broke the OTS unforgeability. Therefore, a simulator B_1 can always use the decryption oracle to answer A's opening queries. Then, B_1 returns 0 if A wins the experiment and 1 otherwise. Then a simulator B can break either the chosen-ciphertext simulatability of the partial verifiable encryption or the strong unforgeability of the OTS with advantage:

$$\begin{aligned}
\epsilon_1 &= \epsilon_{CCS} + \epsilon_{OTS} \\
&= \left| \mathsf{Pr}\left[b' = b \,:\, b \xleftarrow{\$} \{0,1\},\ b' \leftarrow \mathsf{Exp}_{\mathsf{B}_1}^{\mathsf{ccas}-b}(1^\lambda)\right] - \frac{1}{2} \right| \\
&= \frac{1}{2} \left| \mathsf{Pr}\left[b' = 0 \,:\, b' \leftarrow \mathsf{Exp}_{\mathsf{B}_1}^{\mathsf{ccas}-0}(1^\lambda)\right] + \mathsf{Pr}\left[b' = 1 \,:\, b' \leftarrow \mathsf{Exp}_{\mathsf{B}_1}^{\mathsf{ccas}-1}(1^\lambda)\right] - 1 \right| \\
&= \frac{1}{2} \left| \mathsf{Pr}\left[b' = 0 \,:\, b' \leftarrow \mathsf{Exp}_{\mathsf{B}_1}^{\mathsf{ccas}-0}(1^\lambda)\right] - \mathsf{Pr}\left[b' = 0 \,:\, b' \leftarrow \mathsf{Exp}_{\mathsf{B}_1}^{\mathsf{ccas}-1}(1^\lambda)\right] \right| \\
&= \frac{1}{2} |\mathsf{Pr}[\mathbf{Game_0}] - \mathsf{Pr}[\mathbf{Game_1}]| .
\end{aligned}$$

Game 2. In Game 2 the relaxed NIZK is simulated too. As the simulator now simply samples a uniform $\mathbf{F} \xleftarrow{\$} R_q^{1\times m}$ in sig^*, instead of one that hides the identity $\mathbf{m} = id_b$, the game is independent of the bit b' and A can win it with probability exactly $\frac{1}{2}$, i.e., $\mathsf{Pr}[\mathbf{Game_2}] = \frac{1}{2}$. Moreover, distinguishing this game from the previous one implies to break the zero-knowledge property of the relaxed NIZK.

Indeed, if an adversary A was able to distinguish Game 2 from Game 1, then a simulator B_2 could exploit it to distinguish whether a proof was generated by a simulator or honestly. Remark that B_2 can now generate the encryption keys and implement the opening oracle honestly, as the only ciphertext that is simulated is the one contained in sig^*, and the unforgeability of the OTS prevents that to be queried to the opening oracle. Whenever it is necessary to generate a proof using (H_0, P), B_2 gets it from the proof oracle (which implements either the honest prover or the simulator $(\Sigma_1(\cdot), \Sigma_2(\cdot,\cdot)))$. Then, B_2 returns 0 if A wins the experiment and 1 otherwise. Again, the advantage of B_2 in distinguishing or breaking the strong unforgeability of the OTS can be computed as before to be

$$\begin{aligned}\epsilon_2 &= \epsilon_{ZK} + \epsilon_{OTS} \\ &= \frac{1}{2}\left|\mathsf{Pr}\left[0 \leftarrow \mathsf{B}_1^{H_0(\cdot),\mathsf{P}^{H_0}(\cdot,\cdot)}(1^\lambda)\right] - \mathsf{Pr}\left[0 \leftarrow \mathsf{B}_1^{\Sigma_1(\cdot),\Sigma_2(\cdot,\cdot)}(1^\lambda)\right]\right| \\ &= \frac{1}{2}\left|\mathsf{Pr}[\mathbf{Game_1}] - \mathsf{Pr}[\mathbf{Game_2}]\right| .\end{aligned}$$

From the previous considerations it yields that the advantage of A is

$$\begin{aligned}\epsilon_{\mathsf{A}} &= \left|\mathsf{Pr}\left[b = b' \,:\, b \xleftarrow{\$} \{0,1\},\ b' \leftarrow \mathsf{Exp}_{\mathsf{A}}^{\mathsf{anon}-b}(1^\lambda)\right] - \frac{1}{2}\right| \\ &= \left|\mathsf{Pr}[\mathbf{Game_0}] - \mathsf{Pr}[\mathbf{Game_2}]\right| \\ &= \left|\mathsf{Pr}[\mathbf{Game_0}] - \mathsf{Pr}[\mathbf{Game_1}] + \mathsf{Pr}[\mathbf{Game_1}] - \mathsf{Pr}[\mathbf{Game_2}]\right| \\ &= 2\epsilon_{CCS} + 2\epsilon_{ZK} + 4\epsilon_{OTS} .\end{aligned}$$

□

5.3.4 Parameters and Storage Requirements

We show how to compute parameters for a security level of 128 bits where security is based on complexity leveraging and standard RSIS/RLWE assumptions. Let the polynomial ring be $R_q := \mathbb{Z}_q[x]/\langle x^n + 1\rangle$. We choose $n = 2048$, a prime q such that $\log q \leq 2^{116}$, $m = 22$, and users' identities in $R_3^{(16)}$, so that Assumption 3 is based on a hard instance of RSIS. To guarantee this, we compute the Hermite root factor δ_s of an instance of $\mathrm{RSIS}_{(3+m),q,\beta_s}$ in Theorem 5.10. The value of δ_s has to be smaller than 1.0014, that is the value of the Hermite root factor guaranteeing 430 bits of security (this number includes the compensation for the tightness loss in Theorem 5.10). The computation of this bound is done following the approach by Alkim et al. [2016]. Finally, we set p to be a prime such

that $\log p \leq 2^{50}$ to satisfy all the inequalities that guarantee correctness of the decryption and simulation soundness of the verifiable encryption scheme.

The secret key of the group manager is generated sampling a matrix $\mathbf{X}$ in $R_q^{2\times m}$ where $m = 22$ from a discrete Gaussian with standard deviation $\sigma_t = 4$, hence it has infinity norm bounded by $8\sigma_t = 2^5$. Therefore, the secret key of the group manager can be stored in less than $2 \cdot m \cdot n \cdot \log(8\sigma_t) = 2 \cdot 22 \cdot 2^{11} \cdot 5 \leq 56.32$ kB.

The public key of the group manager is composed by vectors $\mathbf{A} \in R_q^{1\times 2}$, $\mathbf{B}, \mathbf{C} \in R_q^{1\times m}$ and $\mathbf{u} \in R_q$, i.e. it can be stored in $(2+2m+1)n\log(q) = 47 \cdot 2^{11} \cdot 116 \leq 1.396$ MB.

The user secret key is an element of $R_3^{(16)}$, i.e. a polynomial with 16 nonzero coefficients in $\{\pm 1\}$, and by a signature on it. Such signature is a vector with $2 + 2m = 46$ components distributed as a Gaussian with standard deviation $\sigma = q^{1/m}\frac{\sigma_t}{\sqrt{\pi}}\sqrt{n} \cdot (\sqrt{2}+\sqrt{m}+\log(n)) = 2^{116/22}\frac{4}{\sqrt{3.1415}}\sqrt{2048}(\sqrt{2}+\sqrt{22}+11) = 6.75335 \cdot 10^4$. Hence the signature has infinity norm bounded by 8σ and both signature and the user id can be stored in $16 \cdot 2 + 46 \cdot n \cdot \log(8\sigma) = 32 + 46 \cdot 2^{11} \cdot (3 + log\sigma) \leq 224.26$ kB.

The opener keys are the encryption and decryption keys of the verifiable encryption: the public key is composed by three random polynomials in R_q, while the secret key is a polynomial in R_3 (cf. Section 3.8.4). Hence, the opener's public key can be stored in $3 \cdot n \cdot \log q \leq 3 \cdot 2048 \cdot 116 = 89.088$ kB, and the secret key in $n \cdot 2 = 512$ B.

A signature produced by a group member is composed by a vector $\mathbf{F} \in R_q^{1\times m}$, an encryption of the identity $\mathbf{v}_1, \mathbf{w}_1, \mathbf{v}_2, \mathbf{w}_2 \in R_q$ and by the transcripts of the three relaxed NIZKs Π_0, Π_1, Π_2. Each transcript is composed by a challenge $\mathbf{c}_i$ and a response vector $\mathbf{Z}_i$, $i = 0, 1, 2$. The length of each challenge depends on the challenge set it is sampled from (i.e., either $\mathscr{C}_0 = \mathscr{C}_1 = \{\mathbf{c} \in R_3 \ : \ \|\mathbf{c}\|_1 \leq 32\}$ or $\mathscr{C}_2 = R_3^{(16)}$). The length of the responses are determined by the standard deviations σ_i used in rejection sampling. Indeed, the infinity norm of the response $\mathbf{Z}_i$ is bounded by $8\sigma_i$ (cf. Section 3.4). Denoting by $\mathbf{V}_i$ the secret vector that the prover is proving knowledge of, the standard deviations can be derived by a bound T_i on the norm of $\mathbf{V}_i\mathbf{c}_i$, i.e. $\sigma_i = 12T_i$. Hence, using Lemma 2.12 we obtain the following bounds:

$$
\begin{aligned}
\|\mathbf{c}_0\mathbf{V}_0\| &\leq \max\|\mathbf{c}_0\|_1\sqrt{n(\|\mathbf{S}_1\|_\infty^2 + m\|\mathbf{S}_2\|_\infty^2 + m\|\mathbf{bS}_{3,i}\|_\infty^2 + \|-\mathbf{ES}_3\|_\infty^2)} \\
&\leq 32 \cdot 8\sigma\sqrt{n(1+m+2n^2m^2)} =: T_0 \\
\|\mathbf{c}_1\mathbf{V}_1\| &\leq 32\sqrt{n(7+m)} =: T_1 \\
\|\mathbf{c}_2\mathbf{V}_2\| &\leq 16\sqrt{6n} =: T_2 \ .
\end{aligned}
$$

Hence, it follows that $\|\mathbf{Z}_0\|_\infty \leq 8\sigma_0 = 8 \cdot 12 \cdot 32 \cdot 8\sigma\sqrt{n(1+m+2n^2m^2)} =: N_0$, $\|\mathbf{Z}_1\|_\infty \leq 8\sigma_1 = 8 \cdot 12 \cdot 32\sqrt{n(7+m)} =: N_1$ and $\|\mathbf{Z}_2\|_\infty \leq 8\sigma_2 = 8 \cdot 12 \cdot 16\sqrt{6n} =: N_2$. Therefore, the transcript of Π_0 can be stored in $32 \cdot \lceil \log 3 \rceil + n(3+2m)\log N_0$, the transcript of Π_1 in $32 \cdot \lceil \log 3 \rceil + n(m+7)\log N_1$ and the transcript of Π_2 in $11 \cdot (16 \cdot \lceil \log 3 \rceil + 6n \log N_2)$, as the last one has to be repeated 11 times to have negligible soundness error. Hence a signature takes at most $32 \cdot \lceil \log 3 \rceil + n(3+2m)\log N_0 + 32 \cdot \lceil \log 3 \rceil + n(m+7)\log N_1 + 11 \cdot (16 \cdot \lceil \log 3 \rceil + 6n \log N_2) + (m+4)n \log q \leq 1.7181$ MB.

5.4 New Building Blocks for better Group Signature

The final construction is the most recent, and will appear at PQCrypto 2020.

From the previous constructions we concluded that the large part of the blowup in the signature dimension is due to the relaxed NIZK proof. Therefore, we worked on improving that building block, in particular to get rid of the relaxed special soundness. Eventually, we decided to adapt an already existing SNARK called Aurora [Ben-Sasson et al., 2019] to prove relations over lattices. Such proof allows to easily and efficiently encode the linear-algebraic statements that arise in lattice schemes and to side-step the issue of "relaxed extractors", meaning extractors that only recover a witness for a larger relation than the one for which completeness is guaranteed. Applying our approach to the example use case of dynamic group signatures yields the first efficient lattice-based group signature that protects users against corrupted issuers, and that is far more efficient than the state of the art, with signature sizes of less than 300 KB for the comparably secure version of the scheme. To obtain our argument size estimates for proof of knowledge of RLWE secret, we implemented the NIZK using libiop.

Regarding the choice of the signature, we also decided to combine it with the ideal lattice version of the original signature by Boyen, which had not been proved secure until now.

In this section we then first introduce the two new building blocks, then we present the group signature.

Remark 15. Differently from all the previous constructions, in this section we represent $\mathbb{Z}_q$ as the ring of elements $\{0, \ldots, q-1\}$.

5.4.1 Boyen's signature on ideal lattices

In this section we describe the variant of Boyen's signature [Boyen, 2010] by Micciancio and Peikert [2012], adapted to have security based on hardness assump-

tions on ideal lattices. Such variant has been claimed to be secure since long time, but, to the best of our knowledge, this is the first time in which a security proof is given explicitly. In particular, we prove the signature secure when defined over the $(2n)$-th cyclotomic ring.

We restate the trapdoors theorems, as in this work we chose a different parameter setup.

Theorem 5.14 (Trapdoor generation, from [Micciancio and Peikert, 2012])**.** *Let R_q be a power of 2 cyclotomic ring and set parameters $m = 2$, $k = \lceil \log q \rceil$, $\bar{m} = m+k$. There exists an algorithm* GenTrap *that outputs a vector $\bar{\mathbf{A}} \in R_q^{1\times \bar{m}}$ and a trapdoor $\mathbf{R} \in R_q^{m\times k}$ with tag $\mathbf{h} \in R_q$ such that:*

- $\bar{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{AR} + \mathbf{hG} \end{bmatrix}$*, where* $\mathbf{G}$ *is the* gadget matrix*,* $\mathbf{G} = \begin{bmatrix} 1 & 2 & 4 & \dots & 2^{k-1} \end{bmatrix}$ *and* $\mathbf{A} = \begin{bmatrix} \mathbf{a} & \mathbf{1} \end{bmatrix} \in R_q^{1\times 2}$*,* $\mathbf{a} \xleftarrow{\$} R_q$*.*
- $\mathbf{R}$ *is distributed as a Gaussian $\mathcal{D}_{R,s}^{2\times k}$ for some $s = \alpha q$, where $\alpha > 0$ is a RLWE error term, $\alpha q > \omega(\sqrt{\log n})$ (cf. [Lyubashevsky et al., 2013, Theorem 2.22]).*
- $\mathbf{h}$ *is an invertible element in R_q.*
- $\bar{\mathbf{A}}$ *is computationally pseudrandom (ignoring the component set to* $\mathbf{1}$*) under (decisional) RLWE$_D$ where $D = \mathcal{D}_{R,s}$.*

Genise and Micciancio [2018] give an optimal sampling algorithm for the previous trapdoor construction.

Theorem 5.15 (Gaussian sampler, adapted from [Micciancio and Peikert, 2012] and [Genise and Micciancio, 2018])**.** *Let R_q, m, k, $\bar{m}$ be as in Theorem 5.14,* $\mathbf{G}$ *be the* gadget matrix $\mathbf{G} = \begin{bmatrix} 1 & 2 & 4 & \dots & 2^{k-1} \end{bmatrix}$*, $\mathbf{A} \in R_q^{1\times m}$ and $\mathbf{R} \in R_q^{2\times k}$ be the output of GenTrap, and $\mathbf{B}$ a vector in $R_q^{1\times d}$ for some $d \geq 0$. Then, there is an algorithm that can sample from the distribution $\mathcal{D}^{\perp}_{[\mathbf{A}\ \mathbf{AR}+\mathbf{G}\ \mathbf{B}],\mathbf{u},s}$ for any $s = O(\sqrt{n \log q}) \cdot \omega(\sqrt{\log n})$ for any $\mathbf{u} \in R_q$ in time $\tilde{O}(n \log q)$ for the offline phase and $\tilde{O}(n^2)$ for the online phase.*

The original signature was proved existentially unforgeable against adaptive chosen-message attacks *eu-acma* under SIS. Micciancio and Peikert proved their variant to be strongly unforgeable against static chosen-message attack (*su-scma*) under SIS with a tighter reduction, and then made it strongly unforgeable against adaptive chosen-message attacks *su-acma* using chameleon hash functions [Shamir and Tauman, 2001]. For our purposes adaptive existential unforgeability is enough, so our aim is to prove the scheme *eu-acma* under RSIS combining the techniques used in the proofs of these two papers.

Parameters. $spar \leftarrow \mathsf{SParGen}(1^\lambda)$

Let $\mathbf{f}$ be the $(2n)$-th cyclotomic polynomial, $\mathbf{f} = \mathbf{x}^n + 1$. Construct the polynomial rings $R = \mathbb{Z}[X]/\langle \mathbf{f} \rangle$ and $R_q = \mathbb{Z}_q[X]/\langle \mathbf{f} \rangle$. Let $k = \lceil \log_2 q \rceil$, $m = 2$, and $\bar{m} = m + k = 2 + \lceil \log q \rceil$ be the length of the public matrices, and ℓ be the length of the message. Let $s_{ssk} = \sqrt{\log(n^2)} + 1$ and $s_\sigma = \sqrt{n \log n} \cdot \sqrt{\log n^2}$ be the standard deviations of the distributions of the signing key and of the signature respectively (their values are determined following Theorem 5.14 and 5.15 respectively).

Key Generation. $(svk, ssk) \leftarrow \mathsf{SKeyGen}(spar)$

Run the algorithm $\mathsf{GenTrap}$ from Theorem 5.14 to get a vector $\begin{bmatrix}\mathbf{A} & \mathbf{B}\end{bmatrix} = \begin{bmatrix}\mathbf{A} & \mathbf{AR} + \mathbf{G}\end{bmatrix}$ and a trapdoor $\mathbf{R}$. The public key is composed by $\ell + 1$ random matrices $\mathbf{A}_0, \ldots, \mathbf{A}_\ell \xleftarrow{\$} R_q^{1\times k}$, a random vector $\mathbf{u} \xleftarrow{\$} R_q$ and the vector $\begin{bmatrix}\mathbf{A} & \mathbf{B}\end{bmatrix} \in R_q^{1\times \bar{m}}$. i.e., $svk = (\mathbf{A}, \mathbf{B}, \mathbf{A}_0, \ldots, \mathbf{A}_\ell, \mathbf{u})$, and the (secret) signing key is $ssk = \mathbf{R}$. Remark that the probability distribution of $\mathbf{R}$ is $\mathscr{D}_{R, s_{ssk}}^{2\times k}$.

Signing. $\sigma \leftarrow \mathsf{Sign}(\mu, ssk)$

To sign a message $\mu = (\mu_1, \ldots, \mu_\ell) \in \{0,1\}^\ell$, the signer constructs a message dependent public vector $\mathbf{A}_\mu = \begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{A}_0 + \sum_{i=1}^{\ell} (-1)^{\mu_i} \mathbf{A}_i\end{bmatrix}$ and then it samples a short vector $\mathbf{S} \in R_q^{\bar{m}+k}$ running the algorithm $\mathsf{SampleD}$ from Theorem 5.15 on input $(\mathbf{A}_\mu, \mathbf{u}, \mathbf{R})$. The algorithm outputs the signature $\sigma = \mathbf{S}$. Remark that the probability distribution of the signature $\mathbf{S}$ is $\mathscr{D}_{\mathbf{A}_\mu, \mathbf{u}, s_\sigma}^{\perp}$.

Verification. $\{0,1\} \leftarrow \mathsf{SVerify}(\sigma, \mu, svk)$

The verifier checks that the vector $\mathbf{S}$ has small norm, i.e., $\|\mathbf{S}\|_\infty \leq 8 s_\sigma$. Then, it constructs $\mathbf{A}_\mu = \begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{A}_0 + \sum_{i=1}^{\ell} (-1)^{\mu_i} \mathbf{A}_i\end{bmatrix}$ and checks that $\mathbf{S}$ satisfies the verification equation, i.e., $\mathbf{A}_\mu \mathbf{S} = \mathbf{u} \bmod q$.

Correctness follows from Theorem 5.14 and 5.15 and from Lemma 2.16. We prove the *eu-acma* security of the scheme under RSIS by proving that if there exists a PPT adversary A that can break the signature scheme we can construct an algorithm B that can solve RSIS exploiting A. The proof is obtained combining the message guessing technique in the proof of Theorem 25 in [Boyen, 2010] with the proof of Theorem 6.1 in [Micciancio and Peikert, 2012].

Theorem 5.16. *(eu-acma security) If there exists a PPT adversary* A *that can break the eu-acma security of the signature scheme* $(\mathsf{SParGen}, \mathsf{SKeyGen}, \mathsf{Sign}, \mathsf{SVerify})$ *in time* t_A *with probability* ϵ_A *asking* q_A *queries to the signing oracle, then there exists a PPT algorithm* B *that can solve* $RSIS_{\bar{m}+1,q,\beta}$ *for a large enough* $\beta = 8 s_\sigma + (\ell + 1) kn 8 s_\sigma$ *exploiting* A *in time* $t_\mathsf{B} \sim t_\mathsf{A}$ *with probability* $\epsilon_\mathsf{B} = \epsilon_\mathsf{A} \times (1 - \epsilon_{RLWE}) \times$

$\frac{1}{q}\left(1-\frac{q_{\mathsf{A}}}{q}\right)$ *or a PPT algorithm that solves* $\mathrm{RLWE}_{(\ell+1)k,\mathscr{U}(\mathscr{S}_1)}$ *with probability* ϵ_{A} *in time* t_{A}.

Proof. The algorithm B has access to a RSIS oracle that outputs a RSIS instance $\mathbf{A}_{RSIS}$ when prompted. To exploit A to solve RSIS, B has to plug in the RSIS instance in the verification key of the signature, and to exploit the forgery that A produces to build a solution for RSIS. The verification key generated by B should be indistinguishable from a honestly generated key. Moreover, B should also implement a signing oracle that, on input a message from A, outputs a signature on that message.

Upon receiving the instance $\mathbf{A}_{RSIS}$ from the RSIS oracle, B generates the public parameters for the signature scheme as it follows. First, it parses[5] $\mathbf{A}_{RSIS}$ as $\mathbf{A}_{RSIS} = \begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{u}\end{bmatrix} = \begin{bmatrix}\mathbf{a} & \mathbf{1} & \mathbf{B} & \mathbf{u}\end{bmatrix}$. Rearranging its components, this corresponds to the normal form of RSIS (cf. beginning of Section 4 in [Lyubashevsky et al., 2013]).

In the eu-acma scenario, the adversary chooses the message μ^* it will forge a signature for after receiving the public key of the scheme and having possibly queried the signing oracle. Hence, the game is as follows. The simulator B generates the verification key $(\mathbf{A}, \mathbf{B}, \mathbf{A}_0, \ldots, \mathbf{A}_\ell, \mathbf{u})$ from the RSIS instance $\mathbf{A}_{RSIS}$ as follows. First, it samples random $\mathbf{R}_i \xleftarrow{\$} \mathscr{S}_1^{2\times k}$ for $i = 0, \ldots, \ell$, the random integers $h_i \xleftarrow{\$} \mathscr{U}(\mathbb{Z}_q)$ for $i = 1, \ldots, \ell$, and sets $h_0 := 1$. Then, it sets:

$$\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{u}\end{bmatrix} := \mathbf{A}_{RSIS}\ , \tag{5.2}$$

$$\mathbf{A}_i := \mathbf{A}\mathbf{R}_i + h_i\mathbf{G} \text{ for } i = 1, \ldots, \ell\ . \tag{5.3}$$

Then, B sends $svk = (\mathbf{A}, \mathbf{B}, \mathbf{A}_0, \ldots, \mathbf{A}_\ell, \mathbf{u})$ to A. The key svk generated by the simulator is indistinguishable from a honestly generated one under $\mathrm{RLWE}_{(\ell+1)k,\mathscr{U}(\mathscr{S}_1)}$ (cf. Lemma 5.17).

The adversary is allowed to query signatures on at most q_{A} messages μ of its

[5] This can be done with high probability, if the ring R_q contains enough invertible elements (cf. for example [Lyubashevsky and Neven, 2017, Lemma 2.2]). Indeed, assume that R_q contains N invertible elements. Then with probability $(m+2)N/q^n$ at least one of the components of $\mathbf{A}_{RSIS} = [\mathbf{a}_1 \ \ldots \ \mathbf{a}_{m+2}]$ is invertible. Assume w.l.o.g. such component to be $\mathbf{b} := \mathbf{a}_{m+2}$. Then $\mathbf{b}^{-1}\mathbf{a}_i$ are iid (uniform) random variables over R_q, and solving the RSIS instance $[\mathbf{b}^{-1}\mathbf{a}_1 \ \ldots \ \mathbf{b}^{-1}\mathbf{a}_{m+1} \ \mathbf{1}]$ implies finding a solution for the $\mathbf{A}_{RSIS}$ instance of RSIS. In case no component of $\mathbf{A}_{RSIS}$ is invertible, B aborts. This happens with probability $1-(m+2)N/q^n$, that is negligible if the number N of invertible elements in R_q is large enough.

choice. Upon receiving the message μ, B constructs

$$\begin{aligned}
h_\mu &:= h_0 + \sum_{i=1}^{\ell} (-1)^{\mu_i} h_i \\
\mathbf{R}_\mu &:= \mathbf{R}_0 + \sum_{i=1}^{\ell} (-1)^{\mu_i} \mathbf{R}_i \\
\mathbf{A}_\mu &= \left[\mathbf{A} \quad \mathbf{B} \quad \mathbf{A}_0\mathbf{R}_\mu + h_\mu \mathbf{G}\right] \\
&= \left[\mathbf{A} \quad \mathbf{B} \quad (\mathbf{A}\mathbf{R}_0 + \textstyle\sum_{i=1}^{\ell} (-1)^{\mu_i}\mathbf{R}_i) + (h_0 + \sum_{i=1}^{\ell} (-1)^{\mu_i} h_i)\mathbf{G}\right] \\
&= \left[\mathbf{A} \quad \mathbf{B} \quad \mathbf{A}_0 + \textstyle\sum_{i=1}^{\ell} (-1)^{\mu_i}\mathbf{A}_i\right] .
\end{aligned}$$

If $h_\mu = 0$, B aborts. The product $h_\mu \mathbf{G}$ can be written as $\mathbf{H}_\mu \mathbf{G}$, where $\mathbf{H}_\mu = h_\mu \mathbb{I}_k$ is an invertible matrix, as $h_\mu \in \mathbb{Z}_q$ and q is a prime. Hence $\mathbf{G}$ is a $\mathbf{H}_\mu$-trapdoor for $\mathbf{A}$, and B can use it to sample a short element $\mathbf{S}$ in the lattice $\Lambda^{\perp}_{\mathbf{u}}(\mathbf{A}_\mu)$ thanks to Theorem 5.15. In fact, B samples an element of $\Lambda^{\perp}_{\mathbf{u}}(\mathbf{A}_\mu)$ distributed as a Gaussian with standard deviation α, hence the norm of the coefficients of the elements of $\mathbf{S}$ is bounded by $B = 8\alpha$. Hence, the vector $\mathbf{S}$ is a valid signature on μ, and B can send it back to A.

Upon receiving a forgery (μ^*, σ^*), B aborts if $0 \leftarrow \mathsf{SVerify}(\sigma^*, \mu^*, svk)$ or if $h_{\mu^*} \not\equiv 0 \bmod q$. Otherwise, B can extract a solution to RSIS from σ, as it can be seen from the verification equation:

$$\begin{aligned}
\mathbf{A}_{\mu^*}\mathbf{S}^* &= \mathbf{u} \bmod q \\
&\Rightarrow \left[\mathbf{A} \quad \mathbf{B} \quad \mathbf{A}_0 + \textstyle\sum_{i=1}^{\ell} (-1)^{\mu_i^*}\mathbf{A}_i\right] \begin{bmatrix} \mathbf{S}_1^* \\ \mathbf{S}_2^* \\ \mathbf{S}_3^* \end{bmatrix} = \mathbf{u} \bmod q \\
&\Rightarrow \left[\mathbf{A} \quad \mathbf{B} \quad \mathbf{A}_0\mathbf{R}_{\mu^*}\right] \begin{bmatrix} \mathbf{S}_1^* \\ \mathbf{S}_2^* \\ \mathbf{S}_3^* \end{bmatrix} = \mathbf{u} \bmod q \\
&\Rightarrow \left[\mathbf{A} \quad \mathbf{B} \quad \mathbf{u}\right] \underbrace{\begin{bmatrix} \mathbf{S}_1^* + \mathbf{R}_{\mu^*}\mathbf{S}_3^* \\ \mathbf{S}_2^* \\ -1 \end{bmatrix}}_{\mathbf{S}_{RSIS}} = \mathbf{0} \bmod q
\end{aligned}$$

where $\mathbf{R}_{\mu^*} := \mathbf{R}_0 + \sum_{i=1}^{\ell} (-1)^{\mu_i^*}\mathbf{R}_i$. The norm of the vector $\mathbf{S}_{RSIS}$ is dominated by

$$\|\mathbf{S}_1^* + \mathbf{R}_{\mu^*}\mathbf{S}_3^*\|_\infty \leq 8s_\sigma + (\ell+1)\max_i \|\mathbf{R}_i\mathbf{S}_3^*\|_\infty \leq 8s_\sigma + (\ell+1)kn8s_\sigma ,$$

as the norm $\max_i \|\mathbf{R}_i\mathbf{S}_3^*\|_\infty \leq k\max_{i,j,h}\|\mathbf{r}_{h,j}^i\mathbf{s}_j^*\|_\infty \leq kn8s_\sigma$, where $\mathbf{r}_{h,j}^i$ and $\mathbf{s}_j^*$ are the components of $\mathbf{R}_i$ and $\mathbf{S}_3^*$ respectively, and the last inequality follows by standard bounds on the infinity norm of the product of polynomials (cf. for example [Boschini et al., 2018b, Lemma 1]). The success probability of B is $\epsilon_\mathsf{A} \times (1-\epsilon_{RLWE}) \times \Pr[\mathsf{B} \text{ does not abort}]$, where ϵ_{RLWE} is the probability that the adversary can distinguish the verification key generated by B from a honestly generated one.The abort probability of algorithm B can be bounded from below as

$$\Pr[\mathsf{B} \text{ does not abort}] \geq \frac{1}{q}\left(1-\frac{q_\mathsf{A}}{q}\right)$$

following the same augment in the proof of [Boyen, 2010, Lemma 27]. Hence the success probability of B is

$$\epsilon_\mathsf{B} = \epsilon_\mathsf{A} \times (1-\epsilon_{RLWE}) \times \frac{1}{q}\left(1-\frac{q_\mathsf{A}}{q}\right)$$

and it is negligible assuming that $q \gg 2q_\mathsf{A}$ (where recall that q_A is the number of queries that A is allow to ask the signing oracle). □

Lemma 5.17. *Assume there exist a PPT algorithm* A *playing the eu-acma experiment that can distinguish the verification key as generated in Equation* (5.2) *from a honestly generated one with probability* ϵ_A *in time* t_A*. Then there exists an algorithm* B *that can solve* $RLWE_{(\ell+1)k,\mathscr{U}(\mathscr{S}_1)}$ *with probability* $\epsilon_B =$ *in time* $t_B = poly(t_\mathsf{A})$ *exploiting* A*.*

Proof. The proof is essentially equal to the proof of Lemma 3.29, hence we omit it. □

Remark 16. Observe that to have a non-negligible success probability it should hold that $q \gg 2q_\mathsf{A}$, where q_A is the number of signing queries the adversary is allowed to do. When using this signature as a building block for a group signature, this is actually not limiting. Recall that in the group signature this signature scheme is used to authenticate an encryption of a user's identity. Hence, a limit in the number of users, e.g., assuming 2^{32} users (i.e., more than the Earth population), implies that the adversary can make at most 2^{32} queries to the signing oracle. Therefore, choosing $q \gg 2^{33}$ would be enough to ensure a non-negligible success probability. The value of q could be improved using the technique shown in Section 3.5 of [Boyen, 2010], which allows to relax the requirement to $q^t \gg 2q_\mathsf{A}$, for some divisor t of n. We decided against the use of this technique as the improvement would not be significant (the parameter q has to be quite large anyway to ensure the hardness of the RSIS instance underlying

the security proof). Finally, remark that for $q \gg 2^{\ell}$, the security proof can also be done using complexity leveraging, along the lines of the proofs of Theorems 7, 8, 9 in [Boschini et al., 2017], and results in better parameters.

5.4.2 Efficient NIZK Proofs for Lattice-Based Relations

Our NIZK proofs for lattices are based on Aurora, that is an Interactive Oracle Proof (IOP) for Rank 1 Constraints Satisfaction (R1CS).

An Interactive Oracle Proof (IOP, a generalization of Interactive Proofs (IP) and Probabilistically Checkable Proofs (PCP), cf. [Ben-Sasson et al., 2016a]) is k-round protocols between a prover and a verifier, where the verifier is given oracle access to the prover's messages $f_1, \ldots, f_k$, i.e., the verifier does not read in full the prover's responses, but only access (random) parts of them. IOPs being proof systems means that they guarantee completeness (if the instance x is in the language, the verifier outputs 1 at the end of the interaction with probability 1), and soundness $\epsilon(x)$ (i.e., the probability that the verifier outputs 1 at the end of the protocol when x is not in the language is $\epsilon(x)$). IOP can have *Honest-Verifier Zero-Knowledge* (i.e., it is possible to build a simulator S that simulates the view of a honest verifier), and they can be *proofs of knowledge* (i.e., it is possible to build an extractor E that extract a valid witness for a given x by interacting with the prover). Finally, an IOP is *public-coin* if the queries of the verifier in round i only depend on the messages $m_1, \ldots, m_{i-1}$ it sent previously to the prover, and if such messages are random in $\{0,1\}^*$.

Alongside the definition of IOP, Ben-Sasson et al. [2016a] introduced also a polynomial-time transformation T to transform a public-coin IOP (P, V) into non-interactive random oracle proofs systems $(\mathsf{P}', \mathsf{V}')$ that relies on Merkle trees. The intuition is very similar to the Fiat–Shamir heuristic: essentially P' executes the interaction between P and V internally, generating the verifier's messages m_i through the random oracle (possible, as the protocol is public coin). Then, it uses the Merkle tree to guarantee that the verifier was run honestly: it "authenticates" each query through computing the root for each of the prover's answers f_i and computing a path for each of the verifier's queries to the oracles f_{y_j}, where y_j is the oracle queried in the j-th query. The choice of using the Merkle tree as an authentication system allows the transformation to only rely on the assumption that random oracles exist. As IOPs are multi round, the prover gets the opportunity to reset the verifier multiple times before the end of the interaction, and could potentially exploit this when generating a non-interactive proof. Hence the authors define a stronger version of soundness called *state restoration soundness*, that is defined as the probability that a malicious prover succeeds in making a

honest verifier accept without knowing a valid witness in an interactive protocol where the prover may rewind the verifier to a previous round. The concept is similar to special soundness for Σ-protocols.

Aurora is a new IOP for R1CS by Ben-Sasson et al. [2019]. The choice of the R1CS comes from the fact that such language generalizes circuits by allowing "native" field arithmetic while still being easy to work with, and it is used in practical applications [ZCash Company, 2014].

Definition 5.18 (R1CS relation)**.** The relation $\mathscr{R}_{R1CS}$ consists of the set of all pairs $((\mathbb{F}, k, m, n, A, B, C, v), w)$ where $\mathbb{F}$ is a finite field, k is the number of inputs, n is the number of variables, m is the number of constraints, A, B, C are matrices in $\mathbb{F}^{m\times(n+1)}$, $v \in \mathbb{F}^k$, and $w \in \mathbb{F}^{n-k}$ such that $Az \circ Bz = Cz$ where $z = (1, v, w) \in \mathbb{F}^{n+1}$ and $\circ$ denotes entry-wise (Hadamard) product.

The matrices A, B and C define a system of constraints; each row corresponds to a constraint, and the columns correspond to the variables. The vector z then represents an assignment to the variables.

The following theorem summarizes the properties of Aurora when compiled to a SNARK via the transform by Ben-Sasson et al. (cf. Theorem 7.1 in [Ben-Sasson et al., 2016b]). In the statement below, $N := \max(m, n)$; generally n and m will be of roughly the same magnitude.

Theorem 5.19 (informal, cf. Theorem 1.2 in [Ben-Sasson et al., 2018])**.** *There exists a non-interactive zero-knowledge argument for R1CS that is unconditionally secure in the Random Oracle Model with proof length $O(\lambda^2 \log^2 N)$ and soundness error $2^{-\lambda}$ against adversaries making at most 2^λ queries to the oracle. The prover runs in time $O_\lambda(N \log N)$ and the verifier in time $O_\lambda(N)$.*

Remark 17 (Simulation soundness)**.** To use the above construction in the Naor–Yung paradigm, as we later do, requires one-time simulation soundness (OTSS). This is shown as follows; we assume some familiarity with [Ben-Sasson et al., 2016a]. Let π be a proof output by the simulator for a statement x supplied by the adversary. First recall that to achieve adaptive soundness and zero knowledge, the oracle queries of the verifier and honest prover are prefixed with the statement x and a fresh random string $r \in \{0,1\}^\lambda$. Since with high probability no efficient adversary can find $x' \neq x, q, q'$ such that $\rho(x\|r\|q) = \rho(x'\|r\|q')$, if the adversary in the OTSS game chooses an instance different from that of the simulated proof, the success probability of the extractor is affected only by a negligible amount.

Now suppose that an adversary generates a different proof $\pi' \neq \pi$ of the same statement x. In the Aurora IOP, the query locations for the first oracle are

a uniformly random subset of $[\ell]$ (where ℓ is the oracle length, $\ell = \Omega(N)$) of size $\Omega(\lambda)$. This is determined by the verifier's final randomness, which in the compiled NIZK depends on all of the Merkle tree roots; these are all included in π. Moreover, these collectively depend on every symbol of π; hence no efficient adversary can find a valid $\pi' \neq \pi$ whose query set is the same as that of π. In particular, the Merkle tree root corresponding to the *first* round has some query in π' which is not in π; since it is infeasible to find an accepting authentication path for this query relative to the root provided by the simulator, the value of this root must differ between π and π'. It follows that, with high probability, the extractor only 'programs' queries which were not already programmed by the simulator, and so one-time simulation soundness holds.

We build the NIZKs required for our group signature scheme from simple, reusable building blocks for NIZKs on lattice primitives. When composing these building blocks, it will often be necessary to make explicit inputs private. Generally this involves no additional complication; if changes are needed to ensure soundness, we will point them out. When we construct R1CS instances (cf. Definition 5.18), we typically write down a list of variables and constraints, rather than explicitly constructing the matrices.

Basic Operations

Let R_q be a polynomial ring whose coefficients are in $\mathbb{Z}_q$. For the entirety of this section, we will take $R_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$; recall that the elements of R_q are polynomials of degree less than n. We describe how to express some basic lattice operations in R_q as arithmetic operations over $\mathbb{F}_q \cong \mathbb{Z}_q$ for prime q. With $\mathsf{BitD}(\mathbf{a})$ we denote an algorithm that on input elements $\mathbf{a}_i \in R_q$, outputs vectors $\vec{a}_i$ containing the binary expansion of the coefficients of $\mathbf{a}_i$.

Representation of Ring Elements We will generally represent ring elements as vectors in $\mathbb{F}_q^n$ with respect to some basis of R_q. Note that regardless of the choice of basis, addition in R_q corresponds exactly to component-wise addition of vectors.

Bases We will use two bases: the coefficient basis and the evaluation or number-theoretic transform (NTT) basis. The NTT basis, which is the discrete Fourier basis over $\mathbb{F}_q$, allows polynomial multiplication to be expressed as pointwise multiplication of vectors. Transforming from the coefficient basis to the NTT basis is a linear transformation $T \in \mathbb{F}_q^{n\times n}$. The choice of basis depends on the type of

constraint we wish to check; generally we will represent inputs in the coefficient basis.

An issue with the NTT basis is that to multiply ring elements $\mathbf{a}, \mathbf{b} \in R_q$ naively requires us to compute the degree-$2n$ polynomial $\mathbf{ab} \in \mathbb{F}_q[X]$ and then reduce modulo $X^n + 1$. This would make multiplying ring elements quite expensive. For our choice of R_q, however, so long as q has $2n$-th roots of unity we can employ the *negative wrapped convolution* [Lyubashevsky et al., 2008], which is a linear transform T such that if $\vec{a}, \vec{b}, \vec{c}$ represent the coefficients of $\mathbf{a}, \mathbf{b}, \mathbf{c} \in R_q$ respectively, $T\vec{a} \circ T\vec{b} = T\vec{c}$ if and only if $\mathbf{c} = \mathbf{ab}$ in R_q. From here on, T is the negative wrapped convolution.

Multiplication Following the above discussion, in the NTT basis multiplication is componentwise over $\mathbb{F}_q$. Hence to check that $\mathbf{a} \cdot \mathbf{b} = \mathbf{c}$ in R_q when $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are represented in the coefficient basis as $\vec{a}, \vec{b}, \vec{c}$, we use the constraint system:

$$T\vec{a} \circ T\vec{b} = T\vec{c} \ .$$

Bit Decomposition A simple but very important component of many primitives is computing the bit decomposition of a $\mathbb{Z}_q$-element. The following simple constraint system enforces that $b_0, \ldots, b_{\ell-1}$, where $\ell = \lceil \log q \rceil$, is the bit decomposition of $a \in \mathbb{F}_q$.

$$\begin{aligned} b_i(1 - b_i) &= 0 \quad \forall i \in \{0, \ldots, \ell - 1\} \\ \sum_{i=0}^{\ell-1} b_i 2^i - a &= 0 \end{aligned}$$

We will use the notation $\vec{b} = \mathsf{BitDec}(a)$ to represent this constraint system. For a vector $\vec{a} \in \mathbb{F}_q^n$ and matrix $B \in \mathbb{F}_q^{n \times \ell}$ we write $B = \mathsf{BitDec}(\vec{a})$ for the constraint system "$B_j = \mathsf{BitDec}(a_j) \quad \forall j \in [k]$", for B_j the j-th row of B.

Proof of Shortness Showing that $a \in \mathbb{Z}_q$ is bounded by $\beta = 2^k < (p-1)/2$, i.e. $-\beta < a < \beta$, can be achieved using its bit decomposition. We have that $|a| < \beta$ if and only if there exist $b_0, \ldots, b_{k-1} \in \{0, 1\}, c \in \{-1, 1\}$ such that $c \sum_{i=0}^{k-1} b_i 2^i = a$. The prover will supply $b_0, \ldots, b_{k-1}$ as part of the witness. This introduces the following constraints:

$$\begin{aligned} b_i(1 - b_i) &= 0 \quad \forall i \\ (\sum_{i=0}^{k-1} b_i 2^i - a)(\sum_{i=0}^{k-1} b_i 2^i + a) &= 0 \end{aligned}$$

The number of new variables is k; the number of constraints is $k+1$. When we describe R1CS instances we will write the above constraint system as "$|a| < B$". For $\vec{a} \in \mathbb{Z}_q^n$, we will write "$\|\vec{a}\| < \beta$" for the constraint system "$|a_i| < \beta \quad \forall i \in [n]$", i.e. n independent copies of the above constraint system, one for each entry of $\vec{a}$.

Proof of Knowledge of RLWE Secret Key

We give a proof of knowledge for the following relation.

$$\mathscr{R} = \{(\mathbf{c}, \mathbf{d}; \mathbf{t}, \mathbf{e}) \in R_q^4 \ : \ \mathbf{d} = \mathbf{ct} + \mathbf{e} \bmod q \ \wedge \ \|\mathbf{e}\|_\infty < \beta = 2^k\}$$

Let $\vec{c}, \vec{d}, \vec{t}, \vec{e} \in \mathbb{F}_q^n$ encode $\mathbf{c}, \mathbf{d}, \mathbf{t}, \mathbf{e}$ in the coefficient basis. The condition is encoded by the following constraint system:

$$T\vec{c} \circ T\vec{t} = T\vec{f}$$
$$\vec{f} + \vec{e} = \vec{d}$$
$$\|\vec{e}\|_\infty \leq \beta$$

where $\vec{f} \in \mathbb{F}_q^n$ should be the coefficient representation of $\mathbf{ct}$. In practice, because we choose the RLWE error distribution to be a Gaussian with standard deviation σ_{RLWE}, we set $\beta = 8\sigma_{RLWE}$. The number of variables is $n(\log\beta + 6)$; the number of constraints is $n(\log\beta + 4)$. We write $\mathsf{RLWE}_\beta(\vec{c}, \vec{d}, \vec{t}, \vec{e})$ as shorthand for the above system of constraints. Note that we did not use the fact that the verifier knows $\vec{c}, \vec{d}$; this will allow us to later use the same constraint system when $\vec{c}, \vec{d}$ are also secret.

Hence, applying the results by Ben-Sasson et al. [2018] yields the following result.

Lemma 5.20. *There is a NIZK proof (SNARK) for the relation $\mathscr{R}$, secure and extractable in the Random Oracle Model, with proof length $O(\log^2\left(n\log\beta\right)\log q)$.*

With our parameters as given in Section 5.5.6, the size of a NIZK for a single proof of knowledge of an RLWE secret key is 72 kB (obtained from our implementation 5.4.3 using libiop). Constraint systems for RSIS, Module-RSIS and Module-RLWE can be derived similarly.

Remark 18. The constraint systems for RSIS, Module-RSIS and Module-RLWE can be derived from the previous system by defining the same constraints for vectors instead of ring elements (cf. the definition of these problems in Section 2.3.4), and imposing the norm check on the whole witness vector (at least in the case of Module-RSIS and RSIS).

Proof of Knowledge of Plaintext

We give a proof of knowledge for the following relation.

$$\begin{aligned}\mathscr{R} = \{&(\mathbf{a},\mathbf{b},\mathbf{v},\mathbf{w};\mathbf{e},\mathbf{f},\mathbf{r},\mu) \in R_q^7 \times \mathscr{S}_1 \\ &: \mathbf{v} = p(\mathbf{ar}+\mathbf{e}) \wedge \mathbf{w} = p(\mathbf{br}+\mathbf{f}) + \mu \wedge \|\mathbf{e}\|_\infty, \|\mathbf{f}\|_\infty < \beta\}\end{aligned}$$

Recall that $\mathscr{S}_1 \subseteq R_q$ is the set of all polynomials of degree less than n whose coefficients are in $\{0,1\}$, which is in natural bijection with the set $\{0,1\}^n$. Again, as before we set $\beta = 8\sigma_{RLWE}$.

Let $\vec{a},\vec{b},\vec{v},\vec{w},\vec{e},\vec{f},\vec{r},\vec{\mu} \in \mathbb{F}_q^n$ be the coefficient representations of the corresponding ring elements. The condition is encoded by the following constraint system:

$$\begin{aligned}\mathsf{RLWE}_\beta(\vec{g},\vec{a},\vec{r},\vec{e}) \\ \mathsf{RLWE}_\beta(\vec{h},\vec{b},\vec{r},\vec{f}) \\ \vec{w} = p\cdot\vec{g} \\ \vec{v} = p\cdot\vec{h} + \vec{\mu} \\ \mu_i(\mu_i - 1) = 0 \quad \forall i\end{aligned}$$

The number of variables is $n(2\log\beta+10)$; the number of constraints is $n(2\log\beta+15)$. This constraint system (repeated twice) is also used to build the NIZK required for the Dolev-Dwork-Naor construction. We write "$\vec{v},\vec{w} = \mathsf{Enc}_p(\vec{a},\vec{b},\vec{r},\vec{\mu})$" to denote the above system of constraints; $\vec{e}$ and $\vec{f}$ will be fresh variables for each instance of the system. Once again, we do not use the fact that the verifier knows $\vec{a},\vec{b},\vec{v},\vec{w}$, which will be useful later.

To encrypt tn bits, we simply encrypt t n-bit blocks separately. The constraint system is then given by t copies of the above system. We will use the notation $V,W = \mathsf{Enc}_p(\vec{a},\vec{b},\vec{r},\vec{\mu})$ to represent this, where V,W are $n\times k$ matrices whose rows are the encryptions of each n-bit block.

Proof of valid signature

An important component of the group signature scheme is proving knowledge of a message $\mu \in \{0,1\}^\ell$ together with a Boyen signature on μ (see Section 5.4.1). We first consider a simpler relation, where we prove knowledge of a signature on a publicly-known message. In the Boyen signature scheme, this corresponds to checking an inner product of ring elements, along with a proof of shortness for the signature. This corresponds to checking the following relation.

$$\mathscr{R} = \{(\mathbf{A}_\mu, \mathbf{u}; \mathbf{S}) \in (R_q^{1\times k} \times R_q) \times R_q^k : \mathbf{A}_\mu \mathbf{S} = \mathbf{u} \bmod q \wedge \|\mathbf{S}\|_\infty < \beta\}$$

Let $A, S \in \mathbb{F}_q^{k\times n}$ be the matrices whose rows are the coefficients of the entries of $\mathbf{A}_\mu, \mathbf{S}$, and let $\vec{u} \in \mathbb{F}_q^n$ be the coefficient representation of $\mathbf{u}$. We obtain the following constraint system:

$$\begin{aligned} TA_i \circ TS_i &= TF_i \quad \forall i \in [k] \\ \sum_{i=1}^{k} F_i &= \vec{u} \\ \|S_i\|_\infty &< \beta \quad \forall i \in [k] \end{aligned}$$

where $F \in \mathbb{F}_q^{k\times n}$, and A_i, S_i, F_i are the i-th rows of the corresponding matrices.

Now we turn to the more complex task of proving knowledge of a (secret) message and a signature on that message. Here the verifier can no longer compute $\mathbf{A}_\mu$ by itself, and so the work must be done in the proof. In particular, we check the following relation.

$$\mathscr{R} = \left\{ \begin{array}{c} (\begin{bmatrix}\mathbf{A} & \mathbf{B}\end{bmatrix}, \mathscr{A}, \mathbf{u}; \mu, \mathbf{S}) \in R_q^{1\times m} \times \\ \times (R_q^{1\times k})^{\ell+1} \times R_q \times \{0,1\}^\ell \times R_q^{m+k} \end{array} : \mathbf{A}_\mu \mathbf{S} = \mathbf{u} mod q \wedge \|\mathbf{S}\|_\infty < B \right\},$$

where $\mathscr{A} = (\mathbf{A}_0, \ldots, \mathbf{A}_\ell)$ and $\mathbf{A}_\mu = \begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{A}_0 + \sum_{i=1}^{\ell} \mu_i \mathbf{A}_i\end{bmatrix}$. Let $M \in \mathbb{F}^{m\times n}$ be the matrix whose rows are the coefficients of the entries of $\begin{bmatrix}\mathbf{A} & \mathbf{B}\end{bmatrix}$, and let $A_0, \ldots, A_\ell \in \mathbb{F}^{k\times n}$ be matrices whose rows are the coefficients of the entries of $\mathbf{A}_0, \ldots, \mathbf{A}_\ell$ respectively. Let $A'_i \in \mathbb{F}^{n\times(\ell+1)}$ be such that the j-th column of A'_i is the i-th row of A_j (i.e., the coefficients of the i-th entry of A_j). Observe that $A'_i \cdot (1, \mu)$ is the coefficient representation of the i-th entry of $\mathbf{A}_0 + \sum_{j=1}^{\ell} \mu_j \mathbf{A}_j$. Given this, the following constraint system captures the relation we need.

$$\begin{aligned} TM_i \circ TS_i &= TF_i & \forall i \in [m] \\ (TA'_i)(1, \mu) \circ TS_{m+i} &= TF_{m+i} & \forall i \in [k] \\ \sum_{i=1}^{k+m} F_i &= \vec{u} \\ \mu_i \cdot (1 - \mu_i) &= 0 & \forall i \in [\ell] \\ \|S_i\|_\infty &< \beta & \forall i \in [m+k] \end{aligned}$$

with $F \in \mathbb{F}_q^{(m+k)\times n}$. We will denote the above constraint system by $\mathsf{SVerify}_\beta(M, \mathscr{A}, \vec{u}, S, \mu)$, with $\mathscr{A} = (A_0, \ldots, A_\ell)$. The number of variables and constraints are bounded by $(4 + \log\beta)(m+k)n + k(\max(n, \ell+1))$. Recall that $\beta = 8s_\sigma = 8\sqrt{n\log n} \cdot \sqrt{\log n^2}$ (cf. Section 5.4.1).

Signature generation

Here we specify the relation whose proof constitutes a signature for our group signature scheme; see 5.5 for details. We repeat its formal description below.

$$\mathscr{R}_S = \left\{ \begin{array}{l} \big(\mathbf{A}, \mathbf{B}, \mathscr{A}, \mathbf{u}, (\mathbf{a}_0, \mathbf{b}_0, \mathbf{a}_1, \mathbf{b}_1), \\ (\mathbf{V}_0, \mathbf{W}_0), (\mathbf{V}_1, \mathbf{W}_1);\ \mathbf{t}, i, \mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{S}\big) \end{array} \right.$$

$$\text{s.t.} \quad \left. \begin{array}{c} 1 \leftarrow \mathsf{SVerify}(\mathbf{S}, (\mathbf{c}, \mathbf{d}, i), \mathbf{A}, \mathbf{B}, \mathbf{A}_0, \ldots, \mathbf{A}_\ell, \mathbf{u}) \\ \wedge\ \mathbf{d} = \mathbf{ct} + \mathbf{e}\ \wedge\ \|\mathbf{e}\| \le \beta' \\ \wedge\ (\mathbf{V}_0, \mathbf{W}_0) \leftarrow \mathsf{Enc}_{RLWE}(i, \mathbf{c}, \mathbf{d}, (\mathbf{a}_0, \mathbf{b}_0)) \\ \wedge\ (\mathbf{V}_1, \mathbf{W}_1) \leftarrow \mathsf{Enc}_{RLWE}(i, \mathbf{c}, \mathbf{d}, (\mathbf{a}_1, \mathbf{b}_1)) \end{array} \right\}$$

We now describe the constraint system which represents this relation. The variables $\vec{c}, \vec{d}, \vec{e}, i, A, B, \mathscr{A}, \vec{u}, S, \vec{a}_0, \vec{a}_1, \vec{b}_0, \vec{b}_1, V_0, W_0, V_1, W_1$ are the coefficient representations of the corresponding variables in the relation. Using the notation defined in the previous subsections, the constraint system is as follows.

$$\begin{aligned} C &= \mathsf{BitDec}(\vec{c}) \\ D &= \mathsf{BitDec}(\vec{d}) \\ \vec{i} &= \mathsf{BitDec}(i) \\ &\mathsf{SVerify}_\beta([A|B], \mathscr{A}, \vec{u}, S, (C, D, \vec{i})) \\ &\mathsf{RLWE}_{\beta'}(\vec{c}, \vec{d}, \vec{e}) \\ V_0, W_0 &= \mathsf{Enc}_p(\vec{a}_0, \vec{b}_0, \vec{r}, (C, D, \vec{i})) \\ V_1, W_1 &= \mathsf{Enc}_p(\vec{a}_1, \vec{b}_1, \vec{r}, (C, D, \vec{i})) \end{aligned}$$

The number of variables and constraints are bounded by $(4+\log\beta)(m+k)n + 2kn\log q + 5n\log\beta + 30n + 6$. With our parameters this yields approximately 10 million variables and constraints.

By applying the proof system of [Ben-Sasson et al., 2018], we obtain the following lemma.

Lemma 5.21. *There is a NIZK proof (SNARK) for the relation $\mathscr{R}_S$, secure and extractable in the Random Oracle Model, with proof length $O(\log^2\big((m+k)n\log\beta + n^2\log q\big)\log q)$.*

Proof of valid decryption

The following relation captures the statement that the prover knows the RLWE secret key corresponding to a given public key, and that a given ciphertext decrypts to a given message under this key.

$$\mathscr{R} = \Big\{(\mathbf{v},\mathbf{w},\mu,\mathbf{a},\mathbf{b};\mathbf{s},\mathbf{e}) \\ : \ (\mathbf{w}-\mathbf{sv}) \bmod p = \mu \ \wedge \ \mathbf{b} = \mathbf{as}+\mathbf{e} \wedge \|\mathbf{s}\|_\infty, \|\mathbf{e}\|_\infty \leq \beta \Big\},$$

The constraint system is as follows.

$$\mathsf{RLWE}_\beta(\vec{a},\vec{b},\vec{s},\vec{e})$$
$$T\vec{w} - T\vec{s} \circ T\vec{v} = T(\vec{\mu} + p\vec{h})$$
$$\|\vec{h}\|_\infty < (q-1)/2p$$

The final constraint is to ensure that $\vec{\mu} + p\vec{h}$ does not 'wrap around' modulo q. Since $\vec{v},\vec{w}$ are public, the verifier can incorporate these into the constraint system. The number of variables and constraints is bounded by $n(\log\beta + \log(q/p) + 5)$.

Parameter choices

In this section we discuss how the parameter choices in 5.5.6 relate to the relations described in the above sections, and the resulting constraint system sizes given by our implementation (5.4.3). Throughout we let q be a prime with $\log_2 q \approx 65$, and $R_q = \mathbb{F}_q/\langle X^n+1\rangle$ with $n = 1024$. We have $\log\beta = 10$.

Proof of knowledge of RLWE secret key Our implementation yields a constraint system with 16,383 variables and 15,361 constraints for the parameters specified. The resulting proof is 72kB in size, and is produced in roughly 40 seconds on a consumer laptop (MacBook Pro).

Proof of knowledge of plaintext Our implementation yields a constraint system with 32,769 variables and 29,696 constraints for the parameters specified. The resulting proof is 87kB in size, and is produced in roughly three minutes.

Proof of valid signature Here $k = \bar{m} = 67$, $m = 2\bar{m} = 134$. Proving knowledge of a message $\mu \in \{0,1\}^\ell$ and signature on μ yields at most $3 \times 10^6 + 67\ell$ constraints, for $\ell > n$. Our message size is $\ell = 2nk + \log N$, where N is the number of users in the system; we obtain roughly $12 \times 10^6 + 67 \log N$ constraints. Since the number of users will always be bounded by (say) 2^{40}, the number of constraints is bounded by 12 million.

Our implementation yields a constraint system of 2,663,451 variables and 2,530,330 constraints. This is too large to produce a proof for on our Google Cloud instance, but extrapolating from known proof sizes we expect this to be at most 150kB.

Signature generation Our implementation yields a constraint system with 10,196,994 variables and 10,460,226 constraints. This is too large to produce a proof for, but extrapolating from known proof sizes we expect at most 250kB.

5.4.3 Optimization and Implementation of the Proof System

Faster NIZKs for NTT

In this section we describe an optimization which could improve verification time for the number-theoretic transform.

Recall that we wish to prove that for $a, b \in \mathbb{F}_p^n$, $a = Tb$, where T is the NTT matrix; $T_{ij} = \omega^{ij}$ for some n-th primitive root of unity $\omega \in \mathbb{F}_p$. The lincheck protocol of [Ben-Sasson et al., 2018] achieves this, but with verification time complexity $\Omega(n^2)$. There is an optimization, however, arising from the relation of T to the multiplicative subgroup $H \subseteq \mathbb{F}^*$ of size n, leading to an exponential speedup.

Recall that we choose some embedding $\gamma : H \to [n]$ so that we can consider $a, b \in \mathbb{F}_p^H$ and $T \in \mathbb{F}_p^{H \times H}$, with $T_{ab} = \omega^{\gamma(a)\gamma(b)}$. In particular, we will choose the embedding $\gamma(\omega^i) = i$. This has the property that $\omega^{\gamma(\omega^i)} = \omega^i$. The lincheck polynomial $\hat{p}_\alpha^{(2)}$ for T is defined as the polynomial of minimal degree such that $\hat{p}_\alpha^{(2)}(b) = \sum_{a \in H} T_{ab} \alpha^{\gamma(a)} = \sum_{a \in H} \omega^{\gamma(a)\gamma(b)} \alpha^{\gamma(a)}$ for all $b \in H$. Since $\omega^{\gamma(b)} = b$, we have that $\hat{p}_\alpha^{(2)}(b) = \sum_{a \in H} (\alpha b)^{\gamma(a)} = \sum_{i=0}^{n-1} (\alpha b)^{\gamma(a)} = \frac{1-(\alpha b)^n}{1-\alpha b}$. Hence $\hat{p}_\alpha^{(2)}(b)(X) = \frac{1-(\alpha X)^n}{1-\alpha X}$.

This polynomial can be evaluated in $O(\log n)$ field operations by repeated squaring, everywhere except for the point α^{-1}. To evaluate at the point α^{-1} we can use techniques for dividing arithmetic circuits in time polylog(n) (or ignore this event since it happens with very low probability).

Implementation

The implementation was written in C++, primarily using the following libraries:

- `libff` (`https://github.com/scipr-lab/libfff`)
- `libiop` (`https://github.com/scipr-lab/libiop`)

`libff` is a C++ implementation of finite fields, and libiop includes a C++ implementation of the Aurora IOP and SNARK. The implementation took advantage of the `libiop`-provided APIs to construct the R1CS encodings of the various relations detailed in 5.4.2. Once these R1CS constraint systems were constructed, `libiop` was used to construct the Aurora IOPs, which where then compiled to zkSNARKs. Finally, the proof size of these SNARKs was measured directly.

`libiop` does not currently provide primitives to organize very large constraint systems as in this paper. To prevent the constraint systems from getting unwieldy, an additional class `ring_poly` was created to represent ring elements as vectors of R1CS variables. This class also contains an implementation of the negative wrapped convolution (along with its inverse), which was tested by comparing with multiplication of polynomials in the 'long-form' method. In addition, now polynomial multiplication using the negative wrapped convolution could be represented as a basic constraint and be composed as part of a larger constraint system. Similarly, bit decompositions and proofs of shortness were also represented as basic constraints.

Mirroring the definition of the relations themselves, the implementations for 5.20, 5.21 were composed by referencing the relevant smaller relations.

Several small utilites were also created in order to compute the parameters `libff` requires for the specific prime fields used in this paper, and to generate other prime fields to test how proof sizes varied with number of bits of the underlying prime field.

The constraint systems were compiled and run on a consumer-grade 2016 Macbook Pro, when running the prover and verifier could fit in memory. For the larger constraint systems such as for 5.21, a Google Cloud large-memory compute instance was used to finish constructing the proofs.

5.5 Group Signature with Blind Issuance from Lattices

This group signature is build from a lattice-based hash-and-sign type of signature $(\mathsf{SParGen}, \mathsf{SKeyGen}, \mathsf{Sign}, \mathsf{SVerify})$ (cf. Section 5.4.1), SNARKs (P, V), a post-

quantum one-time signature scheme $(\mathsf{OTSGen}, \mathsf{OTSSign}, \mathsf{OTSVf})$ (e.g., Lamport's signature scheme with key length 2λ bits) and a CCA2-secure encryption scheme $(\mathsf{EParGen}_{RLWE}, \mathsf{EKeyGen}_{RLWE}, \mathsf{Enc}_{RLWE}, \mathsf{Dec}_{RLWE})$ (in particular, we choose the RLWE encryption scheme [Lyubashevsky et al., 2013] made CCA2-secure via the Dolev-Dwork-Naor paradigm [Dolev et al., 1991], cf. Section 3.8.2).

There are two main differences between the construction by Bellare et al. and ours. First, the user secret is not a signing key but a RLWE secret. In the construction by Bellare et al. the i-th user has an authenticated (by a PKI) key pair (ssk_i, svk_i) and a second key pair (ssk'_i, svk'_i). During the joining phase, the user sends a signature $\sigma \leftarrow \mathsf{Sign}(svk'_i, ssk_i)$ on svk'_i and svk_i to the issuer, to prove that it has a valid key pair. Indeed, the issuer can verify the signature and check that the i-th entry of the public list **upk** of authenticated users contains the public key svk_i. The signature σ and the verification keys svk_i, svk'_i are stored in a list **reg** that is used by the opener and the judge to verify the identities of the signers. This construction protects the user from framing attempts in which a corrupted issuer tries to substitute svk'_i with another verification key. Indeed, to do that the adversary would have to forge a signature on svk'_i that is valid w.r.t. svk_i, thus breaking the unforgeability of the signature scheme. Our scheme is quite similar, but instead of two signature key pairs the i-th user has a OTS key pair $(otssk_i, otsvk_i)$ and a RLWE pair $(\mathbf{a}_i, \mathbf{b}_i)$ with the corresponding secret $\mathbf{t}_i$. To join, it sends a signature $ots_i \leftarrow \mathsf{OTSSign}(otssk_i, (\mathbf{a}_i, \mathbf{b}_i))$ to the issuer, along with the verification key $otsvk_i$ and a proof that it knows $\mathbf{t}_i$. The OTS verification key is authenticated (e.g., by a PKI) and published as the i-th entry in **upk** to prevent framing, while the RLWE is part of the user's group signing key.

The second difference is that in our case we require the NIZK proofs to be arguments of knowledge, so that in the security proofs we can exploit their extractability property. This makes the security proof slightly less tight, but avoids us the need to encrypt the whole credential.

In the following we present the scheme. Recall that BitD outputs the bit decomposition of the input.

5.5.1 Key Generation and Joining Protocol

Let N be the maximum number of users supported by the scheme. We assume there exists a publicly available list **upk** containing the personal (OTS) verification keys of the users, i.e., $\mathbf{upk}[i] = otsvk_i$.

GKg: A trusted third party generates the parameters of the signature scheme $spar \leftarrow \mathsf{SParGen}(1^\lambda)$ and of the encryption scheme running $epar \leftarrow \mathsf{EParGen}_{RLWE}$

(1^λ). We choose the error distribution of the RLWE encryption scheme to be a Gaussian distribution with standard deviation $\sigma_{RLWE} = 2\sqrt{\log q}$. Then it sets $\ell = 2n\lceil \log q \rceil + \lceil \log N \rceil$, and checks that $q \geq 4p\sqrt{\log q \log n}\sqrt{64 \log q + n}$. If that's not the case, it aborts and restarts the parameter generation. It generates the group manager's secret signing key T_{A} with corresponding public key $(\mathbf{A}, \mathbf{B}, \mathbf{A}_0, \ldots, \mathbf{A}_\ell, \mathbf{u})$ running the key generation algorithm $\mathsf{SKeyGen}$ of the signature scheme. Finally, it generates the opener's keys by first generating two pairs of encryption and decryption keys of the encryption scheme, $((\mathbf{a}_i, \mathbf{b}_i), \mathbf{s}_i) \leftarrow \mathsf{EKeyGen}_{RLWE}(epar)$ for $i = 0, 1$, and then setting $opk = (\mathbf{a}_0, \mathbf{b}_0, \mathbf{a}_1, \mathbf{b}_1)$ and $osk = \mathbf{s}_0$; $\mathbf{s}_1$ is discarded. Recall that the RLWE error distribution χ is set to be a discrete Gaussian with standard deviation s_{RLWE}. Hence an element $\mathbf{e} \xleftarrow{\$} \chi$ has norm bounded by $B_I = 8s_{RLWE}$ by Lemma 2.16.

UKg: The i-th user generates an OTS keys running $(otssk_i, otsvk_i) \leftarrow \mathsf{OTSGen}(1^\lambda)$. The verification key $otsvk_i$ is added as the i-th entry to the public list **upk**. The keys of the user are $(usk_i, upk_i) = (otssk_i, otsvk_i)$.

Join and Iss: The joining protocol is composed by a pair of algorithms $(\mathsf{Join}, \mathsf{Iss})$ that are run by the user and the group manager respectively.

To join a group, a user U_i interacts with the group manager M to obtain a certify on its public key usk_i. The user runs the algorithm Join, while the manager runs Iss. The resulting credential and the user's public key are stored by M in a list **reg**. Such list is necessary to guarantee that the opener actually recovered a valid identity. The complete protocol is showed in Figure 5.9 and explained in the following:

- The user starts by running Join on input its key pair $((\mathbf{c}_i, \mathbf{d}_i), \mathbf{t}_i)$. The algorithm ends outputting $(\mathbf{c}_i, \mathbf{d}_i, ots_i, otsvk_i)$ to M along with a proof Π_i that the user knows $\mathbf{t}_i, \mathbf{e}_i$, i.e., a proof that $(\mathbf{c}_i, \mathbf{d}_i)$ is a RLWE pair. The signature is generated running $\mathsf{OTSSign}((\mathbf{c}_i, \mathbf{d}_i), otssk_i)$, while the proof is generated running $\mathsf{P}_I(\mathbf{c}_i, \mathbf{d}_i; \mathbf{t}_i, \mathbf{e}_i)$ that is the prover algorithm of a SNARK $(\mathsf{P}_I, \mathsf{V}_I)$ for the following relation:

$$\mathscr{R}_I = \{(\mathbf{c}_i, \mathbf{d}_i; \mathbf{t}_i, \mathbf{e}_i) \in R_q^4 \ : \ \mathbf{d}_i = \mathbf{c}_i\mathbf{t}_i + \mathbf{e}_i \bmod q \ \wedge \ \|\mathbf{e}_i\|_\infty \leq B_I\}$$

  where $B_I = 8\sigma_{RLWE}$ is an upper bound on the absolute value of the coefficients of $\mathbf{e}_i$ computed in the parameters generation phase.

- M runs $\mathsf{V}_I(\mathbf{c}_i, \mathbf{d}_i, \Pi_i)$ and $\mathsf{OTSVf}(ots_i, (\mathbf{c}_i, \mathbf{d}_i), otsvk_i)$. If any of them outputs 0, the group manager aborts. Otherwise, it signs $(\mathbf{c}_i, \mathbf{d}_i, i)$ using the

signature scheme, i.e., it generates $\mathbf{S}_i$ with small norm such that

$$\left[\mathbf{A} \quad \mathbf{B} \quad \mathbf{A}_0 + \sum_{j=1}^{\ell} \mu_j \mathbf{A}_j\right] \mathbf{S}_i = \mathbf{u} \bmod q$$

where $\mu = (\mu_1, \ldots, \mu_\ell)$ is the binary expansion of $(\mathbf{c}_i, \mathbf{d}_i, i)$. Then, M sends $\mathbf{S}_i$ to U_i.

- the user verifies that $\mathbf{S}_i$ is a valid signature on $(\mathbf{c}_i, \mathbf{d}_i, i)$. If this is the case, it sends *accept* to the issuer, and sets its signing key to be $(\mathbf{t}_i, \mathbf{c}_i, \mathbf{d}_i, i, \mathbf{S}_i)$. Otherwise, it aborts.
- on input *accept*, the issuer stores the registration information $\mathbf{reg}[i] = (\mathbf{c}_i, \mathbf{d}_i, ots_i,)$ and concludes the protocol.

5.5.2 Signing Algorithm

The signature algorithm is shown in Figure 5.10. In the following, we explain the rationale behind it.

To produce a valid signature, a user has to prove that it has a valid credential. This means it has to prove that it has a signature by M on its user public key and group identity $(\mathbf{c}_i, \mathbf{d}_i, i)$. Abusing the notation (because the signing algorithm signs in fact the binary expansion $\mathsf{BitD}(\mathbf{c}_i, \mathbf{d}_i, i)$), we can say that the i-th user has to prove the following relation:

$$\mathscr{R} = \left\{ \begin{array}{c} (\mathbf{A}, \mathbf{B}, \mathbf{A}_0, \ldots, \mathbf{A}_\ell, \mathbf{u}; \\ \mathbf{t}_i, i, \mathbf{c}_i, \mathbf{d}_i, \mathbf{e}_i, \mathbf{S}_i) \end{array} : \begin{array}{c} 1 \leftarrow \mathsf{SVerify}(\mathbf{S}_i, (\mathbf{c}_i, \mathbf{d}_i, i), \mathbf{A}, \mathbf{B}, \mathbf{A}_0, \ldots, \mathbf{A}_\ell, \mathbf{u}) \\ \wedge \ \mathbf{d}_i = \mathbf{c}_i \mathbf{t}_i + \mathbf{e}_i \ \wedge \ \|\mathbf{e}_i\|_\infty \leq B \end{array} \right\} \tag{5.4}$$

where $B = B_I = 8\sigma_{RLWE}$. Moreover, to allow the opener to output a proof of honest opening, it is necessary that it can extract $\mathbf{c}_i$ and $\mathbf{d}_i$ from the signature. Hence, the user attaches to the NIZK proof also two encryptions $(\mathbf{V}_0, \mathbf{W}_0)$, $(\mathbf{V}_1, \mathbf{W}_1)$ of the user's identity i and of the RLWE sample $(\mathbf{c}_i, \mathbf{d}_i)$ w.r.t the two RLWE encryption keys in the opener public key. Remark that this does not compromise the user, as the opener never gets the user's secret key nor the user's signing key. To guarantee that the user is not cheating by encrypting a fake credential or by encrypting different plaintexts in the two ciphertexts, the relation in Equation (5.4) is not enough. In fact, the user has to prove that the two ciphertexts encrypt the same $(i, \mathbf{c}_i, \mathbf{d}_i)$ on which it proved it has a credential. The relation becomes:

$$\mathscr{R}_S = \left\{ \begin{array}{l} (\mathbf{A}, \mathbf{B}, \mathbf{A}_0, \ldots, \mathbf{A}_\ell, \mathbf{u}, \\ opk, (\mathbf{V}_0, \mathbf{W}_0), \\ (\mathbf{V}_1, \mathbf{W}_1); \\ \mathbf{t}_i, i, \mathbf{c}_i, \mathbf{d}_i, \mathbf{e}_i, \mathbf{S}_i) \end{array} : \begin{array}{c} 1 \leftarrow \mathsf{SVerify}(\mathbf{S}_i, \mathsf{BitD}(i, \mathbf{c}_i, \mathbf{d}_i), \mathbf{A}, \mathbf{B}, \mathbf{A}_0, \ldots, \mathbf{A}_\ell, \mathbf{u}) \\ \wedge \ \mathbf{d}_i = \mathbf{c}_i \mathbf{t}_i + \mathbf{e}_i \ \wedge \ \|\mathbf{e}_i\|_\infty \leq B_I \\ \wedge \ (\mathbf{V}_0, \mathbf{W}_0) \leftarrow \mathsf{Enc}_{RLWE}(\mathsf{BitD}(i, \mathbf{c}_i, \mathbf{d}_i), (\mathbf{a}_0, \mathbf{b}_0)) \\ \wedge \ (\mathbf{V}_1, \mathbf{W}_1) \leftarrow \mathsf{Enc}_{RLWE}(\mathsf{BitD}(i, \mathbf{c}_i, \mathbf{d}_i), (\mathbf{a}_1, \mathbf{b}_1)) \end{array} \right\} \tag{5.5}$$

and $(\mathsf{P}_S, \mathsf{V}_S)$ is a non-interactive SNARK for $\mathscr{R}_S$ (cf. Section 5.4.2). The user outputs the signature $\sigma = (\Pi_S, \mathbf{V}_0, \mathbf{W}_0, \mathbf{V}_1, \mathbf{W}_1)$.

Remark 19. In Bellare et al. [2005] the user has to add to the signature the encryption of the credential too. This becomes crucial in the security proof of traceability and non-frameability, as it allows the simulator to recover a forgery for the signature scheme from a valid forgery for the group signature scheme output by the adversary. Instead, in the security proofs of our scheme the credential is extracted through extraction. This is not possible in the construction by Bellare et al., as their NIZK proof is not required to be a proof of knowledge (while ours is an AoK).

5.5.3 Signature Verification and Opening

To verify a signature σ on a message μ, the algorithm GVerify checks the Proof Π_S by running $\mathsf{V}_S(\Pi_S, \mu, \mathbf{A}, \mathbf{B}, \mathbf{A}_0, \ldots, \mathbf{A}_\ell, \mathbf{u}, opk, (\mathbf{V}_0, \mathbf{W}_0, \mathbf{V}_1, \mathbf{W}_1))$. It outputs 1 if the verifier outputs 1.

The opener first runs GVerify on the signature. If GVerify returns 0 the opener outputs $(0, \epsilon)$. Otherwise it decrypts the ciphertext $(\mathbf{V}_0, \mathbf{W}_0)$ using its secret key $\mathbf{s}_0$ to recover the identity i and public key $(\mathbf{c}'_i, \mathbf{d}'_i)$ of the signer using its secret key $\mathbf{s}$. Then, to prove that the user's identity it extracted is valid, it recovers the i-th entry of the list $\mathbf{reg}[i] = (\mathbf{c}_i, \mathbf{d}_i, ots_i)$ and checks that $(\mathbf{c}'_i, \mathbf{d}'_i) = (\mathbf{c}_i, \mathbf{d}_i)$. If that is true, it includes in the output $\mathbf{reg}[i]$ along the $(\mathbf{c}'_i, \mathbf{d}'_i)$ it recovered from the signature. Finally, the opener produces a proof that the opening procedure was performed honestly using the decryption key *osk* corresponding to the opener's public key *opk*, i.e., it outputs a proof Π_O for the following relation:

$$\mathscr{R} = \left\{ \begin{array}{c} (\mathbf{V}_0, \mathbf{W}_0, i, \mathbf{c}'_i, \\ \mathbf{d}'_i, \mathbf{a}_0, \mathbf{b}_0, ; osk) \end{array} \; : \; \begin{array}{c} (i, \mathbf{c}'_i, \mathbf{d}'_i) \leftarrow \mathsf{Dec}_{RLWE}((\mathbf{V}_0, \mathbf{W}_0), osk) \\ \wedge \;\; osk \text{ is valid} \end{array} \right\}. \tag{5.6}$$

When instantiating the encryption scheme with the RLWE encryption scheme relation (5.6) becomes:

$$\mathscr{R}_O = \left\{ \begin{array}{c} (\mathbf{V}_0, \mathbf{W}_0, i, \mathbf{c}'_i, \mathbf{d}'_i, \\ \mathbf{a}_0, \mathbf{b}_0; \; \mathbf{s}_0, \mathbf{e}_0) \end{array} \; : \; \begin{array}{c} (\mathbf{W}_0 - \mathbf{s}_0\mathbf{V}_0) \bmod p = \begin{pmatrix} \hat{i} \\ \hat{\mathbf{c}}'_i \\ \hat{\mathbf{d}}'_i \end{pmatrix} \\ \wedge \;\; \mathbf{b}_0 = \mathbf{a}_0\mathbf{s}_0 + \mathbf{e}_0 \bmod q \\ \wedge \;\; \|\mathbf{s}_0\|_\infty, \|\mathbf{e}_0\|_\infty \leq B_O \end{array} \right\}, \tag{5.7}$$

where $\hat{i}, \hat{\mathbf{c}}'_i, \hat{\mathbf{d}}'_i$ are the binary polynomials obtained from the binary expansions of $i, \mathbf{c}'_i, \mathbf{d}'_i$ and $B_O = 8\sigma_{RLWE}$. If every check and the decryption go through, the

output of the opener is $(i, \tau) = (i, (\mathbf{c}'_i, \mathbf{d}'_i, \mathbf{c}_i, \mathbf{d}_i, ots_i, \Pi_O))$. Otherwise, the opener outputs $(i, \tau) = (0, \epsilon)$.

5.5.4 The Judge Algorithm

The Judge algorithm verifies the opener claims of having opened correctly a signature. Hence, it has to verify Π_O and that the decrypted public key, the entry in the list and the certified public key of the user coincides. It takes as input $(gpk, opk, \mathbf{upk}[i], \mu, \sigma, \tau, i)$, i.e., the group public key, the signature $\sigma = (\Pi_S, (\mathbf{V}_0, \mathbf{W}_0, \mathbf{V}_1, \mathbf{W}_1))$ and the respective message μ, and the output of the opener $(i, \tau) = (i, (\mathbf{c}'_i, \mathbf{d}'_i, \mathbf{c}_i, \mathbf{d}_i, ots_i, \Pi_O))$. It recovers the public key upk_i of user i from the public list, and outputs 1 if all of the following conditions hold:

- $(i, \tau) \neq (0, \epsilon)$
- $1 \leftarrow \mathsf{GVerify}(\sigma, \mu, gpk)$
- $(\mathbf{c}, \mathbf{d}) = (\mathbf{c}', \mathbf{d}')$
- $1 \leftarrow \mathsf{V}_O(\Pi_O, \mathbf{V}_0, \mathbf{W}_0, i, \mathbf{c}'_i, \mathbf{d}'_i, \mathbf{a}_0, \mathbf{b}_0)$
- $1 \leftarrow \mathsf{OTSVf}(ots_i, (\mathbf{c}_i, \mathbf{d}_i), otsvk_i)$.

Otherwise, the algorithm outputs 0.

5.5.5 Correctness and Security

Correctness follows from the correctness of the building blocks, as shown in the proof of Theorem 5.22.

Theorem 5.22 (Correctness)**.** *If the signature, OTS, RLWE encryption and NIZK proof system are correct, the group signature described above is correct.*

Sketch of the proof of Theorem 5.22. The proof consists of 5 steps

- proving that the joining protocol results in the group manager producing a signature on the user's RLWE pair,
- proving that a user can produce a signature through the NIZK and the RLWE encryption,
- proving that verification accepts honestly generated signatures,

- proving that the decryption of the ciphertext contained in a honestly generated signature outputs the identity of the signer and a NIZK,

- proving that the judge always accepts the output of a honest opener.

We start from the joining procedure. A user can prove it has a RLWE pair $(\mathbf{c}_i, \mathbf{d}_i)$ running P_I as $\mathbf{e}_i$ is sampled from a Gaussian (as specified in the Parameter Generation), hence it has infinity norm less than $8\sigma_{RLWE}$ thanks to Lemma 2.16. The signing algorithm succeeds as all the parameters are generated according to specifications. On the user's side, the correctness of the signature guarantees that the verification algorithm outputs 1 w.h.p. .

During the signing procedure we only need to make sure that the prover P_S can in fact output a NIZK for relation $\mathscr{R}_S$. This follows from the correctness of the signature, encryption and NIZK proof system. The bounds in relation $\mathscr{R}_S$ are set in Section 5.4.2 and follow from Lemma 2.16.

As the NIZK is generated honestly, the verification is guaranteed to output 1 with overwhelming probability by the correctness of the NIZK proof system. The correctness of the decryption of the RLWE ciphertext holds as long as $q \geq p\sigma_{RLWE}\sqrt{2 \cdot 16\sigma_{RLWE}^2 + n} \cdot 2\sqrt{logn}$ by Theorem 3.39. As $\sigma_{RLWE} = 2\sqrt{\log q}$, q as chosen in Section 5.5.1 satisfies the inequality. Finally, the opener can generate a NIZK proof to guarantee the opening was performed correctly, as the secret opening key $\mathbf{s}_O$ and $\mathbf{s}_O$ are again sampled from a Gaussian with standard deviation σ_{RLWE}, hence they have infinity norm bound by $8\sigma_{RLWE}$ w.h.p. by Lemma 2.16.

The judge outputs 1 when receiving as input a honestly generated opening thanks to the correctness of the OTS and of the NIZK proof system. □

Our group signature guarantees anonymity, traceability and non-frameability, meaning that it protects also against a corrupted group manager trying to frame a honest user. More precisely, the scheme is proven secure in the Random Oracle Model under quantum-safe assumptions. This essentially means that the scheme is provably secure against a classical adversary (i.e., an adversary that can only ask classical queries to the random oracle) that has access to a quantum computer.

Anonymity relies on simulation soundness and zero-knowledge of the proof system, and on the IND-CPA property of the encryption. The construction does not require the IND-CCA2 security of the encryption, as in the proof we exploit a step similar to the Sahai extension [Sahai, 1999] of the Naor-Yung approach [Naor and Yung, 1990].

Theorem 5.23 (Anonymity)**.** *The group signature scheme* GS *is anonymous in the Random Oracle Model under the zero-knowledge and simulation soundness property of the NIZK proof system and under the IND-CPA security of the encryption scheme.*

Proof. Assume A is an adversary against anonymity interacting with a simulator B. We prove through game hops that the experiment $\mathsf{Exp}^{an}_{\mathsf{A},0}(1^\lambda)$ is indistinguishable from the experiment $\mathsf{Exp}^{an}_{\mathsf{A},1}(1^\lambda)$ (cf. Figure 5.4). The success probability of A is:

$$\begin{aligned}
\epsilon_A &= \left|\mathsf{Pr}\left[b = b' \ : \ b \xleftarrow{\$} \{0,1\}, b' \leftarrow \mathsf{Exp}^{an}_{\mathsf{A},b}(1^\lambda)\right] - \frac{1}{2}\right| \\
&= \frac{1}{2}\left|\mathsf{Pr}\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}^{an}_{\mathsf{A},0}(1^\lambda)\right] + \mathsf{Pr}\left[b' = 1 \ : \ b' \leftarrow \mathsf{Exp}^{an}_{\mathsf{A},1}(1^\lambda)\right] - 1\right| \\
&= \frac{1}{2}\left|\mathsf{Pr}\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}^{an}_{\mathsf{A},0}(1^\lambda)\right] + \mathsf{Pr}\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}^{an}_{\mathsf{A},1}(1^\lambda)\right]\right| .
\end{aligned}$$

We proof that this probability is negligible if the rNIZK is zero-knowledge through a standard sequence of game hops. Let $\mathbf{Game_i}$ the probability that A outputs 0 at the end of the i-th game.

Game 0. Game 0 executes $\mathsf{Exp}^{an}_{\mathsf{A},0}(1^\lambda)$. Hence,

$$\mathsf{Pr}[\mathbf{Game_0}] = \mathsf{Pr}\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}^{an}_{\mathsf{A},0}(1^\lambda)\right] .$$

Game 1. In game 1 B simulates the NIZK proof in the oracle Chall using the simulator Σ_S of the proof system $(\mathsf{P}_S, \mathsf{V}_S)$. An adversary that can distinguish this game from the previous one can be used to break the ZK property of the NIZK proof system. Indeed, an algorithm B_1 with access to an oracle $\mathcal{O}_{NIZK}$ that outputs either simulated or honestly generated proofs can exploit A to distinguish the outputs of such oracle as follows. B_1 runs the anonymity experiment honestly but Chall. When it has to generate the challenge signature on μ_0, B_1 queries it to the oracle instead. It is clear that if the oracle outputs a simulated proof, this is exactly Game 1 and if the proof is honestly generated, A is playing exactly Game 0. At the end of the interaction, B_1 outputs exactly the same bit b' output by A. Hence, the success probability ϵ_1 of B_1 is

$$\begin{aligned}
\epsilon_1 &= \left|\mathsf{Pr}\left[b' = b \ : \ b \xleftarrow{\$} \{0,1\}, \ b' \leftarrow \mathsf{Exp}^{ZK-b}_{\mathsf{B}_1}(1^\lambda)\right] - \tfrac{1}{2}\right| \\
&= \tfrac{1}{2}\left|\mathsf{Pr}\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}^{ZK-0}_{\mathsf{B}_1}(1^\lambda)\right] + \mathsf{Pr}\left[b' = 1 \ : \ b' \leftarrow \mathsf{Exp}^{ZK-1}_{\mathsf{B}_1}(1^\lambda)\right] - 1\right| \\
&= \tfrac{1}{2}\left|\mathsf{Pr}\left[b' = 0 \wedge b' \leftarrow \mathsf{Exp}^{ZK-0}_{\mathsf{B}_1}(1^\lambda)\right] - \mathsf{Pr}\left[b' = 0 \wedge b' \leftarrow \mathsf{Exp}^{ZK-1}_{\mathsf{B}_1}(1^\lambda)\right]\right| \\
&= \tfrac{1}{2}\left|\mathsf{Pr}[\mathbf{Game_0}] - \mathsf{Pr}[\mathbf{Game_1}]\right| ,
\end{aligned}$$

where we denoted by $\mathsf{Exp}_{\mathsf{B}_1}^{ZK-b}$ the experiment in which B_1 has access to an oracle $\mathcal{O}_{NIZK}$ that implements either the honest prover when $b = 0$, or the simulator when $b = 1$.

Game 2. In Game 2 B does everything as in Game 1, except that now it generates $(\mathbf{V}_1, \mathbf{W}_1)$ as an encryption of $1^{\bar{\ell}}$ (where $\bar{\ell}$ is the length of the plaintext) while $(\mathbf{V}_0, \mathbf{W}_0)$ is still an encryption of $id_0^* = (i_0^*, \mathbf{c}_0^*, \mathbf{d}_0^*)$. The IND-CPA property of the encryption guarantees the indistinguishability of the games. Namely, let A be an adversary such that $\mathsf{Pr}[\mathbf{Game_1}] - \mathsf{Pr}[\mathbf{Game_2}]$ is non-negligible. Then B_2 can win the IND-CPA experiment in Figure 3.12 exploiting A as follows. Upon receiving $(\mathbf{a}_1, \mathbf{b}_1)$ from the oracle, B_2 generates $(\mathbf{a}_0, \mathbf{b}_0)$ honestly and sends $opk = (\mathbf{a}_0, \mathbf{b}_0, \mathbf{a}_1, \mathbf{b}_1)$ to A. When A sends back the identities (id_0^*, id_1^*), B_2 generates $(\mathbf{V}_0, \mathbf{W}_0)$ as an encryption of $id_0^* = (i_0^*, \mathbf{c}_0^*, \mathbf{d}_0^*)$, sends id_1^* to the encryption oracle and generates the proof Π_S using the simulator. B_2 outputs the same bit b' output by A. Remark that if the encryption outputs an encryption of id_1^*, then B_2 is implementing Game 1, otherwise this is exactly Game 2. Hence, the success probability ϵ_2 of B_2 is

$$\epsilon_2 = \frac{1}{2} \left| \mathsf{Pr}[\mathbf{Game_1}] - \mathsf{Pr}[\mathbf{Game_2}] \right| .$$

Game 3. In Game 3 B does everything as in Game 2, except that now it generates $(\mathbf{V}_1, \mathbf{W}_1)$ as an encryption of $id_1^* = (i_1^*, \mathbf{c}_1^*, \mathbf{d}_1^*)$ ($(\mathbf{V}_0, \mathbf{W}_0)$ is still an encryption of $id_0^* = (i_0^*, \mathbf{c}_0^*, \mathbf{d}_0^*)$). Again, the IND-CPA property of the encryption guarantees the indistinguishability of the games. Indeed, an adversary A such that $\mathsf{Pr}[\mathbf{Game_2}] - \mathsf{Pr}[\mathbf{Game_3}]$ is non-negligible can be exploited by B_3 to win the IND-CPA experiment in Figure 3.12 exactly as before. Hence, the success probability ϵ_3 of B_3 is

$$\epsilon_3 = \frac{1}{2} \left| \mathsf{Pr}[\mathbf{Game_2}] - \mathsf{Pr}[\mathbf{Game_3}] \right| .$$

Game 4. In Game 4 the simulator does everything as in Game 3 except the generation of the opening keys and of the opening oracle $\mathcal{O}_{\mathsf{GOpen}}$. Indeed, when generating the opening keys, B preserves $\mathbf{s}_1$ instead of $\mathbf{s}_0$, and then it performs the decryption in $\mathcal{O}_{\mathsf{GOpen}}$ w.r.t. $\mathbf{s}_1$. The only way that an adversary can distinguish the games is if it can submit a valid signature σ whose two RLWE ciphertexts encrypt different messages. This would break the simulation soundness of the NIZK proof system. Indeed, an algorithm B_4 would break the soundness exploiting A as follows. When generating the opening keys, it would keep $\mathbf{s}_0$ as well. Whenever it receives a decryption query, it would decrypt both $(\mathbf{V}_0, \mathbf{W}_0)$ and $(\mathbf{V}_1, \mathbf{W}_1)$,

checking that the resulting plaintexts are equal. If that is not the case, then B_4 can return $((\mu, otsvk, gpk, opk, (\mathbf{V}_0, \mathbf{W}_0), (\mathbf{V}_1, \mathbf{W}_1), \mathbf{t}_i, i, \mathbf{c}_i, \mathbf{d}_i, \mathbf{e}_i, \mathbf{S}_i), \Pi_S)$ as a proof of false statement. Since making this query s the only way A can distinguish the two games, the algorithm B_3 has success probability

$$\epsilon_4 = \left|\mathsf{Pr}[\mathbf{Game_3}] - \mathsf{Pr}[\mathbf{Game_4}]\right| .$$

Game 5. In Game 5 B does everything as in Game 4, except that now it generates $(\mathbf{V}_0, \mathbf{W}_0)$ as an encryption of $1^{\bar{\ell}}$ (where $\bar{\ell}$ is the length of the plaintext) while $(\mathbf{V}_1, \mathbf{W}_1)$ is an encryption of $id_1^* = (i_1^*, \mathbf{c}_1^*, \mathbf{d}_1^*)$. As before, the IND-CPA property of the encryption guarantees the indistinguishability of the games. Remark that a simulator B_5 trying to win the IND-CPA experiment exploiting A never has to decrypt $(\mathbf{V}_0, \mathbf{W}_0)$, thanks to the key switching in the previous game. The success probability ϵ_5 of B_5 is

$$\epsilon_5 = \frac{1}{2}\left|\mathsf{Pr}[\mathbf{Game_4}] - \mathsf{Pr}[\mathbf{Game_5}]\right| .$$

Game 6. In Game 6 B does everything as in Game 5, except that now it generates $(\mathbf{V}_0, \mathbf{W}_0)$ as an encryption of $id_1^* = (i_1^*, \mathbf{c}_1^*, \mathbf{d}_1^*)$. As before, a simulator B_5 trying to win the IND-CPA experiment exploiting A has success probability

$$\epsilon_6 = \frac{1}{2}\left|\mathsf{Pr}[\mathbf{Game_5}] - \mathsf{Pr}[\mathbf{Game_6}]\right| .$$

Game 7. In Game 7, B reverts back to decrypt using $\mathbf{s}_0$ in the opening oracle. As in Game 4, an algorithm B_7 breaks the soundness exploiting an adversary such that $\mathsf{Pr}[\mathbf{Game_6}] - \mathsf{Pr}[\mathbf{Game_7}$ is non negligible has success probability

$$\epsilon_7 = \left|\mathsf{Pr}[\mathbf{Game_6}] - \mathsf{Pr}[\mathbf{Game_7}]\right| .$$

Game 8. In Game 8 B reverts back to generating the proof Π_S^* in the challenge signature honestly. This is equivalent to B running Chall_1, hence Game 8 is exactly $\mathsf{Exp}_{\mathsf{A},1}^{an}(1^\lambda)$, and $\mathsf{Pr}[\mathbf{Game_8}] = \mathsf{Pr}\left[b' = 0 \ : \ b' \leftarrow \mathsf{Exp}_{\mathsf{A},1}^{an}(1^\lambda)\right]$. Moreover, analogously to Game 1, an adversary distinguishing Game 8 from Game 7 allows to construct a distinguisher B_8 that has advantage

$$\epsilon_8 = \frac{1}{2}\left|\mathsf{Pr}\left[\mathbf{Game_7}\right] - \mathsf{Pr}\left[\mathbf{Game_8}\right]\right| ,$$

in breaking the zero-knowledge property of the proof system.

This yields that the advantage of A in breaking anonymity is bounded by

$$\begin{aligned}
\epsilon &= \frac{1}{2}\left|\mathsf{Pr}\left[b'=0 \;:\; b' \leftarrow \mathsf{Exp}^{an}_{\mathsf{A},0}(1^\lambda)\right] + \mathsf{Pr}\left[b'=0 \;:\; b' \leftarrow \mathsf{Exp}^{an}_{\mathsf{A},1}(1^\lambda)\right]\right| \\
&= \frac{1}{2}\left|\mathsf{Pr}\left[\mathbf{Game_0}\right] - \mathsf{Pr}\left[\mathbf{Game_8}\right]\right| \\
&= \frac{1}{2}\left|\sum_{i=0}^{7} \mathsf{Pr}\left[\mathbf{Game_i}\right] - \mathsf{Pr}\left[\mathbf{Game_{i+1}}\right]\right| \\
&\leq \sum_{i=0}^{7} \frac{1}{2}\left|\mathsf{Pr}\left[\mathbf{Game_i}\right] - \mathsf{Pr}\left[\mathbf{Game_{i+1}}\right]\right| \\
&= \epsilon_1 + \epsilon_2 + \epsilon_3 + \frac{1}{2}\epsilon_4 + \epsilon_5 + \epsilon_6 + \frac{1}{2}\epsilon_7 + \epsilon_8 \\
&= 2\epsilon_{ZK} + \epsilon_{SS} + 4\epsilon_{CPA}\,,
\end{aligned}$$

where ϵ_{ZK}, ϵ_{SS}, and ϵ_{CPA} are the advantage in breaking the zero-knowledge property of the NIZK proof system, the simulation soundness property of the NIZK proof system and the IND-CPA security of the encryption scheme. □

Theorem 5.24 (Traceability)**.** *The group signature scheme* GS *satisfies traceability in the Random Oracle Model if the signature scheme is eu-acma secure and the proof system is a sound argument of knowledge.*

Proof. Assume A is a PPT algorithm that breaks the traceability of the GS with non-negligible advantage $\mathsf{Adv}^{\mathsf{A}}_{trac}(\lambda)$. We define an adversary B (shown in Figure 5.11) that breaks the eu-acma security of the signature with non negligible probability, assuming the NIZK proof is a sound argument of knowledge.

The simulator B runs the eu-acma experiment in Figure 5.5. At the beginning of the experiment, B receives the signature verification key *svk*, and access to a signing oracle $\mathcal{O}_{\mathsf{Sign}}$.

B instantiates the group signature GS as follows. It generates honestly the opener's keys (opk, osk) running EKeyGen, and sets the issuer's public key to be $gpk = svk$. Then it implements the oracles according to their definition (cf. Section 5.1.2), except for $\mathcal{O}_{\mathsf{Iss}}$ and AddU. When answering these oracles, B produces the signature $\mathbf{S}$ by querying the signature oracle $\mathcal{O}_{\mathsf{Sign}}$ as shown in Figure 5.11.

At the end of the experiment, the adversary A outputs a valid pair message-signature (μ^*, σ^*). Consider the following events:

E_1**:** the signature is valid but the opener cannot recover a valid signer's identity: $1 \leftarrow \mathsf{GVerify}(\mu^*, \sigma^*, gpk, opk) \;\wedge\; (i^* = 0)$;

E_2: the signature is valid, the opener can recover a valid identity but it cannot prove that the opening was performed correctly: $1 \leftarrow \mathsf{GVerify}(\mu^*, \sigma^*, gpk, opk) \wedge (i^* \neq 0) \wedge 0 \leftarrow \mathsf{GJudge}(gpk, opk, upk_{i^*}, \mu^*, \sigma^*, i^*, \tau^*)$;

E: $E_1 \vee E_2$;

S: the signature is valid and the opening is correct: $(gpk, opk, c_0, c_1, gsk_i, \mathbf{e}) \in \mathscr{R}_S$;

where $(i^*, \tau^*) \leftarrow \mathsf{GOpen}(\mu^*, \sigma^*, gpk, osk)$. Notice that E_1 and E_2 are disjoint. Then we can write the advantage of A in breaking the traceability of GS as:

$$\mathsf{Adv}^{\mathsf{A}}_{trac}(\lambda) = \Pr[E] = \Pr[E \wedge \bar{S}] + \Pr[E_1 \wedge S] + \Pr[E_2 \wedge S] \,.$$

We now compute the three probabilities.

The event $E \wedge \bar{S}$ (remark that $\bar{S}$ means that the signature is valid, as we assume A outputs a signature that passes verification, and the opening is *not* correct) corresponds to A breaking the soundness of the SNARK. Indeed, if the opening is not correct either there is no entry $\mathbf{reg}[i]$ or $\mathbf{reg}[i]$ is not equal to the output of the decryption (i is the decrypted identity). In both cases B can parse $\sigma^* = (\Pi^*_S, \mathbf{V}^*_0, \mathbf{W}^*_0, \mathbf{V}^*_1, \mathbf{W}^*_1)$ and send $(\Pi^*_S, \mu^*, gpk, opk, (\mathbf{V}^*_0, \mathbf{W}^*_0, \mathbf{V}^*_1, \mathbf{W}^*_1))$ as a proof of false statement. From the previous analysis it holds:

$$\begin{aligned} \Pr[E \wedge \bar{S}] &= \Pr[1 \leftarrow \mathsf{GVerify}(\mu^*, \sigma^*, gpk, opk) \wedge (gpk, opk, c_0, c_1, gsk_i, \mathbf{e}) \notin \mathscr{R}_S] \\ &\leq 2^{-\lambda} \end{aligned}$$

In the event $E_1 \wedge S$ the opener decrypts the ciphertext contained in the signature to obtain a user's identity and public key $(i, \mathbf{c}'_i, \mathbf{d}'_i)$. Then, it recovers the i-th entry of the list $\mathbf{reg}[i]$. There can be two cases: either $\mathbf{reg}[i] = \perp$ or $\mathbf{reg}[i] = (\mathbf{c}_i, \mathbf{d}_i, ots_i)$ and $(\mathbf{c}_i, \mathbf{d}_i) \neq (\mathbf{c}'_i, \mathbf{d}'_i)$. Both cases implies that A did not query the signing oracle on $(\mathbf{c}'_i, \mathbf{d}'_i)$. Hence, B can break the unforgeability of the signature scheme using the extractor of the SNARK to obtain a valid signature $\bar{\mathbf{S}}$ on $(\mathbf{c}'_i, \mathbf{d}'_i)$. Therefore,

$$\Pr[E_1 \wedge S] \leq 8(1 - \nu(\lambda))\mathsf{Adv}^{eu-acma}_{\mathsf{B}}(\lambda) \,,$$

where $1 - \nu(\lambda)$ comes from the success probability of the extractor and the factor 8 from the Generalized Forking Lemma (Lemma 2.6), that is needed to get the input for the extractor. The runtime of B is $t_{\mathsf{A}} \cdot 8n^2 q_H / \epsilon_{\mathsf{A}} \cdot \ln(8n/\epsilon_{\mathsf{A}})$, where t_{A} is the runtime of A, $\epsilon_{\mathsf{A}} = \Pr[E_1 \wedge S]$, and $q_H = poly(\lambda)$ as an algorithm that runs in polynomial time can query the random oracle at most a polynomial number of times. Remark that B makes N queries to the signing oracle as N is the bound on the number of users supported by the scheme.

Finally, it holds that $\mathsf{Pr}[E_2 \wedge S] = 0$. To verify this claim, let us analyze the algorithm GJudge. Upon receiving $(gpk, opk, upk_i, \mu, \sigma, i, \tau)$, the algorithm parses $\tau = (\mathbf{V}_0, \mathbf{W}_0, i, \mathbf{c}'_i, \mathbf{d}'_i, \mathbf{a}_0, \mathbf{b}_0, \Pi_O)$, and recovers $\mathbf{reg}[i]$. Given that the opener outputs $(i, \tau) \neq (0, \epsilon)$, the user's public key obtained by the opener from the decryption $(\mathbf{c}'_i, \mathbf{d}'_i)$ should be equal to the entry $\mathbf{reg}[i] = (\mathbf{c}, \mathbf{d})$ output by the Opener, i.e., $(\mathbf{c}'_i, \mathbf{d}'_i) = (\mathbf{c}_i, \mathbf{d}_i)$. All the verification algorithms go through, as the opener executed honestly the verification of the one-time signature *ots*, and the event S implies that $(gpk, opk, c_0, c_1, gsk_i, \mathbf{e}) \in \mathscr{R}_S$, hence the verification of the proof Π_O produced by the opener outputs 1. Therefore, if the opener is honest the algorithm GJudge outputs 1 and the claim follows.

In conclusion, the advantage of A against GS is

$$\mathsf{Adv}^{trac}_{\mathsf{A}}(\lambda) \leq 2^{-\lambda} + 8(1 - \nu(\lambda))\mathsf{Adv}^{eu-acma}_{\mathsf{B}}(\lambda) + \nu(\lambda)\,,$$

that is negligible if the signature is eu-acma secure and the proof system is sound and a proof of knowledge. □

Theorem 5.25 (Non-Frameability)**.** *The group signature scheme* GS *satisfies non-frameability in the Random Oracle Model if the proof system is a zero-knowledge argument of knowledge, the OTS is a OTS, and* $RLWE_{1,\mathscr{U}(\mathscr{S}_1)}$ *is hard.*

Proof. Let A be a PPT algorithm which wins the non-frameability experiment in Figure 5.6 with advantage $\epsilon_{\mathsf{A}} = \mathsf{Adv}^{non-fr}_{\mathsf{A}}(\lambda)$. We start analyzing the event $E =$"A succeeds". Consider the following events:

F: $1 \leftarrow \mathsf{GVerify}(\mu^*, \sigma^*, gpk, opk)$, $\mathbf{HU}[i^*] \neq \perp$, A did not query $\mathsf{USK}(i)$, $1 \leftarrow \mathsf{GJudge}(gpk, opk, upk_{i^*}, \mu^*, \sigma^*, i^*, \tau^*)$, $(i^*, \mu^*) \notin \mathbf{GSig}$, where $(i^*, \tau^*) \leftarrow \mathsf{GOpen}(\mu^*, \sigma^*, gpk, osk)$;

S_1: $(\mu^*, \sigma^*, gpk, opk) \in \mathscr{R}_S$;

S_2: $(gpk, opk, \mu^*, \sigma^*, \tau^*, i^*) \in \mathscr{R}_O$;

P: $(\mathbf{c}'_i, \mathbf{d}'_i) = (\mathbf{c}_i, \mathbf{d}_i)$, where $(\mathbf{c}_i, \mathbf{d}_i)$ is obtained from $\mathbf{reg}[i]$ and $(\mathbf{c}'_i, \mathbf{d}'_i)$ from the opening.

Then the advantage of the adversary in winning the non-frameability experiment is

$$\mathsf{Adv}^{nf}_{\mathsf{A}}(\lambda) = \mathsf{Pr}[F] \leq \mathsf{Pr}[F \wedge \bar{S}_1] + \mathsf{Pr}[F \wedge \bar{S}_2] + \mathsf{Pr}[F \wedge P \wedge S_1 \wedge S_2] + \mathsf{Pr}[F \wedge \bar{P} \wedge S_1 \wedge S_2]$$

because the events S_1 and S_2 are not disjoint[6]. In the following we compute the probabilities of these events.

The soundness of the proof systems yields that $\mathsf{Pr}[F \wedge \bar{S}_1] \leq 2^{-\lambda}$ and $\mathsf{Pr}[F \wedge \bar{S}_1] \leq 2^{-\lambda}$.

If the event $F \wedge P \wedge S_1 \wedge S_2$ happens, we can construct an algorithm B_1 that solves the Search version of RLWE, i.e., that given $(\bar{\mathbf{c}}, \bar{\mathbf{d}})$ finds $\bar{\mathbf{t}}$ such that $\bar{\mathbf{d}} = \bar{\mathbf{c}}\bar{\mathbf{s}} + \bar{\mathbf{e}}$ for some small error $\bar{\mathbf{e}}$.

Lemma 5.26. $\mathsf{Pr}[F \wedge P \wedge S_1 \wedge S_2] \leq \mathsf{Adv}_{\mathsf{B}}^{RLWE}(\lambda) \cdot 8N(\lambda)(1 - \nu(\lambda))$.

Proof. The algorithm B_1 (cf. Figure 5.12) is given access to an oracle $\mathcal{O}_{RLWE}$ that outputs RLWE pairs $(\mathbf{c}, \mathbf{b})$ when prompted, and simulates all oracles according to their definitions but USK, $\mathcal{O}_{\mathsf{Iss}}$, SndToU and $\mathcal{O}_{\mathsf{GSign}}$. Indeed, B samples a random u and simulates all honest users honestly but the u-th, whose RLWE pair is set to be the pair $(\bar{\mathbf{c}}, \bar{\mathbf{d}})$ output by $\mathcal{O}_{RLWE}$. Then, B simulates USK honestly unless A queries for the user u; in such case, B aborts. If the adversary runs $\mathcal{O}_{\mathsf{Iss}}$ with user identity u (where u has not been assigned yet), B_1 samples another u and then runs the algorithm Iss honestly. Whenever A queries the oracle SndToU for the user u, B sends $(\bar{\mathbf{c}}, \bar{\mathbf{d}})$ as user's keys, and simulates the SNARK of $\bar{\mathbf{t}}$. In a similar way, whenever the user queries $\mathcal{O}_{\mathsf{GSign}}$ for a signature by the user u, B simulates the proof Π_S. If the adversary successfully outputs a forgery, this means that A was able to generate a SNARK of, among other things, small $\bar{\mathbf{t}}$ and $\bar{\mathbf{e}}$ such that $\bar{\mathbf{d}} = \bar{\mathbf{c}}\bar{\mathbf{s}} + \bar{\mathbf{e}}$. Hence, B can recover $\bar{\mathbf{t}}$ rewinding A and exploiting the extractor of the proof system $(\mathsf{P}_S, \mathsf{V}_S)$. Given that solving the search version of RLWE is equivalent to solving the decisional version (cf. [Lyubashevsky et al., 2010]), the advantage of B in solving RLWE is $\mathsf{Pr}[F \wedge P \wedge S_1 \wedge S_2] \leq \mathsf{Adv}_{\mathsf{B}}^{RLWE}(\lambda) \cdot 8N(\lambda)(1 - \nu(\lambda))$ where the factor 8 comes from the Generalized Forking Lemma 2.6, $N(\lambda)$ is the number of users and $(1 - \nu(\lambda))$ comes from the success probability of the extractor of the SNARK. □

Finally, consider the event $F \wedge \bar{P} \wedge S_1 \wedge S_2$. We construct an algorithm B_2 that breaks the unforgeability of the OTS exploiting A.

Lemma 5.27. $\mathsf{Adv}_{\mathsf{B}}^{OTS}(\lambda) \geq \mathsf{Pr}[F \wedge \bar{P} \wedge S_1 \wedge S_2] \cdot 1/N(\lambda)$ *where* $N(\lambda)$.

[6] Indeed, if S_1 and S_2 are not disjoint events it holds that, for all sets P in the set space Ω:

$$\bar{S}_1 \cup \bar{S}_2 \cup \big[(P \cap S_1 \cap S_2) \cup (\bar{P} \cap S_1 \cap S_2)\big] = \bar{S}_1 \cup \bar{S}_2 \cup (S_1 \cap S_2) = \Omega \,,$$

hence $\{\bar{S}_1, \bar{S}_2, (P \cap S_1 \cap S_2), (\bar{P} \cap S_1 \cap S_2)\}$ covers the space Ω.

Proof. B_2 (cf. Figure 5.13) has access to an oracle $\mathcal{O}_{OTS}$ that, when prompted, outputs a verification key *otsvk*, and that allows querying (only) one signature w.r.t. such key pair on a message of B_2's choice. Again, B_2 samples a random user identity u, and simulates all the oracle honestly but USK, $\mathcal{O}_{\mathsf{Iss}}$, SndToU, and AddU. If A queries USK for the user u, B_2 aborts. Otherwise, B_2 simulates all other users honestly (according to the definitions in Section 5.1.2). If it runs $\mathcal{O}_{\mathsf{Iss}}$ with user identity u (where u has not been assigned yet), B_2 samples another u and then runs the algorithm Iss honestly. If A prompts SndToU for user u, B_2 generates the RLWE pair $(\mathbf{c}_u, \mathbf{d}_u)$, and prompts $\mathcal{O}_{OTS}$ to get the key pair $(otsvk, otssk)$ and a signature on $(\mathbf{c}_u, \mathbf{d}_u)$. Then it generates the proof Π_I and sends everything to A. Otherwise, B_2 executes SndToU according to the definition in Section 5.1.2. Finally, if A asks to add a honest user i to AddU, B_2 behaves honestly but in case $i = u$, when it gets the one-time signature from $\mathcal{O}_{OTS}$.

When A outputs the forged signature σ^*, B runs the opening algorithm on it to find the targeted identity and aborts if the target user is not u. Otherwise, according to the definition of the event $F \wedge \bar{P} \wedge S_1 \wedge S_2$, the pair $(\mathbf{c}'_u, \mathbf{d}'_u)$ output by the opening is not equal to the pair $(\mathbf{c}_u, \mathbf{d}_u)$ contained in $\mathbf{reg}[u]$. Therefore, the one-time signature ots'_u output by the opener (which is valid w.r.t. , as GJudge outputs 1) is a valid signature w.r.t. the user public key $upk_u = otsvk$ on a message $(\mathbf{c}'_u, \mathbf{d}'_u)$ that was not queried to the signing oracle (as $\mathcal{O}OTS$ was only queried for a signature on $(\mathbf{c}_u, \mathbf{d}_u)$). Hence, B can output $((\mathbf{c}'_u, \mathbf{d}'_u), ots'_u)$ to win the eu-acma experiment, and it holds $\mathsf{Adv}^{OTS}_{\mathsf{B}}(\lambda) \geq \mathsf{Pr}[F \wedge \bar{P} \wedge S_1 \wedge S_2] \cdot 1/N(\lambda)$ where $N(\lambda)$. □

Hence, the advantage of A is

$$\mathsf{Adv}^{nf}_{\mathsf{A}}(\lambda) \leq 2^{-\lambda+1} + 8N(k)(\mathsf{Adv}^{RLWE}_{\mathsf{B}}(\lambda)(1-\nu(\lambda)) + N(\lambda)\mathsf{Adv}^{OTS}_{\mathsf{B}'}(\lambda)) \,.$$

□

5.5.6 Parameters and Storage Requirements

We compute parameters for $\lambda = 128$ bits of security in the "paranoid" framework of Alkim et al. [2016], that in particular requires $\delta \leq 1.00255$. We intend "security" here as the claim that the underlying hardness assumptions are hard to solve for a quantum computer. We choose as ring the polynomial ring R_q defined by $n = 2^{10}$ and a prime $2^{64} < q < 2^{65}$. Such choice of degree guarantees that the set $\mathcal{S}_1$ contains more than 2^{256} elements, hence finding the user's secret $\mathbf{t}_i$ through a brute-force attack is not possible. The number N of supported users is

2^{26}. For technical reasons, Aurora requires that $\mathbb{F}_q$ has a large power-of-2 multiplicative subgroup, and so we choose q accordingly (most choices of q satisfy this requirement).

This implies that the unforgeability of the signature scheme is based on a $\text{RSIS}_{d,\beta}$ instance where $d = 68$ and $\beta \leq 2^{46}$, and on a $\text{RLWE}_{l,\chi}$ instance with $l < 2^{25}$. To estimate their hardness, we use the root Hermite factor δ (cf. [Micciancio and Regev, 2009]), and we obtained a $\delta_{RSIS} \leq 1.00062$ and $\delta_{RLWE} \leq 1.00001$.

The delta is chosen taking into account the tightness loss in the unforgeability proof of the signature due to the use of complexity leveraging.

We now compute the length of the keys and of a signature output by the group signature. An element in R_q can be stored in $nk \leq 8.32$ KB. The opener's secret key is composed by one ring element, hence it can be stored in 8.32 KB, while the opener's public key in 33.28 KB (as it is composed by 4 ring elements).

The group manager's public key requires a bit of care. Indeed, the key $(\mathbf{A}, \mathbf{B}, \mathbf{A}_0, \ldots, \mathbf{A}_\ell, \mathbf{u})$ includes $\mathbf{A} = \begin{bmatrix} \mathbf{a} & \mathbf{1} \end{bmatrix}$ and $\mathbf{B} \in R_q^{1 \times \bar{m}}$ that are generated with the trapdoor (cf. Section 5.4.1), ℓ random vectors with $\bar{m} = 67$ components in R_q, where $\ell = 2nk + \lceil \log N \rceil$, and a random element $\mathbf{u} \in R_q$. Storing these would require $nk \cdot (1 + \bar{m} + \bar{m} \cdot \ell + 1) = 2^{10} \cdot 65 \cdot (1 + 67 + 67 \cdot 2^{18} + 1) = 146$ GB, and it is clearly infeasible. Instead, the issuer can send a condensed (pseudorandom) representation of the random elements $\mathbf{A}_0, \ldots, \mathbf{A}_\ell, \mathbf{u}$, having considerably smaller size. The size of the public key then becomes the size of such a representation plus $(\bar{m} + 1)nk \leq 0.57$ MB.

The group manager's secret key is the trapdoor $\mathbf{T}_\mathbf{A}$, that has components with coefficients smaller than $8s_{ssk} = 8\sqrt{\log(n^2)} + 1$ (cf Lemma 2.16 and Theorem 5.14). Hence the size of $\mathbf{T}_\mathbf{A}$ is $2kn \log(8s_{ssk}) \leq 91$ KB.

At the end of the joining phase the user obtains the credential $(\mathbf{c}_i, \mathbf{d}_i, \mathbf{t}_i, i, \mathbf{S}_i)$, where the vector $\mathbf{S}_i$ is composed by $2\bar{m}+2$ ring elements with coefficients smaller than $8s_\sigma = 8\sqrt{n \log n \log n^2}$ (cf. Section 5.4.1). Hence it has size $3nk + \lceil \log N \rceil + (2\bar{m} + 2)n \log(8\sqrt{n \log n \cdot \log n^2}) \leq 231$ KB. The secret signing key of the OTS can be discarded after the joining phase.

Finally, a signature is composed by the NIZK proof Π_S, and 4 vectors of elements in the ring. The proof length is around 200 KB (estimate from [Ben-Sasson et al., 2018]). The vectors $\mathbf{V}_0, \mathbf{W}_0, \mathbf{V}_1, \mathbf{W}_1$ are the encryptions of two ring elements $(\mathbf{c}_i, \mathbf{d}_i)$ and a number $i < N$. As the encryption algorithm converts them into polynomials in $\mathcal{S}_1$ whose coefficients are the bits of their binary expansions, each vector is composed by $\lceil (2nk + \lceil \log N \rceil)/n \rceil = 2k + \lceil \lceil \log N \rceil / n \rceil$ elements in R_q, hence $(\mathbf{V}_0, \mathbf{W}_0, \mathbf{V}_1, \mathbf{W}_1)$ has size $(2k + \lceil \lceil \log N \rceil / n \rceil) \cdot nk \leq (2 \cdot 65 + \lceil 26 \cdot 2^{-10} \rceil) \cdot 2^{10} \cdot 65 = 131 \cdot 1024 \cdot 65 = 1.09$ MB. Hence a signature is roughly 1.29 MB long.

To compare our scheme with previous ones (such as [del Pino et al., 2018]

or [Boschini et al., 2018a]), we compute the length of the signature for the case in which the group manager is assumed to be honest (by default our scheme guarantees a stronger notion of security). Modifying our signature to have a honest group manager essentially means that it is enough that during issuance the user gets a signature by the group manager on the user identity i. Hence, opening only requires the signature to contain an encryption of the user's identity i, whose bit decomposition can be encoded as one element of $\mathscr{S}_1$. Therefore, the vectors $\mathbf{V}_0, \mathbf{W}_0, \mathbf{V}_1, \mathbf{W}_1$ actually are just ring elements, hence the size of the signature is at most $200 + 4 \cdot 2^{10} \cdot 65 \leq 250$ KB (obviously, the size of the proof should shrink too, as the number of variables is smaller, but we mean this number as a rough upper bound), plus the dimension of the OTS signature and its verification key.

Simulator $\mathsf{B}^{\mathsf{A},\mathscr{O}_s}(1^\lambda)$
$\mathscr{Q}_{\mathsf{UKg}} \leftarrow \emptyset$, $\mathscr{Q}_{\mathsf{GSign}} \leftarrow \emptyset$
$(spar, svk) \leftarrow \mathscr{O}_s(1^\lambda)$
$epar \leftarrow \mathsf{DerivePar}_{\mathsf{pve}}(spar)$
Compute the parameters $ppar$ of the relaxed Σ-protocol.
$par \leftarrow (spar, epar, ppar)$
$\bar{\alpha} \xleftarrow{\$} \{0,1\}^\lambda$
$gpk \leftarrow (svk, \bar{\alpha})$
$(opk, osk) \leftarrow \mathsf{EKeyGen}(epar)$
$(\mu^*, sig^*, sig') \leftarrow \mathsf{GF}_{\mathsf{A}}^{\mathscr{O}_{\mathsf{UKg}},\mathscr{O}_{\mathsf{GSign}}}(gpk, opk, osk)$
$(\bar{\mathbf{c}}_1, \bar{\mathbf{S}}, \bar{\mathbf{c}}_2) \leftarrow \mathsf{E}(sig^*, sig')$
$id \leftarrow \mathsf{GOpen}(\mu^*, sig^*, osk)$
If $1 \leftarrow \mathsf{GVerify}(\mu^*, sig^*, gpk, opk)$
and $id \notin \mathscr{Q}_{\mathsf{UKg}} \wedge \nexists\, usk \ :\ (id, usk, \mu^*) \in \mathscr{Q}_{\mathsf{GSign}}$
Then If $\exists\ (id, \mu, sig)\ :\ sig = (\mathbf{F}, \Pi_0, t, \pi, otsvk^*, ots)$ abort.
Else return $((\mu^*, \bar{\alpha}), (\bar{\mathbf{c}}_1, \bar{\mathbf{S}}, \bar{\mathbf{c}}_2))$.
Else abort.

Oracle $\mathscr{O}_{\mathsf{UKg}}(id)$
$\mathbf{S} \leftarrow \mathscr{O}_s(id, \bar{\alpha})$
$\mathscr{Q}_{\mathsf{UKg}} \leftarrow \mathscr{Q}_{\mathsf{UKg}} \cup \{id\}$
Return usk .

Oracle $\mathscr{O}_{\mathsf{GSign}}(id, \mu)$
If $id \in \mathscr{Q}_{\mathsf{UKg}}$ abort.
If $\exists\ (id, \mu, sig) \in \mathscr{Q}_{\mathsf{GSign}}$ return sig.
$(otssk, otsvk) \leftarrow \mathsf{OTSGen}(1^\lambda)$
$\mathbf{E} \xleftarrow{\$} R_3^{1\times m}$
$\mathbf{b} \xleftarrow{\$} R_3$
$\mathbf{F} \leftarrow \mathbf{b}^{-1}(\mathbf{C} + \mathbf{mG} + \mathbf{E}) \bmod q$
$\Pi_0 = (\mathbf{T}_0, \mathbf{c}_0, \mathbf{Z}_0) \leftarrow \Sigma_0(ppar, \begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{F} & \mathbf{1}\end{bmatrix}, H(\bar{\alpha}))$
Program $H_0(\mathbf{F}, \mathbf{T}_0, \mathbf{A}, \bar{\alpha}, otsvk) \leftarrow \mathbf{c}_0$ (aborts if it was already set).
$(t, \pi) \leftarrow \mathsf{Enc}(opk, (\begin{bmatrix}\mathbf{G}^\mathsf{T} & \mathbf{F}^\mathsf{T} & \mathbb{I}_m\end{bmatrix}, -\mathbf{C}^\mathsf{T}), (\mathbf{m}, \begin{bmatrix}-\mathbf{b} & ; & \mathbf{E}^\mathsf{T}\end{bmatrix}, \mathbf{1}), otsvk)$
$ots \leftarrow \mathsf{OTSSign}(otssk, (\mathbf{A}, \mathbf{B}, \mathbf{F}, \mathbf{u}, \Pi_0, t, \pi, \mu))$
$sig = (\mathbf{F}, \Pi_0, t, \pi, otsvk, ots)$
$\mathscr{Q}_{\mathsf{GSign}} \leftarrow \mathscr{Q}_{\mathsf{GSign}} \cup \{(id, \mu, sig)\}$
Return sig .

Figure 5.8. Simulator of the traceability experiment.

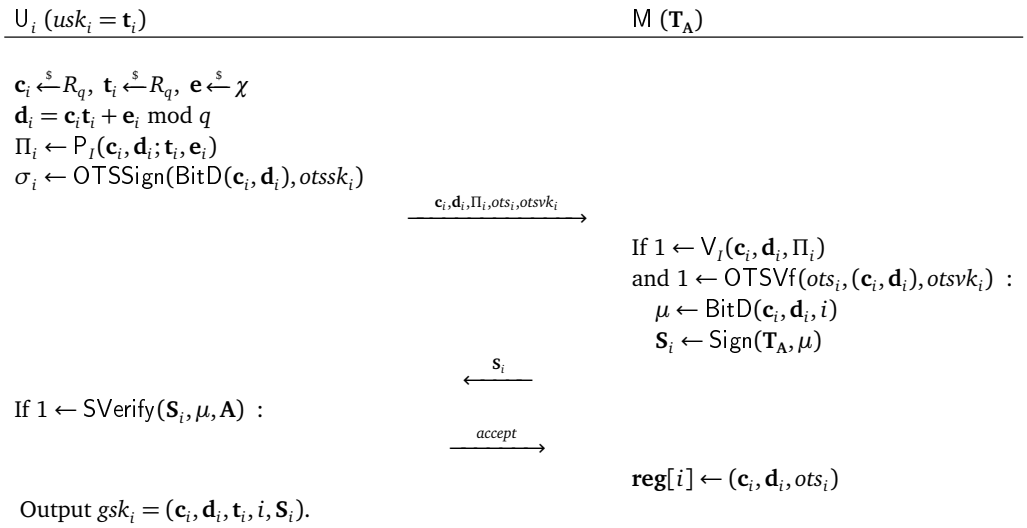

Figure 5.9. Joining protocol.

$\mathsf{GSign}(gsk_i, gpk, opk, \mu)$

Parse $gsk_i = (\mathbf{c}_i, \mathbf{d}_i, \mathbf{t}_i, i, \mathbf{S}_i)$ and $opk = (\mathbf{a}_0, \mathbf{b}_0, \mathbf{a}_1, \mathbf{b}_1)$

For $b = 0, 1$

$(\mathbf{V}_i, \mathbf{W}_i) \leftarrow \mathsf{Enc}_{RLWE}(\mathsf{BitD}(\mathbf{c}_i, \mathbf{d}_i, i), (\mathbf{a}_b, \mathbf{b}_b))$

$\Pi_S \leftarrow \mathsf{P}_S(\mu; gpk, opk, (\mathbf{V}_0, \mathbf{W}_0), (\mathbf{V}_1, \mathbf{W}_1), \mathbf{t}_i, i, \mathbf{c}_i, \mathbf{d}_i, \mathbf{e}_i, \mathbf{S}_i)$

Return $\sigma = (\Pi_S, \mathbf{V}_0, \mathbf{W}_0, \mathbf{V}_1, \mathbf{W}_1)$.

Figure 5.10. Signing algorithm

$\mathsf{B}^{\mathcal{O}_{\mathsf{Sign}},\,\mathsf{A}}(gpk)$
$(opk, osk) \leftarrow \mathsf{EKeyGen}_{RLWE}(1^\lambda)$
$\mathbf{CU} \leftarrow \emptyset$, $\mathbf{HU} \leftarrow \emptyset$, $\mathbf{GSig} \leftarrow \emptyset$
$(\mu^*, sig^*, sig') \leftarrow \mathsf{GF}_{\mathsf{A}}^{\mathcal{O}_{\mathsf{Iss}},\,\mathsf{AddU},\,\mathsf{USK},\,\mathsf{CrptU},\,\mathsf{RReg}}(gpk, opk, osk)$
If $0 \leftarrow \mathsf{GVerify}(\mu^*, \sigma^*, gpk, opk)$, abort.
Else $(i^*, \tau^*) \leftarrow \mathsf{GOpen}(\mu^*, \sigma^*, gpk, osk)$.
If $(i^* = \epsilon) \vee 0 \leftarrow \mathsf{GJudge}(gpk, opk, upk_{i^*}, \mu^*, \sigma^*, i^*, \tau^*)$ abort.
Else $((\bar{\mathbf{c}}_i, \bar{\mathbf{d}}_i, \bar{i}), \bar{\mathbf{S}}) \leftarrow \mathsf{E}(sig^*, sig')$.
Return $((\bar{\mathbf{c}}_i, \bar{\mathbf{d}}_i, \bar{i}), \bar{\mathbf{S}})$.

$\mathcal{O}_{\mathsf{Iss}}(\mathbf{c}_i, \mathbf{d}_i, \Pi_i, ots_i, otsvk_i)$
$b \leftarrow \mathsf{V}_I(\mathbf{c}_i, \mathbf{d}_i, \Pi_i)$
$b' \leftarrow \mathsf{OTSVf}(ots_i, (\mathbf{c}_i, \mathbf{d}_i), otsvk_i)$
If $(b = 0 \ \vee \ b' = 0)$ abort.
$\mu \leftarrow \mathsf{BitD}(\mathbf{c}_i, \mathbf{d}_i, i)$
$\mathbf{S}_i \leftarrow \mathcal{O}_{\mathsf{Sign}}(\mu)$
$\mathbf{reg}[i] \leftarrow (\mathbf{c}_i, \mathbf{d}_i, ots_i, otsvk_i)$
$l[i] \leftarrow (\mathbf{c}_i, \mathbf{d}_i, ots_i, otsvk_i)$
Return $i, \mathbf{S}_i$

$\mathsf{AddU}(i)$
If $i \in \mathbf{CU} \cup \mathbf{HU}$ abort.
$(otssk, otsvk) \leftarrow \mathsf{OTSGen}(1^\lambda)$
$\mathbf{HU}[i] \leftarrow (otsvk, otssk)$
$\mathbf{c}_i \xleftarrow{\$} R_q$, $\mathbf{t}_i \xleftarrow{\$} R_q$, $\mathbf{e} \xleftarrow{\$} \chi$
$\mathbf{d}_i = \mathbf{c}_i \mathbf{t}_i + \mathbf{e}_i \bmod q$
$\Pi_i \leftarrow \mathsf{P}_I(\mathbf{c}_i, \mathbf{d}_i; \mathbf{t}_i, \mathbf{e}_i)$
$ots_i \leftarrow \mathsf{OTSSign}(\mathsf{BitD}(\mathbf{c}_i, \mathbf{d}_i), otssk_i)$
$\mu \leftarrow \mathsf{BitD}(\mathbf{c}_i, \mathbf{d}_i, i)$
$\mathbf{S}_i \leftarrow \mathcal{O}_{\mathsf{Sign}}(\mu)$
$\mathbf{reg}[i] \leftarrow (\mathbf{c}_i, \mathbf{d}_i, ots_i)$
$l[i] \leftarrow (\mathbf{c}_i, \mathbf{d}_i, ots_i, otsvk_i)$
$gsk_i = (\mathbf{c}_i, \mathbf{d}_i, \mathbf{t}_i, i, \mathbf{S}_i)$.
$\mathbf{HK}[i] \leftarrow gsk_i$
Return $upk_i = otsvk$.

Figure 5.11. Simulator for the proof of traceability.

$\mathsf{B}_1^{\mathcal{O}_{RLWE},\,\mathsf{A}}(1^\lambda)$
$(gpk, gsk, opk, osk) \leftarrow \mathsf{GKg}(1^\lambda)$
$\mathbf{CU} \leftarrow \emptyset$, $\mathbf{HU} \leftarrow \emptyset$, $\mathbf{GSig} \leftarrow \emptyset$
$u \xleftarrow{\$} \{1, \ldots, N\}$
$(\mu^*, \sigma_1^*, \sigma_2^*) \leftarrow \mathsf{GF}_{\mathsf{A}}^{\mathcal{O}_{\mathsf{GSign}},\,\mathsf{AddU},\,\mathsf{USK},\,\mathsf{CrptU},\,\mathsf{RReg},\,\mathsf{WReg},\,\mathsf{SndToU}}(gpk, gsk, opk, osk)$
$(i, \tau) \leftarrow \mathsf{GOpen}(\sigma_1^*, osk)$
If $i \neq u$ abort.
For $j = 1, 2$, parse $\sigma_j^* = (\Pi_{S,j}^*, \mathbf{V}_{0,j}^*, \mathbf{W}_{0,j}^*, \mathbf{V}_{1,j}^*, \mathbf{W}_{1,j}^*)$
$(\bar{\mathbf{t}}_u, \bar{u}, \bar{\mathbf{c}}_u, \bar{\mathbf{d}}_u, \bar{\mathbf{e}}_u, \bar{\mathbf{S}}_u) \leftarrow \mathsf{E}_S(\Pi_{S,1}^*, \Pi_{S,2}^*)$
Return $(\bar{\mathbf{t}}_u, \bar{\mathbf{e}}_u)$.

$\mathcal{O}_{\mathsf{SndToU}}(i, aux)$
If $\mathbf{HU}[i] = \perp$ abort.
Parse $\mathbf{HU}[i] = (upk_i, usk_i)$.
If $aux = \perp$
If $i = u$
$(\bar{\mathbf{c}}, \bar{\mathbf{d}}) \leftarrow \mathcal{O}_{RLWE}$
$\Pi_u \leftarrow \Sigma_I(\bar{\mathbf{c}}, \bar{\mathbf{d}})$
$ots \leftarrow \mathsf{OTSSign}(\mathsf{BitD}(\bar{\mathbf{c}}, \bar{\mathbf{d}}), usk_u)$
Return $\bar{\mathbf{c}}, \bar{\mathbf{d}}, \Pi_u, ots_u, otsvk_u$.
Else
$\mathbf{c}_i \xleftarrow{\$} R_q$, $\mathbf{t}_i \xleftarrow{\$} R_q$, $\mathbf{e} \xleftarrow{\$} \chi$
$\mathbf{d}_i = \mathbf{c}_i \mathbf{t}_i + \mathbf{e}_i \bmod q$
$\Pi_i \leftarrow \mathsf{P}_I(\mathbf{c}_i, \mathbf{d}_i; \mathbf{t}_i, \mathbf{e}_i)$
$ots_i \leftarrow \mathsf{OTSSign}(\mathsf{BitD}(\mathbf{c}_i, \mathbf{d}_i), usk_i)$
Return $\mathbf{c}_i, \mathbf{d}_i, \Pi_i, ots_i, otsvk_i$.
Else
Parse $aux = \mathbf{S}_i$
$\mathsf{BitD}(\mathbf{c}_i, \mathbf{d}_i, i)$
If $0 \leftarrow \mathsf{SVerify}(\mathbf{S}_i, \mu, \mathbf{A})$ abort.
$\mathbf{reg}[i] \leftarrow (\mathbf{c}_i, \mathbf{d}_i, ots_i)$
$gsk_i = (\mathbf{c}_i, \mathbf{d}_i, \mathbf{t}_i, i, \mathbf{S}_i)$
$\mathbf{HU}[i] \leftarrow (upk_i, usk_i)$, $\mathbf{HK}[i] \leftarrow gsk_i$
Return *accept*

$\mathsf{USK}(i)$
If $i = u$ abort.
If $\mathbf{HU}[i] = \perp \vee \mathbf{HK}[i] = \perp$ abort.
Return usk_i, gsk_i.

$\mathcal{O}_{\mathsf{GSign}}(i, \mu)$
If $\mathbf{HU}[i] = \perp$ abort.
Parse $\mathbf{HK}[i] = gsk_i$.
If $i = u$
For $b = 0, 1$
$(\mathbf{V}_i, \mathbf{W}_i) \leftarrow \mathsf{Enc}_{RLWE}(\mathsf{BitD}(\mathbf{c}_i, \mathbf{d}_i, i), (\mathbf{a}_b, \mathbf{b}_b))$
$\Pi_S \leftarrow \Sigma_S(\mu; gpk, opk, (\mathbf{V}_0, \mathbf{W}_0), (\mathbf{V}_1, \mathbf{W}_1))$
Else
For $b = 0, 1$
$(\mathbf{V}_i, \mathbf{W}_i) \leftarrow \mathsf{Enc}_{RLWE}(\mathsf{BitD}(\mathbf{c}_i, \mathbf{d}_i, i), (\mathbf{a}_b, \mathbf{b}_b))$
$\Pi_S \leftarrow \mathsf{P}_S(\mu; gpk, opk, (\mathbf{V}_0, \mathbf{W}_0), (\mathbf{V}_1, \mathbf{W}_1), \mathbf{t}_i, i, \mathbf{c}_i, \mathbf{d}_i, \mathbf{e}_i, \mathbf{S}_i)$
$\mathbf{GSig} \leftarrow \mathbf{GSig} \cup \{(i, \mu)\}$
Return $\sigma = (\Pi_S, \mathbf{V}_0, \mathbf{W}_0, \mathbf{V}_1, \mathbf{W}_1)$.

Figure 5.12. Simulator that solves RLWE exploiting an adversary against non-frameability.

$\mathsf{B}_2^{\mathcal{O}_{OTS}}(1^\lambda)$
$(gpk, gsk, opk, osk) \leftarrow \mathsf{GKg}(1^\lambda)$
$\mathbf{CU} \leftarrow \emptyset$, $\mathbf{HU} \leftarrow \emptyset$, $\mathbf{GSig} \leftarrow \emptyset$
$u \xleftarrow{\$} \{1, \dots, N\}$
$(\mu^*, \sigma^*) \leftarrow \mathsf{A}^{\mathcal{O}_{\mathsf{GSign}},\ \mathsf{AddU},\ \mathsf{USK},\ \mathsf{CrptU},\ \mathsf{RReg},\ \mathsf{WReg},\ \mathsf{SndToU}}(gpk, gsk, opk, osk)$
$(i, \tau) \leftarrow \mathsf{GOpen}(\sigma_1^*, osk)$
Parse $\tau = (\bar{\mathbf{c}}_u, \bar{\mathbf{d}}_u, \mathbf{c}'_u, \mathbf{d}'_u, ots'_u, otsvk_u, \Pi_O))$
Parse $\mathbf{reg}[u] = (\mathbf{c}_u, \mathbf{d}_u, ots_u)$.
If $i \neq u \ \vee\ (\mathbf{c}_u, \mathbf{d}_u) = (\mathbf{c}'_u, \mathbf{d}'_u) \ \vee\ 0 \leftarrow \mathsf{GJudge}(gpk, opk, upk_i, \mu^*, \sigma^*, i, \tau)$ abort.
Return $(\mathbf{c}'_u, \mathbf{d}'_u, ots'_u)$.

$\mathcal{O}_{\mathsf{SndToU}}(i, aux)$
If $\mathbf{HU}[i] = \perp$ abort.
Parse $\mathbf{HU}[i] = (upk_i, usk_i)$.
If $aux = \perp$
$\mathbf{c}_i \xleftarrow{\$} R_q,\ \mathbf{t}_i \xleftarrow{\$} R_q,\ \mathbf{e} \xleftarrow{\$} \chi$
$\mathbf{d}_i = \mathbf{c}_i \mathbf{t}_i + \mathbf{e}_i \bmod q$
$\Pi_i \leftarrow \mathsf{P}_I(\mathbf{c}_i, \mathbf{d}_i; \mathbf{t}_i, \mathbf{e}_i)$
If $i = u$
$ots_u \leftarrow \mathcal{O}_{OTS}(\mathsf{BitD}(\mathbf{c}_u, \mathbf{d}_u))$
Else
$ots_i \leftarrow \mathsf{OTSSign}(\mathsf{BitD}(\mathbf{c}_i, \mathbf{d}_i), usk_i)$
Return $\mathbf{c}_i, \mathbf{d}_i, \Pi_i, ots_i, otsvk_i$.
Else
Parse $aux = \mathbf{S}_i$
$\mathsf{BitD}(\mathbf{c}_i, \mathbf{d}_i, i)$
If $0 \leftarrow \mathsf{SVerify}(\mathbf{S}_i, \mu, \mathbf{A})$ abort.
$\mathbf{reg}[i] \leftarrow (\mathbf{c}_i, \mathbf{d}_i, ots_i)$
$gsk_i = (\mathbf{c}_i, \mathbf{d}_i, \mathbf{t}_i, i, \mathbf{S}_i)$
$\mathbf{HU}[i] \leftarrow (upk_i, usk_i)$, $\mathbf{HK}[i] \leftarrow gsk_i$
Return *accept*

$\mathsf{USK}(i)$
If $i = u$ abort.
If $\mathbf{HU}[i] = \perp \ \vee\ \mathbf{HK}[i] = \perp$ abort.
Return usk_i, gsk_i.

$\mathsf{AddU}(i)$
If $i \in \mathbf{HU} \cup \mathbf{CU}$ abort.
If $i = u$
$usk_u \leftarrow \perp$
$upk_i \leftarrow \mathcal{O}_{OTS}$
Else
$(otssk_i, otsvk_i) \leftarrow \mathsf{OTSGen}(1^\lambda)$
$gsk_i \leftarrow \langle \mathsf{Iss}(gsk), \mathsf{SndToU}(i, aux) \rangle$
$\mathbf{HU}[i] \leftarrow (upk_i, usk_i)$
$\mathbf{HK}[i] \leftarrow gsk_i$
Return upk_i.

Figure 5.13. Simulator that breaks the unforgeability of the OTS exploiting an adversary against non-frameability.

Chapter 6

Conclusions

In this work, we have investigated how to build lattice-based privacy-preserving signature with the goal to get shorter signatures.

The basic building block we have used was the relaxed NIZK prof obtainable from the digital signature scheme presented in [Lyubashevsky, 2012]. As already observed in literature [Benhamouda et al., 2015; Lyubashevsky and Neven, 2017], modifying such scheme required new, generic, relaxed definitions, in particular of special soundness, as the scheme did not allow to extract a witness for the original relation. Hence, in this work we have given formal definitions for lattice based primitive that take into account this relaxation, and gave a suite of relaxed lattice based primitives to be combined with such NIZK proof. In particular, we defined a relaxed signature and commitment scheme, where the first is obtained as a mix of the signature scheme by Boyen [Boyen, 2010] and the signature by Gentry et al. [2008]. With these three building blocks we were able to construct a lattice-based Anonymous Attribute Tokens, which improved the original construction by Camenisch et al. [2012] in terms of size of the token. Adding a modified version of the verifiable encryption scheme by Lyubashevsky and Neven [2017] and improving the underlying relaxed NIZK proof, we were able to construct a dynamic group signature that allowed to produce signatures of less than 2 MB, thus improving the state of the art.

Finally, we constructed a more efficient SNARK for lattice relations by adapting Aurora [Ben-Sasson et al., 2019]. This resulted in a group signature that produce the shortest signatures (estimated to be less than 0.3 MB) among the group signatures with comparable security guarantee that have been proposed so far (to the best of our knowledge).

Besides the obvious conclusions (lattice-based protocols produce cryptographic artifacts that have in general bigger size than their classical counterpart),

comparing the schemes we have built yielded quite an interesting insight on how to build privacy-preserving signatures from lattice-based building blocks.

For example, comparing the group signature we have obtained in Section 5.2 with the one in Section 5.3, we found that more efficient direct construction can be built by breaking open the different building blocks. This is analogous to the non-lattice-based world where generic, modular constructions [Camenisch et al., 2016] are often considerably less efficient than direct schemes [Ateniese et al., 2000; Camenisch and Lysyanskaya, 2003].

Moreover, we found that exact extractability is not a requirement to build privacy-preserving signature from lattice hardness assumptions, but that in fact it is possible to compensate the tightness loss with a careful choice of the parameters. However, due to the lack of clarity on how to exactly set the value of these parameters (as it can be observed following the NIST standardization process), this approach might be not the way to go (even more so considering the need for either complexity leveraging or new hardness assumptions).

We believe that some contributions are of independent interest, in particular the formal definitions for lattice based primitive that allow to exploit the inherent relaxed extractability of the NIZK proof obtained from [Lyubashevsky, 2012], as the lack of appropriate definitions for primitives that allow for relaxations in correctness or soundness had been overlooked so far.

Finally, we believe that our SNARK for lattice problems might be a good tool to build other types of privacy-preserving protocols, as it guarantees quite short transcripts when compared to lattice-based NIZK proofs.

Bibliography

Abdalla, M., An, J. H., Bellare, M. and Namprempre, C. [2002]. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security, *in* L. R. Knudsen (ed.), *EUROCRYPT 2002*, Vol. 2332 of *LNCS*, Springer, Heidelberg, pp. 418–433.

Abe, M. and Ohkubo, M. [2009]. A framework for universally composable non-committing blind signatures, *in* M. Matsui (ed.), *ASIACRYPT 2009*, Vol. 5912 of *LNCS*, Springer, Heidelberg, pp. 435–450.

Agrawal, S., Boneh, D. and Boyen, X. [2010]. Efficient lattice (H)IBE in the standard model, *in* H. Gilbert (ed.), *EUROCRYPT 2010*, Vol. 6110 of *LNCS*, Springer, Heidelberg, pp. 553–572.

Ajtai, M. [1996]. Generating hard instances of lattice problems (extended abstract), *28th ACM STOC*, ACM Press, pp. 99–108.

Alkim, E., Ducas, L., Pöppelmann, T. and Schwabe, P. [2016]. Post-quantum key exchange - A new hope, *in* T. Holz and S. Savage (eds), *USENIX Security 2016*, USENIX Association, pp. 327–343.

An, J. H., Dodis, Y. and Rabin, T. [2002]. On the security of joint signature and encryption, *in* L. R. Knudsen (ed.), *EUROCRYPT 2002*, Vol. 2332 of *LNCS*, Springer, Heidelberg, pp. 83–107.

Arora, S. and Ge, R. [2011]. New algorithms for learning in presence of errors, *in* L. Aceto, M. Henzinger and J. Sgall (eds), *ICALP 2011, Part I*, Vol. 6755 of *LNCS*, Springer, Heidelberg, pp. 403–415.

Ateniese, G., Camenisch, J., Joye, M. and Tsudik, G. [2000]. A practical and provably secure coalition-resistant group signature scheme, *in* M. Bellare (ed.), *CRYPTO 2000*, Vol. 1880 of *LNCS*, Springer, Heidelberg, pp. 255–270.

Bagherzandi, A., Cheon, J. H. and Jarecki, S. [2008]. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma, *in* P. Ning, P. F. Syverson and S. Jha (eds), *ACM CCS 2008*, ACM Press, pp. 449–458.

Banaszczyk, W. [1993]. New bounds in some transference theorems in the geometry of numbers, *Mathematische Annalen* **296**(1): 625–635.

Banaszczyk, W. [1995]. Inequalites for convex bodies and polar reciprocal lattices in R^n, *Discrete & Computational Geometry* **13**: 217–231.

Baum, C., Bootle, J., Cerulli, A., del Pino, R., Groth, J. and Lyubashevsky, V. [2018]. Sub-linear lattice-based zero-knowledge arguments for arithmetic circuits, *in* H. Shacham and A. Boldyreva (eds), *CRYPTO 2018, Part II*, Vol. 10992 of *LNCS*, Springer, Heidelberg, pp. 669–699.

Belenkiy, M., Chase, M., Kohlweiss, M. and Lysyanskaya, A. [2008]. P-signatures and noninteractive anonymous credentials, *in* R. Canetti (ed.), *TCC 2008*, Vol. 4948 of *LNCS*, Springer, Heidelberg, pp. 356–374.

Bellare, M., Micciancio, D. and Warinschi, B. [2003]. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions, *in* E. Biham (ed.), *EUROCRYPT 2003*, Vol. 2656 of *LNCS*, Springer, Heidelberg, pp. 614–629.

Bellare, M. and Rogaway, P. [1993]. Random oracles are practical: A paradigm for designing efficient protocols, *in* D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu and V. Ashby (eds), *ACM CCS 93*, ACM Press, pp. 62–73.

Bellare, M., Shi, H. and Zhang, C. [2005]. Foundations of group signatures: The case of dynamic groups, *in* A. Menezes (ed.), *CT-RSA 2005*, Vol. 3376 of *LNCS*, Springer, Heidelberg, pp. 136–153.

Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M. and Ward, N. P. [2018]. Aurora: Transparent succinct arguments for R1CS, Cryptology ePrint Archive, Report 2018/828. `https://eprint.iacr.org/2018/828`.

Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M. and Ward, N. P. [2019]. Aurora: Transparent succinct arguments for R1CS, *in* Y. Ishai and V. Rijmen (eds), *EUROCRYPT 2019, Part I*, Vol. 11476 of *LNCS*, Springer, Heidelberg, pp. 103–128.

Ben-Sasson, E., Chiesa, A. and Spooner, N. [2016a]. Interactive oracle proofs, *in* M. Hirt and A. D. Smith (eds), *TCC 2016-B, Part II*, Vol. 9986 of *LNCS*, Springer, Heidelberg, pp. 31–60.

Ben-Sasson, E., Chiesa, A. and Spooner, N. [2016b]. Interactive oracle proofs, *IACR Cryptology ePrint Archive* **2016**: 116.
URL: *http://eprint.iacr.org/2016/116*

Benhamouda, F., Camenisch, J., Krenn, S., Lyubashevsky, V. and Neven, G. [2014]. Better zero-knowledge proofs for lattice encryption and their application to group signatures, *in* P. Sarkar and T. Iwata (eds), *ASIACRYPT 2014, Part I*, Vol. 8873 of *LNCS*, Springer, Heidelberg, pp. 551–572.

Benhamouda, F., Krenn, S., Lyubashevsky, V. and Pietrzak, K. [2015]. Efficient zero-knowledge proofs for commitments from learning with errors over rings, *in* G. Pernul, P. Y. A. Ryan and E. R. Weippl (eds), *ESORICS 2015, Part I*, Vol. 9326 of *LNCS*, Springer, Heidelberg, pp. 305–325.

Bichsel, P., Camenisch, J., Groß, T. and Shoup, V. [2009]. Anonymous credentials on a standard java card, *in* E. Al-Shaer, S. Jha and A. D. Keromytis (eds), *ACM CCS 2009*, ACM Press, pp. 600–610.

Blum, M., Feldman, P. and Micali, S. [1988]. Non-interactive zero-knowledge and its applications (extended abstract), *20th ACM STOC*, ACM Press, pp. 103–112.

Boneh, D. and Boyen, X. [2004]. Secure identity based encryption without random oracles, *in* M. Franklin (ed.), *CRYPTO 2004*, Vol. 3152 of *LNCS*, Springer, Heidelberg, pp. 443–459.

Boneh, D., Boyen, X. and Goh, E.-J. [2005]. Hierarchical identity based encryption with constant size ciphertext, *in* R. Cramer (ed.), *EUROCRYPT 2005*, Vol. 3494 of *LNCS*, Springer, Heidelberg, pp. 440–456.

Boneh, D., Boyen, X. and Shacham, H. [2004]. Short group signatures, *in* M. Franklin (ed.), *CRYPTO 2004*, Vol. 3152 of *LNCS*, Springer, Heidelberg, pp. 41–55.

Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C. and Zhandry, M. [2011]. Random oracles in a quantum world, *in* D. H. Lee and X. Wang (eds), *ASIACRYPT 2011*, Vol. 7073 of *LNCS*, Springer, Heidelberg, pp. 41–69.

Bootle, J., Cerulli, A., Chaidos, P., Ghadafi, E. and Groth, J. [2016]. Foundations of fully dynamic group signatures, *in* M. Manulis, A.-R. Sadeghi and S. Schneider (eds), *ACNS 16*, Vol. 9696 of *LNCS*, Springer, Heidelberg, pp. 117–136.

Boschini, C., Camenisch, J. and Neven, G. [2017]. Relaxed lattice-based signatures with short zero-knowledge proofs, Cryptology ePrint Archive, Report 2017/1123. `https://eprint.iacr.org/2017/1123`.

Boschini, C., Camenisch, J. and Neven, G. [2018a]. Floppy-sized group signatures from lattices, *in* B. Preneel and F. Vercauteren (eds), *ACNS 18*, Vol. 10892 of *LNCS*, Springer, Heidelberg, pp. 163–182.

Boschini, C., Camenisch, J. and Neven, G. [2018b]. Relaxed lattice-based signatures with short zero-knowledge proofs, *in* L. Chen, M. Manulis and S. Schneider (eds), *ISC 2018*, Vol. 11060 of *LNCS*, Springer, Heidelberg, pp. 3–22.

Boschini, C., Camenisch, J., Ovsiankin, M. and Spooner, N. [2020]. Efficient post-quantum SNARKs for RSIS and RLWE and their applications to privacy.

Boyen, X. [2010]. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more, *in* P. Q. Nguyen and D. Pointcheval (eds), *PKC 2010*, Vol. 6056 of *LNCS*, Springer, Heidelberg, pp. 499–517.

Brakerski, Z., Langlois, A., Peikert, C., Regev, O. and Stehlé, D. [2013]. Classical hardness of learning with errors, *in* D. Boneh, T. Roughgarden and J. Feigenbaum (eds), *45th ACM STOC*, ACM Press, pp. 575–584.

Brickell, E. F., Camenisch, J. and Chen, L. [2004]. Direct anonymous attestation, *in* V. Atluri, B. Pfitzmann and P. McDaniel (eds), *ACM CCS 2004*, ACM Press, pp. 132–145.

Buchmann, J. and Lindner, R. [2009]. Secure parameters for SWIFFT, *in* B. K. Roy and N. Sendrier (eds), *INDOCRYPT 2009*, Vol. 5922 of *LNCS*, Springer, Heidelberg, pp. 1–17.

Camenisch, J., Dubovitskaya, M., Neven, G. and Zaverucha, G. M. [2011]. Oblivious transfer with hidden access control policies, *in* D. Catalano, N. Fazio, R. Gennaro and A. Nicolosi (eds), *PKC 2011*, Vol. 6571 of *LNCS*, Springer, Heidelberg, pp. 192–209.

Camenisch, J., Kiayias, A. and Yung, M. [2009]. On the portability of generalized Schnorr proofs, *in* A. Joux (ed.), *EUROCRYPT 2009*, Vol. 5479 of *LNCS*, Springer, Heidelberg, pp. 425–442.

Camenisch, J., Krenn, S., Lehmann, A., Mikkelsen, G. L., Neven, G. and Pedersen, M. Ø. [2016]. Formal treatment of privacy-enhancing credential systems, *in* O. Dunkelman and L. Keliher (eds), *SAC 2015*, Vol. 9566 of *LNCS*, Springer, Heidelberg, pp. 3–24.

Camenisch, J. and Lysyanskaya, A. [2001]. An efficient system for non-transferable anonymous credentials with optional anonymity revocation, *in* B. Pfitzmann (ed.), *EUROCRYPT 2001*, Vol. 2045 of *LNCS*, Springer, Heidelberg, pp. 93–118.

Camenisch, J. and Lysyanskaya, A. [2003]. A signature scheme with efficient protocols, *in* S. Cimato, C. Galdi and G. Persiano (eds), *SCN 02*, Vol. 2576 of *LNCS*, Springer, Heidelberg, pp. 268–289.

Camenisch, J., Neven, G. and Rückert, M. [2012]. Fully anonymous attribute tokens from lattices, *in* I. Visconti and R. D. Prisco (eds), *SCN 12*, Vol. 7485 of *LNCS*, Springer, Heidelberg, pp. 57–75.

Camenisch, J., Neven, G. and shelat, a. [2007]. Simulatable adaptive oblivious transfer, *in* M. Naor (ed.), *EUROCRYPT 2007*, Vol. 4515 of *LNCS*, Springer, Heidelberg, pp. 573–590.

Camenisch, J. and Shoup, V. [2003]. Practical verifiable encryption and decryption of discrete logarithms, *in* D. Boneh (ed.), *CRYPTO 2003*, Vol. 2729 of *LNCS*, Springer, Heidelberg, pp. 126–144.

Canetti, R., Goldreich, O. and Halevi, S. [1998]. The random oracle methodology, revisited (preliminary version), *30th ACM STOC*, ACM Press, pp. 209–218.

Cash, D., Hofheinz, D., Kiltz, E. and Peikert, C. [2010]. Bonsai trees, or how to delegate a lattice basis, *in* H. Gilbert (ed.), *EUROCRYPT 2010*, Vol. 6110 of *LNCS*, Springer, Heidelberg, pp. 523–552.

Chase, M. and Lysyanskaya, A. [2006]. On signatures of knowledge, *in* C. Dwork (ed.), *CRYPTO 2006*, Vol. 4117 of *LNCS*, Springer, Heidelberg, pp. 78–96.

Chaum, D. [1991]. Some weaknesses of "weaknesses of undeniable signatures" (rump session), *in* D. W. Davies (ed.), *EUROCRYPT'91*, Vol. 547 of *LNCS*, Springer, Heidelberg, pp. 554–556.

Chaum, D., Fiat, A. and Naor, M. [1990]. Untraceable electronic cash, *in* S. Goldwasser (ed.), *CRYPTO'88*, Vol. 403 of *LNCS*, Springer, Heidelberg, pp. 319–327.

Chaum, D. and van Heyst, E. [1991]. Group signatures, *in* D. W. Davies (ed.), *EUROCRYPT'91*, Vol. 547 of *LNCS*, Springer, Heidelberg, pp. 257–265.

Chen, Y. and Nguyen, P. Q. [2011]. BKZ 2.0: Better lattice security estimates, *in* D. H. Lee and X. Wang (eds), *ASIACRYPT 2011*, Vol. 7073 of *LNCS*, Springer, Heidelberg, pp. 1–20.

Chiesa, A., Manohar, P. and Spooner, N. [2019]. Succinct arguments in the quantum random oracle model, *in* D. Hofheinz and A. Rosen (eds), *TCC 2019, Part II*, Vol. 11892 of *LNCS*, Springer, Heidelberg, pp. 1–29.

Cohen, H. [2013]. *A course in computational algebraic number theory*, Vol. 138, Springer Science & Business Media.

Cooley, J. W. and Tukey, J. W. [1965]. An algorithm for the machine calculation of complex fourier series, *Mathematics of computation* **19**(90): 297–301.

Coppersmith, D. and Shamir, A. [1997]. Lattice attacks on NTRU, *in* W. Fumy (ed.), *EUROCRYPT'97*, Vol. 1233 of *LNCS*, Springer, Heidelberg, pp. 52–61.

Council of European Union [2016]. Regulation (eu) 2016/679 of the european parliament and of the council, https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679.

Cramer, R. [1996]. *Modular design of secure, yet practical cryptographic protocols*, PhD thesis, University of Amsterdam.

Cramer, R., Damgård, I. and Schoenmakers, B. [1994]. Proofs of partial knowledge and simplified design of witness hiding protocols, *in* Y. Desmedt (ed.), *CRYPTO'94*, Vol. 839 of *LNCS*, Springer, Heidelberg, pp. 174–187.

Damgård, I. [2002]. On σ-protocols, *Lecture Notes, University of Aarhus, Department for Computer Science* .

del Pino, R., Lyubashevsky, V. and Seiler, G. [2018]. Lattice-based group signatures and zero-knowledge proofs of automorphism stability, *in* D. Lie, M. Mannan, M. Backes and X. Wang (eds), *ACM CCS 2018*, ACM Press, pp. 574–591.

Diffie, W. and Hellman, M. E. [1976]. New directions in cryptography, *IEEE Transactions on Information Theory* **22**(6): 644–654.

Dolev, D., Dwork, C. and Naor, M. [1991]. Non-malleable cryptography (extended abstract), *23rd ACM STOC*, ACM Press, pp. 542–552.

Don, J., Fehr, S., Majenz, C. and Schaffner, C. [2019]. Security of the Fiat-Shamir transformation in the quantum random-oracle model, *in* A. Boldyreva and D. Micciancio (eds), *CRYPTO 2019, Part II*, Vol. 11693 of *LNCS*, Springer, Heidelberg, pp. 356–383.

Erdös, P. [1946]. On the coefficients of the cyclotomic polynomial, *Bull. Amer. Math. Soc.* **52**(2): 179–184.

Faust, S., Kohlweiss, M., Marson, G. A. and Venturi, D. [2012]. On the non-malleability of the Fiat-Shamir transform, *in* S. D. Galbraith and M. Nandi (eds), *INDOCRYPT 2012*, Vol. 7668 of *LNCS*, Springer, Heidelberg, pp. 60–79.

Fiat, A. and Shamir, A. [1987]. How to prove yourself: Practical solutions to identification and signature problems, *in* A. M. Odlyzko (ed.), *CRYPTO'86*, Vol. 263 of *LNCS*, Springer, Heidelberg, pp. 186–194.

Fischlin, M. [2006]. Round-optimal composable blind signatures in the common reference string model, *in* C. Dwork (ed.), *CRYPTO 2006*, Vol. 4117 of *LNCS*, Springer, Heidelberg, pp. 60–77.

Gama, N. and Nguyen, P. Q. [2008]. Predicting lattice reduction, *in* N. P. Smart (ed.), *EUROCRYPT 2008*, Vol. 4965 of *LNCS*, Springer, Heidelberg, pp. 31–51.

Gauss, C. F. [2006]. *Untersuchungen über höhere Arithmetik*, Vol. 191, American Mathematical Soc.

Genise, N. and Micciancio, D. [2018]. Faster Gaussian sampling for trapdoor lattices with arbitrary modulus, *in* J. B. Nielsen and V. Rijmen (eds), *EUROCRYPT 2018, Part I*, Vol. 10820 of *LNCS*, Springer, Heidelberg, pp. 174–203.

Gennaro, R., Minelli, M., Nitulescu, A. and Orrù, M. [2018]. Lattice-based zk-SNARKs from square span programs, *in* D. Lie, M. Mannan, M. Backes and X. Wang (eds), *ACM CCS 2018*, ACM Press, pp. 556–573.

Gentry, C., Peikert, C. and Vaikuntanathan, V. [2008]. Trapdoors for hard lattices and new cryptographic constructions, *in* R. E. Ladner and C. Dwork (eds), *40th ACM STOC*, ACM Press, pp. 197–206.

Goldreich, O. [2001]. *The Foundations of Cryptography - Volume 1: Basic Techniques*, Cambridge University Press.

Goldreich, O., Goldwasser, S. and Halevi, S. [1996]. Collision-free hashing from lattice problems, *Electronic Colloquium on Computational Complexity (ECCC)* **3**(42).

Goldwasser, S., Micali, S. and Rivest, R. L. [1988]. A digital signature scheme secure against adaptive chosen-message attacks, *SIAM Journal on Computing* **17**(2): 281–308.

Gordon, S. D., Katz, J. and Vaikuntanathan, V. [2010]. A group signature scheme from lattice assumptions, *in* M. Abe (ed.), *ASIACRYPT 2010*, Vol. 6477 of *LNCS*, Springer, Heidelberg, pp. 395–412.

Hanrot, G., Pujol, X. and Stehlé, D. [2011]. Terminating BKZ, Cryptology ePrint Archive, Report 2011/198. `http://eprint.iacr.org/2011/198`.

Hirt, M. and Sako, K. [2000]. Efficient receipt-free voting based on homomorphic encryption, *in* B. Preneel (ed.), *EUROCRYPT 2000*, Vol. 1807 of *LNCS*, Springer, Heidelberg, pp. 539–556.

Katsumata, S. and Yamada, S. [2019]. Group signatures without NIZK: From lattices in the standard model, *in* Y. Ishai and V. Rijmen (eds), *EUROCRYPT 2019, Part III*, Vol. 11478 of *LNCS*, Springer, Heidelberg, pp. 312–344.

Katz, J. and Lindell, Y. [2014]. *Introduction to Modern Cryptography, Second Edition*, CRC Press.

Kiayias, A. and Yung, M. [2005]. Group signatures with efficient concurrent join, *in* R. Cramer (ed.), *EUROCRYPT 2005*, Vol. 3494 of *LNCS*, Springer, Heidelberg, pp. 198–214.

Kirchner, P. and Fouque, P.-A. [2015]. An improved BKW algorithm for LWE with applications to cryptography and lattices, *in* R. Gennaro and M. J. B. Robshaw (eds), *CRYPTO 2015, Part I*, Vol. 9215 of *LNCS*, Springer, Heidelberg, pp. 43–62.

Laguillaumie, F., Langlois, A., Libert, B. and Stehlé, D. [2013]. Lattice-based group signatures with logarithmic signature size, *in* K. Sako and P. Sarkar (eds), *ASIACRYPT 2013, Part II*, Vol. 8270 of *LNCS*, Springer, Heidelberg, pp. 41–61.

Lamport, L. [1979]. Constructing digital signatures from a one-way function, *Technical report*, Technical Report CSL-98, SRI International Palo Alto.

Lang, S. [2002]. *Algebra*, Vol. 211 of *Graduate Texts in Mathematics*, Springer-Verlag, New York.

Langlois, A. and Stehlé, D. [2015]. Worst-case to average-case reductions for module lattices, *Des. Codes Cryptography* **75**(3): 565–599.

Libert, B., Ling, S., Nguyen, K. and Wang, H. [2016]. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors, *in* M. Fischlin and J.-S. Coron (eds), *EUROCRYPT 2016, Part II*, Vol. 9666 of *LNCS*, Springer, Heidelberg, pp. 1–31.

Libert, B., Ling, S., Nguyen, K. and Wang, H. [2018]. Lattice-based zero-knowledge arguments for integer relations, *in* H. Shacham and A. Boldyreva (eds), *CRYPTO 2018, Part II*, Vol. 10992 of *LNCS*, Springer, Heidelberg, pp. 700–732.

Libert, B., Mouhartem, F. and Nguyen, K. [2016]. A lattice-based group signature scheme with message-dependent opening, *in* M. Manulis, A.-R. Sadeghi and S. Schneider (eds), *ACNS 16*, Vol. 9696 of *LNCS*, Springer, Heidelberg, pp. 137–155.

Ling, S., Nguyen, K. and Wang, H. [2015]. Group signatures from lattices: Simpler, tighter, shorter, ring-based, *in* J. Katz (ed.), *PKC 2015*, Vol. 9020 of *LNCS*, Springer, Heidelberg, pp. 427–449.

Ling, S., Nguyen, K., Wang, H. and Xu, Y. [2017a]. Lattice-based group signatures: Achieving full dynamicity with ease, Cryptology ePrint Archive, Report 2017/353. `http://eprint.iacr.org/2017/353`.

Ling, S., Nguyen, K., Wang, H. and Xu, Y. [2017b]. Lattice-based group signatures: Achieving full dynamicity with ease, *in* D. Gollmann, A. Miyaji and H. Kikuchi (eds), *ACNS 17*, Vol. 10355 of *LNCS*, Springer, Heidelberg, pp. 293–312.

Ling, S., Nguyen, K., Wang, H. and Xu, Y. [2018]. Constant-size group signatures from lattices, *in* M. Abdalla and R. Dahab (eds), *PKC 2018, Part II*, Vol. 10770 of *LNCS*, Springer, Heidelberg, pp. 58–88.

Lyubashevsky, V. [2009]. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures, *in* M. Matsui (ed.), *ASIACRYPT 2009*, Vol. 5912 of *LNCS*, Springer, Heidelberg, pp. 598–616.

Lyubashevsky, V. [2012]. Lattice signatures without trapdoors, *in* D. Pointcheval and T. Johansson (eds), *EUROCRYPT 2012*, Vol. 7237 of *LNCS*, Springer, Heidelberg, pp. 738–755.

Lyubashevsky, V. [2016]. Digital signatures based on the hardness of ideal lattice problems in all rings, *in* J. H. Cheon and T. Takagi (eds), *ASIACRYPT 2016, Part II*, Vol. 10032 of *LNCS*, Springer, Heidelberg, pp. 196–214.

Lyubashevsky, V. and Micciancio, D. [2006]. Generalized compact Knapsacks are collision resistant, *in* M. Bugliesi, B. Preneel, V. Sassone and I. Wegener (eds), *ICALP 2006, Part II*, Vol. 4052 of *LNCS*, Springer, Heidelberg, pp. 144–155.

Lyubashevsky, V., Micciancio, D., Peikert, C. and Rosen, A. [2008]. SWIFFT: A modest proposal for FFT hashing, *in* K. Nyberg (ed.), *FSE 2008*, Vol. 5086 of *LNCS*, Springer, Heidelberg, pp. 54–72.

Lyubashevsky, V. and Neven, G. [2017]. One-shot verifiable encryption from lattices, *in* J. Coron and J. B. Nielsen (eds), *EUROCRYPT 2017, Part I*, Vol. 10210 of *LNCS*, Springer, Heidelberg, pp. 293–323.

Lyubashevsky, V., Peikert, C. and Regev, O. [2010]. On ideal lattices and learning with errors over rings, *in* H. Gilbert (ed.), *EUROCRYPT 2010*, Vol. 6110 of *LNCS*, Springer, Heidelberg, pp. 1–23.

Lyubashevsky, V., Peikert, C. and Regev, O. [2013]. A toolkit for ring-LWE cryptography, *in* T. Johansson and P. Q. Nguyen (eds), *EUROCRYPT 2013*, Vol. 7881 of *LNCS*, Springer, Heidelberg, pp. 35–54.

Micciancio, D. [2002]. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions, *43rd FOCS*, IEEE Computer Society Press, pp. 356–365.

Micciancio, D. and Goldwasser, S. [2002]. *Complexity of lattice problems - a cryptograhic perspective*, Vol. 671 of *The Kluwer international series in engineering and computer science*, Springer.

Micciancio, D. and Peikert, C. [2012]. Trapdoors for lattices: Simpler, tighter, faster, smaller, *in* D. Pointcheval and T. Johansson (eds), *EUROCRYPT 2012*, Vol. 7237 of *LNCS*, Springer, Heidelberg, pp. 700–718.

Micciancio, D. and Regev, O. [2009]. *Lattice-based Cryptography*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 147–191.

Naor, M. and Yung, M. [1990]. Public-key cryptosystems provably secure against chosen ciphertext attacks, *22nd ACM STOC*, ACM Press, pp. 427–437.

National Institute of Standards and Technology [2016]. Announcing request for nominations for public-key post-quantum cryptographic algorithms, https://federalregister.gov/a/2016-30615.

Nguyen, P. Q., Zhang, J. and Zhang, Z. [2015]. Simpler efficient group signatures from lattices, *in* J. Katz (ed.), *PKC 2015*, Vol. 9020 of *LNCS*, Springer, Heidelberg, pp. 401–426.

Paquin, C. and Zaverucha, G. [n.d.]. U-prove cryptographic specification v1. 1 (revision 3, december 2013), https://www.microsoft.com/en-us/research/publication/u-prove-cryptographic-specification-v1-1-revision-3.

Peikert, C. [2008]. Limits on the hardness of lattice problems in l_p norms, *Computational Complexity* **17**(2): 300–351.

Peikert, C. and Rosen, A. [2006]. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices, *in* S. Halevi and T. Rabin (eds), *TCC 2006*, Vol. 3876 of *LNCS*, Springer, Heidelberg, pp. 145–166.

Pointcheval, D. and Stern, J. [1996]. Security proofs for signature schemes, *in* U. M. Maurer (ed.), *EUROCRYPT'96*, Vol. 1070 of *LNCS*, Springer, Heidelberg, pp. 387–398.

Regev, O. [2005]. On lattices, learning with errors, random linear codes, and cryptography, *in* H. N. Gabow and R. Fagin (eds), *37th ACM STOC*, ACM Press, pp. 84–93.

Rivest, R. L., Shamir, A. and Tauman, Y. [2001]. How to leak a secret, *in* C. Boyd (ed.), *ASIACRYPT 2001*, Vol. 2248 of *LNCS*, Springer, Heidelberg, pp. 552–565.

Sahai, A. [1999]. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security, *40th FOCS*, IEEE Computer Society Press, pp. 543–553.

Schnorr, C.-P. [1990]. Efficient identification and signatures for smart cards, *in* G. Brassard (ed.), *CRYPTO'89*, Vol. 435 of *LNCS*, Springer, Heidelberg, pp. 239–252.

Shamir, A. and Tauman, Y. [2001]. Improved online/offline signature schemes, *in* J. Kilian (ed.), *CRYPTO 2001*, Vol. 2139 of *LNCS*, Springer, Heidelberg, pp. 355–367.

Shor, P. W. [1994]. Algorithms for quantum computation: Discrete logarithms and factoring, *35th FOCS*, IEEE Computer Society Press, pp. 124–134.

Snowden, E. [2019]. *Permanent Record*, Metropolitan Books.

Stern, J. [1994]. A new identification scheme based on syndrome decoding, *in* D. R. Stinson (ed.), *CRYPTO'93*, Vol. 773 of *LNCS*, Springer, Heidelberg, pp. 13–21.

Unruh, D. [2017]. Post-quantum security of Fiat-Shamir, *in* T. Takagi and T. Peyrin (eds), *ASIACRYPT 2017, Part I*, Vol. 10624 of *LNCS*, Springer, Heidelberg, pp. 65–95.

Xagawa, K. and Tanaka, K. [2009]. Zero-knowledge protocols for NTRU: Application to identification and proof of plaintext knowledge, *in* J. Pieprzyk and F. Zhang (eds), *ProvSec 2009*, Vol. 5848 of *LNCS*, Springer, Heidelberg, pp. 198–213.

Xue, R., Li, N. and Li, J. [2008]. Algebraic construction for zero-knowledge sets, *J. Comput. Sci. Technol.* **23**(2): 166–175.

ZCash Company [2014]. `https://z.cash/`.