UNIVERSITY OF MACEDONIA

GRADUATE PROGRAM

DEPARTMENT OF APPLIED INFORMATICS

LATTICE-BASED CRYPTOGRAPHY: PROTOCOLS AND APPLICATIONS

A dissertation submitted in partial satisfaction of the requirements for the degree
Doctor of Philosophy

in

Applied Informatics

by

Dimitrios Papachristoudis

Committee in charge:
Retired Professor George Stephanides, Chair
Professor Dimitrios Hristu-Varsakelis
Professor Maria Satratzemi

Thessaloniki, May 2022

LATTICE-BASED CRYPTOGRAPHY: PROTOCOLS AND APPLICATIONS

Dimitrios Papachristoudis

B.Sc. in Mathematics, University of Ioannina, 2011
M.Sc. in Applied Informatics, University of Macedonia, 2014

Doctoral Dissertation

A dissertation submitted in partial fulfillment for the requirements for the degree of

DOCTOR OF PHILOSOPHY IN APPLIED INFORMATICS

Doctoral Advisor
George Stephanides, Retired Professor

The members of the Committee appointed to examine the dissertation of Dimitrios Papachristoudis find it satisfactory and recommend that it be accepted 3/5/2022.

George Stephanides, Retired Professor, Chair	Dimitrios Hristu-Varsakelis, Professor	Maria Satratzemi, Professor
.....	.....	.....
Foteini Baldimtsi, Assistant Professor	Ioannis Stamatiou, Professor	Ioannis Mavridis, Professor
.....	.....	.....
	Sofia Petridou, Assistant Professor	

.....

Dimitrios Papachristoudis

.....

University of Macedonia, Thessaloniki

To my mother Natalie and my father George.

Contents

List of Figures

List of Tables

List of Algorithms

Prologue – Acknowledgements

I am deeply grateful to my supervisor and mentor Professor George Stephanides for his guidance, support, and patience during my years as his Ph.D. student. Under his tutelage I had the privilege of learning a lot about Cryptography and Mathematics in general, and to develop my teaching style while serving as his T.A. for two consecutive years. I highly appreciate the fact that he entrusted me with researching an ambitious, demanding, and cutting-edge topic such as lattice-based Cryptography, because of the great promise it holds for the immediate decade(s).

I am also very grateful to my dissertation's co-supervisor, Professor Dimitrios Hristu-Varsakelis for his cooperation, steady support and overall positive attitude during my years as a Ph.D. student, as well as Professor Maria Satratzemi for participating in the committee in charge.

I am indebted to Professor Foteini Baldimtsi for her guidance and support during my Ph.D.; it is through her motivation and deep knowledge of the landscape of Applied Cryptography that I was able to kick-start and advance my research. I am also very grateful to her for giving me the opportunity to conduct a research visit to the Department of Computer Science of George Mason University (GMU), and to give a seminar talk about the construction presented in Chapter 4 at the department's Crypto Group. While visiting, I also had the distinct honour of personally meeting Professors Jonathan Katz and Dov Gordon, to both of whom I am very thankful for their very encouraging feedback on my work.

I am also extremely thankful to Dr. Julian Loss for his guidance, patience, and encouragement. Working with him on the construction of Chapter 5 for the past 1.5 year has been a profoundly didactical and eye-opening experience for me.

I would also like to thank Professor Ioannis Stamatiou for being a good role model for me during my years as an undergraduate student at the Department of Mathematics of the University of Ioannina. It is thanks to his influence and encouragement that I decided on pursuing a Ph.D. from early on during my studies.

I have to certainly acknowledge my friends in the department and outside of it with

whom I have spent many enjoyable non-working hours. It gives me great pleasure to thank you all in alphabetical order. Thank you, Alexandros, Antonis, Arkadios, Charis, Dimitris, Dimosthenis, Dionysis, Efi, George, Giannis, Ioanna, Konstantinos, Kostas, Lysander, Panagiotis, Sakis, Tasos, Thanasis, and last but not least Thomas and Vangelis. I would also like to give special thanks to Georgios Sidiropoulos for his steadfast support in my effort and to my teacher Athanasios Valavanis for helping me to appreciate the beauty of Mathematics.

Finally, the biggest thank you goes to my family for always being there for me, throughout all my years as a student. It is thanks to their *unwavering* support and *numerous* sacrifices that this effort comes to a close.

Chapter 2 sets the required theoretical and notational groundwork that is required for presenting this work.

Chapter 3 is, in part, a reprint of the paper "A Survey on Lattice-based Blind Signatures and their Feasibility" which is set to appear in the peer-reviewed journal Archives of Economic History. The dissertation author was the primary investigator and author of this paper.

Chapter 4 is, in part, a reprint of the paper "Leakage-resilient lattice-based partially blind signatures" co-authored with Dimitrios Hristu-Varsakelis, Foteini Baldimtsi and George Stephanides, which appears in the peer-reviewed journal IET Information Security. The dissertation author was the primary investigator and author of this paper.

Chapter 5 is an improved and expanded version of a manuscript titled "A Framework for Blind Signatures with Revocable Sessions" which was co-authored with Julian Loss, Foteini Baldimtsi and George Stephanides and was originally submitted to Asiacrypt 2021. The dissertation author was the primary investigator and author of this paper.

Chapter 6 is, in part, a reprint of the paper "Proxy Signatures from Ideal Lattices–Secure in All Rings" [145] co-authored with George Stephanides, which appears in the Proceedings of the 35th Panhellenic Conference on Mathematical Education "Mathematics: Research and Education in the 21st century". The dissertation author was the primary investigator and author of this paper.

Περίληψη

Η κρυπτογραφία με βάση τα δικτυώματα ξεκίνησε με τη ρηξικέλευθη εργασία του M. Ajtai [14] κατά τη δεκαετία του '90 και έκτοτε έχει εδραιωθεί ως μία από τις πλέον πολύπλευρες προσεγγίσεις για την κατασκευή αποδεδειγμένα ασφαλών, αποδοτικών και παραλληλοποιήσιμων κρυπτογραφικών εργαλείων τα οποία μπορούν να ανθίστανται επιθέσεις ακόμη και από κβαντικούς υπολογιστές. Εκτός τούτου, η κρυπτογραφία με βάση τα δικτυώματα παρέχει το μοναδικό χαρακτηριστικό ότι επιτρέπει αναγωγές χείριστης σε μέση περίπτωση, το οποίο *απαιτείται* για κρυπτογραφικές εφαρμογές καθώς η απλή ύπαρξη ενός στιγμιοτύπου ενός δύσκολου υπολογιστικά προβλήματος στη χείριστη περίπτωση εγγυάται την ασφάλεια στη μέση περίπτωση. Το γεγονός αυτό όχι μόνο επιτρέπει την αξιοποίηση της δυσκολίας προβλημάτων χείριστης περίπτωσης σε δικτυώματα, αλλά τυπικά απλοποιεί και την επιλογή κλειδιών.

Μολονότι κάποια από τα πλέον διαδεδομένα κρυπτογραφικά εργαλεία όπως οι ψηφιακές υπογραφές έχουν ερευνηθεί εκτενώς τις τελευταίες δύο δεκαετίες, σημαντικά μικρότερη πρόοδος έχει γίνει με τα πιο προηγμένα εργαλεία τα οποία διαθέτουν επιπλέον χαρακτηριστικά, παρά τη σημασία τους για πληθώρα πραγματικών εφαρμογών που εστιάζουν στη διατήρηση της ιδιωτικότητας όπως: το ψηφιακό χρήμα, οι ψηφιακές εκλογές, οι ψηφιακές δημοπρασίες, οι τυφλά υπογεγραμμένες συμβάσεις (χρησιμοποιούνται στα κρυπτονομίσματα), τα δίκτυα ασύρματων αισθητήρων και τα ανώνυμα διαπιστευτήρια (χρησιμοποιούνται στην τεχνολογία U-Prove της Microsoft).

Εμφορούμενοι από αυτή τη σχετική έλλειψη αποτελεσμάτων, καθώς και από το πολυποίκιλο και υποδειγματικό φάσμα των σεναρίων εφαρμογής, η παρούσα διατριβή στοχεύει όχι απλώς στο να επιτρέψει τέτοιου είδους φιλικές προς την ιδιωτικότητα εφαρμογές βασισμένες σε προβλήματα δικτυωμάτων, αλλά και στο να τα καταστήσουμε *πρακτικά* και *αποδοτικά*. Πιστεύουμε πως οι τεχνικές μας μπορούν να μεταφερθούν και σε άλλα σενάρια εφαρμογών.

Λέξεις Κλειδιά: Κρυπτογραφία βασισμένη στα δικτυώματα, Μετακβαντική κρυπτογραφία, Εφαρμογές προσανατολισμένες στην ιδιωτικότητα, Τυφλές υπογραφές, Ευαπόδεικτη ασφάλεια, Υπογραφές πληρεξουσίου.

Abstract

Lattice-based cryptography began with the groundbreaking work of M. Ajtai [14] back in the 90's and has since then proven to be one of the most versatile approaches for constructing provably secure, efficient, and highly parallelizable cryptographic primitives that can withstand attacks even by quantum computers. Moreover, lattice-based cryptography offers the unique feature of allowing for worst-case to average-case reductions, which is *needed* for cryptographic applications because the mere existence of a computationally hard problem instance in the worst case guarantees security in the average case. This not only allows us to harness the hardness of worst-case lattice problems, but it also typically simplifies key selection.

While some of the more ubiquitous cryptographic primitives like digital signatures have been extensively explored during the past two decades, far less progress has been made with more advanced primitives which possess additional features, despite their significance in a plethora of real-world privacy-preserving applications like e-cash, e-voting, e-auctions, blindly signed contracts (used in cryptocurrencies), wireless sensor networks, and anonymous credentials (used in Microsoft's U-Prove technology).

Motivated by this relative dearth of results, as well as the diverse and exemplary spectrum of application scenarios, this dissertation aims on not only enabling such privacy-friendly applications from lattice assumptions, but also making them both *practical*, and *efficient*. We believe that our techniques can be transferred to other application scenarios as well.

Keywords: Lattice-based cryptography, Post-Quantum cryptography, Privacy-oriented applications, Blind signatures, Provable security, Proxy signatures.

Notation

SETS

$\emptyset$	the empty set
$\mathbb{N}$	the set of natural numbers $\{1, 2, \ldots\}$
$\mathbb{N}_0$	the set of natural numbers, including zero i.e., $\{0, 1, 2, \ldots\}$
$\mathbb{Z}$	the set of integer numbers $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$
$\mathbb{Z}_q$	the ring of integers modulo q (also a field if q is prime)
$\mathbb{Q}$	the set of rational numbers $\{p/q : p \in \mathbb{Z}, q \in \mathbb{N} \text{ and } \gcd(p, q) = 1\}$
$\mathbb{R}$	the set of real numbers
$\mathbb{R}^+$	the set of positive real numbers i.e., $(0, +\infty)$
$\mathbb{R}_0^+$	the set of non-negative real numbers i.e., $[0, +\infty)$
$[n]$	the set $\{1, \ldots, n\}$, where $n \in \mathbb{N}$
$\{v_i\}_{i=1}^k$	abbreviation for $\{v_1, \ldots, v_k\}$

OPERATORS

$\cup$	set union
$\cap$	set intersection
$\setminus$	set difference
$\times$	Cartesian product
$\vee$	logical disjunction (or)
$\wedge$	logical conjunction (and)
$\oplus$	bitwise XOR
$\otimes$	tensor product of matrices
$\|\|$	concatenation of strings or matrix columns
$[\![statement]\!]$	boolean operator (evaluates to 1 if *statement* is true, and to 0 otherwise)

RELATIONS

$\equiv$	is equivalent/congruent to
$\cong$	is isomorphic to
$P_1 \preceq P_2$	Problem P_1 (classically) reduces to problem P_2
$P_1 \preceq_Q P_2$	Problem P_1 quantumly reduces to problem P_2

$x \sim D$	x is distributed according to distribution D
FUNCTIONS	
$\lfloor . \rfloor$	floor function
$\lceil . \rceil$	ceiling function
$poly(n)$	some fixed but unspecified polynomial in variable n
$negl(n)$	a function that is negligible in n
PROBABILITY DISTRIBUTIONS	
$\mathsf{Unif}(S)$	uniform distribution over some (countable) set S
$\Delta(X, Y)$	statistical distance of random variables X and Y
COMPLEXITY THEORY	
BPP	class of decision problems solvable by a probabilistic Turing machine in polynomial time with an error probability bounded away from $\frac{1}{3}$ for all instances
P	class of decision problems solvable by a deterministic Turing machine in polynomial time
NP	class of decision problems that solvable in polynomial time by a nondeterministic Turing machine
coNP	class of decision problems whose complement is in NP
NPC	class of all NP-complete problems
AM	class of decision problems that can be decided in polynomial time by an Arthur-Merlin protocol with two messages
coAM	class of decision problems whose complement is in AM

Abbreviations

AM	Arthur-Merlin
BPP	Bounded-error Probabilistic Polynomial time
BSS	Blind Signature Scheme(s)
coAM	co-Arthur-Merlin
coNP	co-NP
CVP	Closest Vector Problem
DLP	Discrete Logarithm Problem
DSS	Digital Signature Scheme(s)
FPBSS	Fair Partially-Blind Signature Scheme(s)
FSS	Fair Blind Signature Scheme(s)
HE	Homomorphic Encryption
iff	if and only if
LHF	Linear Hash Function
LWE	Learning with Errors
NIST	National Institute of Standards and Technology
NIZK	Non-Interactive Zero-Knowledge
NP	Non-deterministic Polynomial time
NPC	Non-deterministic Polynomial time Complete
OMUF	One-More Unforgeability
OWF	One-Way Function(s)
P	Polynomial time
PP	Probabilistic Polynomial time
PBSS	Partially-Blind Signature Scheme(s)
PoR	Proof of Revocation
PPT	Probabilistic Polynomial Time
PRF	Pseudo-Random Function
QROM	Quantum (Accessible) Random Oracle Model
resp.	respectively
RO	Random Oracle
ROM	Random Oracle Model
RSA	Rivest, Shamir, Addleman
RSR	Random Self-Reducibility
SIS	Short Integer Solution
SIVP	Shortest Independent Vectors Problem
s.t.	such that
SVP	Shortest Vector Problem
UF-CMA	Universal Forgery under Chosen-Message Attack
w.l.o.g.	without loss of generality
w.r.t.	with regards/respect to
ZKPoK	Zero-Knowledge Proof of Knowledge

Chapter 1

Introduction

"Privacy is one of the biggest
problems in this new electronic age."

Andrew Grove, former Intel CEO

Cryptography has always been an integral part of any society dealing with sensitive information. Historically, one of the earliest uses of cryptography can be traced all the way to ancient Egypt 1900 BC, where complex hieroglyphics were carved on the walls of tombs as a means of pictorially obfuscating information [105]. The primary use of cryptography throughout the ages has mainly been in warfare, diplomacy, and espionage, where it enabled two parties to communicate in confidentiality. Individuals and even commercial organizations rarely considered it necessary to resort to encryption for protecting their communications.

Until the modern era, cryptography was essentially synonymous with encryption (i.e., primarily concerned with developing techniques that would enable confidential communication between two parties over great distances). However, the techniques employed to this end were *severely* hindered by the fact that both parties needed to share a pre-agreed secret key (*secret key* or *symmetric* cryptography) which would help them transform their messages accordingly. The celebrated work of Diffie and Hellman [63] in the 1970's completely revolutionized the landscape of cryptography by introducing for the first time the notion of *public-key* (or *asymmetric*) cryptography. In this context, *only* the receiver of the message needs to be in possession of a secret key,

whereas the sender just needs to know the receiver's public key. The first implementation of a public key cryptosystem soon followed with the work of Rivest, Shamir, and Adleman [157]. These groundbreaking works combined with advances in computing and telecommunications enabled cryptography to broaden its scope far beyond that of just encryption. In recent decades, a plurality of cryptographic primitives, schemes, and applications have been proposed. These range from simple primitives such as one-way hash functions and zero-knowledge proofs, to secure multiparty computation and a broad array of digital signature schemes.

A special branch of contemporary cryptography is concerned with designing primitives and protocols that protect the privacy and anonymity of communicating parties within complex transactions. Where digital signatures are typically concerned with simply producing a signature on an arbitrary message, primitives such as blind signatures aim to accomplish this task: a) without forcing the recipient to reveal its message (privacy), and b) by preventing the authority responsible for issuing signatures (signer) from linking the blinded message it signs to a later "unblinded" version that it may be called upon to verify. For instance, in a (remote) electronic voting scenario, we may want each ballot to be certified by an election authority before it can be accepted for counting. The authority can check the credentials of the voter to ensure that they are eligible to vote (and that they are not submitting multiple ballots). At the same time, a voter's selections in the ballot remain outside the authority's view and the authority is unable to link any given vote to the identity of the voter that cast it (e.g.: during counting).

Naturally, with such a high level of privacy comes great risk for abuse. For example, blind signatures have been argued to provide a gateway for committing "perfect" crimes [181] such as money laundering, blackmailing, etc. At the same time, blind signatures provide no guarantees to the signer that the blinded message he signed, has an appropriate "format" or even contains some valid information that should be included in the message (e.g.: the denomination of a digital coin, the date a voucher was issued, etc.). These challenges further motivate the study of schemes with additional features and more complex frameworks to support them.

1.1 Hard Computational Problems and Post-Quantum Cryptography

Early cryptographic methods crucially relied on *security through obscurity* (i.e., by keeping the encryption and decryption algorithms secret). However, as history has shown time and time again, encryption/decryption machines can be captured and analyzed, people can defect or be "convinced" to divulge information, etc. This led to the formulation of the most fundamental principle in contemporary cryptography called *Kerckhoff's principle*: "In assessing the security of a cryptosystem, one should always assume the enemy knows the method being used". As such, most cryptographic schemes nowadays rely on the secrecy of the key rather than the algorithm. Furthermore, schemes are brought under scrutiny to ensure that they do not suffer from any design flaws which could compromise their security. One prominent way to design cryptographic schemes is by relating their security to the complexity hardness assumptions of computational problems. If these problems are computationally hard to solve, then breaking the scheme's security is (at least) as hard as these problems.

But what exactly does it mean for a problem to be "hard"? Roughly speaking, we say that a problem is *efficiently solvable* if it can be solved by an algorithm running in polynomial time in the size of the input. Under this definition, problems are classified as problems that are: a) possible to solve efficiently, b) impossible to solve efficiently, or c) impossible to solve at all. Cryptography is generally concerned with the latter two classes. While problems that are impossible to solve are of some interest to cryptography, they typically lead to *extremely inefficient* constructions. On the other hand, problems that are computationally hard to solve offer a nice balance between security and efficiency.

One of the most well-studied and widely used problems in cryptography is the *factoring problem*. Given a large positive integer n, we are asked to factor it into a product of primes. A special case of this problem where $n = p \cdot q$ with p and q being large primes of roughly the same bit-size is at the heart of the RSA cryptosystem

[157]. A second, also widely used problem is the so-called *discrete logarithm problem (DLP)*. Here, we are given a finite cyclic group G, a generator $g \in G$, and an element $h = g^x \in G$, and are tasked with finding $x \in \mathbb{Z}$. DLP is most commonly used with elliptic curve groups and some of its applications include end-to-end encryption in VoIP telephony (e.g.: Viber), signing transactions in cryptocurrencies (e.g.: Bitcoin, Ethereum), and many more.

A question that naturally arises at this point is: Does just an assumption offer any tangible security guarantees? This concern is indeed valid; just because no efficient algorithm currently exists for solving a problem does not necessarily mean that one will never be found. More theoretically sound arguments can be made based on complexity theory which classifies problems based on their theoretical hardness. Hence, a problem that is, say, NP-complete would give us strong indications about its (theoretical) hardness. However, there is often a noticeable gap between theory and practice. Indeed, just because a problem is considered NP-hard does not mean that a specific instance of it is also hard (it is often hard to even reliably generate hard instances of problems). The factoring problem in particular is *not known* to be NP-complete and in fact, the related problem of primality testing is actually solvable in polynomial time [9].

To make matters worse, in 1994, P. Shor showed a quantum algorithm capable of factoring large numbers into primes exponentially faster than a classical computer can [169]. The quantum step in Shor's algorithm can efficiently break DLP, too. Hence, the construction of a reasonable-scale quantum computer would signal the compromise of any cryptographic scheme that relied on these well-established number theoretic hardness assumptions (cybersecurity experts call this Q-Day). Given the ubiquity of cryptographic schemes in our everyday online activities, this could have catastrophic implications for individuals, companies and countries around the world. Hackers would have free reign to view everything including emails, medical histories, old banking records, etc. Frauds could impersonate a user's identity, potentially gaining access to his/her bank account. Data thieves could already be accumulating encrypted data, with the view of unlocking them once quantum computers become

available. A corrupt government would be able to gain insight in voters' political views simply by linking their electronic ballots to their identities. Foreign governments could influence the outcome of elections abroad by tampering electronic ballots. This alarming discovery by Shor goaded researchers into investigating alternative hardness assumptions that would remain secure even against a quantum computer.

The original title of my PhD dissertation is "Post-Quantum Cryptography: Protocols and Applications". That is, cryptography that will guarantee security (and privacy) even after Q-Day. This differs from *quantum cryptography* in that the goal is to defeat quantum computers *without* quantum computers. Indeed, it is fairly reasonable to assume that even after the first large-scale quantum computers become a reality, their availability to the general public may be limited (e.g.: very high costs to build and maintain). However, cryptographic algorithms and protocols need to already be in place to protect even those that cannot afford a quantum computer. This reason motivated me to steer my thesis towards the fairly young and promising line of cryptographic research where security is based on hard computational problems in point lattices, which are conjectured to be immune to quantum attacks.

1.2 Lattice-Based Cryptography

When designing secure cryptographic schemes, one has to be mindful of developments both in technology and also in the field of cryptanalysis. Indeed, following the formulation of Shor's algorithm [169], the need for alternative hardness assumptions that remain intractable even in the presence of quantum computers became as imperative as ever. By now, lattice-based cryptography is one of the predominant approaches for constructing provably secure and efficient cryptographic primitives that can withstand attacks even by a quantum computer. This is largely due to the fact that unlike number-theoretic hardness assumptions, there are no known algorithms for solving the lattice problems that are typically used at the foundation of cryptographic constructions, which has led to their conjectured intractability even against quantum computer attacks. Aside from *quantum-resistance*, lattices additionally have

the unique feature of allowing for worst-case to average-case reductions. Phrased differently, a randomly selected (according to some distribution) problem instance, is at least as hard to solve as some related lattice problem in the worst case. This feature not only allows us to *reliably* base security on worst-case hardness, but also greatly simplifies key selection for constructed cryptosystems. This extraordinary observation was first made by Ajtai in [14]. Moreover, lattice-based constructions are characterized by simplicity, efficiency, and parallelizability as one typically has to perform linear operations on vectors and matrices, as well as reductions modulo some small integer. Finally, lattice-based cryptography offers great versatility and is suitable for a plethora of advanced applications like: fully-homomorphic encryption (FHE), attribute-based encryption (ABE), general-purpose code obfuscation, hierarchical ID-based constructions, and much more. For a more detailed listing of applications, the reader is referred to surveys like [132, 148].

1.3 Summary of Results

We begin by introducing the necessary notation, basic definitions, and cryptographic tools that will be used throughout this dissertation in Chapter 2. Chapter 3 offers a critical overview of the literature on blind signature schemes from lattice assumptions. The remaining chapters 4 to 6 are devoted to constructing various lattice-based signature schemes with additional features, while some also contain results of independent interest.

Chapter 3: An Overview of Lattice-Based Blind Signature Schemes and their Feasibility. Blind signatures, introduced by Chaum [55], have become an important primitive for privacy-preserving cryptography. Roughly speaking, these schemes allow a signer to sign a message without seeing it, while retaining a certain amount of control over the number of issued signatures (unforgeability). For the receiver of the signature, this process provides perfect anonymity (blindness), e.g., his spendings remain anonymous when using blind signatures for electronic money. However, designing blind signature schemes in the lattice regime has proven to be a *very* diffi-

cult task. In this Chapter, we review the most prominent literary developments that have been made towards this goal. In particular, we provide a comparative overview of these developments and point out security flaws, bottlenecks in performance and applicability of known impossibility results.

Chapter 4: Leakage-Resilient Partially-Blind Signatures from Lattices. Blind signature schemes play a pivotal role in privacy-oriented cryptography. However, with blind signature schemes, the signed message remains unintelligible to the signer, giving them no guarantee that the blinded message he signed actually contained valid information. Partially-blind signature schemes were introduced to address precisely this problem. In this Chapter we present the first leakage-resilient, lattice-based partially blind signature scheme in the literature. Our construction is provably secure in the random oracle model and offers quasilinear complexity w.r.t. key/signature sizes and signing speed. In addition, it offers statistical partial-blindness and its unforgeability is based on the computational hardness of worst-case ideal lattice problems for approximation factors in $\tilde{O}(n^4)$ in dimension n. Our scheme benefits from the subexponential hardness of ideal lattice problems and remains secure even if a $(1 - o(1))$ fraction of the signer's secret key leaks to an adversary via arbitrary side-channels. Several extensions of the security model, such as honest-user unforgeability and selective failure blindness, are also considered and concrete parameters for instantiation are proposed.

Chapter 5: A Framework for Blind Signatures with Revocable Sessions. We revisit the problem of rendering blind signatures from linear hash function families with noticeable correctness error in the random oracle model [98]. We propose for the first time a new cryptographic primitive called *blind signatures with revocable sessions* (BSRS) to model four-move schemes in which the user is able to prove that he did not obtain a valid signature from a session and can ask for the session to be revoked (e.g.: [159]). We provide a general framework for constructing BSRS schemes from *any* linear hash function family with noticeable correctness error and prove its security in the ROM by expanding upon the techniques introduced in [98]. We then instantiate our general framework from the standard SIS assumption. Our lattice-based scheme

greatly outperforms the blind signature scheme of [98] in terms of communication overhead and achieves slightly more compact signature sizes. While our construction is not practical enough to be used in practice, we believe that it may motivate future works in this area.

Chapter 6: Proxy Signatures from Ideal Lattices - Secure in All Rings. Proxy Signatures are a variant of digital signatures that allow one entity to delegate its signing rights to a second entity. This is particularly common in distributed computing where an entity (e.g.: a server) typically wants to grant privileges to other entities (e.g.: other servers) in order to be able to handle traffic, or to be able to undergo maintenance without denying its services. Over the past two decades, considerable work has been put into designing such post-quantum cryptographic primitives. In this Chapter, we propose a proxy signature scheme that offers strong security guarantees by relying on the *simultaneous* hardness of lattice problems over exponentially-many rings.

1.4 Conclusion and Open Research Questions

In this thesis, we have demonstrated the great versatility of lattices in cryptographic constructions that are geared towards preserving privacy and anonymity. Former construction principles from the area of number theoretic cryptography cannot be applied directly. In particular, because the size of objects is a crucial factor for proving a reduction from lattice problems leads to major complications. We have shown how to overcome some of them.

We believe that our constructions will, as building blocks, lead to even more complex schemes in an effort to provide a comprehensive cryptographic landscape from lattices. While it seems unlikely for these constructions to achieve simplicity and elegance that is comparable to designs from, say, pairings, we believe that ideal and module lattices hold a lot of potential that has not been fully exploited yet.

In the individual chapters, we discuss further research directions in the respective area of research.

Chapter 2

Definitions, Preliminaries and Basic Tools

This preliminary chapter, summarizes the basic definitions and results from the literature that are both relevant and necessary for presenting the constructions in later chapters.

2.1 General Notation

Throughout this thesis, we use bold, non-italic, lower-case letters like $\mathbf{x}$ to denote column vectors; for row vectors we use the transpose $\mathbf{x}^T$. Let $\mathbf{x}$ be a vector. We denote the length of $\mathbf{x}$ by $|\mathbf{x}|$. The inner product of vectors $\mathbf{x}, \mathbf{y}$ is denoted by $\mathbf{x} \cdot \mathbf{y}$ or directly as $\mathbf{x}^T\mathbf{y}$. For $1 \leq j \leq |\mathbf{x}|$, we refer to its j-th entry as $\mathbf{x}_j$. We denote the (sub)vector consisting of the first j entries of $\mathbf{x}$ by $\mathbf{x}[j]$. We generally use bold, non-italic, upper-case letters like $\mathbf{A}$ to denote matrices, and sometimes identify a matrix with its ordered set of column vectors. We denote the $n \times n$ identity matrix by $\mathbf{I}_n$. Let $\mathbf{A}$ be a matrix. We denote the i-th row of $\mathbf{A}$ as $\mathbf{A}_i$ and the j-th entry of $\mathbf{A}_i$ as $\mathbf{A}_{i,j}$. We denote the horizontal concatenation of vectors, matrices and strings with two vertical bars, e.g.: $[\mathbf{A}\|\mathbf{b}]$. Let S be a set. For $1 \leq i \leq n$ and $\mathbf{v} \in S^{i-1}$, we write $\mathbf{h}' \leftarrow_{\$} S^n|\mathbf{v}$ to denote that vector $\mathbf{h}'$ is uniformly sampled from S^n, conditioned on $\mathbf{h}'[i-1] = \mathbf{v}$. This can trivially be implemented by fixing the first $i-1$ components of $\mathbf{h}'$ to be identical to $\mathbf{v}$ and sampling

the remaining $n - i + 1$ uniformly from S.

2.2 Algorithms and Asymptotic Notation

An *algorithm* is a finite sequence of well-defined steps that given some input, produces some output. We will model all algorithms as Turing machines. Let A be an algorithm. By $\mathsf{A}(x)$ we denote the output of A on input x. If A is a *deterministic* algorithm, it will always output the same result for a given input and we will write $y \leftarrow \mathsf{A}(x)$ to denote that the output is assigned to y. On the other hand, if A uses random bits as part of its logic, it is called *probabilistic* (or *randomized*) [171]. Running a probabilistic algorithm on the exact same input will most likely result in an entirely different output. We will write $y \leftarrow_{\$} \mathsf{A}(x)$ to denote that the output is assigned to y and that A uses uniformly random coins. We will write $\mathsf{A}(x;\rho)$ to specify the random coins ρ. Notice that by fixing ρ, $\mathsf{A}(.;\rho)$ becomes deterministic. We write $\mathsf{A}(x) = v$ if A outputs v on input x.

An *interactive algorithm* is an algorithm that, before producing its final output, may produce some intermediate outputs or wait for additional inputs (possibly from some other interactive algorithm). Let X and Y be two interactive algorithms. We denote by $(a, b) \leftarrow_{\$} \langle \mathsf{X}(x), \mathsf{Y}(y) \rangle$, the joint execution of X and Y in an *interactive protocol* with private inputs x and y respectively. The respective private outputs are a for X and b for Y. By $\langle \mathsf{X}(x), \mathsf{Y}(y) \rangle^k$, we mean that the interaction can occur at most k times, where $k \in \mathbb{N} \cup \{\infty\}$. Accordingly, if Y can invoke an unbounded number of executions of an interactive protocol with X in arbitrarily interleaved order, we write $\mathsf{Y}^{\langle \mathsf{X}(x),. \rangle^{\infty}}(y)$. Finally, $\mathsf{Y}^{\langle \mathsf{X}(x_0),. \rangle^1, \langle \mathsf{X}(x_1),. \rangle^1}(y)$ means that Y can invoke arbitrarily ordered executions with $\mathsf{X}(x_0)$ and $\mathsf{X}(x_1)$, but interact with each algorithm only once.

A positive function $f(n)$ is called *negligible* in n if for any polynomial $p(n)$, there exists a $n_0 \in \mathbb{N}$, such that $f(n) \leq 1/p(n), \forall n \geq n_0$. We denote a negligible function in n by $negl(n)$. A positive function $f(n)$ is called *noticeable* (or *non-negligible*), if there exists a positive polynomial $p(n)$ and a $n_0 \in \mathbb{N}$, such that $f(n) \geq 1/p(n), \forall n \geq n_0$. A function $f(n)$ is called *overwhelming* if $1 - f(n)$ is negligible.

For asymptotics, we assume the standard Knuth notation [59], as summarized

Table 2.1: Asymptotic Notation.

Growth	Description	Limit Condition
$f(n) = O(g(n))$	f grows no faster than g asymptotically	$\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$
$f(n) = o(g(n))$	f grows slower than g asymptotically	$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$
$f(n) = \Omega(g(n))$	f grows no slower than g asymptotically	$\lim_{n\to\infty} \frac{f(n)}{g(n)} > 0$
$f(n) = \omega(g(n))$	f grows faster than g asymptotically	$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$
$f(n) = \Theta(g(n))$	f grows about as fast as g asymptotically	$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$, for some constant c.
$f(n) \approx g(n)$	f grows as fast as g asymptotically	$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$

in Table 2.1. Additionally, we will use soft-O notation to "hide" any polylogarithmic factors. For example, if $f(n) = n \log n$ and $g(n) = n^3 \log^2 n$, we will write $f(n) = \tilde{O}(n)$ and $g(n) = \tilde{O}(n^3)$, respectively. All logarithms are considered to be base 2. An algorithm is considered *efficient* if it runs in probabilistic polynomial time (PPT).

2.3 Information Theory

2.3.1 Probability Distributions

Let S be a finite set. We will write $x \leftarrow_{\$} S$ to express the fact that x is sampled uniformly at random from S. We denote the uniform distribution over S by $\mathsf{Unif}(S)$. If D is an arbitrary probability distribution, we write $x \leftarrow_{\$} D$ to denote that x is sampled according to D, and $x \sim D$ to denote that x is distributed according to distribution D.

Statistical distance provides us with a means of quantifying how "far apart" two probability distributions (or random variables) are. Although there are many definitions of statistical distance in the literature, our analysis uses the following:

Definition 2.3.1. (Statistical Distance) Let X and Y be two discrete random variables over a (countable) set S. The statistical distance $\Delta(X, Y)$ between X and Y is defined as

$$\Delta(X, Y) := \frac{1}{2} \sum_{v \in S} \left| \Pr[X = v] - \Pr[Y = v] \right|$$

A well-known property of statistical distance is that it does not increase if we apply a function f to its arguments [134].

Lemma 2.3.1. *Let S and T be finite sets, X and Y are random variables taking values in S, and $f : S \to T$ be a function. Then $\Delta(f(X), f(Y)) \leq \Delta(X, Y)$.*

We will say that X and Y are *perfectly indistinguishable* (resp. *statistically indistinguishable*) iff $\Delta(X, Y) = 0$ (resp. $\Delta(X, Y) = negl(n)$).

2.3.2 Conditional Min-Entropy

Min-entropy measures how likely one is to correctly guess a value of a random variable on his first try. If the most likely value of a random variable $W \sim D$, occurs with probability at most $2^{-\alpha}$, then we say that W has α bits of min-entropy and write $H_\infty(W) = \alpha$. More concretely, we have:

Definition 2.3.2. (Min-Entropy) Let W be a discrete random variable with possible outcomes in $\{v_1, \ldots, v_n\}$, distributed according to D, and which maps $v_i \mapsto p_i := \Pr[W = v_i], \forall i \in [n]$. The min-entropy of W is defined as: $H_\infty(W) := -\log_i(\max p_i)$.

Conditional min-entropy quantifies the amount of information needed to describe the outcome of a random variable Y, given that the value of another random variable X is known.

Definition 2.3.3. (Conditional Min-Entropy) Let X and Y be two discrete random variables. The min-entropy of Y conditioned on X is defined as:

$$H_\infty(Y|X) := \min_y \{-\log(\Pr[Y = y \mid X])$$

2.4 Computational Complexity Theory

The P = NP problem [83] plays an important role in many areas of Applied Mathematics like: Algorithmic research, Artificial Intelligence (AI), Game theory, and of course Cryptography. This section briefly recalls basic concepts and definitions from computational complexity theory.

Problems typically come in two variants: search and decision. In a *search* problem, we are given some input parameters called the *problem instance* and a specification

Algorithmica	P = NP. Cryptography is dead and AI reigns supreme.
Heuristica	NP-complete problems are hard in the worst-case (P ≠ NP) but are efficiently solvable in the average-case.
Pessiland	There exist average-case NP-complete problems but one-way functions do not exist. It is easy to create hard NP problems, but not hard NP problems where we know the solution. Hence, not only can we not solve hard problems on average, but their "hardness" offers no cryptographic advantage whatsoever. This is the worst possibility.
Minicrypt	One-way functions exist but public-key cryptosystems are impossible.
Cryptomania	Public-key cryptography is possible and secure communication is possible.

Figure 2.1: Brief description of Impagliazzo's five worlds and their implications [101].

of what a *solution* looks like. We are asked to *find* a solution matching this specification. In a *decision* problem, we are given a problem instance, and we have to *decide* whether the answer is YES or NO. Every search problem has a corresponding decision problem, and given a solution to a search variant of a problem, it is trivial to solve the corresponding decision problem. However, the converse does not hold [29]. For these reasons, computational complexity theory focuses on the study of decision problems instead of search problems. Decision problems can be generalized to allow for inputs that are neither YES, nor NO instances. Phrased differently, one has to decide, under the promise that the given problem instance is either a YES or a NO instance, which is the case [59]. If the input problem instance is neither, any output is considered a valid answer. These are called *promise* problems.

Class P consists of all (decision) problems that can be solved by a *deterministic* Turing machine in polynomial time as a function of the problem instance's size. Problems in P, are typically referred to as *tractable*, whereas problems not in P are referred to as *intractable*. Class NP consists of all problems, for which a YES instance can be verified in polynomial time by a deterministic Turing machine i.e., given a "certificate" of a solution, one can verify that the certificate is correct in polynomial time in the size

of the input to the problem. It is straightforward to show that $\mathsf{P} \subseteq \mathsf{NP}$. However, the question of whether $\mathsf{P} = \mathsf{NP}$ is the most important open question in Computer Science [89].

The *complement* $\bar{P}$ of a decision problem P is the decision problem resulting from reversing the YES and NO instances. We say that a decision problem P_1 *reduces* to decision problem P_2 if there exists a deterministic algorithm (i.e., a mathematical function) for transforming any YES (resp. NO) instance of P_1, into a YES (resp. NO) instance of P_2. We will write $P_1 \preceq P_2$ after the fact. If in addition, the reduction function is computable in polynomial time, then we say that P_1 is *polynomially reducible* to P_2. Notice that a polynomial-time reduction proves that P_1 is no harder than P_2, because whenever an efficient algorithm exists for P_2, one exists for P_1 as well. Conversely, if no efficient algorithm exists for P_1, none exists for P_2 either. Moreover, it is straightforward to prove that $\preceq$ is a preorder[1] relation among decision problems.

A problem P_0 is called NP-*hard* if any problem $P' \in \mathsf{NP}$ is polynomially-reducible to P_0. If in addition the problem itself is in NP, we say that the problem is NP-*Complete* and denote the respective class by NPC. Class NPC thus consists of the hardest problems in NP. If any problem in NPC is solvable in PPT, then $\mathsf{P} = \mathsf{NP}$ and the Polynomial-time Hierarchy would collapse. This would imply that not even something as rudimentary as one-way functions exists. This corresponds to Impagliazzo's "Algorithmica" (see Figure 2.1) and it is hence impossible to construct *any* secure cryptographic primitives. This is indicated by the following theorem:

Theorem 2.4.1. *(Theorem 34.4 from [59]) If any problem in* NPC *can be solved in polynomial time, then* $\mathsf{P} = \mathsf{NP}$.

It is for this very reason that cryptography builds upon the assumption that $\mathsf{P} \neq \mathsf{NP}$.

We also mention classes PP, coNP, BPP, AM, and coAM. Class PP consists of all decision problems that can be solved by a *probabilistic* polynomial time (PPT) Turing machine with probability error less than $1/2$. coNP consists of all decision problems in which NO instances can be verified in polynomial time by a non-deterministic

[1] i.e., reflexive and transitive.

Turing machine. Phrased differently, coNP contains all decision problems P_0 whose complements $\overline{P_0}$ are in NP. Problems that belong to both NP and coNP, are generally believed to not be NP-hard (otherwise we would have NP = coNP) [90].

Class BPP was proposed in [88] and contains all decision problems solvable by a probabilistic Turing machine in polynomial time with an error probability bounded away from 1/3 for all instances. It is not hard to see that P $\subseteq$ BPP $\subseteq$ PP. In fact, if access to randomness is removed from the definition of BPP, then we arrive at the definition for class P. BPP's relation to NP is currently unknown.

Class AM consists of all decision problems for which a YES instance can be verified by an *Arthur-Merlin protocol*, as follows: A BPP verifier called Arthur, generates a "challenge" based on the input, and sends it together with its random coins to a prover called Merlin. Merlin sends back a response, and then Arthur decides whether to accept or not based on a deterministic computation (depending on the common input and the two exchanged messages). Given an algorithm for Arthur, we require that:

- If the answer is YES, then Merlin can act in such a way that Arthur accepts with probability at least 2/3 (over the choice of Arthur's random bits).
- If the answer is NO, then regardless of how Merlin acts, Arthur will reject with probability at least 2/3.

Requiring Arthur to reveal its random bits does not weaken the system [92]. Hence, Arthur never needs to withhold information from Merlin. Class AM contains classes NP and BPP. Boppana et al. [41] show that if coNP $\subseteq$ AM, then the Polynomial-time Hierarchy collapses. Finally, class coAM is the complement of AM.

2.5 Lattice Theory

2.5.1 Cryptographic Lattices

A *lattice* is an infinite set of points in n-dimensional space with a periodic structure. The easiest way to represent a lattice is as the set of all *integer* linear combinations

$$\Lambda := \{\sum_{i=1}^{d} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z}\}$$

of d linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d \in \mathbb{R}^n$.[2] These vectors are called a *basis* for the lattice Λ and are often represented more compactly as a matrix $\mathbf{B} = [\mathbf{b}_1 \| \ldots \| \mathbf{b}_d] \in \mathbb{R}^{n \times d}$. We will write $\Lambda = \Lambda(\mathbf{B})$ to express this fact. We say that the *rank* of the lattice is d (i.e., $\text{rank}(\Lambda) = d$) and its *dimension* is n (i.e., $\dim(\Lambda) = n$). If $d = n$, the lattice is called *full-rank*. Throughout this work, we will focus on full-rank *integer lattices*, i.e., $\mathbf{B} \in \mathbb{Z}^{n \times n}$. If $d > 1$, a lattice has infinitely many bases. The quality of different bases is typically measured by the length of the basis $\|\mathbf{B}\|_p := \max_{i=1,\ldots,d} \|\mathbf{b}_i\|_p$ for some $p \in \{1, 2, \ldots\} \cup \{\infty\}$, depending on the norm used to measure the magnitude of vectors. For any basis $\mathbf{B}$, we define the *fundamental parallelepiped* $\mathcal{P}(\mathbf{B}) := \{\mathbf{Bx} \mid x_i \in [0, 1), \forall i \in [d]\}$ (see Figure 2.2 for an example). The *volume* of $\mathcal{P}(\mathbf{B})$ is defined as $\text{vol}(\mathcal{P}(\mathbf{B})) := \sqrt{\det(\mathbf{B}^T\mathbf{B})}$ (we typically refer to $\mathbf{B}^T\mathbf{B}$ as the *Gram matrix* of $\mathbf{B}$), and it can be easily proven that it is *invariant* w.r.t. basis selection (see e.g.: [46]). As such, for any lattice $\Lambda = \Lambda(\mathbf{B})$, we define its *determinant* as $\det(\Lambda) := \text{vol}(\mathcal{P}(\mathbf{B}))$.

The dual (or reciprocal) of a lattice $\Lambda \subset \mathbb{R}^n$ is defined as the set of points whose inner products with all the vectors in Λ are all integers. I.e.,

$$\Lambda^* := \{\mathbf{w} \in \text{span}(\Lambda) : \mathbf{w} \cdot \mathbf{v} \in \mathbb{Z}, \forall \mathbf{v} \in \Lambda\}$$

If $\mathbf{B}$ is a basis for Λ, then $(\mathbf{B}^{-1})^T$ is a basis for Λ^*. Another useful property states that $(c\Lambda)^* = c^{-1}\Lambda^*, \forall c \in \mathbb{R} \setminus \{0\}$ for any lattice Λ. It is easy to verify that $\mathbb{Z}^n$ is a lattice and

[2] Notice that if it were not for the restriction imposed on the coefficients, we would arrive at the well-known definition of the *linear span* of $\{\mathbf{b}_1, \ldots, \mathbf{b}_d\}$, i.e., $\text{span}(\mathbf{b}_1, \ldots, \mathbf{b}_d) := \{\sum_{i=1}^{d} x_i \mathbf{b}_i \mid x_i \in \mathbb{R}\}$.

that $(\mathbb{Z}^n)^* = \mathbb{Z}^n$.

2.5.2 Lattices and Discrete Gaussians

For any vector $\mathbf{c} \in \mathbb{R}^n$ and any real $s > 0$, the *Gaussian function* with standard deviation s and center $\mathbf{c}$ is defined as $\rho_{s,\mathbf{c}}(\mathbf{x}) := \exp\left(-\frac{\pi\|\mathbf{x}-\mathbf{c}\|^2}{s^2}\right), \forall \mathbf{x} \in \mathbb{R}^n$. The *Gaussian distribution* is defined as $D_{\mathbf{c},s}(\mathbf{x}) := \rho_{s,\mathbf{c}}(\mathbf{x})/s^n, \forall \mathbf{x} \in \mathbb{R}^n$.

Many modern works on lattices in complexity theory and cryptography rely on Gaussian-like probability distributions defined over lattices, called *discrete Gaussians*. We recall their formal definition below.

Definition 2.5.1. (Discrete Gaussian distribution over a lattice) The *discrete Gaussian distribution* over a lattice $\Lambda \subseteq \mathbb{R}^n$, with standard deviation $s > 0$ and center $\mathbf{c}$ is defined as $D_{\Lambda,s,\mathbf{c}}(\mathbf{x}) = \frac{\rho_{s,\mathbf{c}}(\mathbf{x})}{\sum_{\mathbf{v}\in\Lambda}\rho_{s,\mathbf{c}}(\mathbf{v})}, \forall \mathbf{x} \in \Lambda$. We omit $\mathbf{c}$ from the subscript when $\mathbf{c} = \mathbf{0}$.

Another very important lattice invariant is the so-called *smoothing parameter* $\eta_\epsilon(\Lambda)$ [134]. Informally, it is the amount of Gaussian "blur" required to "smooth out" essentially all the discrete structure of Λ. More formally, it is defined as follows:

Definition 2.5.2. (Smoothing Parameter) For an n-dimensional lattice Λ, and $\epsilon \in \mathbb{R}^+$, we define the *smoothing parameter* as: $\eta_\epsilon(\Lambda) := \min\{s \in \mathbb{R}^+ : \rho_{1/s}(\Lambda^* \setminus \{\mathbf{0}\}) \leq \epsilon\}$.

Finally, a set of lattice invariants are the *successive minima*. The i-th successive minimum $\lambda_i^p(\Lambda)$ is defined as the smallest radius of a n-dimensional sphere, containing at least i linearly independent lattice vectors. More concretely:

$$\lambda_i^p(\Lambda) := \min\{r \in \mathbb{R}^+ : \dim(\text{span}(\Lambda(\mathbf{B}) \cap C(\mathbf{0}, r))) \geq i\}, \forall i \in [n]$$

where: $C(\mathbf{0}, r) := \{\mathbf{v} \in \mathbb{Z}^n : \|\mathbf{v}\|_p \leq r\}$ is the closed n-dimensional ball, centered at the origin, with radius r. Notice that the successive minima are defined w.r.t. any l_p-norm, $p \in \{1, 2, \ldots\} \cup \{\infty\}$. The first successive minimum is of particular interest, because it directly ties to a computationally hard lattice problem and we call $\lambda_1^p(\Lambda) := \min\{\|\mathbf{x}-\mathbf{y}\|_p : \mathbf{x}, \mathbf{y} \in \Lambda(\mathbf{B}) \wedge \mathbf{x} \neq \mathbf{y}\} = \min\{\|\mathbf{v}\|_p : \mathbf{v} \in \Lambda(\mathbf{B}) \setminus \{\mathbf{0}\}\}$ the *minimum distance* of lattice $\Lambda(\mathbf{B})$.

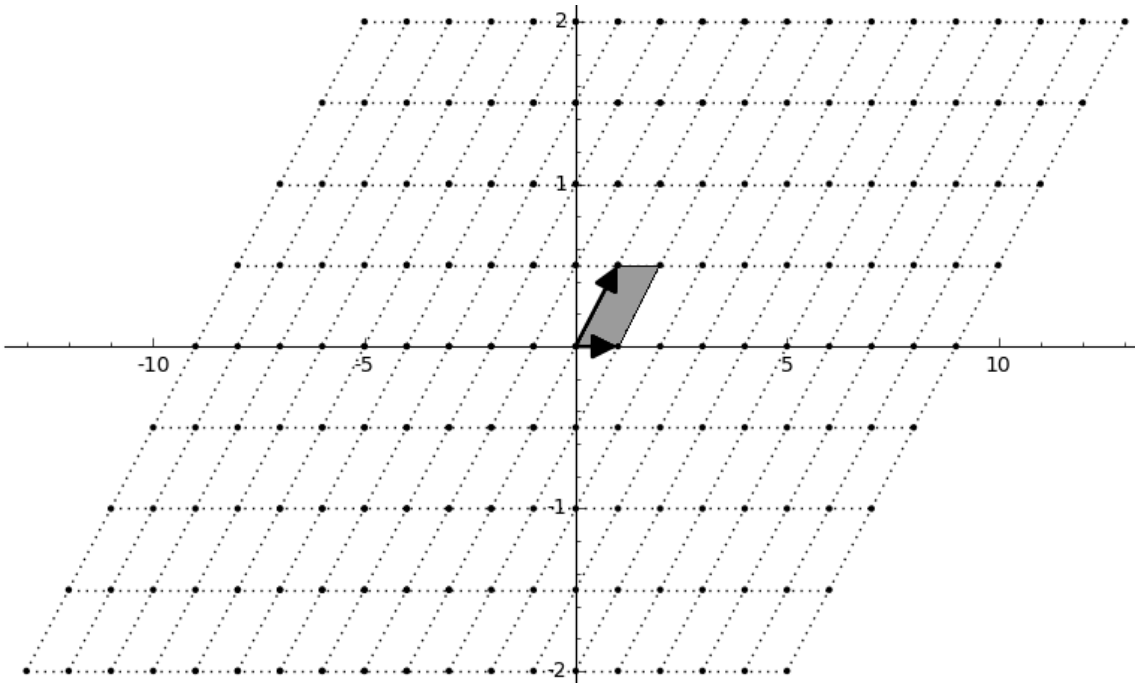

Figure 2.2: Example of the 2-dimensional lattice spanned by vectors $\mathbf{b}_1 = [1, 0]^T$ and $\mathbf{b}_2 = [1/2, 1/2]^T$. The fundamental parallelogram is shown in grey.

2.5.3 Hardness Assumptions

One of the main computationally hard problems involving lattices is the *Shortest Vector Problem* (SVP) [179, 14]. Informally, it states that given a lattice basis $\mathbf{B}$ of some lattice $\Lambda(\mathbf{B}) \subseteq \mathbb{Z}^n$, one has to find a non-zero lattice vector, whose magnitude (w.r.t. some l_p-norm) is no longer than the magnitude of all other lattice vectors. More formally:

Definition 2.5.3. (The Shortest Vector Problem - SVP^p) Let $\Lambda = \Lambda(\mathbf{B})$ be a lattice. Find a vector $\mathbf{v} \in \Lambda \setminus \{\mathbf{0}\}$, such that $\|\mathbf{v}\|_p = \min_{\mathbf{w} \in \Lambda \setminus \{\mathbf{0}\}}(\|\mathbf{w}\|_p) = \lambda_1^p(\Lambda)$.

A relaxed version of SVP allows for a relaxation factor $\gamma = \gamma(n) \in [1, \infty)$ to be taken into account. We thus obtain the following approximation problem:

Definition 2.5.4. (The Approximate Shortest Vector Problem - SVP$_\gamma^p$) Let $\Lambda = \Lambda(\mathbf{B})$ be a lattice and $\gamma \geq 1$. Find a vector $\mathbf{v} \in \Lambda \setminus \{\mathbf{0}\}$, such that $\|\mathbf{v}\|_p \leq \gamma(n) \cdot \min_{\mathbf{w} \in \Lambda \setminus \{\mathbf{0}\}}(\|\mathbf{w}\|_p) = \gamma(n) \cdot \lambda_1^p(\Lambda)$.

Notice that for an approximation factor of $\gamma = 1$, we obtain the exact version of SVP^p from Definition 2.5.3. A generalization of SVP is the *Shortest Independent*

Vectors Problem (SIVP).

Definition 2.5.5. (Approximate SIVP_γ^p) Given a basis $\mathbf{B}$ of a full-rank n-dimensional lattice $\Lambda = \Lambda(\mathbf{B})$, find a set $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\} \subset \Lambda$ of n linearly independent lattice vectors s.t.: $\|\mathbf{v}_i\|_p \leq \gamma(n) \cdot \lambda_n(\Lambda), \forall i \in [n]$.

The above problems are *search* problems since we are asked to find a solution satisfying some criteria. While several cryptosystems can be proved secure assuming the hardness of certain lattice problems, in the worst case, we are not aware of any such proof from the search version of SVP_γ^p. Instead, security proofs are more commonly based on either the search version of SIVP, or the following *promise* variant of approximate SVP:

Definition 2.5.6. (Decisional Approximate Shortest Vector Problem GapSVP_γ) An input to GapSVP_γ is a pair $(\mathbf{B}, d)$ where $\mathbf{B}$ is a basis for a full-rank n-dimensional lattice and $d \in \mathbb{R}^+$. It is a YES instance if $\lambda_1(\Lambda(\mathbf{B})) \leq d$, and is a NO instance if $\lambda_1(\Lambda(\mathbf{B})) > \gamma(n) \cdot d$. The objective is to determine which is the case.

In the above definition, $\gamma : \mathbb{N} \to \mathbb{R}^+$ acts as a *gap* between the YES instance and the NO instance. The most commonly used version of GavSVP_γ is the one in which $d = 1$.

Complexity and Algorithms

From a computational complexity perspective, lattice problems are quite fascinating and have been the object of study since antiquity.[3] Lattice problems are known to be NP-hard, even to approximate to within various sub-polynomial factors $\gamma(n) = n^{o(1)}$. In particular, the exact version of the SVP_1^∞ problem was proved to be NP-complete by Boas in [179]. Ajtai [15] later showed that SVP_1^2 is NP-hard under randomized reductions. In [131], Micciancio establishes that SVP_γ^p remains NP-hard to approximate for any $\gamma(n) < 2^{1/p}$ and w.r.t. any l_p-norm ($1 \leq p \leq \infty$), under *reverse unfaithful random* reductions. Dinur [64] later proved a considerably stronger result which states that $\mathsf{SVP}_\gamma^\infty$ is NP-hard for approximation factors up to $\gamma(n) = n^{c/\log(\log(n))}$ in the lattice

[3] Indeed, results can be traced back to Euclid for 1-dimensional lattices and to Gauss for 2-dimensional lattices.

dimension n, where $c > 0$ is a constant. Khot [110] further improved this by showing that approximating SVP_γ^p to within arbitrarily large constants under randomized reductions for any $1 < p < \infty$ is NP-hard. Furthermore, he shows that under randomized quasipolynomial-time reductions, the approximation factor becomes $2^{(\log(n))^{1/2-\epsilon}}$ for any $\epsilon > 0$. Finally, Haviv and Regev [99] further refined the state-of-the-art by proving that for any $\epsilon > 0$ there is no polynomial-time algorithm approximating the SVP in n-dimensional lattices in the l_p-norm ($1 \leq p < \infty$) to within a factor of $2^{(\log(n))^{1-\epsilon}}$. On the other hand, the SIVP problem is NP-hard for any approximation factor $\gamma(n) \in O(1)$ [36]. However, the aforementioned hardness results are not very relevant to "real-world" lattice-based cryptography because lattice-based cryptographic constructions typically rely on approximation factors $\gamma(n) \geq n$. Indeed, as the groundbreaking works of [14, 130] suggest (see Figure 2.3), constructing something as primordial as one-way functions (OWFs), involves approximation factors starting from $\tilde{O}(n)$.

In terms of algorithms, the most well-known algorithm for solving SVP (as well as most other lattice problems) is the LLL algorithm [117] which only provides a polynomial-time approximation for SVP for very large approximation factors of $2^{n/2}$ in the lattice dimension n. Schnorr [165] presented an improvement of the LLL algorithm leading to somewhat better approximation factors $\gamma(n) = 2^{n\log(\log(n))^2/\log(n)}$. By allowing randomization, Ajtai et al. [16] slightly improved the approximation factor to $\gamma(n) = 2^{n\log(\log(n))/\log(n)}$. While there exist algorithms for $\gamma(n) = poly(n)$ [16, 136, 8], their running times are either super-exponential $2^{\Theta(n\log(n))}$ or exponential $2^{\Theta(n)}$ in *both* time and space. It is possible to allow tradeoffs between the running time and the approximation factors by interpolating between these two classes. This results to approximation factors of $\gamma(n) = 2^k$ and a time complexity of $2^{\tilde{O}(n/k)}$ [165]. This represents the state of the art for quantum algorithms (also see [141] for an experimental evaluation of some LLL-type algorithms). For SIVP, the best known algorithms (including quantum algorithms) for obtaining an exact solution and an approximation to within any factor $\gamma(n) = poly(n)$ all have exponential complexities [135].

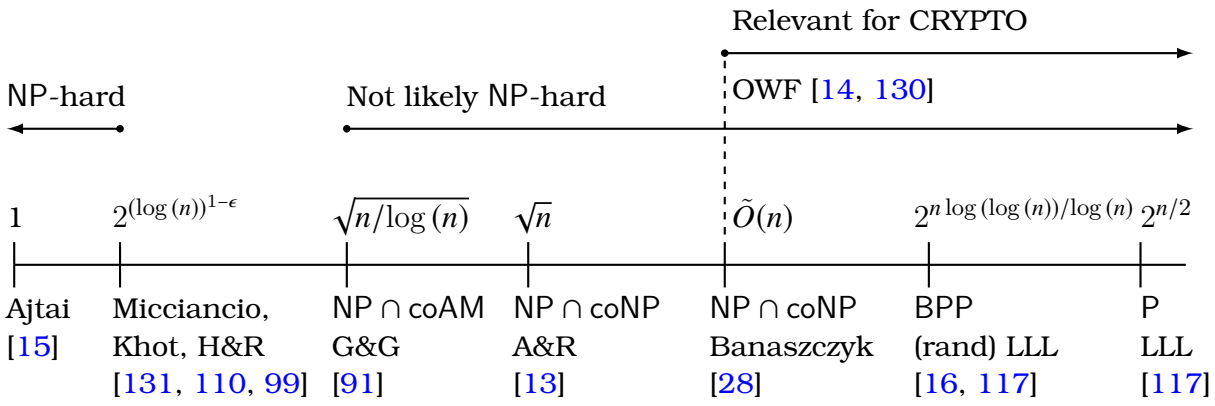

Figure 2.3: Complexity of the Shortest Vector Problem (constants are omitted).

From the above discussion, and as one can verify from Figure 2.3, there is a *very wide* gap between the best known hardness result (i.e., $2^{(\log(n))^{1-\epsilon}}$) approximation factor and the best approximation that can be obtained in polynomial time (i.e., $2^{n\log(\log(n))/\log(n)}$). In particular, as we mentioned earlier, we are interested in polynomial approximation factors $\gamma(n) = n^{\kappa}, \kappa \in \mathbb{N}$ (preferably of small degree κ). However, such factors exceed our state of knowledge for NP-hardness. In fact, *limits on inapproximability* exist which point towards the contrary. In particular, for approximation factors $\gamma(n) \geq \sqrt{n}$ the works of [91, 13] prove that GapSVP_{γ} is in $\mathsf{NP} \cap \mathsf{coNP}$, and is thus (likely) no longer NP-hard (see Figure 2.3).[4] Similarly, for $\gamma(n) = \tilde{O}(n)$, [28] shows containment of GapSVP_{γ} in $\mathsf{NP} \cap \mathsf{coNP}$. Finally, the work of Goldreich and Goldwasser [91] provides us with the currently best known limit on inapproximability, stating for $\gamma(n) = \sqrt{n/\log(n)}$, GapSVP_{γ} lies in $\mathsf{NP} \cap \mathsf{coAM}$ and is thus also unlikely to be NP-hard.

The above discussion justifiably gives rise to the question: Can we even hope to base the security of cryptosystems on the hardness of approximating lattice problems to within polynomial factors since the latter are unlikely to be NP-hard? The answer is an astounding YES! Given how stagnant progress has been in coming up with algorithms that perform even slightly better than the exponential factor achieved by LLL [117], many people conjecture that there do not exist efficient algorithms for

[4] Indeed, if GapSVP_{γ} were NP-hard, then NP = coNP and the polynomial hierarchy would collapse.

approximating lattice problems to within polynomial factors. Therefore, lattice-based cryptography relies on the following core conjecture:

Conjecture 1. *There exists no polynomial time (even quantum) algorithm for approximating lattice problems to within a polynomial factor.*

Worst-case to Average-case Reducibility

Conjecture 1 makes worst-case problems like GapSVP and SIVP attractive hardness hypotheses for lattice-based cryptographic schemes. While basing the security of cryptosystems on the worst-case hardness of a problem is the safest thing to assume, this is still not good enough for applications. Indeed, worst-case problems are considered hard on the basis that they have *at least one* intractable instance. Unfortunately, it is not uncommon for problems that appear hard in the worst case to turn out to be easier on the average. This is especially true for distributions that produce instances with some extra "structure". Instead, what cryptography requires in practice is average-case hardness. In other words, we need problems for which *randomly* drawn instances from a specified probability distribution (for example, uniform) are hard to solve (see Figure 2.4).

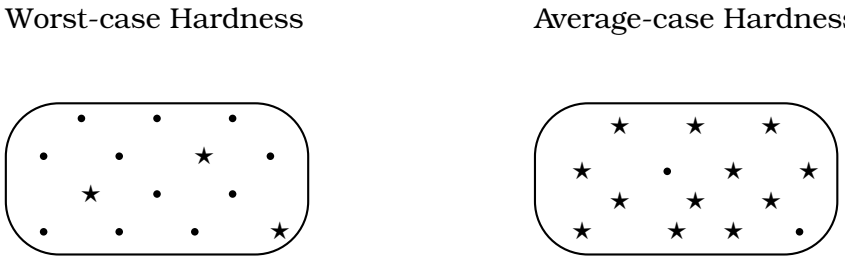

Figure 2.4: Difference between worst-case hardness and average-case hardness (hard instances are denoted by a star).

In his seminal work, Ajtai [14] gave a groundbreaking connection between the worst case and the average case for lattices. In particular, he proved that for cryptographically relevant distributions, certain problems are hard on the average as long as some

related lattice problems are hard in the worst case. Using results of this kind, one can design cryptographic constructions and prove that they are infeasible to break, unless *all* instances of certain lattice problems are easy to solve. This hardness guarantee is a unique feature of lattice-based cryptography.

The reader familiar with the *discrete logarithm problem* (DLP) might notice similarities to the well-known notion of *random self-reducibility (RSR)*[5]. However, it should be noted that while DLP allows us to choose a random instance that is at least as hard to solve as any other instance, it keeps the cyclic group fixed.

The average-case lattice problems that are heavily featured in lattice-based cryptography are the *short integer solution* (SIS) problem and the *learning with errors* (LWE) problem.

2.5.4 The Short Integer Solution Problem

SIS was first introduced in [14] and since then has been used as the foundation for lattice-based "minicrypt" i.e., cryptographic constructions that can be built upon one-way functions (e.g.: one-way hash functions, collision-resistant hash functions, identification schemes, digital signature schemes, etc). Informally, the SIS problem asks, given many uniformly random elements of a certain large finite additive group, to find a sufficiently "short" nontrivial *integer* combination of them that sums to zero. Below we give a formal definition w.r.t. to the Euclidean norm (i.e., the l_2-norm) but the problem can easily be rephrased for any l_p-norm, $p \in \mathbb{N} \cup \{\infty\}$.

Definition 2.5.7. (Short Integer Solution $\mathsf{SIS}_{q,n,m,\beta}$) We define the SIS problem in terms of the following game $\mathbf{SIS}_{q,n,m,\beta}$:

Game $\mathbf{SIS}_{q,n,m,\beta}$:

- **Setup.** On input prime modulus q, lattice dimension $n \in \mathbb{N}$, solution dimension $m \in \mathbb{N}$, and a norm bound $\beta \in \mathbb{R}^+$, game $\mathbf{SIS}_{q,n,m,\beta}$ samples m uniformly random vectors $\mathbf{a}_i \leftarrow_{\$} \mathbb{Z}_q^n$ forming the columns of a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. It then runs adversary A on input q, n, m, β and $\mathbf{A}$.

[5] I.e., the existence of an efficient algorithm for solving a random instance of DLP implies the existence of an efficient algorithm for solving any instance of DLP.

- **Output Determination.** When A outputs a vector $\mathbf{z} = (z_1, \ldots, z_m) \in \mathbb{Z}^m$, the game returns 1 iff: (i) $\|\mathbf{z}\| \leq \beta$, (ii) $\mathsf{F}_{\mathbf{A}}(\mathbf{z}) := \mathbf{A}\mathbf{z} = \sum_{i=1}^{m} \mathbf{a}_i z_i = \mathbf{0} \in \mathbb{Z}_q^n$, and (iii) $\mathbf{z} \neq \mathbf{0}$. Otherwise, it outputs 0.

We define the *advantage* of adversary A in Game $\mathbf{SIS}_{q,n,m,\beta}$ as:

$$\mathbf{Adv}^{\mathbf{SIS}}_{q,n,m,\beta}(\mathsf{A}) := \Pr_{\mathbf{A} \leftarrow_{\$} \mathbb{Z}_q^{n \times m}} \left[\mathbf{SIS}^{\mathsf{A}}_{q,n,m,\beta} = 1 \right].$$

The function $\mathsf{F}_{\mathbf{A}}$ is often called *Ajtai's function* or the *SIS function*. When clear from context, $\mathbf{A}$ is omitted from the subscript.

We now make some important remarks about the SIS problem:

1. The lattice dimension n is considered the main hardness parameter and all other parameters are expressed as functions of n.

2. The norm bound should satisfy $\beta < q$. Indeed, if that were not the case, the problem would trivially admit $\mathbf{z} = (0, \ldots, 0, q) \in \mathbb{Z}^m$ as a solution.

3. Without the shortness constraint $\|\mathbf{z}\| \leq \beta$, SIS would belong in class P because we would be able to employ the Gaussian elimination algorithm.

4. The norm bound β and the solution dimension m must be set in a way that a solution actually exists. By the pigeonhole principle, this is the case if $\beta \geq \sqrt{\lceil n \log(q) \rceil}$ and $m \geq \lceil n \log(q) \rceil$. Indeed, assume that $m = \lceil n \log(q) \rceil$. Since there exist more than q^n vectors $\mathbf{z} \in \{0,1\}^m$, there must also exist two vectors $\mathbf{z}_1, \mathbf{z}_2$ s.t. $\mathsf{F}_{\mathbf{A}}(\mathbf{z}_1) = \mathsf{F}_{\mathbf{A}}(\mathbf{z}_2) \in \mathbb{Z}_q^n$, so their difference $\mathbf{z}' := \mathbf{z}_1 - \mathbf{z}_2 \in \{0, \pm 1\}^m$ is a solution s.t. $\|\mathbf{z}'\| \leq \beta$.

5. The above observation also implies that the induced function family $\{\mathsf{F}_{\mathbf{A}} : \{0,1\}^m \rightarrow \mathbb{Z}_q^n\}$ is collision-resistant based on the hardness of SIS. It is trivial to see that if $\mathbf{z}_1, \mathbf{z}_2 \in \{0,1\}^m$ is a collision for $\mathsf{F}_{\mathbf{A}}$, then $\mathbf{z}_1 - \mathbf{z}_2$ is a solution for SIS.

The SIS problem can be seen as an average-case short-vector problem on the

following family of lattices:

$$\Lambda_q^{\perp}(\mathbf{A}) := \{\mathbf{v} \in \mathbb{Z}^m \mid \mathbf{Av} = \mathbf{0} \pmod q\} \supseteq q\mathbb{Z}^m.$$

These lattices are often referred to as *q-ary* or *SIS lattices*. $\mathbf{A}$ can be seen as a "parity-check" matrix that defines the lattice $\Lambda_q^{\perp}(\mathbf{A})$ in a manner similar to coding theory. Therefore, the SIS problem asks to find a sufficiently short (w.r.t. some norm) non-zero vector in $\Lambda_q^{\perp}(\mathbf{A})$, where $\mathbf{A}$ is chosen uniformly at random.

In his groundbreaking paper [14], Ajtai described for the first time how *any* instance of the SIVP problem in dimension n can be phrased as a *random* instance of the SIS problem. This worst-case to average-case reduction implies that a random instance of SIS is *at least as hard* to solve as the hardest instance of SIVP in dimension n. Hence, solving a non-negligible portion of instances of SIS in PPT directly yields a PPT algorithm for solving *all* instances of SIVP in dimension n. Later works by Micciancio and Regev [130], and Gentry et al. [86] improved upon this by showing tighter reductions:

Lemma 2.5.1. ($\mathsf{SIVP} \leq \mathsf{SIS}$, *from [86]*) *For any $\nu \leq poly(n)$, prime $q \leq \nu g(n)$ for $g(n) = \omega(\sqrt{n \log(n)})$, and $m \geq 2n \log(q) = \Omega(n \log(n))$, the average-case problem $\mathsf{SIS}_{q,n,m,\beta}$ is at least as hard as SIVP_γ in dimension n in the worst case with $\gamma = \nu\tilde{O}(\sqrt{n})$.*

A variant of SIS called the k-SIS problem was introduced in [39].

Definition 2.5.8. ($k - \mathsf{SIS}_{q,n,m,\beta,\sigma}$) For any integer $k \in \mathbb{N}$, we define the $k - \mathsf{SIS}$ problem in terms of the following game $k - \mathbf{SIS}_{q,n,m,\beta}$:
Game $k - \mathbf{SIS}_{q,n,m,\beta}$:

- **Setup.** On input a prime modulus q, lattice dimension $n \in \mathbb{N}$, solution dimension $m \in \mathbb{N}$, and norm bound $\beta \in \mathbb{R}^+$, game $k - \mathbf{SIS}_{q,n,m,\beta}$ samples m uniformly random vectors $\mathbf{a}_i \leftarrow_\$ \mathbb{Z}_q^n$ forming the columns of a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. It also samples a set of k vectors $\mathbf{e}_1, \ldots, \mathbf{e}_k \in \Lambda_q^{\perp}(\mathbf{A})$ via $\mathbf{e}_i \leftarrow_\$ D_{\Lambda_q^{\perp}(\mathbf{A}),\sigma}, \forall i \in [k]$. It then runs adversary A on input $q, n, m, \beta, \mathbf{A}$, and $\mathbf{e}_1, \ldots, \mathbf{e}_k$.

- **Output Determination.** When A outputs a vector $\mathbf{z} = (z_1, \ldots, z_m) \in \mathbb{Z}^m$, the game returns 1 iff: (i) $\|\mathbf{z}\| \leq \beta$, (ii) $\mathsf{F}_{\mathbf{A}}(\mathbf{z}) := \mathbf{Az} = \sum_{i=1}^{m} \mathbf{a}_i z_i = \mathbf{0} \in \mathbb{Z}_q^n$, (iii) $\mathbf{z} \neq \mathbf{0}$, and (iv) $\mathbf{v} \notin \mathbb{Q} - \text{span}(\{\mathbf{e}_1, \ldots, \mathbf{e}_k\})$. Otherwise, it outputs 0.

We define the *advantage* of adversary A in Game $k - \mathbf{SIS}_{q,n,m,\beta}$ as:

$$\mathbf{Adv}^{k-\mathbf{SIS}}_{q,n,m,\beta}(\mathsf{A}) := \Pr_{\mathbf{A} \leftarrow_{\$} \mathbb{Z}_q^{n \times m}, \mathbf{e}_i \leftarrow_{\$} D_{\Lambda_q^{\perp}(\mathbf{A}),\sigma}} \left[k - \mathbf{SIS}^{\mathsf{A}}_{q,n,m,\beta} = 1 \right].$$

Notice that for $k = 0$, $k-\mathsf{SIS}$ is identical to the SIS problem. Boneh and Freeman also provide a reduction from the SIS problem in dimension $m - k$ to the $k - \mathsf{SIS}$ problem in dimension m. Worst-case to average-case reducibility is deduced by combining lemmas 2.5.1 and 2.5.2.

Lemma 2.5.2. *($\mathsf{SIS} \leq k - \mathsf{SIS}$ [39]) Let q be a prime, and let m, β, σ, and k, be polynomial functions of a security parameter n. Suppose that $m \geq 2n \log(q), m/k > n, \sigma > \omega(\sqrt{\log(m)}), t > \omega(\sqrt{\log(n)})$, and $q > \sigma \cdot \omega(\sqrt{\log(m)})$. Let $\beta' := \beta \cdot (k^{3/2} + 1)k!(t\sigma)^k$. Let* A *be a polynomial-time adversary for the $k - \mathsf{SIS}_{q,n,m,\beta,\sigma}$ problem. Then there exists a polynomial-time algorithm* B *for solving $\mathsf{SIS}_{n,q,m-k,\beta'}$, s.t.*

$$\boldsymbol{Adv}^{\mathbf{SIS}}_{q,n,m-k,\beta'}(\mathsf{B}) \geq \boldsymbol{Adv}^{k-\mathbf{SIS}}_{q,n,m,\beta}(\mathsf{A}) - \mathit{negl}(n).$$

Since the SIS problem is only assumed to be hard for parameters $\beta = \mathit{poly}(n)$, the fact that the above reduction degrades exponentially in k means that k must be chosen to be small enough so that β' is still polynomial in n.

2.5.5 The Learning With Errors Problem

Another very prominent average-case problem for lattice-based cryptography is the *learning with errors* (LWE) problem which was introduced by Regev in [156]. Despite its syntactical similarities to SIS, the LWE problem enables lattice-based "cryptomania" [101]. Applications include public key encryption schemes [156], as well as other advanced constructions like identity-based encryption [86, 52, 10, 11], homomorphic

encryption [85], fully dynamic multi-key FHE [45], predicate encryption [94], obfuscation [40], etc.

LWE is parameterized by integers $n, q \in \mathbb{N}$, and an error/noise distribution χ over $\mathbb{Z}$. For concreteness, n and q can be thought of as roughly the same as in SIS, and χ is usually taken to be a discrete Gaussian of width αq for some $\alpha < 1$, which is often called the relative "error rate".

Definition 2.5.9. (LWE distribution) For a *secret* vector $\mathbf{s} \in \mathbb{Z}_q^n$, the LWE distribution $A_{\mathbf{s},\chi}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ is sampled by choosing $\mathbf{a} \leftarrow_\$ \mathbb{Z}_q^n$, choosing error via $e \leftarrow_\$ \chi$, and outputting $(\mathbf{a}, b := \mathbf{a} \cdot \mathbf{s} + e \pmod q)$.

The LWE problem comes in two version: *search*, which asks to find the secret given multiple LWE samples, and *decision*, which is to distinguish between samples from the LWE distribution and uniformly random ones. Both versions are parameterized by the number $m \in \mathbb{N}$ of available samples, which is typically set so that the secret is uniquely defined with high probability.

Definition 2.5.10. (Decision version of Learning With Errors ($\mathsf{LWE}_{n,q,\chi,m}$)) The decision version of the LWE problem is defined in terms of the following game $\mathbf{LWE}_{n,q,\chi,m}$:

Game $\mathbf{LWE}_{n,q,\chi,m}$:

- **Setup.** On input an dimension $n \in \mathbb{N}$, integer modulus $q \geq 2$, error distribution χ, and number of samples $m \in \mathbb{N}$, game $\mathbf{LWE}_{n,q,\chi,m}$ samples a secret $\mathbf{s} \leftarrow_\$ \mathbb{Z}_q^n$ and $b \leftarrow_\$ \{0,1\}$. If $b = 0$ it samples m independent LWE samples via $(\mathbf{a}_i, b_i) \leftarrow_\$ A_{\mathbf{s},\chi}, \forall i \in [m]$. Otherwise (i.e., $b = 1$), it samples $(\mathbf{a}_i, b_i) \leftarrow_\$ \mathbb{Z}_q^n \times \mathbb{Z}_q, \forall i \in [m]$. It then runs adversary A on input $(\mathbf{a}_i, b_i), i \in [m]$.

- **Output Determination.** When the adversary outputs a bit b^*, the game returns 1 iff $b = b^*$. Otherwise, it returns 0.

We define the *advantage* of adversary A in Game $\mathbf{LWE}_{n,q,\chi,m}$ as:

$$\mathbf{Adv}^{\mathbf{LWE}}_{n,q,\chi,m}(\mathsf{A}) := \Pr\left[\mathbf{LWE}^{\mathsf{A}}_{n,q,\chi,m} = 1\right].$$

The search version of LWE differs only in that during the Output Determination phase, the adversary A has to output a vector $\mathbf{s}^* \in \mathbb{Z}_q^n$ s.t. $\mathbf{s}^* = \mathbf{s}$.

We now make some important remarks about the LWE problem:

1. If not for the error terms from χ, both problems would be trivial to solve. Indeed, the underlying secret vector $\mathbf{s}$ can be retrieved from the LWE samples via Gaussian elimination. In the case of decision-LWE where $b = 1$, no solution $\mathbf{s}$ will exist with high probability.

2. It is often convenient to combine the given samples into a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ (whose columns are the vectors $\mathbf{a}_i \in \mathbb{Z}_q^n$) and a vector $\mathbf{b} = (b_1, \ldots, b_m)^T \in \mathbb{Z}_q^m$, so that for LWE samples we have $\mathbf{b}^T = \mathbf{s}^T\mathbf{A} + \mathbf{e}^T \pmod q$, where $\mathbf{e} \leftarrow_\$ \chi^m$. In the decision version of LWE with $b = 1$, vector $\mathbf{b}$ is uniformly random and independent of $\mathbf{A}$.

In his groundbreaking work [156], Regev proved the following worst-case to average-case reduction for LWE:

Lemma 2.5.3. ($\mathsf{GapSVP}, \mathsf{SIVP} \leq_Q \mathsf{LWE}$ *[156]*) *Let $\alpha = \alpha(n)$ with $0 < \alpha < 1$, and prime $q > 2\sqrt{n}/\alpha$, and $m = m(n) \leq poly(n)$. The average-case decision problem $\mathsf{LWE}_{n,q,\chi,m}$ with noise distribution χ (with parameter α) is at least as hard as quantumly solving SIVP_γ and GapSVP_γ in dimension n in the worst case with $\gamma = \tilde{O}(n/\alpha)$.*

It is important to note that the exact values of m and q are irrelevant for the hardness guarantee (apart from the constraint $q \geq 2\sqrt{n}/\alpha$). However, the approximation factor γ is essential and in fact degrades with the inverse error rate $1/\alpha$ of the LWE problem.

Lemma 2.5.3 is proved by giving a *quantum* polynomial-time reduction that uses an oracle for LWE to solve GapSVP_γ and SIVP_γ in the worst case. Hence, any algorithm (classical or quantum) that solves LWE can be transformed into a quantum algorithm for GapSVP_γ and SIVP_γ. The quantum nature of the reduction is meaningful because there are no known quantum algorithms for GapSVP_γ or SIVP_γ that significantly outperform classical ones, beyond generic quantum speedups.

A later work by Peikert [147] partially dequantized Regev's worst-case to average-case reduction [156] from Lemma 2.5.3. In particular, Peikert proved that LWE with

an error rate of α is classically at least as hard as worst-case GapSVP_γ, for the same $\gamma = \tilde{O}(n/\alpha)$ factor as in Lemma 2.5.3. His reduction however comes with two notable caveats:

1. Unlike Regev's quantum reduction [156], the classical reduction works *only* for GapSVP.

2. Peikert's classical reduction requires an exponential modulus $q \geq 2^{n/2}$ for the LWE problem, whereas Regev's quantum reduction can work for any modulus $q \geq 2\sqrt{n}/\alpha$. In practice, a larger modulus means that LWE samples require many more bits to represent. This in turn affects key sizes and leads to less-efficient cryptographic schemes.

Peikert's classical reduction can be adapted to work for polynomial moduli at the expense of relying on a non-standard variant of GapSVP. Influenced by techniques like "key-switching" and "modulus reduction" from the literature of fully homomorphic encryption, Brakerski et al. [44] gave a general dimension-modulus tradeoff for LWE, which roughly says that hardness for a particular error rate α is determined almost entirely by $n \log(q)$, and not by the particular choices of n and q, provided that q is lower bounded by some small polynomial. The reduction is informally stated below:

Lemma 2.5.4. *(*$\mathsf{GapSVP} \preceq \mathsf{LWE}$*, from [44]) Solving* $\mathsf{LWE}_{n,q,\chi,m}$ *with* $q = \mathit{poly}(n)$ *implies an equally efficient solution to* GapSVP_γ*, where* $\gamma = \sqrt{n}$*.*

2.5.6 Ideal Lattices

Despite their elegance, q-ary lattices are impractical to use. Indeed, matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ takes $mn \log(q) = \Omega(n^2)$ space (to represent) and time (simply to read).[6] We can ameliorate this caveat by restricting the underlying lattice problems to a subclass of lattices, called *ideal lattices*, which possess some specialized algebraic structure.

Recall that if $\mathcal{R}$ is a ring, then $I \subseteq \mathcal{R}$ is an *ideal* in $\mathcal{R}$ iff it is an additive subgroup of $\mathcal{R}$ that is closed under multiplication with $\mathcal{R}$ [79]. Informally, an ideal lattice is simply a lattice corresponding to an ideal in some ring $\mathcal{R}$, under some fixed choice of

[6] Recall that we must have $m > n$ in order for Ajtai's function to be a compressing function.

geometric embedding. What this means is that any lattice vector can be interpreted – under the geometric embedding – as an element in I and vice versa. Since I is closed under addition and multiplication, we obtain lattices with a much richer structure.[7]

Throughout this thesis, $\mathcal{R}$ will denote the polynomial ring $\mathbb{Z}[X]/\langle \mathbf{f} \rangle$ and $\mathcal{R}_q$ will denote the polynomial ring $\mathcal{R}/q\mathcal{R} = \mathbb{Z}_q[X]/\langle \mathbf{f} \rangle$, where q is a prime and $\mathbf{f} \in \mathbb{Z}[X]$ is any monic, irreducible polynomial of degree n. For efficiency reasons, the preferred choice for $\mathbf{f}$ is X^n+1, where n is a power of 2 (although the ring-structure induced by this choice of $\mathbf{f}$ allows for much shorter key-sizes and makes operations more efficient through the Fast Fourier Transform, it provides no further functionality [123, p. 2]). Furthermore, the ring of integers modulo q will be identified with the set $\{-\frac{q-1}{2}, \ldots, \frac{q-1}{2}\}$. We will use *coefficient embedding* and will identify any polynomial $\mathbf{g} \in \mathcal{R}_q$ with its coefficient vector $\mathbf{g} = (g_0, \ldots, g_{n-1}) \in \mathbb{Z}_q^n$ (i.e., we will treat polynomials of $\mathcal{R}_q$ and vectors of $\mathbb{Z}_q^n$ as equivalent). Conventionally, we will denote polynomials in $\mathcal{R}_q$ with boldface, non-italic letters and m-tuples of polynomials in $\mathcal{R}_q^m$ with boldface, non-italic letters and a hat. It is not hard to see that under coefficient embedding, $\mathcal{R} \cong \mathbb{Z}^n$ and $\mathcal{R}_q^m \cong \mathbb{Z}_q^{mn}, \forall m \in \mathbb{N}$ with vector addition corresponding to polynomial addition, and matrix-vector multiplication corresponding to the *convolution product* $\hat{\mathbf{a}} \cdot \hat{\mathbf{b}} := \sum_{i=0}^{m-1} \mathbf{a}_i \mathbf{b}_i$ (modulo $X^n + 1$ and q) of polynomials in $\mathcal{R}_q$. For all $\hat{\mathbf{a}}, \hat{\mathbf{b}} \in \mathcal{R}_q^m$ and $\mathbf{c} \in \mathcal{R}_q$, the product $\hat{\mathbf{a}} \cdot \hat{\mathbf{b}}$ is a *scalar product* and satisfies the properties: (i) $\hat{\mathbf{a}} \cdot \hat{\mathbf{b}} = \hat{\mathbf{b}} \cdot \hat{\mathbf{a}}$, and (ii) $\mathbf{c}(\hat{\mathbf{a}} \cdot \hat{\mathbf{b}}) = (\mathbf{c}\hat{\mathbf{a}}) \cdot \hat{\mathbf{b}} = \hat{\mathbf{a}} \cdot (\mathbf{c}\hat{\mathbf{b}})$. We slightly abuse notation and define $\|\mathbf{g}\|_\infty := \max_i |g_i|$ and $\|\hat{\mathbf{g}}\|_\infty := \max_i(\|\mathbf{g}_i\|_\infty)$.

As we stated above, a lattice corresponds to an ideal $I \subset \mathcal{R}_q$, iff every lattice vector is the coefficient vector of a polynomial in I. Using the definition of an ideal, it is easy to interpret an ideal lattice as a q-ary lattice with special structure. In particular, an element $\hat{\mathbf{a}} = (\mathbf{a}_1, \ldots, \mathbf{a}_m) \in \mathcal{R}_q^m$ defines the ideal lattice:

$$\Lambda^\perp_{\mathcal{R}_q} := \{\hat{\mathbf{x}} \in \mathcal{R}^m \mid \hat{\mathbf{a}} \cdot \hat{\mathbf{x}} \equiv \mathbf{0} \in \mathcal{R}_q^m\}$$

Given $\Lambda^\perp_{\mathcal{R}_q}$, the corresponding q-ary lattice $\Lambda^\perp_{\mathbf{A}}$ is defined via the matrix $\mathbf{A} := \mathbf{A}_1 \| \ldots \| \mathbf{A}_m$,

[7] Phrased differently, lattices are groups, while ideal lattices are ideals.

where:

$$\mathbf{A}_i := \begin{bmatrix} a_{i,0} & -a_{i,n-1} & \dots & -a_{i,1} \\ a_{i,1} & a_{i,0} & \dots & -a_{i,2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i,n-1} & a_{i,n-2} & \dots & a_{i,0} \end{bmatrix}, \forall i \in [m].$$

Notice that the two lattices are identical. However, as we will discuss shortly, $\hat{\mathbf{a}}$ provides us with a much more succinct means of representation.

Ideal Lattice Problems

Worst-case problems like SVP and SIVP easily translate to ideal lattices and are called Ring – SVP and Ring – SIVP, respectively. For typical choices of rings, and for cryptographically relevant approximation factors γ, the Ring – SVP_γ and Ring – SIVP_γ problems on ideal lattices appear to be intractable in the worst case. All currently known algorithms (classical or quantum), seem to only achieve meagre speedups by exploiting the specialized algebraic structure of ideal lattices for these problems. In particular, the best known (quantum) algorithms for Ring – SVP_γ, where $\gamma = poly(n)$ on ideal lattices in typical choices of rings take exponential $2^{\Omega(n)}$ time.

Average-case problems can also be phrased w.r.t ideal lattices. Within the context of ideal lattices, the k – SIS problem is called Ring k – SIS and is formally defined as follows:

Definition 2.5.11. (Ring k – $\mathsf{SIS}_{q,n,m,\beta}$) For any integer $k \geq 0$, we define the Ring k – $\mathsf{SIS}_{q,n,m,\beta}$ problem in terms of the following game **Ring** k – $\mathbf{SIS}_{q,n,m,\beta}$:
Game **Ring** k – $\mathbf{SIS}_{q,n,m,\beta}$:

- **Setup.** On input modulus q, lattice dimension $n \in \mathbb{N}$, solution dimension $m \in \mathbb{N}$, norm bound $\beta \in \mathbb{R}^+$, game **Ring** k – $\mathbf{SIS}_{q,n,m,\beta}$ samples a vector $\hat{\mathbf{a}} \leftarrow_\$ \mathcal{R}^m$ and k short polynomial vectors $\hat{\mathbf{e}}_1, \dots, \hat{\mathbf{e}}_k$ s.t. $\hat{\mathbf{a}} \cdot \hat{\mathbf{e}}_i = \mathbf{0} \pmod q, \forall i \in [k]$ and runs adversary A on input $\hat{\mathbf{a}}$, q, n, m, β and $\hat{\mathbf{e}}_1, \dots, \hat{\mathbf{e}}_k$.

- **Output Determination.** When A outputs a vector $\hat{\mathbf{v}} \in \mathcal{R}^m$, the game returns 1 iff: (i) $\hat{\mathbf{v}} \neq \mathbf{0}$, (ii) $\|\hat{\mathbf{v}}\| \leq \beta$, (iii) $\hat{\mathbf{a}} \cdot \hat{\mathbf{v}} = \mathbf{0} \pmod q$, and (iv) $\hat{\mathbf{v}} \notin \mathcal{R} \setminus \text{span}(\hat{\mathbf{e}}_1, \dots, \hat{\mathbf{e}}_k)$. Otherwise, it outputs 0.

We define the *advantage* of adversary A in Game **Ring** $k - \mathbf{SIS}_{q,n,m,\beta}$ as:

$$\mathbf{Adv}^{\mathbf{Ring}\ k-\mathbf{SIS}}_{q,n,m,\beta}(\mathsf{A}) := \Pr\left[\mathbf{Ring}\ k - \mathbf{SIS}^{\mathsf{A}}_{q,n,m,\beta} = 1\right].$$

The Ring-SIS problem ($\mathsf{R}-\mathsf{SIS}$) can be seen as a special case of Ring $k - \mathsf{SIS}_{q,n,m,\beta}$, where $k = 0$. In order to guarantee that a solution exists for $\mathsf{R}-\mathsf{SIS}$, we typically set $m = O(\log(n))$ (as opposed to $m = \Omega(n\log(q))$ for standard SIS). This enables a more succinct representation for ideal lattices since we only require $mn\log(q) = O(n\log^2(n)) = \tilde{O}(n)$ bits of storage. Computing the convolution product $\hat{\mathbf{a}} \cdot \hat{\mathbf{b}} \in \mathcal{R}_q$ can also be achieved in quasi-linear $O(n\log(n)) = \tilde{O}(n)$ time by using FFT-like techniques, while addition takes $O(n\log(q)) = O(n\log(n)) = \tilde{O}(n)$ time.

Inspired by Ajtai's construction [14] and Micciancio's one-way function [134], Lyubashevsky and Micciancio introduce a family $\mathcal{H}(\mathcal{R}_q, m)$ of collision-resistant compression functions in ideal lattices [125]. The functions $\mathsf{F} \in \mathcal{H}$ map $\mathcal{R}_q^m \to \mathcal{R}_q$. A random element of the family is indexed with $\hat{\mathbf{a}} \leftarrow_{\$} \mathcal{R}_q^m$ and the associated function $\mathsf{F} = \mathsf{F}_{\hat{\mathbf{a}}}$ maps:

$$\hat{\mathbf{x}} \in \mathcal{R}_q^m \mapsto \mathsf{F}_{\hat{\mathbf{a}}}(\hat{\mathbf{x}}) := \hat{\mathbf{a}} \cdot \hat{\mathbf{x}} := \sum_{i=0}^{m-1} \mathbf{a}_i \mathbf{x}_i \in \mathcal{R}_q.$$

Evidently, F is homomorphic (i.e., linear) over $\mathcal{R}_q^m$, i.e., it satisfies $\mathsf{F}(\mathbf{a}\hat{\mathbf{x}} + \mathbf{b}\hat{\mathbf{y}}) = \mathbf{a}\mathsf{F}(\hat{\mathbf{x}}) + \mathbf{b}\mathsf{F}(\hat{\mathbf{y}}), \forall \mathbf{a}, \mathbf{b} \in \mathcal{R}_q$, and $\forall \hat{\mathbf{x}}, \hat{\mathbf{y}} \in \mathcal{R}_q^m$. When the domain is restricted to a set $\mathcal{D} \subset \mathcal{R}_q^m$ of small-norm polynomials (i.e., the coefficients of the input are restricted to $\{-d, \ldots, d\}$, where $d \in \mathbb{N}$), the function family is collision-resistant [125]. Phrased differently, finding short vectors in the kernel of the family $\mathcal{H}(\mathcal{R}_q, m)$ of module homomorphisms $\mathsf{F}_{\hat{\mathbf{a}}} : \mathcal{R}_q^m \to \mathcal{R}_q$, when the domain is restricted to $\mathcal{D} \subset \mathcal{R}_q$, is computationally hard and is known as the *collision problem* $\mathsf{Col}(\mathcal{H}(\mathcal{R}_q, m), \mathcal{D})$, which we now formally state:

Definition 2.5.12. (Collision Problem $\mathsf{Col}(\mathcal{H}(\mathcal{R}_q, m), \mathcal{D})$, from [125]) We define the $\mathsf{Col}(\mathcal{H}(\mathcal{R}_q, m), \mathcal{D})$ problem in terms of the following game $\mathbf{Col}_{\mathcal{H}(\mathcal{R}_q,m),\mathcal{D}}$:

Game $\mathbf{Col}_{\mathcal{H}(\mathcal{R}_q,m),\mathcal{D}}$:

- **Setup.** On input ring $\mathcal{R}_q$, solution dimension $m \in \mathbb{N}$, and subset $\mathcal{D} \subset \mathcal{R}_q$, game

$\mathbf{Col}_{\mathcal{H}(\mathcal{R}_q,m),\mathcal{D}}$ samples $\mathsf{F} \leftarrow_{\$} \mathcal{H}(\mathcal{R}_q, m)$ and runs adversary A on input $\mathsf{F}, \mathcal{R}_q, \mathcal{D}$, and m.

- **Output Determination.** When A outputs a pair $(\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2)$, the game returns 1 iff: (i) $\mathsf{F}(\hat{\mathbf{z}}_1) = \mathsf{F}(\hat{\mathbf{z}}_2)$, (ii) $\hat{\mathbf{z}}_1 \neq \hat{\mathbf{z}}_2$, and (iii) $(\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2) \in \mathcal{D} \times \mathcal{D}$. Otherwise, it outputs 0.

We define the *advantage* of adversary A in Game $\mathbf{Col}_{\mathcal{H}(\mathcal{R}_q,m),\mathcal{D}}$ as:

$$\mathbf{Adv}^{\mathbf{Col}}_{\mathcal{R}_q,m,\mathcal{D}}(\mathsf{A}) := \Pr_{\mathsf{F}\leftarrow_{\$}\mathcal{H}(\mathcal{R}_q,m)}\left[\mathbf{Col}^{\mathsf{A}}_{\mathcal{H}(\mathcal{R}_q,m),\mathcal{D}} = 1\right].$$

The Collision Problem can trivially be shown to be as hard as $\mathsf{R-SIS}$ [134] in the average case and transitively, at least as hard as $\mathsf{Ring-SVP}$ in the worst case. The next theorem from [125] provides this connection.

Theorem 2.5.5. ($\mathsf{Ring-SVP} \leq \mathsf{Col}$) *Let* $\mathcal{D} = \{\mathbf{f} \in \mathcal{R}_q : \|\mathbf{f}\|_\infty \leq d\}$, *where* $m > \log(q)/\log(2d)$, *and* $q \geq 4dmn\sqrt{n}\log(n)$. *An adversary* A *that solves the* $\mathsf{Col}(\mathcal{H}(\mathcal{R}_q, m), \mathcal{D})$ *problem for* $\mathsf{F} \leftarrow_{\$} \mathcal{H}(\mathcal{R}_q, m)$, *i.e., finds preimages* $\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}} \in \mathcal{D}$ *such that* $\hat{\boldsymbol{x}} \neq \hat{\boldsymbol{y}}$ *and* $\mathsf{F}(\hat{\boldsymbol{x}}) = \mathsf{F}(\hat{\boldsymbol{y}})$, *can then use them to solve* $\mathsf{Ring-SVP}^{\infty}_{\gamma}$ *with approximation factors* $\gamma \geq 16dmn\log^2(n)$ *in the worst case.*

The Ring-LWE problem was introduced in [126] as the algebraic analogue of standard LWE. In its original proposal in [126], it is stated w.r.t. rings of the form $\mathbb{Z}[X]/\langle\Phi_m(X)\rangle$ of integer polynomials modulo a cyclotomic polynomial $\Phi_m(X)$. A concurrent and independent work by Stehlé et al. [174] focuses on the special case of ring-LWE for rings of the form $\mathcal{R} = \mathbb{Z}[X]/\langle X^n + 1\rangle$, where $n = 2^k, k \in \mathbb{N}$ but lacks a hardness proof for the decision form of the problem. Ring-LWE is parameterized by a ring $\mathcal{R}$ of degree n over $\mathbb{Z}$, a modulus $q \in \mathbb{N}$ defining the quotient ring $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$, and an error distribution χ over $\mathcal{R}$. Typically, $\mathcal{R}$ is picked to be a cyclotomic ring, and χ to be some kind of discretized Gaussian in the canonical embedding of $\mathcal{R}$, which we can roughly think of as having an "error rate" $\alpha < 1$ relative to q.

Definition 2.5.13. (Ring-LWE distribution) For a *secret* $\mathbf{s} \in \mathcal{R}_q$, the Ring-LWE distribution $A_{\mathbf{s},\chi}$ over $\mathcal{R}_q \times \mathcal{R}_q$ is sampled by choosing $\mathbf{a} \leftarrow_{\$} \mathcal{R}_q$, choosing error via $\mathbf{e} \leftarrow_{\$} \chi$, and outputting $(\mathbf{a}, \mathbf{b} := \mathbf{a} \cdot \mathbf{s} + \mathbf{e} \pmod q)$.

Informally, the decision version of the Ring-LWE problem in $\mathcal{R}$, denoted $\mathsf{R-LWE}$, states that: let χ be a fixed error distribution over $\mathcal{R}$ that is concentrated on "small" elements, and let $\mathbf{s} \leftarrow_{\$} \mathcal{R}_q$ be the *secret*. Analogously to LWE, the goal is to distinguish arbitrarily many independent "random noisy ring equations" from truly uniform pairs. More specifically, the noisy equations are of the form $(\mathbf{a}, \mathbf{b} \approx \mathbf{a} \cdot \mathbf{s} \pmod{q}) \in \mathcal{R}_q \times \mathcal{R}_q$, where each $\mathbf{a}$ is uniformly random, and each product $\mathbf{a} \cdot \mathbf{s}$ is perturbed by a term drawn independently from the error distribution χ over $\mathcal{R}$. More formally, we have the following definition:

Definition 2.5.14. (Decision version of Ring Learning With Errors ($\mathsf{R-LWE}_{n,q,\chi,m}$)) The decision version of the Ring-LWE problem is defined in terms of the following game $\mathbf{R\text{-}LWE}_{n,q,\chi,m}$:
Game $\mathbf{R\text{-}LWE}_{n,q,\chi,m}$:

- **Setup.** On input lattice dimension $n \in \mathbb{N}$, an integer modulus $q \geq 2$, error distribution χ, and the number of samples $m \in \mathbb{N}$, game $\mathbf{R\text{-}LWE}_{n,q,\chi,m}$ samples a secret $\mathbf{s} \leftarrow_{\$} \mathcal{R}_q$ and $b \leftarrow_{\$} \{0,1\}$. If $b = 0$ it samples m independent Ring-LWE samples via $(\mathbf{a}_i, \mathbf{b}_i) \leftarrow_{\$} A_{\mathbf{s},\chi}, \forall i \in [m]$. Otherwise (i.e., $b = 1$), it samples $(\mathbf{a}_i, \mathbf{b}_i) \leftarrow_{\$} \mathcal{R}_q \times \mathcal{R}_q, \forall i \in [m]$. It then runs adversary A on input $(\mathbf{a}_i, \mathbf{b}_i), i \in [m]$.

- **Output Determination.** When the adversary outputs a bit b^*, the game returns 1 iff $b = b^*$. Otherwise, it returns 0.

We define the *advantage* of adversary A in Game $\mathbf{R\text{-}LWE}_{n,q,\chi,m}$ as:

$$\mathbf{Adv}^{\mathbf{R\text{-}LWE}}_{n,q,\chi,m}(\mathsf{A}) := \Pr\left[\mathbf{R\text{-}LWE}^{\mathsf{A}}_{n,q,\chi,m} = 1\right].$$

The search version of Ring-LWE differs only in that during the Output Determination phase, the adversary A has to output a polynomial $\mathbf{s}^* \in \mathcal{R}_q$ s.t. $\mathbf{s}^* = \mathbf{s}$ (which is in fact unique).

Just like in LWE, the problem would be easy to solve without the error terms. Indeed, if $b = 0$ in Game $\mathbf{R\text{-}LWE}_{n,q,\chi,m}$, we can efficiently find $\mathbf{s}$: given a sample $(\mathbf{a}_i, \mathbf{b}_i)$ where $\mathbf{a}_i \in \mathcal{R}_q$ is invertible (most elements in $\mathcal{R}_q$ are), we have $\mathbf{s} = \mathbf{b}_i \cdot \mathbf{a}_i^{-1}$, whereas if

$b = 1$, there will almost never be a single $\mathbf{s}$ that is consistent with all samples. Ring-LWE also has a *normal form*, in which the secret $\mathbf{s}$ is picked from the error distribution (modulo q) instead of uniformly. It is not hard to show that this form of the problem is at least as hard as the one defined above.

Ring-LWE offers more compactness and efficiency compared to LWE. Indeed, each sample $(\mathbf{a}_i, \mathbf{b}_i)$ yields an n-dimensional pseudorandom ring element $\mathbf{b}_i \in \mathcal{R}_q$, rather than just a single pseudorandom scalar value $b_i \in \mathbb{Z}_q$ which is the case in LWE. Furthermore, like with $\mathsf{R-SIS}$, ring multiplication can be performed in only quasi-linear $\tilde{O}(n)$ time using Fast Fourier Transform. Hence, we can generate these n pseudorandom scalars in just $\tilde{O}(1)$ amortized time each.

As for its hardness, Lyubashevsky et al. [126] show that the Ring-LWE problem is at least as hard on the average as solving SVP on arbitrary ideal lattices in the worst case, using a *quantum* algorithm:

Theorem 2.5.6. *($\mathsf{Ring-SVP} \leq_Q \mathsf{R-LWE}$, from [126]) For any $m = poly(n)$, cyclotomic ring $\mathcal{R}$ of degree n (over $\mathbb{Z}$), and appropriate choices of modulus q and error distribution χ of error rate $\alpha < 1$, solving the $\mathsf{R-LWE}_{n,q,\chi,m}$ problem is at least as hard as quantumly solving the $\mathsf{Ring-SVP}_\gamma$ problem on arbitrary ideal lattices in $\mathcal{R}$, for some $\gamma = poly(n)/\alpha$.*

2.5.7 Rejection Sampling

Rejection sampling is a technique originally introduced by von Neumann [180], and it allows us to draw samples from arbitrarily complex probability distributions. In [122], it was shown for the first time, how this technique can be utilized to construct a canonical identification scheme from lattices. Because this technique is a crucial component to understanding the constructions of Chapters 3, 4, and 5 we give here a brief overview.

Rejection Sampling with Uniform Samples

Let $0 < A \leq B$ be two integer numbers. Now, consider the set of constant random variables $\{X_c := c : c \in \{-A, \ldots, A\}\}$ with respective probability mass functions: $f_{X_c}(x) := 1$, if $x = c$, and 0 otherwise. Furthermore, let Y be an independent, discrete

uniform random variable, taking values in the set $\{-B, \ldots, B\} \supseteq \{-A, \ldots, A\}$ and with probability mass function: $g_Y(y) := \frac{1}{2B+1}$, if $y \in \{-B, \ldots, B\}$, and 0 otherwise.
We now define a new random variable Z_c as the sum of X_c and Y, for any fixed $c \in \{-A, \ldots, A\}$. Obviously, Z_c takes values in the set $\{-(A+B), \ldots, A+B\}$. The distribution h_{Z_c} of Z_c is thus the *convolution* of distributions f_{X_c} and g_Y, and its probability mass function is given from the formula [150]:

$$h_{Z_c}(z) = \sum_{k=-\infty}^{\infty} f_{X_c}(k) g_Y(z-k) = \sum_{k=-A+B}^{A+B} f_{X_c}(k) g_Y(z-k) = \Pr[Y = z - c].$$

Notice that if $|z-c| > B$, then the above probability is zero. On the other hand, if $|z-c| \leq B$, i.e., if $-B+c \leq z \leq B+c$, then the above probability equals $\frac{1}{2B+1}$. Therefore, the probability mass function of h_{Z_c} is $h_{Z_c}(z) := \frac{1}{2B+1}$, if $z \in \{-B+c, \ldots, B+c\}$, and 0 otherwise.
Thus, h_{Z_c} is just a "shifted" version of g_Y by c "places". It is not difficult to notice that Z_c is uniformly distributed over $\{-(B-A), \ldots, B-A\}, \forall c \in \{-A, \ldots, A\}$. Thus, if we compute $Z_c := X_c + Y = c + Y$, and only output the result if it falls within $\{-(B-A), \ldots, B-A\}$ (and resample Y otherwise), then each value $z \in \{-(B-A), \ldots, B-A\}$ will be equally likely to occur. As a result, we can use this technique to "hide" the value of c (In other words, Z_c is distributed independently of c). The following lemma formalizes the above idea and can be viewed as a Rejection Sampling Lemma w.r.t. uniform samples.

Lemma 2.5.7. (Lemma 3.4 in [159, p. 29]) *Let $k \in \mathbb{N}$, $\boldsymbol{a}, \boldsymbol{a}', \boldsymbol{b} \in \mathbb{Z}^k$ with arbitrary $\boldsymbol{a}, \boldsymbol{a}' \in \{\boldsymbol{v} \in \mathbb{Z}^k : \|\boldsymbol{v}\|_\infty \leq A\}$, and a random $\boldsymbol{b} \leftarrow_\$ \{\boldsymbol{v} \in \mathbb{Z}^k : \|\boldsymbol{v}\|_\infty \leq B\}$ for $B > A$. If $\boldsymbol{b}$ is such that $\max\{\|\boldsymbol{a}+\boldsymbol{b}\|_\infty, \|\boldsymbol{a}'+\boldsymbol{b}\|_\infty\} \leq B - A$, we define the random variables $\boldsymbol{c} \leftarrow \boldsymbol{a}+\boldsymbol{b}$ and $\boldsymbol{c}' \leftarrow \boldsymbol{a}'+\boldsymbol{b}$, otherwise, re-sample $\boldsymbol{b}$. Then, $\Delta(\boldsymbol{c}, \boldsymbol{c}') = 0$.*

The following lemma from [159] provides a relation between bounds A and B, and the probability of succeeding to mask a constant vector $\mathbf{a}$ s.t. $\|\mathbf{a}\|_\infty \leq A$ by adding to it a vector $\mathbf{b} \sim \mathsf{Unif}(\{\mathbf{v} : \|\mathbf{v}\|_\infty \leq B\})$.

Lemma 2.5.8. (Lemma 3.1 in [159, p. 28]) *Let $k = \Omega(n), \boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}^k$ with arbitrary*

$\boldsymbol{a} \in \{\boldsymbol{v} \in \mathbb{Z}^k : \|\boldsymbol{v}\|_\infty \leq A\}$ *and random* $\boldsymbol{b} \leftarrow_\$ \{\boldsymbol{v} \in \mathbb{Z}^k : \|\boldsymbol{v}\|_\infty \leq B\}$. *If* $B \geq \phi k A$ *for* $\phi \in \mathbb{N}$, *then* $\Pr_{\boldsymbol{b}}[\|\boldsymbol{a} + \boldsymbol{b}\|_\infty \leq B - A] > e^{-1/\phi} - o(1)$.

Rejection Sampling with Discrete Gaussian Samples

Similarly, rejection sampling can be performed using discrete Gaussian samples. The following two lemmas from [123] are used throughout this work. The first provides a tail-cut bound for Gaussian distributed elements, while the second one concerns rejection sampling:

Lemma 2.5.9. (Lemma 4.4 in [123]) *We have:*

1. $\Pr[|x| > t \cdot \sigma \mid x \leftarrow_\$ D_{\mathbb{Z},\sigma}] \leq 2\exp(-t^2/2), \forall t > 0$.
2. $\Pr[\|\boldsymbol{x}\| > \eta\sigma\sqrt{m} \mid \boldsymbol{x} \leftarrow_\$ D_{\mathbb{Z}^m,\sigma}] \leq \eta^m \exp(\frac{m}{2}(1-\eta^2)), \forall \eta > 1$.

Lemma 2.5.10. (Theorem 4.6 in [123]) *Let* $V \subseteq \mathbb{Z}^m$ *be a set whose elements' norms are bounded by* T, $\sigma = \omega(T\sqrt{\log m})$, *and* $h : V \to \mathbb{R}$ *be a probability distribution. Then there exists a constant* $M = O(1)$ *s.t.* $\forall \boldsymbol{v} \in V : \Pr[D_{\mathbb{Z}^m,\sigma}(\boldsymbol{z}) \leq M \cdot D_{\mathbb{Z}^m,\sigma,\boldsymbol{v}}(\boldsymbol{z}) \mid \boldsymbol{z} \leftarrow_\$ D_{\mathbb{Z}^m,\sigma}] \geq 1 - \epsilon$, *where* $\epsilon = 2^{-\omega(\log m)}$. *Furthermore, the following two algorithms are within a statistical distance of* $\delta = \epsilon/M$.

1. $\boldsymbol{v} \leftarrow_\$ h, \boldsymbol{z} \leftarrow_\$ D_{\mathbb{Z}^m,\sigma,\boldsymbol{v}}$, *output* $(\boldsymbol{z}, \boldsymbol{v})$ *with probability* $\frac{D_{\mathbb{Z}^m,\sigma}(\boldsymbol{z})}{M \cdot D_{\mathbb{Z}^m,\sigma,\boldsymbol{v}}(\boldsymbol{z})}$.
2. $\boldsymbol{v} \leftarrow_\$ h, \boldsymbol{z} \leftarrow_\$ D_{\mathbb{Z}^m,\sigma}$, *output* $(\boldsymbol{z}, \boldsymbol{v})$ *with probability* $1/M$.

Moreover, the probability that the first algorithm produces an output is at least $(1-\epsilon)/M$. *If* $\sigma = \alpha T$ *for any* $\alpha > 0$, *then* $M = \exp(\frac{12}{\alpha} + \frac{1}{2\alpha^2})$ *with* $\epsilon = 2^{-100}$.

Algorithm 1 shows how rejection sampling is performed w.r.t. discrete Gaussian samples.

2.5.8 Lattice Trapdoors

Informally, a trapdoor function is a function that is easy to evaluate but hard to invert on its own, but which can be generated together with some additional "trapdoor" information that makes inversion easy. Ajtai showed that it is possible to construct

Algorithm 1 Rejection_Sample($\hat{\mathbf{z}}, \hat{\mathbf{c}}, \phi, T; \rho$)

$\sigma := \phi T$; $M(\phi) := e^{12/\phi + 1/(2\phi^2)}$
if $\rho = \perp$ **then**
    $u \leftarrow_{\$} [0, 1)$; ▷ Sample uniformly
else
    $u \leftarrow_{\rho} [0, 1)$; ▷ Sample with fixed randomness ρ
if $(u \leq \frac{1}{M(\phi)} e^{(\frac{-2\hat{\mathbf{z}}\cdot\hat{\mathbf{c}} + \|\hat{\mathbf{c}}\|^2}{2\sigma^2})})$ **then**
    return 1; ▷ Accept sample
return 0; ▷ Reject sample

certain types of trapdoor functions from the hardness of lattice problems [14]. Further advances [52, 86, 23] proposed more elegant and efficient constructions of "strong trapdoors" from the SIS/LWE assumptions.

Roughly speaking, the idea was to generate a lattice basis $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ of the SIS lattice $\Lambda_q^{\perp}(\mathbf{A}) := \{\mathbf{v} \in \mathbb{Z}^m \mid \mathbf{Av} = \mathbf{0} \pmod q\}$, along with an associated lattice trapdoor that enables efficiently sampling *short* Gaussian vectors $\mathbf{x} \in \mathbb{Z}^m$ s.t. $\mathsf{F}_{\mathbf{A}}(\mathbf{x}) = \mathbf{u} \pmod q$, where $\mathsf{F}_{\mathbf{A}}(\mathbf{x}) := \mathbf{Ax}$ denotes the SIS function, and $\mathbf{u} \in \mathbb{Z}_q^n$. This sampling process is known as *preimage sampling*. For the purposes of this thesis, we focus on a particular category of lattice trapdoors called *"gadget"-based trapdoors* [133].

Gadget-based trapdoor

The main idea behind the construction in [133] is to start with a fixed, public lattice, defined by a "gadget" matrix $\mathbf{G}$. This matrix should allow for a fast, parallel and offline way of solving equation $\mathbf{Gx} = \mathbf{u} \pmod q$ for short vectors $\mathbf{x} \in \mathbb{Z}^m$. The trapdoored lattice basis $\mathbf{A}$ is obtained by applying a unimodular transformation which involves the gadget matrix $\mathbf{G}$, along with other randomly sampled matrices. The transformation itself serves as the trapdoor for $\mathbf{A}$. We now describe how to construct an (almost) uniform random matrix $\mathbf{A}$ together with an associated trapdoor $\mathbf{T_A}$, having a desired tag $\mathbf{H}$.

Trapdoor Generation. On input a modulus q, gaussian parameter σ, an optional $\mathbf{A'} \in \mathbb{Z}_q^{n \times \bar{m}}$ and an optional (but invertible) *tag* $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$, the trapdoor generation algorithm outputs a lattice basis $\mathbf{A}$, along with a short trapdoor $\mathbf{T_A}$. Algorithm TrapGen

is summarized in Algorithm 2.

Algorithm 2 $\mathsf{TrapGen}(q, \sigma, \mathbf{A}' \in \mathbb{Z}_q^{n\times\bar{m}}, \mathbf{H} \in \mathbb{Z}_q^{n\times n})$

$l := \lceil \log(q) \rceil$
$\mathbf{g} := (1, 2, 2^2, \ldots, 2^{l-1}) \in \mathbb{Z}_q^l$ ▷ Gadget vector
$\mathbf{G} := \mathbf{I}_n \otimes \mathbf{g}^T \in \mathbb{Z}_q^{n\times nl}$ ▷ Gadget matrix
if $\mathbf{A}' = \perp$ **then**
    $\mathbf{A}' \leftarrow_{\$} \mathbb{Z}_q^{n\times\bar{m}}$ ▷ Sample uniformly
if $\mathbf{H} = \perp \vee \det(\mathbf{H}) = 0$ **then**
    $\mathbf{H} := \mathbf{I}_n$
$\mathbf{T} \leftarrow_{\$} D_{\mathbb{Z},\sigma}^{\bar{m}\times nl}$ ▷ Sample a short Gaussian matrix
$m := \bar{m} + nl$
$\mathbf{A} := (\mathbf{A}' \| \mathbf{HG} - \mathbf{AT}) \in \mathbb{Z}_q^{m\times n}$ ▷ Form the trapdoored lattice basis
$\mathbf{T_A} := \begin{pmatrix} \mathbf{T} \\ \mathbf{I}_{nl} \end{pmatrix}$ ▷ Trapdoor for $\mathbf{A}$ with tag $\mathbf{H}$
return $(\mathbf{A}, \mathbf{T_A})$

Preimage Sampling. On input basis $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$, trapdoor $\mathbf{T_A} \in \mathbb{Z}_q^{\bar{m}\times nl}$, tag $\mathbf{H} \in \mathbb{Z}_q^{n\times n}$ and syndrome $\mathbf{u} \in \mathbb{Z}_q^n$, algorithm PreSample outputs a (short) vector $\mathbf{x} \in \mathbb{Z}^m$ s.t. $\mathbf{Ax} = \mathbf{u} \pmod q$. Algorithm PreSample is summarized in Algorithm 3.

Algorithm 3 $\mathsf{PreSample}(\mathbf{A} \in \mathbb{Z}_q^{m\times n}, \mathbf{T_A} \in \mathbb{Z}_q^{\bar{m}\times nl}, \mathbf{H} \in \mathbb{Z}_q^{n\times n}, \mathbf{u} \in \mathbb{Z}_q^n)$

Generate a perturbation vector $\mathbf{p} \in \mathbb{Z}^m$ with covariance $\Sigma_2 = \sigma^2\mathbf{I} - \mathbf{T_A}\mathbf{T_A}^T > 0$
$\mathbf{b} := \mathbf{H}^{-1} \cdot (\mathbf{u} - \mathbf{Ap})$
For $i \in [n]$:
    Find $\mathbf{w}_i$ s.t. $\mathbf{g}^T\mathbf{w}_i = b_i \pmod q$
$\mathbf{w} := (\mathbf{w}_1, \ldots, \mathbf{w}_n)$
$\mathbf{x} := \mathbf{p} + \mathbf{T_A}\mathbf{w}$
return $\mathbf{x}$

2.6 Cryptographic Primitives and Tools

In this section, we give an overview of the cryptographic primitives and tools that will be used throughout the following chapters.

2.6.1 Hash Functions and the Random Oracle Model

Many cryptographic constructions in the literature employ cryptographic hash functions as a component. A *hash function* is any function that can be used to map data

of arbitrary size to fixed-size values (called *hash values*). More formally:

Definition 2.6.1. (Hash Function) A function $\mathsf{H} : \{0,1\}^* \rightarrow \{0,1\}^k$, mapping bit strings of arbitrary length to bit strings of a fixed length $k \in \mathbb{N}_0$, is called a *hash function*. Additionally, H is called a *cryptographic hash function*, if $\mathsf{H}(x)$ can be efficiently computed for any string $x \in \{0,1\}^*$, and it has at least one of the following properties:

1. One-wayness (or preimage resistance): given a k-bit string $y \in \{0,1\}^k$, it is hard to find a bit string $x \in \{0,1\}^*$ s.t.: $\mathsf{H}(x) = y$.

2. Weak collision-resistance (or 2nd-preimage resistance): given a bit string $x_1 \in \{0,1\}^*$, it is hard to find a bit string $x_2 \in \{0,1\}^*$ s.t.: $x_2 \neq x_1$ and $\mathsf{H}(x_1) = \mathsf{H}(x_2)$.

3. Collision-resistance (or strong collision-resistance): it is hard to find a pair of bit strings $(x_1, x_2) \in \{0,1\}^* \times \{0,1\}^*$ with $x_1 \neq x_2$ s.t.: $\mathsf{H}(x_1) = \mathsf{H}(x_2)$.

Notice that collision-resistance implies weak collision-resistance, but does not necessarily imply one-wayness. A good cryptographic hash function typically possesses *all* of these properties.

A *random oracle* is a mathematical function chosen uniformly at random, i.e., a function mapping each possible query to a *fixed* random response from its range. In the *random oracle model* (ROM) [31], we assume that a publicly known cryptographic hash function (ideally) behaves like a truly random function and can thus be used as a "drop-in replacement" for the random oracle. As such, a provably-secure construction in the ROM can only be broken if the attacker exploits specific properties of the concrete hash function used (e.g.: SHA-512). This *heuristic* approach was first advocated by [31] and since then has been widely used for designing a plurality of efficient cryptographic primitives. While the confidence placed on random oracles in the context of provable security has spurred a lot of debate in the cryptographic community, it is generally accepted that a proof in the ROM is much better than no proof at all.

It should finally be noted that Boneh et al [38] introduced the notion of *Quantum-accessible Random Oracle Model* (QROM). In this model, a quantum attacker can

access any "offline primitives" (e.g.: the concrete hash function replacing the random oracle) in quantum superposition. In other words, the attacker can, with a single quantum query to all inputs in superposition, learn a superposition of all possible random oracle values. Thus, reductions relying on techniques like adaptive re-programming of the random oracle do not work in this model. The QROM generalizes the ROM, and a proof of security in the QROM is thus stronger than one in the plain ROM. However, as recently shown in [71], security in the QROM *cannot* imply security in the standard model (i.e., without the use of any random oracles). For this dissertation, we limit our study to the classical ROM.

2.6.2 The General Forking Lemma

The Generalized Forking Lemma from [32] is a probabilistic tool for proving security of cryptographic constructions in the ROM. Informally, it states that if an algorithm A outputs a pair of values (I, σ) with $I > 0$ with noticeable probability *acc*, then the forking algorithm F_{A} defined below will, with noticeable probability return $(1, \sigma, \sigma')$ based on two executions of A, sharing an identical prefix up to the I-th query to H. In other words, the probability of getting two related runs with the same value of I, and a common prefix of length $I - 1$ is not too small.

Algorithm 4 $F_{\mathsf{A}}(x)$

Pick coins ρ for A at random;
$h_1, \ldots, h_q \leftarrow_{\$} H$;
$(I, \sigma) \leftarrow \mathsf{A}(x; h_1, \ldots, h_q; \rho)$;
if $(I = 0)$ **then**
    return $(0, \varepsilon, \varepsilon)$;
$h'_I, \ldots, h'_q \leftarrow_{\$} H$;
$(I', \sigma') \leftarrow \mathsf{A}(x; h_1, \ldots, h_{I-1}, h'_I, \ldots, h'_q; \rho)$;
if $(I = I' \wedge h_I \neq h'_I)$ **then**
    return $(1, \sigma, \sigma')$;
else
    return $(0, \varepsilon, \varepsilon)$;

Lemma 2.6.1. *(Lemma 1 in [32]) Fix an integer $q \geq 1$ and a set H of size $h \geq 2$. Let A be a randomized algorithm that on input $x, h_1, \ldots, h_q$ returns a pair, the first*

element of which is an integer in the range $0, \ldots, q$ *and the second element of which we refer to as a side output. Let* IG *be a randomized algorithm that we call the input generator. The accepting probability of* A*, denoted acc, is defined as the probability that* $J \geq 1$ *in the experiment* $x \leftarrow_{\$} \mathsf{IG}; h_1, \ldots, h_q \leftarrow_{\$} H; (J, \sigma) \leftarrow_{\$} \mathsf{A}\left(x; h_1, \ldots, h_q\right)$*. The forking algorithm* F_{A} *associated to* A *is the randomized algorithm that takes input* x *and proceeds as shown in Algorithm 4. Let* $frk := \Pr\left[b = 1 : x \leftarrow_{\$} \mathsf{IG}; (b, \sigma, \sigma') \leftarrow F_{\mathsf{A}}(x)\right]$*. Then,* $frk \geq acc\left(\frac{acc}{q} - \frac{1}{h}\right)$*.*

2.6.3 Merkle Trees

A *Merkle tree* (or *binary hash tree*) is a data structure where each internal (i.e., non-leaf) node is computed as the hash of its child nodes [129]. Merkle trees help maintain data integrity, and select applications of Merkle trees include: file-systems (e.g.: ZFS and IPFS), peer-to-peer networks (e.g.: Bitcoin [172] and Ethereum [24]), distributed version control systems (e.g.: Git and Mercurial), as well as in the branch of *hash-based cryptography* [106].

Let $\mathsf{G} : \{0,1\}^* \rightarrow \{0,1\}^{2\lambda}$ be a collision-resistant hash function. Given a list of values $v_0, \ldots, v_{l-1}$, a Merkle tree is constructed as follows: the leaves of the tree are simply the hashes of v_i under G. Every inner node y is constructed via $y := \mathsf{G}(\text{left child}, \text{right child})$. Using this construction, the values of all inner nodes are fully determined by the leaf nodes. The entire (ordered) list of values is thus represented by a single hash. For proving the inclusion of a leaf node in a Merkle tree, we use an *authentication path*, which consists of the siblings of the nodes on the path from the bottom to the root of the tree. If v is indeed included in the tree and auth is an authentication path for v, then the verifier should obtain the tree's root node root by calculating successive parent nodes. The algorithms HashTree, BuildAuth, and RootCalc associated for a collision-resistant hash function G are described in Table 2.2. Algorithm HashTree takes as input a list of values $v_0, \ldots, v_{l-1}$ and returns a sequence of nodes spanning the tree, along with the root of the tree. Algorithm BuildAuth takes as input an index, as well as a tree and outputs an authentication path auth. Finally, algorithm RootCalc takes as input a node and an

authentication path auth and returns the root of a hash tree. Note that for all nodes $(v_0, \ldots, v_{l-1})$ and for all indices $i \in \{0, \ldots, l-1\}$, we have $\mathsf{RootCalc}(v_i, \mathsf{auth}) = \mathsf{root}$, where $(\mathsf{root}, \mathsf{tree}) \leftarrow \mathsf{HashTree}(v_0, \ldots, v_{l-1})$ and $\mathsf{auth} \leftarrow \mathsf{BuildAuth}(I, \mathsf{tree})$.

Table 2.2: Description of algorithms HashTree, BuildAuth, and RootCalc associated to collision-resistant hash function G.

Algorithm $\mathsf{HashTree}(v_0, \ldots, v_{l-1})$:	Algorithm $\mathsf{BuildAuth}(I, \mathsf{tree})$:	Algorithm $\mathsf{RootCalc}(v, \mathsf{auth})$:
$h \leftarrow \lceil \log(l) \rceil$	$(t_0^{(1)}, \ldots, t_{2^{h-1}-1}^{(h)}) \leftarrow \mathsf{tree}$	$(I, a_0, \ldots, a_{h-1}) \leftarrow \mathsf{auth}$
for $j \in \{0, \ldots, l-1\}$:	for $i \in \{0, \ldots, h-1\}$:	$b_0 \leftarrow \mathsf{G}(v)$
$\quad t_j^{(0)} \leftarrow \mathsf{G}(v_j)$	$\quad s \leftarrow \lfloor I/2^i \rfloor$	for $i \in [h]$:
for $i \in [h]$:	$\quad b \leftarrow s \pmod 2$	$\quad s \leftarrow \lfloor I/2^{i-1} \rfloor$
$\quad$ for $j \in \{0, \ldots, 2^{h-i}-1\}$:	$\quad$ if $b = 1$:	$\quad b \leftarrow s \pmod 2$
$\quad\quad t_j^{(i)} \leftarrow \mathsf{G}(t_{2j}^{(i-1)}, t_{2j+1}^{(i-1)})$	$\quad\quad a_i \leftarrow t_{s-1}$	$\quad$ if $b = 1$:
$\mathsf{root} \leftarrow t_0^{(h)}$	$\quad$ else:	$\quad\quad b_i \leftarrow \mathsf{G}(a_{i-1}, b_{i-1})$
$\mathsf{tree} \leftarrow (t_0^{(1)}, \ldots, t_{2^{h-1}-1}^{(h)})$	$\quad\quad a_i \leftarrow t_{s+1}$	$\quad$ else:
return $(\mathsf{root}, \mathsf{tree})$	$\mathsf{auth} \leftarrow (I, a_0, \ldots, a_{h-1})$	$\quad\quad b_i \leftarrow \mathsf{G}(b_{i-1}, a_{i-1})$
	return auth	return root := b_h

2.6.4 Commitment Schemes

Commitment schemes are fundamental cryptographic primitives that lie at the heart of many modern cryptographic protocols. Informally, they allow a party to commit to a certain value (or statement), while keeping the actual value hidden from all others, with the ability to reveal that value at a later point.

Definition 2.6.2. (Commitment Schemes) Let $\mathsf{com} : \{0,1\}^* \times \{0,1\}^\lambda \to \{0,1\}^*$ be a deterministic polynomial time algorithm, where λ is a security parameter. A (non-interactive) commitment scheme consists of two protocols between two parties which are typically named "sender" and "receiver":

Commit phase. The sender commits to a value $\mu \in \{0,1\}^*$ by computing $C := \mathsf{com}(\mu; r)$, where randomness $r \leftarrow_\$ \{0,1\}^\lambda$, and sends C to the receiver.

Reveal phase. The sender "opens" commitment $C := \mathsf{com}(\mu; r)$ by revealing the "de-commitment" parameter r to the receiver. The receiver can then verify that $C = \mathsf{com}(\mu; r)$.

Commitment schemes need to satisfy two properties: hiding and binding. The *hiding property* requires that C does not reveal any information about the committed message μ, whereas the *binding property* requires that no algorithm can substitute the committed message μ with some other message $\mu' \neq \mu$, in such a way that $C = \mathsf{com}(\mu'; r) = \mathsf{com}(\mu; r')$, for some randomness $r' \in \{0,1\}^{\lambda}$. A commitment scheme is (t, θ)-*hiding* (resp. *binding*) if no algorithm exists running in time at most t, that can break the hiding (resp. binding) property with a probability of at least θ. Both properties can be satisfied computationally or unconditionally. It has been shown that a commitment scheme cannot be unconditionally hiding and unconditionally binding at the same time [62].

Lattice-based cryptographic hash functions such as [25] can be used as a message authentication code to construct purely lattice-based commitment schemes.

2.6.5 Zero-Knowledge Proofs of Knowledge

Zero-Knowledge Proofs of Knowledge (ZKPoK) are a method by which one party (called *prover*) can prove to another party (called *verifier*) that a given statement is true, without revealing *any* information apart from the fact that the statement is indeed true. A zero-knowledge proof needs to satisfy the following properties, which we now state informally:

- **Correcteness:** if the statement is true, an honest verifier is always convinced of this fact by an honest prover.
- **Soundness:** if the statement is false, no cheating prover can convince an honest verifier that it is true, except with some very small probability.
- **Zero-knowledge:** if the statement is true, no verifier is able to learn anything other than the fact that the statement is true.

If no interaction is required between the prover and the verifier in order to prove a statement, then the ZKPoK is said to be a *Non-Interactive Zero-Knowledge (NIZK)* proof of knowledge.

2.6.6 Homomorphic Encryption

A homomorphic encryption (HE) scheme is an encryption scheme that permits computations to be performed on encrypted data (i.e., decrypting is not required before performing the computations).

Definition 2.6.3. (Homomorpic Encryption Scheme) A homomorphic encryption scheme is a tuple of probabilistic polynomial time (PPT) algorithms $\mathsf{HE} = (\mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Eval}, \mathsf{HE.Dec})$ defined as follows:

- $\mathsf{HE.KeyGen}(1^\lambda, 1^\mathsf{d})$: On input security parameter λ and a depth bound d, the algorithm outputs a pair of keys $(\mathsf{sk}, \mathsf{pk})$.
- $\mathsf{HE.Enc}(\mathsf{pk}, \mu)$: On input a public key pk and a plaintext message $\mu \in \{0,1\}$, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{HE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}k)$: On input a public key pk, a circuit $C : \{0,1\}^k \to \{0,1\}$ of depth at most d, and a tuple of ciphertexts $(\mathsf{ct}_1, \ldots, \mathsf{ct}_k)$, the evaluation algorithm outputs an evaluated ciphertext $\hat{\mathsf{ct}}$.
- $\mathsf{HE.Dec}(\mathsf{pk}, \mathsf{sk}, \hat{\mathsf{ct}})$: On input a public key pk, a secret key sk and a ciphertext $\hat{\mathsf{ct}}$, the decryption algorithm outputs a message $\hat{\mu} \in \{0,1\}$ or $\perp$ (in case of failure).

Definition 2.6.4. (Correctness) An HE scheme is correct if for all λ, depth bound d, circuit $C : \{0,1\}^k \to \{0,1\}$ of depth at most d, and $\mu_i \in \{0,1\}$ for $i \in [k]$, the following holds: for $(\mathsf{sk}, \mathsf{pk}) \leftarrow_\$ \mathsf{HE.KeyGen}(1^\lambda, 1^d)$, $\mathsf{ct}_i \leftarrow \mathsf{HE.Enc}(\mathsf{pk}, \mu_i)$ for $i \in [k]$, $\hat{\mathsf{ct}} \leftarrow \mathsf{HE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_k)$, we have $\Pr[\mathsf{HE.Dec}(\mathsf{pk}, \mathsf{sk}, \hat{\mathsf{ct}}) = C(\mu_1, \ldots, \mu_k)] = 1 - \lambda^{-\omega(1)}$.

Definition 2.6.5. (Security) We say that an HE scheme is secure if for all λ and depth bound d, the following holds: for any adversary A with run-time $2^{o(\lambda)}$, the following experiment outputs 1 with probability $2^{-\Omega(\lambda)}$:

1. On input the security parameter λ and a depth bound d, the challenger runs $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{HE.KeyGen}(1^\lambda, 1^d)$ and $\mathsf{ct} \leftarrow \mathsf{HE.Enc}(\mathsf{pk}, b)$ for $b \leftarrow_\$ \{0,1\}$. It sends $(\mathsf{sk}, \mathsf{pk})$ to A.

2. A outputs a guess b'. The experiment outputs 1 iff $b' = b$. Otherwise, it outputs 0.

Definition 2.6.6. (Circuit Privacy) An homomorphic encryption scheme HE is semi-honest circuit private if for $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{HE.KeyGen}(1^\lambda, 1^d)$, any circuit $C : \{0,1\}^k \rightarrow \{0,1\}$ of depth at most d, $\mu_i \in \{0,1\}$ for $i \in [k]$, and $\mathsf{ct}_i \leftarrow \mathsf{HE.Enc}(\mathsf{pk}, \mu_i)$ for $i \in [k]$, the statistical distance between the distributions $(\mathsf{HE.Eval}(\mathsf{pk}, C, \{\mathsf{ct}_i\}_{i\leq k}), \{\mathsf{ct}_i\}_{i\leq k}, \mathsf{pk}, \mathsf{sk})$ and $(\mathsf{HE.Eval}(\mathsf{pk}, C^0, \{\mathsf{ct}'_i\}_{i\leq k}), \{\mathsf{ct}_i\}_{i\leq k}, \mathsf{pk}, \mathsf{sk})$ is $2^{-\Omega(\lambda)}$, where $\mathsf{ct}'_1 = \mathsf{HE.Enc}(\mathsf{pk}, C(\mu_1, \ldots, \mu_k))$, $\mathsf{ct}'_i = \mathsf{HE.Enc}(\mathsf{pk}, 0)$ for $i \in \{2, \ldots, k\}$ and $C^0 : \{0,1\}^k \rightarrow \{0,1\}$ is the circuit of depth d that simply outputs its first input (and ignores the rest).

If the above hold even for keys $(\mathsf{sk}, \mathsf{pk})$ and ciphertexts ct_i for $i \in [k]$ that were not generated honestly, then we say that the HE scheme is *maliciously circuit private*.

2.6.7 Digital Signature Schemes

One of the most ubiquitous cryptographic primitives is that of digital signature schemes (DSS). In this section, we recall the definition of digital signature schemes and their security model.

Syntax and Security Model

Definition 2.6.7. (Digital Signature Scheme) A *digital signature scheme* consists of a triplet of (possibly probabilistic) polynomial-time algorithms $(\mathsf{KG}, \mathsf{Sign}, \mathsf{Ver})$.

The algorithms $\mathsf{KG}, \mathsf{Sign}, \mathsf{Ver}$ are respectively called *key-generation*, *signing*, and *verification* algorithms. A digital signature scheme is said to be *correct* iff:

$$\Pr\left[\mathsf{Ver}(\mathsf{vk}, msg, \mathsf{Sign}(\mathsf{sk}, msg)) = 1 \mid (\mathsf{sk}, \mathsf{vk}) \leftarrow_\$ \mathsf{KG}(1^\lambda)\right] = 1, \forall msg \in \{0,1\}^*$$

The keys sk and vk are called *signing* and *verification key*, respectively.

Informally, a digital signature scheme is *secure* if no forger, after seeing signatures of messages of his choosing, can sign a message whose signature he has not already seen, with noticeable probability [93]. More concretely, we have:

Definition 2.6.8. (Unforgeability) A digital signature scheme $\mathsf{DSS} = (\mathsf{KG}, \mathsf{Sign}, \mathsf{Ver})$ is called *existentially unforgeable* if for every PPT forger U^*, the probability that after seeing vk and $\{(msg_1, \sigma_1), \ldots, (msg_k, \sigma_k)\}$ for any k messages msg_i of its choosing (where $k = poly(\lambda)$ and $\sigma_i := \mathsf{Sign}(\mathsf{sk}, msg_i), \forall i \in [k]$), U^* can produce an additional message-signature pair (msg^*, σ^*) s.t.: $\mathsf{DSS.Ver}(\mathsf{vk}, msg^*, \sigma^*) = 1$, and $msg^* \notin \{msg_1, \ldots, msg_k\}$ is $negl(\lambda)$. The probability is taken over the randomnesses of $\mathsf{KG}, \mathsf{Sign}, \mathsf{Ver}$, and U^* and the security game is summarized in Game $\mathbf{EUF}_{\mathsf{DSS}}$ below:

Game $\mathbf{EUF}_{\mathsf{DSS}}$:

- **Setup.** On input the security parameter 1^λ, the game generates a pair of keys via $(\mathsf{sk}, \mathsf{vk}) \leftarrow_\$ \mathsf{DSS.KG}(1^\lambda)$ and initializes a counter $k := 0$ as well as a list $L := \{\}$. It then runs adversary U^* on input vk.

- **Online Phase.** U^* is given access to the following oracle:

  1. **Oracle** Sign: On input a message msg, the oracle outputs (msg, σ), where σ is obtained by invoking DSS's signing algorithm $\sigma \leftarrow_\$ \mathsf{DSS.Sign}(\mathsf{sk}, msg)$. If $\sigma \neq \perp$, it appends msg to the list of queried messages via $L := L \cup \{msg\}$ and increments the counter via $k := k + 1$.

- **Output Determination.** When the adversary outputs a message-signature pair (msg^*, σ^*), the game returns 1 iff a) $msg^* \notin \{msg_1, \ldots, msg_k\}$, and b) $\mathsf{DSS.Ver}(\mathsf{vk}, msg^*, \sigma^*) = 1$. Otherwise, it returns 0.

We define the *advantage* of adversary U^* in Game $\mathbf{EUF}_{\mathsf{DSS}}$ as:

$$\mathbf{Adv}^{\mathbf{EUF}}_{\mathsf{DSS}}(\mathsf{U}^*) := \Pr\left[\mathbf{EUF}^{\mathsf{U}^*}_{\mathsf{DSS}}(1^\lambda) = 1\right].$$

In the above security definition, the forger should not be able to produce a signature of a new message. A stronger security notion called *strong unforgeability* requires that in addition to the above, a forger should not even be able to come up with a different signature for a message whose signature he has already seen. Replacing condition a) in the output determination phase in game $\mathbf{EUF}_{\mathsf{DSS}}$ with "$(msg^*, \sigma^*) \notin \{(msg_1, \sigma_1) \ldots, (msg_k, \sigma_k)\}$ ", yields this stronger definition.

Chapter 3

An Overview of Lattice-Based Blind Signature Schemes and their Feasibility

3.1 Introduction

We live in the highly digitalized Age of Information. The rise of the Internet heralded a major and rapid shift from the traditional industry towards an economy based upon information technologies. Indeed, all of us employ some form of electronic services in our daily lives (e-mail services, online banking, e-learning through various software, etc). Not surprisingly, there is great effort being put in the digital transformation of the global economy as evidenced by the high precedence set for it by administrative bodies such as the European Commission [1], the wide spread of Information and Communication Technologies across all business sectors as a means of enhancing productivity, and by the dramatic increase of cryptocurrencies in only a few years [2]. This Internet Economy [51] gives rise to a vast new array of opportunities for businesses, it boosts the development of trustworthy technology, enables a vibrant and sustainable economy, and fosters an open and democratic society. A major component of this new economy is the so-called *e-business infrastructure* and includes hardware, software, telecommunication networks, support devices, and human capital used in

electronic business and commerce [182]. However, digital networks are susceptible to hacking, which creates the need for sophisticated cryptographic systems in order to secure transactions over insecure digital networks. On the other hand, it is well-understood that the amount of disclosed personal user information during any form of transaction should be kept at a strict minimum. Hence, it is paramount for every publicly used digital system to strike the right balance between digital security and digital anonymity.

Digital signatures are a cryptographic primitive that enable one party, termed the *signer*, to issue signatures on messages or documents, validating their authenticity to some other party, termed the *user*. Such schemes primarily safeguard against attempts of impersonation, repudiation, and message tampering. However, forfeiting the confidentiality of the message-to-be-signed becomes problematic for privacy-oriented applications in which the message needs to remain unintelligible to the signer.

Blind Signature schemes (BS) are a variant of digital signatures, pioneered by D. Chaum in 1982 [53] in an effort to create an electronic version of conventional cash (e-cash). Since their original conception, they have found a myriad of applications in electronic voting [114], anonymous authentication via digital credentials [26] (like in Microsoft's U-Prove technology [146]), wireless sensor networks (WSN) [186], blindly signed contracts to ensure anonymity and fairness in cryptocurrencies [100], to name a few. The key idea in blind signature schemes is to separate the party owning the message from the party issuing signatures. This is done by allowing the owner of the message to interact through a cryptographic protocol with the signer in order to obtain a signature on it but in a way that does not expose it to the signer's view. The resulting signature can still be verified against the signer's public key, just like with typical digital signatures. However, nobody – not even the signer – can link a message-signature pair to a signing transcript.

There have been numerous BSS proposals in the literature, including early work by [53, 54] and later advances [50, 137, 163, 152, 73, 97]. However, all are based on number-theoretic assumptions, such as the hardness of factoring large integers, computing discrete logarithms, or the quadratic residuosity problem. Unfortunately,

the security assumptions underlying these schemes are known to be vulnerable to quantum attacks thanks to Shor's algorithm [169]. As a result, they are ill-suited candidates for the post-quantum era. There also exist BSS from general complexity assumptions [70, 75, 104] but their efficiency under standard assumptions poses an exceptionally difficult task.

By now, lattice-based cryptography is one of the most versatile approaches for constructing provably secure, efficient, and highly parallelizable cryptographic primitives that can withstand attacks even by quantum computers. This is apparent from the number of lattice-based candidates in the third round of NIST's post-quantum cryptography standardization process [3]. In addition, lattice-based cryptography offers the unique feature of allowing for worst-case to average-case reductions [14, 134, 156, 147, 115], which is *needed* for cryptographic applications. This not only allows us to harness the hardness of worst-case lattice problems, but it also greatly simplifies key selection.

3.1.1 Organization

In Section 3.2 we recall a few preliminary notions that are necessary for presenting the rest of this chapter. The notion of blind signature schemes and their security model which will be relevant to later chapters is also recalled. Section 3.3 provides an overview of the most prominent literary attempts towards designing blind signature schemes from lattice assumptions. Section 3.4 provides an overview of the current state-of-the-art blind signature constructions in the lattice setting, along with a discussion on issues pertaining to their feasibility. Section 3.5 addresses the relationship between all presented proposals and known impossibility results from the literature. Finally, Section 3.6 provides a comparison of concrete sizes between all presented schemes, as well as other post-quantum proposals for blind signature schemes.

This chapter is a partial reprint of [142]. The dissertation author was the primary investigator and author of this paper.

3.2 Preliminaries

3.2.1 Signed Permutations

The notion of *signed permutation monomials* was introduced in [21].

Definition 3.2.1. (Adapted from [21]) For the ring $\mathcal{R}_q$, we define the set of signed permutation monomials as $\mathbb{S} := \{(-1)^s \cdot X^i \mid s \in \{0,1\} \text{ and } 0 \leq i \leq n-1\}$.

In [21], the authors prove that $\mathbb{S}$ forms a group under multiplication in $\mathcal{R}_q$. Furthermore, if we define the set $\mathbb{S}_\kappa^n := \{\mathbf{v} \in \mathcal{R}_q : \|\mathbf{v}\|_1 = \kappa \text{ and } \|\mathbf{v}\|_\infty \leq 1\}$, then any polynomial $\mathbf{c} \in \mathbb{S}_\kappa^n$ can be partitioned into a set of partitioning monomials $\{c_1, \ldots, c_\kappa\}$ s.t. $\mathbf{c} = \sum_{j=1}^{\kappa} c_j$ where c_j contains exactly the j-th non-zero entry of $\mathbf{c}$ at the exact same position. Additionally, the following lemma shows that signed permutations can be used to (individually) "mask" each partitioning monomial:

Lemma 3.2.1. *(Adapted from [21]) Let $\mathbf{c} \in \mathbb{S}_\kappa^n$ be a polynomial and $\{c_1, \ldots, c_\kappa\}$ be its partitioning set. Furthermore, let $p_1, \ldots, p_\kappa$ be random signed permutations in $\mathbb{S}$ and $c_j^* = p_j^{-1} c_j, \forall j = 1, \ldots, \kappa$. Then, for $c_j, c_j^* \in \mathbb{S}$, the following holds:*

$$\Pr_{p_j \leftarrow_\$ \mathbb{S}}[(c_1^*, \ldots, c_\kappa^*) = (p_1^{-1}c_1, \ldots, p_\kappa^{-1}c_\kappa) \mid \mathbf{c}] =$$

$$\Pr_{p_j \leftarrow_\$ \mathbb{S}, \mathbf{c} \leftarrow_\$ \mathbb{S}_\kappa^n}[(c_1^*, \ldots, c_\kappa^*) = (p_1^{-1}c_1, \ldots, p_\kappa^{-1}c_\kappa)] = \frac{1}{(2n)^\kappa}$$

3.2.2 Blind Signature Schemes

In this section, we recall the syntax and security model of blind signature schemes from [53, 104].

Definition 3.2.2. (Blind Signature Schemes) A BSS is a tuple of algorithms $(\mathsf{PG}, \mathsf{KG}, \mathsf{Sign} = \langle \mathsf{S}, \mathsf{U} \rangle, \mathsf{Ver})$, where Sign is an interactive protocol executed between a signer S and a user U. Their specification is the following:

- $\mathsf{PG}(1^\lambda)$ is a PPT algorithm. On input the main security parameter λ, it outputs the scheme's public parameters par.

- $\mathsf{KG}(\mathsf{par})$ is a PPT algorithm. On input parameters par, it outputs a key pair $(\mathsf{sk}, \mathsf{pk})$ where sk is the secret signing key and pk is the public verification key.

- $\mathsf{Sign}(\mathsf{sk}, msg) = \langle \mathsf{S}(\mathsf{sk}), \mathsf{U}(\mathsf{pk}, msg) \rangle$ is an interactive and PPT two-party protocol between a signer S and a user U (who requests the signature) with a public key pk as common input. The private input of S is a private key sk, and the private input of U is a message msg. The signer's private output is a view V consisting of all messages exchanged between the two parties, and the user's private output is a signature σ on message msg under sk. We also assume that the protocol generates a status message like "ok" or $\perp$ for the signer, denoting success or failure, respectively.

- $\mathsf{Ver}(msg, \mathsf{pk}, \sigma)$ is a deterministic polynomial time algorithm. On input a message msg, a public key pk, and a purported signature σ, it determines whether σ is a valid signature on msg with respect to public key pk. If it is valid, the algorithm outputs 1, otherwise it outputs 0.

According to [104], a secure blind signature scheme must satisfy the following three properties: correctness, blindness, and one-more unforgeability.

Correctness states that if both the signer and the user comply with the signing protocol, then the produced blind signature is accepted as a valid signature by the verification algorithm except with probability ε, which denotes the scheme's *correctness error*. More formally, we have:

Definition 3.2.3. (Correctness of BSS) A BSS is *correct* with *correctness error* $\varepsilon \in [0, 1]$, if for all uniformly picked messages $msg \in \{0,1\}^*$, all honestly generated keys $(\mathsf{pk}, \mathsf{sk})$, and any honestly generated signature σ through the signature issuing protocol, σ is a valid signature with probability:

$$\Pr\left[\sigma \neq \perp \wedge \mathsf{BSS.Ver}(\mathsf{pk}, msg, \sigma) = 1 \;\middle|\; \begin{array}{l} msg \leftarrow_{\$} \{0,1\}^*, \\ \mathsf{par} \leftarrow_{\$} \mathsf{BSS.PG}(1^{\lambda}), \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{BSS.KG}(\mathsf{par}), \\ \sigma \leftarrow \langle \mathsf{S}(\mathsf{sk}), \mathsf{U}(\mathsf{pk}, msg) \rangle \end{array}\right] \geq 1 - \varepsilon$$

If $\varepsilon = 0$ in the above definition, we say that BSS has *perfect correctness*. Similarly, if $\varepsilon = negl(\lambda)$, we say that BSS has *statistical correctness*.

Blindness informally states that it is infeasible for a malicious signer to link any valid signature to the exact instance (or session) of the signature issuing protocol in which it was created. More formally, we have:

Definition 3.2.4. (Blindness) We define blindness of a blind signature scheme $\mathsf{BSS} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Sign} = \langle \mathsf{S}, \mathsf{U} \rangle, \mathsf{Ver})$ via the game in Figure 3.1. In the game, the adversarial signer S^* works in three modes. In mode find, it chooses two messages msg_0, msg_1 and interacts with two user sessions in mode issue. Depending on a coin flip $b \leftarrow_\$ \{0,1\}$, the first (resp., second) user obtains a blind signature for msg_b (resp., msg_{1-b}). After seeing the unblinded signatures in the original order, w.r.t. msg_0, msg_1, the signer has to guess the bit b in mode guess. If either of the user algorithms fails in outputting a valid signature, the signer is merely notified of the failure but is not given any signature.

We define the *advantage* of adversary S^* in game $\mathbf{Blind}_{\mathsf{BSS}}$ as $\mathbf{Adv}^{\mathbf{Blind}}_{\mathsf{BSS}}(\mathsf{S}^*) := \left|\Pr[\mathbf{Blind}^{\mathsf{S}^*}_{\mathsf{BSS}}(1^\lambda) = 1] - 1/2\right|$. We say that BSS is *statistically blind* if for all adversaries S^*, $\mathbf{Adv}^{\mathbf{Blind}}_{\mathsf{BSS}}(\mathsf{A}) \approx 0$ (if $\mathbf{Adv}^{\mathbf{Blind}}_{\mathsf{BSS}}(\mathsf{S}^*) = 0$, we say that it is *perfectly blind*). We remark that in the above definition, we consider that the signer behaves honestly and generates its keys through the scheme's key generation algorithm. In the stronger, *malicious-signer* model, the signer should be unable to link the output signatures to the sessions that generated them, even if the signer is allowed to pick the keys on its own [75].

One-more unforgeability [104] states that an adversarial user U^* should be unable to produce even a single signature more than it should be able to learn through interacting with an honest signer S. Phrased differently, each completed interaction between signer and user should yield *at most* one signature. More formally, we have:

Definition 3.2.5. (One-more unforgeability of BSS) We define one-more unforgeability of a blind signature scheme $\mathsf{BSS} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Sign} = \langle \mathsf{S}, \mathsf{U} \rangle, \mathsf{Ver})$ via the game in Figure 3.2, where $\mathcal{H}$ denotes a family of random oracles. In particular, after k successful, complete interactions with honest signer S, the adversarial user U^* wins if it is able

Game $\mathbf{Blind}_{\mathsf{BSS}}(1^\lambda)$

1: $(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{BSS.KG}(1^\lambda)$
2: $(msg_0, msg_1, \mathsf{state}_{\mathsf{find}}) \leftarrow_\$ \mathsf{S}^*(\mathsf{find}, 1^\lambda)$
3: $b \leftarrow_\$ \{0,1\}$
4: $\mathsf{state}_{\mathsf{issue}} \leftarrow_\$ \mathsf{S}^{*\langle \cdot, \mathsf{U}(\mathsf{pk}, msg_b)\rangle^1, \langle \cdot, \mathsf{U}(\mathsf{pk}, msg_{1-b})\rangle^1}(\mathsf{issue}, \mathsf{state}_{\mathsf{find}})$
5: $\sigma_b := \mathsf{U}(\mathsf{pk}, msg_b)$, $\sigma_{1-b} := \mathsf{U}(\mathsf{pk}, msg_{1-b})$
6: If $(\sigma_0 = \perp \vee \sigma_1 = \perp)$
7: $\quad b' \leftarrow_\$ \mathsf{S}^*(\mathsf{guess}, \perp, \perp, \mathsf{state}_{\mathsf{issue}})$
8: Else
9: $\quad b' \leftarrow_\$ \mathsf{S}^*(\mathsf{guess}, \sigma_0, \sigma_1, \mathsf{state}_{\mathsf{issue}})$
10: return $[\![b' = b]\!]$

Figure 3.1: Security game for blindness.

Game $\mathbf{OMUF}_{\mathsf{BSS}}(1^\lambda)$

1: $(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{BSS.KG}(1^\lambda)$
2: $\mathsf{H} \leftarrow_\$ \mathcal{H}(1^\lambda)$
3: $(\mu_1, \sigma_1), \ldots, (\mu_l, \sigma_l) \leftarrow_\$ \mathsf{U}^{*\mathsf{H}(.), \langle \mathsf{S}(\mathsf{sk}), . \rangle^\infty}(\mathsf{pk})$
4: Let k := # successful, complete interactions between U^* and S.
5: $b_1 := [\![\mu_i \neq \mu_j, \forall i, j = 1, \ldots, l \text{ with } i \neq j]\!]$
6: $b_2 := [\![\mathsf{BSS.Ver}(\mathsf{pk}, \mu_i, \sigma_i) = 1, \forall i = 1, \ldots, l]\!]$
7: $b_3 := [\![l = k + 1]\!]$
8: return $b_1 \wedge b_2 \wedge b_3$

Figure 3.2: Security game for (honest-user) one-more unforgeability of BSS.

to output $l > k$ valid signatures, where the output signatures correspond to pairwise distinct messages.

An interactive $\mathsf{BSS} = (\mathsf{PG}, \mathsf{KG}, \langle \mathsf{S}, \mathsf{U} \rangle, \mathsf{Ver})$ is said to be $(\varepsilon, t, Q_{\mathsf{sig}}, Q_{\mathsf{H}})$-*one-more unforgeable* if Ver is deterministic, and for any algorithm U^* running in time at most t, making up to Q_{sig} signing queries and up to Q_{H} queries to oracle H, we have $\Pr\left[\mathbf{OMUF}^{\mathsf{U}^*}_{\mathsf{BSS}}(1^\lambda) = 1\right] \leq \varepsilon$.We define the *advantage* of adversary U^* in game $\mathbf{OMUF}_{\mathsf{BSS}}$ as $\mathbf{Adv}^{\mathbf{OMUF}}_{\mathsf{BSS}}(\mathsf{U}^*) := \Pr[\mathbf{OMUF}^{\mathsf{U}^*}_{\mathsf{BSS}}(1^\lambda) = 1]$.

Remark 1. Requiring all message-signature pairs to be pairwise distinct in the condition on line 5 of Figure 3.2 (i.e., $(\mu_i, \sigma_i) \neq (\mu_j, \sigma_j), \forall i, j \in [l]$ s.t. $1 \leq j < i \leq l$) results in a stronger security notion called *strong one-more unforgeability*.

Remark 2. It should be noted that the above definition for unforgeability is not meaningful for blind signature schemes with noticeable correctness error (i.e., schemes in which either party may abort with noticeable probability) [98]. Indeed, because there is no 1-1 correspondence between a signing session and a message-signature pair, even if the adversary behaves honestly during the signing protocol, it may obtain less signatures than the number of closed sessions. However, it still has to come up with (at least) $k + 1$ signatures in order to win. This results in a significant weakening of the definition.

3.3 Overview of Flawed Lattice-Based BSS

In this section we review a number of blind signature schemes in the literature whose design has been shown to be flawed in one way or another.

3.3.1 Rückert's Blind Signature Scheme

The first attempt towards constructing blind signature schemes from lattice-based assumptions was made in 2008 in the seminal work of Rückert [159]. Following a well-known pattern[1] found in many number-theoretic blind signatures [139, 153, 154, 163, 152], [159] uses Lyubashevsky's identification scheme [122] as its basis (which itself relies on the SIS hash function) to construct a Fiat-Shamir-like blind signature scheme. However, because of the *rejection sampling* technique (which stems from the underlying hash function's enclosedness errors [98]), there is no guarantee that any given protocol run will actually produce a valid blind signature for the user. The novelty introduced in [159] to resolve this issue is to extend the standard 3-move protocol structure with an additional move, in which a user can prove to the signer that it failed to obtain a valid signature (when unblinding).

[1] The recent work of [97] formalizes this pattern but only for hash functions with negligible enclosedness errors.

Construction

We now describe Rückert's blind signature scheme in detail. The parameter definitions are summarized in Table 3.1. The construction makes use of the following cryptographic components:

- the $\mathsf{R}-\mathsf{SIS}$ linear hash function family $\mathsf{F}(\hat{\mathbf{x}}) := \sum_{i=1}^{m} \mathbf{u}_i \mathbf{x}_i \pmod q, \hat{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_m) \in \mathcal{R}_q^m$, where $\hat{\mathbf{u}} \leftarrow_{\$} \mathcal{R}_q^m$.
- a hash function $\mathsf{H} : \{0,1\}^* \to S_{c'}$ (modelled as a programmable random oracle),
- a statistically hiding, and computationally binding commitment function $\mathsf{com} : \{0,1\}^* \times \{0,1\}^n \to \{0,1\}^n$.

Table 3.1: Parameter definitions for Rückert's lattice-based blind signature scheme.

Parameter	Definition and Constraints
n	main security parameter, integer power of 2
d_{sk}	positive integer $< q/(4n)$
$\mathcal{D}_{sk}$	$\{\mathbf{v} \in \mathcal{R}_q : \|\mathbf{v}\|_\infty \leq d_{sk}\}$
m	positive integer $> \lfloor \log q / \log(2d_{sk}) \rfloor + 1$
$S_{c'}$	challenge space $\{\mathbf{v} \in \mathcal{R}_q : \|\mathbf{v}\|_\infty \leq 1 =: d_{c'}\}$
u, v	positive integer constants ≥ 1
S_b	$\{\mathbf{v} \in \mathcal{R}_q : \|\mathbf{v}\|_\infty \leq und_{c'} =: d_b\}$
S_c	$\{\mathbf{v} \in \mathcal{R}_q : \|\mathbf{v}\|_\infty \leq d_b - d_{c'} =: d_c\}$
$\mathcal{D}_r$	$\{\mathbf{v} \in \mathcal{R}_q : \|\mathbf{v}\|_\infty \leq vmn^2 d_{sk} d_c =: d_r\}$
$\mathcal{D}_s$	$\{\mathbf{v} \in \mathcal{R}_q : \|\mathbf{v}\|_\infty \leq d_r - nd_{sk} d_c =: d_s\}$
$\mathcal{D}_a$	$\{\mathbf{v} \in \mathcal{R}_q : \|\mathbf{v}\|_\infty \leq vmnd_s =: d_a\}$
$\mathcal{D}_{s'}$	$\{\mathbf{v} \in \mathcal{R}_q : \|\mathbf{v}\|_\infty \leq d_a - d_s =: d_{s'}\}$
$\mathcal{D}$	$\{\mathbf{v} \in \mathcal{R}_q : \|\mathbf{v}\|_\infty \leq d_s + d_a + nd_{sk} d_{c'} =: d\}$
q	prime s.t. $\geq 4dmn\sqrt{n} \log n$

Key Generation. On input the main security parameter n, the algorithm selects

parameters as specified in Table 3.1.[2] $\mathsf{KeyGen}(1^n)$ samples polynomials $\hat{\mathbf{u}} \leftarrow_{\$} \mathcal{R}_q^m$ defining the homomorphic hash function F and $\hat{\mathbf{z}} \leftarrow_{\$} \mathcal{D}_{sk}^m$. Finally, it sets $\mathsf{sk} := \hat{\mathbf{z}}$, and $\mathsf{pk} := \mathsf{F}(\hat{\mathbf{z}})$, and returns $(\mathsf{sk}, \mathsf{pk})$.

Signing. The interactive signing protocol $\langle \mathsf{S}(\mathsf{sk}), \mathsf{U}(\mathsf{pk}, msg) \rangle$ works as follows:

1. **Signer:** At the outset, the signer samples a masking vector $\hat{\mathbf{r}} \leftarrow_{\$} \mathcal{D}_r^m$ and computes a commitment $\mathbf{R} := \mathsf{F}(\hat{\mathbf{r}})$. It sends $\mathbf{R}$ to the user.

2. **User:** The user receives $\mathbf{R}$ and samples its masking parameters $\hat{\mathbf{a}} \leftarrow_{\$} \mathcal{D}_a^m$ and $\mathbf{b} \leftarrow_{\$} S_b$. It samples randomness $\rho \leftarrow_{\$} \{0,1\}^n$ and uses it to commit to the message-to-be-signed by computing $C := \mathsf{com}(msg, \rho)$. It computes a "masked commitment" $\mathbf{R}' := \mathbf{R} + \mathsf{F}(\hat{\mathbf{a}}) + \mathbf{b} \cdot \mathsf{pk}$ as well as its challenge $\mathbf{c}' := \mathsf{H}(\mathbf{R}', C)$. Because $\mathbf{c}'$ will be part of the produced signature, it cannot be sent in the clear. The user "blinds" $\mathbf{c}'$ as $\mathbf{c} := \mathbf{c}' + \mathbf{b}$. If $\mathbf{c} \notin S_c$ then $\mathbf{c}'$ "leaks" information about $\mathbf{c}$. Hence, to maintain anonymity, the user only sends $\mathbf{c}$ if it falls within S_c and repeats the entire step from scratch otherwise. This rejection sampling step can be performed locally by the user without affecting the scheme's correctness.

3. **Signer:** Upon receiving $\mathbf{c}$, the signer computes its response $\hat{\mathbf{s}} := \hat{\mathbf{r}} + \mathbf{c} \cdot \mathsf{sk}$. To make $\hat{\mathbf{s}}$ independent of the secret key, the signer rejection-samples it and only sends it to the user if $\hat{\mathbf{s}} \in \mathcal{D}_s^m$. Otherwise, the entire protocol restarts. This introduces an expected correctness error of $1 - e^{-1/\nu} + o(1)$ to the blind signature scheme.

4. **User:** Upon receiving $\hat{\mathbf{s}}$, the user checks if $\mathsf{F}(\hat{\mathbf{s}}) \neq \mathbf{c} \cdot \mathsf{pk} + \mathbf{R}$ and $\hat{\mathbf{s}} \notin \mathcal{D}_s^m$. If either condition fails, then the signer sent invalid data, and the user can trivially request a protocol restart. Otherwise, the user "unblinds" the response $\hat{\mathbf{s}}$ by computing $\hat{\mathbf{s}}' := \hat{\mathbf{s}} + \hat{\mathbf{a}}$. A final rejection sampling step is necessary here in order to make $\hat{\mathbf{s}}'$ independent from $\hat{\mathbf{s}}$. Hence, the user outputs $(\mathbf{c}', \hat{\mathbf{s}}', \rho)$ iff $\hat{\mathbf{s}}' \in \mathcal{D}_{s'}^m$. Otherwise, the user reveals the blinding parameters $\hat{\mathbf{a}}, \mathbf{b}$, the challenge $\mathbf{c}'$ and the commitment to the message C (the user only withholds the decommitment

[2] These parameters are *globally* known and implicit inputs to all other algorithms.

parameter ρ to avoid having to reveal *msg*) to the signer and requests a restart. Notice that rejection sampling during this step further amplifies the blind signature scheme's correctness error by an additional factor of $1 - e^{-1/\nu} + o(1)$.

5. **Signer:** The signer receives $\hat{\mathbf{a}}, \mathbf{b}$, the challenge $\mathbf{c}'$ and the commitment to the message C. These allow the signer to trace all computations performed on the user's side and ascertain if a restart is truly necessary. To this end, it computes the blinded commitment $\mathbf{R}' := \mathbf{R} + \mathsf{F}(\hat{\mathbf{a}}) + \mathbf{b} \cdot \mathsf{pk}$, as well as $\tilde{\mathbf{c}}_1 := \mathsf{H}(\mathbf{R}', C)$ and $\tilde{\mathbf{c}}_2 := \mathsf{H}(\mathsf{F}(\hat{\mathbf{s}} + \hat{\mathbf{a}}) - \mathbf{c}' \cdot \mathsf{pk}, C)$. If $\hat{\mathbf{s}} \in \mathcal{D}_s^m$, $\mathbf{c} - \mathbf{b} = \mathbf{c}'$, $\mathbf{c}' = \tilde{\mathbf{c}}_1 = \tilde{\mathbf{c}}_2$, and $\hat{\mathbf{s}} + \hat{\mathbf{a}} \notin \mathcal{D}_{s'}^m$, the signer restarts the entire protocol. Otherwise, the user attempts to cheat by submitting an invalid "proof" and the signer simply dismisses the request.

Verification. On input public key pk, a purported signature $(\mathbf{c}', \hat{\mathbf{s}}', \rho)$ and message *msg*, algorithm $\mathsf{Ver}(\mathsf{pk}, (\mathbf{c}', \hat{\mathbf{s}}', \rho), msg)$ outputs 1 iff $\hat{\mathbf{s}}' \in \mathcal{D}_{s'}^m$ and $\mathsf{H}(\mathsf{F}(\hat{\mathbf{s}}') - \mathsf{pk} \cdot \mathbf{c}', \mathsf{com}(msg, \rho)) = \mathbf{c}'$. Otherwise, it outputs 0.

Claimed Security Results

While rejection-sampling can be used for tailoring the distributions of messages exchanged during the protocol, it comes at the cost of introducing a noticeable correctness error to the scheme:

Lemma 3.3.1. *(Adapted from Theorem 3.3. of [159]) The construction of [159] has an expected correctness error of* $(1 - e^{-1/\nu} + o(1))^2$.

Moreover, because of its reliance on a commitment scheme, the construction of [159] is only as blind as com is hiding.

Lemma 3.3.2. *(Adapted from Theorem 3.5. of [159]) If* com *is a statistically (resp. perfectly) hiding commitment function, then the construction of [159] is statistically (resp. perfectly) blind.*

Finally, unforgeability relies on the collision resistance of the $\mathsf{R-SIS}$ hash function family, as well as the binding property of com:

Lemma 3.3.3. *(Adapted from Theorem 3.8. of [159]) If* com *is a computationally binding commitment function and the* R – SIS *hash function is collision-resistant in* $\mathcal{D}$*, then the construction of [159] is one-more unforgeable.*

3.3.2 BLAZE

BLAZE [21] is a blind signature scheme, structurally similar to [159]. The signing protocol relies on rejection sampling with discrete Gaussian samples, instead of uniform, which allows for smaller key and signature sizes. Furthermore, the use of signed permutations allows their scheme to avoid having to rejection-sample the challenge part of the signature (when unblinding), thus achieving a smaller correctness error. The key result ensuring this is Lemma 3.2.1.

Construction

We now describe the BLAZE blind signature scheme in detail. BLAZE makes use of the following cryptographic ingredients:

- a public (randomly chosen) deterministic function $\mathsf{Expand} : \{0,1\}^\lambda \to \mathcal{R}_q^m$ (can be instantiated for example with a PRF),
- a hash function $\mathsf{H} : \{0,1\}^* \to \mathbb{S}_\kappa^n$ modelled as a random oracle,
- a statistically hiding and computationally binding commitment function $\mathsf{com} : \{0,1\}^* \times \{0,1\}^\lambda \to \{0,1\}^\lambda$,
- functions Compress and Decompress for compressing (resp. decompressing) integers distributed according to $D_{\mathbb{Z},\sigma}$ (for implementation details cf. Table 3 in [67]).

Parameter Generation. On input the main security parameter λ, algorithm $\mathsf{PG}(1^\lambda)$ sets the scheme's public parameters par according to the specifications of Table 3.2, and it outputs par.

Key Generation. On input public scheme parameters par, the key generation algorithm KG samples a seed $\leftarrow_\$ \{0,1\}^\lambda$ and $\hat{\mathbf{z}}_1, \mathbf{z}_2 \leftarrow D^m_{\mathbb{Z}^n,\sigma} \times D_{\mathbb{Z}^n,\sigma}$. It expands the

Table 3.2: Parameter definitions for the BLAZE blind signature scheme.

Parameter	Definition and Constraints
λ	main security parameter
n	integer power of 2
m	number of polynomials in the secret key s.t. $m + 1 \geq 2$
q	prime modulus s.t. $q \equiv 1 \pmod{2n}$
σ	standard deviation for the distribution from which the secret key is drawn s.t. $\sigma > 0, (m+1)\log(t\sigma) > \log(q)$
κ	Hamming weight of H's output s.t. $2^{\kappa}\binom{n}{\kappa} > 2^{\lambda}$
s^*	standard deviation of the distribution from which the signer draws its blinding parameters s.t. $s^* = \alpha^*\sqrt{\kappa}\|\mathsf{sk}\|, \alpha^* > 0$
s	standard deviation of the distribution from which the user draws its blinding parameters s.t. $s = \eta\alpha s^*\sqrt{(m+1)\kappa n}\|\mathsf{sk}\|, \alpha, \eta > 0$ $\eta^{(m+1)n} e^{\frac{(m+1)n}{2}(1-\eta^2)} \leq 2^{-\lambda}$
M	expected number of iterations s.t. $M = e^{\frac{12}{\alpha} + \frac{1}{2\alpha^2} + \frac{12}{\alpha^*} + \frac{1}{2\alpha^{*2}}}$

seed to a vector of polynomials by computing $\hat{\mathbf{a}} := \mathsf{Expand}(\mathsf{seed})$, and it also computes $\mathbf{b} := \hat{\mathbf{a}} \cdot \hat{\mathbf{z}}_1 + \mathbf{z}_2 \pmod{q}$. It sets $\mathsf{sk} = (\hat{\mathbf{z}}_1, \mathbf{z}_2)$, and $\mathsf{pk} = (\mathsf{seed}, \mathbf{b})$, and returns $(\mathsf{sk}, \mathsf{pk})$.

Signing. The interactive signing protocol $\langle \mathsf{S}(\mathsf{sk}, \mathsf{seed}), \mathsf{U}(\mathsf{pk}, \mathsf{seed}, msg) \rangle$ works as follows:

1. **Signer:** On input secret key sk, as well as seed, the signer first expands the seed via $\hat{\mathbf{a}} := \mathsf{Expand}(\mathsf{seed})$. It then samples masking terms $\hat{\mathbf{r}}^*_{1,1}, \ldots, \hat{\mathbf{r}}^*_{\kappa,1} \leftarrow D^m_{\mathbb{Z}^n, s^*}$ and $\mathbf{r}^*_{1,2}, \ldots, \mathbf{r}^*_{\kappa,2} \leftarrow D_{\mathbb{Z}^n, s^*}$ and computes commitments $\mathbf{R}_j := \hat{\mathbf{a}} \cdot \hat{\mathbf{r}}^*_{j,1} + \mathbf{r}^*_{j,2} \pmod{q}, \forall j \in [\kappa]$. It sets $\hat{\mathbf{R}} := (\mathbf{R}_1, \ldots, \mathbf{R}_{\kappa})$, and it sends $\hat{\mathbf{R}}$ to the user.

2. **User:** On input seed, and a message $msg \in \{0,1\}^*$, and commitment $\hat{\mathbf{R}}$, the user computes $\hat{\mathbf{a}} := \mathsf{Expand}(\mathsf{seed})$. It samples randomness $r, r', \rho, \rho' \leftarrow_{\$} \{0,1\}^{\lambda}$, signed permutations $p_1, \ldots, p_{\kappa} \leftarrow_{\$} \mathbb{S}$, and masking terms $(\hat{\mathbf{e}}_1, \mathbf{e}_2) \leftarrow_{\rho} D^{m+1}_{\mathbb{Z}^n, s}$ (notice that the random coins are fixed through ρ). It computes commitments $C_1 := \mathsf{com}(msg, r)$ and $C_2 := \mathsf{com}(\rho', r')$ and a challenge $\mathbf{c} := \mathsf{H}(\hat{\mathbf{a}} \cdot \hat{\mathbf{e}}_1 + \mathbf{e}_2 + \sum_{j=1}^{\kappa} p_j \mathbf{R}_j, C_1, C_2)$. It partitions $\mathbf{c} = \sum_{j=1}^{\kappa} c_j \in \mathbb{S}^n_{\kappa}$ and individually blinds each partitioning term via $c^*_j := p_j^{-1} c_j \in \mathbb{S}$. It sets $\mathbf{c}^* := (c^*_1, \ldots, c^*_{\kappa})$ and sends $\mathbf{c}^*$ to the signer.

3. **Signer:** The signer receives $\mathbf{c}^*$ and computes its responses with the help of

its secret key $\mathsf{sk} = (\hat{\mathbf{z}}_1, \mathbf{z}_2)$. In particular, it computes $\hat{\mathbf{s}}^*_{j,1} := \hat{\mathbf{r}}^*_{j,1} + \hat{\mathbf{z}}_1 c^*_j, \mathbf{s}^*_{j,2} := \mathbf{r}^*_{j,2} + \mathbf{z}_2 c^*_j, \forall j \in [\kappa]$, and sets $\hat{\mathbf{s}}^* := (\hat{\mathbf{s}}^*_{1,1}, \ldots, \mathbf{s}^*_{\kappa,2})$, and it rejection-samples $\hat{\mathbf{s}}^*$ using Algorithm 1 to ensure that it is distributed independently from sk. In case of rejection (which occurs with probability $1 - e^{-12/\alpha^* - 1/(2\alpha^{*2})}$), the entire protocol restarts. Otherwise, $\hat{\mathbf{s}}^*$ is sent to the user.

4. **User:** The user receives $\hat{\mathbf{s}}^* := (\hat{\mathbf{s}}^*_{1,1}, \ldots, \mathbf{s}^*_{\kappa,2})$ and computes $\hat{\mathbf{v}}_1 := \sum_{j=1}^{\kappa} p_j \hat{\mathbf{s}}^*_{j,1}$, $\mathbf{v}_2 := \sum_{j=1}^{\kappa} p_j \mathbf{s}^*_{j,2}$. If $\|(\hat{\mathbf{v}}_1, \mathbf{v}_2)\| > \eta s^* \sqrt{(m+1)\kappa n}$, then the signer submitted an invalid response and the entire protocol must be restarted. Otherwise, it "unblinds" the response via $\hat{\mathbf{s}}_1 := \hat{\mathbf{v}}_1 + \hat{\mathbf{e}}_1$ and $\mathbf{s}_2 := \mathbf{v}_2 + \mathbf{e}_2$. It then invokes $\mathsf{Rejection\_Sample}((\hat{\mathbf{s}}_1, \mathbf{s}_2), (\hat{\mathbf{v}}_1, \mathbf{v}_2); \rho')$ to make $(\hat{\mathbf{s}}_1, \mathbf{s}_2)$ independent from $(\hat{\mathbf{v}}_1, \mathbf{v}_2)$. Notice that the user needs to fix the random coins of Algorithm 1 through ρ' to which it committed to earlier in the protocol. If rejection sampling fails (which occurs with probability $1 - e^{-12/\alpha - 1/(2\alpha^2)}$), the user sends $(C_1, \rho, \rho', r', p_1, \ldots, p_\kappa, \mathbf{c})$ to the signer, requesting a protocol restart. Otherwise, it compresses $(\hat{\mathbf{s}}_1, \mathbf{s}_2)$ via $(\hat{\mathbf{s}}_1, \mathbf{s}_2) := \mathsf{Compress}(\hat{\mathbf{s}}_1, \mathbf{s}_2)$ and it outputs $(C_2, r, \hat{\mathbf{s}}_1, \mathbf{s}_2, \mathbf{c})$ as its signature.

5. **Signer:** Upon receiving $(C_1, \rho, \rho', r', p_1, \ldots, p_\kappa, \mathbf{c})$, the signer uses ρ to retrieve[3] $\hat{\mathbf{e}}_1, \mathbf{e}_2$ via $(\hat{\mathbf{e}}_1, \mathbf{e}_2) \leftarrow_\rho D^{m+1}_{\mathbb{Z}^n, s}$. It then computes $C_2 := \mathsf{com}(\rho', r')$, $\hat{\mathbf{s}}_1 := \hat{\mathbf{e}}_1 + \sum_{j=1}^{\kappa} p_j \hat{\mathbf{s}}^*_{j,1}$, $\mathbf{s}_2 := \mathbf{e}_2 + \sum_{j=1}^{\kappa} p_j \mathbf{s}^*_{j,2}$. If $\sum_{j=1}^{\kappa} p_j c^*_j = \mathbf{c}$, $\mathbf{c} = \mathsf{H}(\hat{\mathbf{a}} \cdot \hat{\mathbf{e}}_1 + \mathbf{e}_2 + \sum_{j=1}^{\kappa} p_j \mathbf{R}_j \pmod q, C_1, C_2)$, $\mathbf{c} = \mathsf{H}(\hat{\mathbf{a}} \cdot \hat{\mathbf{s}}_1 + \mathbf{s}_2 - \mathbf{bc} \pmod q, C_1, C_2)$, and $\mathsf{Rejection\_Sample}((\hat{\mathbf{s}}_1, \mathbf{s}_2); \rho') = 0$, the signer is convinced and restarts the protocol. Otherwise, it ignores the request.

Verification. On input public key $\mathsf{pk} = (\mathsf{seed}, \mathbf{b})$, message $msg \in \{0,1\}^*$ and a purported signature $(C_2, r, \hat{\mathbf{s}}_1, \mathbf{s}_2, \mathbf{c})$, the verifier expands the seed to obtain $\hat{\mathbf{a}} := \mathsf{Expand}(\mathsf{seed})$, and it decompresses $(\hat{\mathbf{s}}_1, \mathbf{s}_2)$ using $\mathsf{Decompress}$. If $\|(\hat{\mathbf{s}}_1, \mathbf{s}_2)\| < \eta s \sqrt{(m+1)n}$ and $\mathbf{c} = \mathsf{H}(\hat{\mathbf{a}} \cdot \hat{\mathbf{s}}_1 + \mathbf{s}_2 - \mathbf{bc} \pmod q, \mathsf{com}(msg, r), C_2)$, it outputs 1 (accept). Otherwise, it outputs 0 (reject).

[3] Essentially, the signer samples from distribution $D^{m+1}_{\mathbb{Z}^n, s}$ with fixed random coins.

Claimed Security Results

Thanks to the use of signed permutations, the correctness error that would be induced when unblinding the challenge part of a signature is completely eliminated:

Lemma 3.3.4. *(Adapted from Theorem 1 of [21]) If the parameters for* BLAZE *are set according to Table 3.2, then* BLAZE *has a correctness error of* $(1 - e^{-12/\alpha^* - 1/(2\alpha^{*2})})(1 - e^{-12/\alpha - 1/(2\alpha^2)})$.

As with [159], reliance on com affects the blindness property:

Lemma 3.3.5. *(Adapted from Theorem 2 of [21]) If* com *is a statistically hiding commitment function, then* BLAZE *is statistically blind.*

Similarly to [159], unforgeability is also conditioned on the binding property of com:

Lemma 3.3.6. *(Adapted from Theorem 3 of [21]) If* com *is a statistically hiding and computationally binding commitment function, and* R – SIS *is hard for the parameters set according to Table 3.2, then* BLAZE *is strongly one-more-unforgeable.*

Attacks and Countermeasures

BLAZE can be attacked by exploiting a subtle design flaw as hinted in [98]. Notice that because a malicious user has complete control over the randomness ρ, it can attack the protocol by rigging ρ so that u (line 5 of Algorithm 1) is always picked *very* close to 1 (e.g., with statistical distance at most $1 - 2^{-128}$), thus causing the Rejection_Sample algorithm to always output 0 when the user tries to unblind. This allows the user to keep asking for new signatures until it can produce its own forgery and win in the unforgeability game. One possible countermeasure to this kind of attack would be to switch the distribution from discrete Gaussian to uniform but this would increase key and signature sizes considerably.

3.3.3 BLAZE+

BLAZE+ [19] introduces a novel technique for reducing the correctness error by performing multiple rejection samplings in parallel. Instead of sampling a single pair of

masking terms $(\hat{\mathbf{e}}_1, \mathbf{e}_2)$, the user samples multiple such pairs and stores (a function of) each such pair as a leaf of a Merkle tree. This offers the advantage that when the user tries to unblind to produce its signature, it will succeed for at least one of these pairs with very high probability. By tuning the number of sampled pairs to a sufficiently large amount, this probability can effectively be made negligible. The optimizations introduced are backwards compatible with BLAZE.

Table 3.3: Parameter definitions for the BLAZE+ blind signature scheme(s).

Parameter	Definition and Constraints
λ	main security parameter
n	integer power of 2
m	number of polynomials in the secret key s.t. $m + 1 \geq 2$
q	prime modulus s.t. $q \equiv 1 \pmod{2n}$
γ	positive integer constant
σ	standard deviation for the distribution from which the secret key is drawn s.t. $\sigma > 0, (m+1)\log(t\sigma) > \log(q)$
κ	Hamming weight of H's output s.t. $2^\kappa \binom{n}{\kappa} > 2^\lambda$
s^*	standard deviation of the distribution from which the signer draws its blinding parameters s.t. $s^* = \alpha^* \gamma \sigma \sqrt{(m+1)\kappa n}, \alpha^* > 0$
s	standard deviation of the distribution from which the user draws its blinding parameters s.t. $s = \eta \alpha s^* \sqrt{(m+1)\kappa n}, \alpha, \eta > 0,$ $\eta^{(m+1)n} e^{\frac{(m+1)n}{2}(1-\eta^2)} \leq 2^{-\lambda}$

Three-move Variant of BLAZE+

We now describe the 3-move variant of the BLAZE+ blind signature scheme in detail. This variant makes use of the following cryptographic ingredients:

- a public (randomly chosen) deterministic function $\mathsf{Expand} : \{0,1\}^\lambda \to \mathcal{R}_q^m$ (can be instantiated for example with a PRF),
- a hash function $\mathsf{H} : \{0,1\}^* \to \mathbb{S}_\kappa^n$ modelled as a random oracle,
- functions Compress and Decompress for compressing (resp. decompressing) integers distributed according to $D_{\mathbb{Z},\sigma}$.

Parameter Generation. On input the main security parameter λ, algorithm $\mathsf{PG}(1^\lambda)$ generates the scheme's public parameters par according to Table 3.3 and it outputs par.

Key Generation. On input the scheme's parameters par, algorithm $\mathsf{KG}(\mathsf{par})$ samples a seed $\leftarrow_\$ \{0,1\}^\lambda$ and $\hat{\mathbf{z}} := (\hat{\mathbf{z}}_1, \mathbf{z}_2) \leftarrow D^m_{\mathbb{Z}^n,\sigma} \times D_{\mathbb{Z}^n,\sigma}$. If $\|(\hat{\mathbf{z}}_1, \mathbf{z}_2)\| > \gamma\sigma\sqrt{(m+1)n}$, the algorithm resamples $(\hat{\mathbf{z}}_1, \mathbf{z}_2)$. It expands the seed to a vector of polynomials by computing $\hat{\mathbf{a}}' := \mathsf{Expand}(\mathsf{seed})$, and sets $\hat{\mathbf{a}} := (1, \hat{\mathbf{a}}')$ It also computes $\mathbf{b} := \hat{\mathbf{a}} \cdot \hat{\mathbf{z}} \pmod q$. It sets $\mathsf{sk} = \hat{\mathbf{z}}$, and $\mathsf{pk} = (\mathsf{seed}, \mathbf{b})$, and returns $(\mathsf{sk}, \mathsf{pk})$.

Signing. The interactive signing protocol $\langle \mathsf{S}(\mathsf{sk}, \mathsf{seed}), \mathsf{U}(\mathsf{pk}, \mathsf{seed}, msg)\rangle$ works as follows:

1. **Signer:** On input secret key sk, as well as seed, the signer expands the seed via $\hat{\mathbf{a}}' := \mathsf{Expand}(\mathsf{seed})$ and sets $\hat{\mathbf{a}} := (1, \hat{\mathbf{a}}')$. It samples masking parameters $\hat{\mathbf{r}}^*_1, \ldots, \hat{\mathbf{r}}^*_\kappa \leftarrow_\$ D^{m+1}_{\mathbb{Z}^n,s^*}$ and computes commitments $\mathbf{R}_j := \hat{\mathbf{a}} \cdot \hat{\mathbf{r}}^*_j \pmod q, \forall j \in [\kappa]$. It sets $\hat{\mathbf{R}} := (\mathbf{R}_1, \ldots, \mathbf{R}_\kappa)$ and sends $\hat{\mathbf{R}}$ to the user.

2. **User:** On input seed, and a message $msg \in \{0,1\}^*$, the user also expands the seed $\hat{\mathbf{a}}' := \mathsf{Expand}(\mathsf{seed})$ and sets $\hat{\mathbf{a}} := (1, \hat{\mathbf{a}}')$. It samples signed permutations $p_1, \ldots, p_\kappa \leftarrow_\$ \mathbb{S}$, randomness $\rho \leftarrow_\$ \{0,1\}^\lambda$, and masking terms $(\hat{\mathbf{e}}_0, \ldots, \hat{\mathbf{e}}_{l-1}) \leftarrow_\rho D^{m+1}_{\mathbb{Z}^n,s}$ (notice that the random coins are fixed through ρ). It computes $\mathbf{y} := \sum_{j=1}^\kappa p_j \mathbf{R}_j \pmod q$ as well as $\mathbf{t}_j := \hat{\mathbf{a}} \cdot \hat{\mathbf{e}}_j + \mathbf{y} \pmod q, \forall j = 0, \ldots, l-1$. It uses the $\mathbf{t}_j$ elements as leaf nodes to construct a Merkle tree via $(\mathsf{tree}, \mathsf{root}) := \mathsf{HashTree}(\mathbf{t}_0, \ldots, \mathbf{t}_{l-1})$. It computes its challenge $\mathbf{c} := \mathsf{H}(\mathsf{root}, msg)$ and partitions it into monomials c_j (this can trivially be done by writing $\mathbf{c}$ in the form $\mathbf{c} = \sum_{j=1}^\kappa c_j, c_j \in \mathbb{S}$). It blinds the partitioning monomials with the help of signed permutations $p_1, \ldots, p_\kappa$ via $c^*_j := c_j p_j^{-1}$. It sets $\mathbf{c}^* := (c^*_1, \ldots, c^*_\kappa)$ as the blinded challenge and transmits $\mathbf{c}^*$ to the signer.

3. **Signer:** On input a blinded challenge $\mathbf{c}^*$, the signer computes its responses $\hat{\mathbf{s}}^*_j := \hat{\mathbf{r}}^*_j + \mathsf{sk} \cdot c^*_j, \forall j \in [\kappa]$. It then invokes $\mathsf{Rejection\_Sample}((\hat{\mathbf{s}}^*_1, \ldots, \hat{\mathbf{s}}^*_\kappa), (\mathsf{sk} \cdot c^*_1, \ldots, \mathsf{sk} \cdot c^*_\kappa))$ to make its response independent of sk. If rejection sampling fails, it restarts the entire protocol. Otherwise, it sets $\hat{\mathbf{s}}^* := (\hat{\mathbf{s}}^*_1, \ldots, \hat{\mathbf{s}}^*_\kappa)$ and sends $\hat{\mathbf{s}}^*$ to the user.

4. **User:** The user receives $\hat{\mathbf{s}}^*$ and computes $\hat{\mathbf{v}} := \sum_{j=1}^\kappa p_j \hat{\mathbf{s}}^*_j$. If $\|\hat{\mathbf{v}}\| > \eta s^* \sqrt{(m+1)\kappa n}$,

the protocol is aborted (this occurs with probability $2^{-\lambda}$). The user then locates the first $\hat{\mathbf{e}}_j, j \in \{0, \ldots, l-1\}$ for which rejection sampling of $\hat{\mathbf{e}}_j + \hat{\mathbf{v}}$ succeeds (if none succeed, the protocol is aborted)[4]. Let I be the index of the first successful rejection sampling. The user sets $\hat{\mathbf{s}} := \mathsf{Compress}(\hat{\mathbf{e}}_I + \hat{\mathbf{v}})$ and also invokes $\mathsf{BuildAuth}(I, \mathsf{tree})$ to compute an authentication path proving that the I-th element under root was used to compute $\hat{\mathbf{s}}$. It outputs $(\hat{\mathbf{s}}, \mathbf{c}, \mathsf{auth})$ as its blind signature.

Verification. On input public key pk, message $msg \in \{0,1\}^*$ and a purported signature $(\hat{\mathbf{s}}, \mathbf{c}, \mathsf{auth})$, the verifier expands the seed to obtain $\hat{\mathbf{a}}' := \mathsf{Expand}(\mathsf{seed}), \hat{\mathbf{a}} := (1, \hat{\mathbf{a}}')$ and decompresses the signature via $\hat{\mathbf{s}} := \mathsf{Decompress}(\hat{\mathbf{s}})$. It computes $\mathbf{w} := \hat{\mathbf{a}} \cdot \hat{\mathbf{s}} - \mathbf{bc} \pmod q$ and $\mathsf{root} := \mathsf{RootCalc}(\mathbf{w}, \mathsf{auth})$. If $\|\hat{\mathbf{s}}\| \leq \eta s \sqrt{(m+1)n}$ and $\mathbf{c} = \mathsf{H}(\mathsf{root}, msg)$, it returns 1 (accept). Otherwise, it outputs 0 (reject).

Claimed Security Results

Correctness is significantly improved over BLAZE thanks to the use of a Merkle tree:

Lemma 3.3.7. *(Adapted from Theorem 1 of [19]) If the parameters are set according to Table 3.3, then* BLAZE+ *has a correctness error of* $2^{-2\lambda}$.

While this variant manages to decouple itself from relying a commitments scheme, it still satisfies blindness in a statistical sense:

Lemma 3.3.8. *(Adapted from Theorem 2 of [19])* BLAZE+ *is statistically blind.*

Finally, as long as the hash function G used for constructing the Merkle tree satisfies collision-resistance and R – SIS is hard, BLAZE+ is *strongly* one-more unforgeable:

Lemma 3.3.9. *(Adapted from Theorem 3 of [19]) If* $\mathsf{G} : \{0,1\}^* \to \{0,1\}^{2\lambda}$ *is a collision-resistant hash function used for constructing the Merkle trees during the signing protocol, and* R – SIS *is hard for the parameters set according to Table 3.3, then* BLAZE+ *is strongly one-more unforgeable in the ROM.*

[4] This occurs with probability $2^{-\lambda}$.

Four-move Variant of BLAZE+

In [19], the authors propose a second, 4-move, "hybrid" variant of BLAZE+ in which the user can prove that a session did not yield a valid signature similarly to BLAZE. This approach however circumvents the attack discussed in Section 3.3.2 because multiple rejection samplings are performed when the user unblinds. The number of such samplings can be set to a sufficiently high value, which guarantees that at least one of them will lead to a successful result, thus preventing the user from claiming otherwise. In addition to the cryptographic ingredients required in the 3-move variant of BLAZE+, this variant also makes use of a statistically hiding and computationally binding commitment function $\mathsf{com} : \{0,1\}^* \times \{0,1\}^\lambda \to \{0,1\}^\lambda$.

Parameter Generation. Algorithm $\mathsf{PG}(1^\lambda)$ is identical to the 3-move variant of BLAZE+.

Key Generation. Algorithm $\mathsf{KG}(\mathsf{par})$ is identical to the 3-move variant of BLAZE+.

Signing. The interactive signing protocol $\langle \mathsf{S}(\mathsf{sk}, \mathsf{seed}), \mathsf{U}(\mathsf{pk}, \mathsf{seed}, msg) \rangle$ works as follows:

1. **Signer:** On input secret key sk, as well as seed, the signer expands the seed $\hat{\mathbf{a}}' := \mathsf{Expand}(\mathsf{seed})$ and sets $\hat{\mathbf{a}} := (1, \hat{\mathbf{a}}')$. It samples masking parameters $\hat{\mathbf{r}}_1^*, \ldots, \hat{\mathbf{r}}_\kappa^* \leftarrow_\$ D_{\mathbb{Z}^n, s^*}^{m+1}$ and computes commitments $\mathbf{R}_j := \hat{\mathbf{a}} \cdot \hat{\mathbf{r}}_j^* \pmod q, \forall j \in [\kappa]$. It sets $\hat{\mathbf{R}} := (\mathbf{R}_1, \ldots, \mathbf{R}_\kappa)$ and sends $\hat{\mathbf{R}}$ to the user.

2. **User:** On input seed, and a message $msg \in \{0,1\}^*$, the user also expands the seed $\hat{\mathbf{a}}' := \mathsf{Expand}(\mathsf{seed})$ and sets $\hat{\mathbf{a}} := (1, \hat{\mathbf{a}}')$. It samples signed permutations $p_1, \ldots, p_\kappa \leftarrow_\$ \mathbb{S}$, random coins $r, r', \rho, \rho' \leftarrow_\$ \{0,1\}^\lambda$, and masking terms $(\hat{\mathbf{e}}_0, \ldots, \hat{\mathbf{e}}_{l-1}) \leftarrow_\rho D_{\mathbb{Z}^n, s}^{m+1}$ (notice that the random coins are fixed through ρ). It commits to the message-to-be-signed msg and the randomness ρ' by computing $C_1 := \mathsf{com}(msg, r)$ and $C_2 := \mathsf{com}(\rho', r')$, respectively. It computes $\mathbf{y} := \sum_{j=1}^\kappa p_j \mathbf{R}_j \pmod q$ as well as $\mathbf{t}_j := \hat{\mathbf{a}} \cdot \hat{\mathbf{e}}_j + \mathbf{y} \pmod q, \forall j = 0, \ldots, l-1$. It uses the $\mathbf{t}_j$ elements as leaf nodes to construct a Merkle tree via $(\mathsf{tree}, \mathsf{root}) := \mathsf{HashTree}(\mathbf{t}_0, \ldots, \mathbf{t}_{l-1})$. It computes its challenge $\mathbf{c} := \mathsf{H}(\mathsf{root}, C_1, C_2)$ by passing C_1 and C_2 as inputs to H, and partitions the challenge into monomials c_j (this can trivially be done by writing $\mathbf{c}$ in the form $\mathbf{c} = \sum_{j=1}^\kappa c_j, c_j \in \mathbb{S}$). It blinds the partitioning monomials with the help of signed permutations $p_1, \ldots, p_\kappa$ via $c_j^* := c_j \cdot p_j^{-1}$. It

sets $\mathbf{c}^* := (c_1^*, \ldots, c_\kappa^*)$ as the blinded challenge and transmits $\mathbf{c}^*$ to the signer.

3. **Signer:** On input a blinded challenge $\mathbf{c}^*$, the signer computes its responses $\hat{\mathbf{s}}_j^* := \hat{\mathbf{r}}_j^* + \mathsf{sk} \cdot c_j^*, \forall j \in [\kappa]$. It then invokes Rejection_Sample$((\hat{\mathbf{s}}_1^*, \ldots, \hat{\mathbf{s}}_\kappa^*), (\mathsf{sk} \cdot c_1^*, \ldots, \mathsf{sk} \cdot c_\kappa^*))$ to make its response independent of sk. If rejection sampling fails, it restarts the entire protocol. Otherwise, it sets $\hat{\mathbf{s}}^* := (\hat{\mathbf{s}}_1^*, \ldots, \hat{\mathbf{s}}_\kappa^*)$ and sends $\hat{\mathbf{s}}^*$ to the user.

4. **User:** The user receives $\hat{\mathbf{s}}^*$ and computes $\hat{\mathbf{v}} := \sum_{j=1}^{\kappa} p_j \hat{\mathbf{s}}_j^*$. If $\|\hat{\mathbf{v}}\| > \eta s^* \sqrt{(m+1)\kappa n}$, the protocol is aborted (this occurs with probability $2^{-\lambda}$). The user expands ρ' into l random coins via $(\rho_0, \ldots, \rho_{l-1}) := \mathsf{Expand}(\rho')$. The user then locates the first pair of the form $(\hat{\mathbf{e}}_j, \rho_j), j \in \{0, \ldots, l-1\}$ for which rejection sampling of $\hat{\mathbf{e}}_j + \hat{\mathbf{v}}$ with fixed randomness ρ_j succeeds. Let I be the index of the first successful rejection sampling. The user sets $\hat{\mathbf{s}} := \mathsf{Compress}(\hat{\mathbf{e}}_I + \hat{\mathbf{v}})$ and also invokes BuildAuth(I, tree) to compute an authentication path proving that the I-th element under root was used to compute $\hat{\mathbf{s}}$ and it outputs $(C_2, r, \hat{\mathbf{s}}, \mathbf{c}, \mathsf{auth})$ as its blind signature. On the contrary, if all rejection samplings fail, it sends $(C_1, \rho, \rho', r', p_1, \ldots, p_\kappa, \mathbf{c})$ to the signer, requesting a restart.

5. **Signer:** Upon receiving $(C_1, \rho, \rho', r', p_1, \ldots, p_\kappa, \mathbf{c})$, the signer computes $C_2 := \mathsf{com}(\rho', r')$, $(\rho_0, \ldots, \rho_{l-1}) := \mathsf{Expand}(\rho')$, $(\hat{\mathbf{e}}_0, \ldots, \hat{\mathbf{e}}_{l-1}) \leftarrow_\rho D_{\mathbb{Z}^n, s}^{m+1}$, $\mathbf{y} := \sum_{j=1}^{\kappa} p_j \mathbf{R}_j \pmod q$, $\mathbf{t}_j := \hat{\mathbf{a}} \cdot \hat{\mathbf{e}}_j + \mathbf{y} \pmod q, \forall j = 0, \ldots, l-1$, $(\mathsf{tree}, \mathsf{root}) := \mathsf{HashTree}(\mathbf{t}_0, \ldots, \mathbf{t}_{l-1})$, $\hat{\mathbf{v}} := \sum_{j=1}^{\kappa} p_j \hat{\mathbf{s}}_j^*$. For each $j \in \{0, \ldots, l-1\}$, the signer computes $\mathbf{w} := \hat{\mathbf{a}} \cdot (\hat{\mathbf{e}}_j + \hat{\mathbf{v}}) - \mathbf{bc} \pmod q$ and $\mathsf{auth}_j := \mathsf{BuildAuth}(j, \mathsf{tree})$. It then verifies that all rejection samplings on the user's side failed by checking whether $\mathbf{c} \neq \mathsf{H}(\mathsf{RootCalc}(\mathbf{w}_j, \mathsf{auth}_j), C_1, C_2)$ or Rejection_Sample$(\hat{\mathbf{e}}_j + \hat{\mathbf{v}}, \rho_j) = 1, \forall j = 0, \ldots, l-1$, and if that is the case, it ignores the user's request. Finally, if $\mathbf{c} = \mathsf{H}(\mathsf{root}, C_1, C_2) = \sum_{j=1}^{\kappa} p_j c_j^*$, the signer restarts the entire protocol. Otherwise, it ignores the user's request.

Verification. On input public key pk, message $msg \in \{0,1\}^*$ and a purported signature $(C_2, r, \hat{\mathbf{s}}, \mathbf{c}, \mathsf{auth})$, the verifier expands the seed to obtain $\hat{\mathbf{a}}' := \mathsf{Expand}(\mathsf{seed}), \hat{\mathbf{a}} := (1, \hat{\mathbf{a}}')$ and decompresses the signature via $\hat{\mathbf{s}} := \mathsf{Decompress}(\hat{\mathbf{s}})$. It computes $\mathbf{w} := \hat{\mathbf{a}} \cdot \hat{\mathbf{s}} - \mathbf{bc} \pmod q$ and $\mathsf{root} := \mathsf{RootCalc}(\mathbf{w}, \mathsf{auth})$. If $\|\hat{\mathbf{s}}\| \leq \eta s \sqrt{(m+1)n}$ and $\mathbf{c} = \mathsf{H}(\mathsf{root}, \mathsf{com}(msg, r), C_2)$, it returns 1 (accept). Otherwise, it outputs 0 (reject).

3.3.4 Ermann's Blind Signature Scheme

Ermann et al. [42] propose a 3-move "Schnorr-like" blind signature scheme from lattice assumptions but with a twist in order to achieve *perfect* correctness. The key idea is to trapdoor the SIS function using a technique from [133]. This allows the signer to sample a preimage for its response until the latter satisfies a certain shortness condition. Since the signer can always come up with an appropriately distributed response, it is freed from having to restart the entire protocol. Moreover, the user is also freed from having to rejection sample when it unblinds to obtain its signature.

Construction

The only cryptographic block required for their scheme is a hash function $\mathsf{H} : \{0,1\}^* \rightarrow \mathcal{R}_2$ modelled as a random oracle.

Table 3.4: Parameter definitions for the Ermann et al. blind signature scheme.

Parameter	Definition and Constraints
λ	main security parameter
n	integer power of 2
m	number of polynomials in the secret key s.t. $m := \lfloor \log(q) \rfloor + 1$
α	$\omega(k\sqrt{\log(n)})$
β	$2^{\omega(\log(n))}\sigma\sqrt{n}$
γ	$n\alpha$
τ	standard deviation of the distribution from which the secret key is drawn
σ	$\omega((n\sqrt{n\alpha})\sqrt{\log(n)})$
D	$t\sqrt{nm}(\beta+\sigma)$
q	prime modulus s.t. $q \geq 4mn\sqrt{n}\log(n)$

Parameter Generation. On input the main security parameter λ, algorithm $\mathsf{PG}(1^\lambda)$ selects the scheme's public parameters par according to Table 3.4 and it outputs par.

Key Generation. Let $\hat{\mathbf{g}} = (1, 2, 2^2, \ldots, 2^{k-1})^T \in \mathcal{R}_q^k$, with $k = \lceil \log_2(q) \rceil$ be the *gadget vector*. On input the scheme's parameters par, algorithm $\mathsf{KG}(\mathsf{par})$ samples $\hat{\mathbf{s}} \leftarrow_\$ \mathcal{R}_3^m$, a vector of polynomials $\hat{\mathbf{a}}' \leftarrow_\$ \mathcal{R}_q^{m-k}$, $\mathbf{h} \leftarrow_\$ \mathcal{R}_q$ and a short trapdoor vector $\hat{\mathbf{T}}_{\hat{\mathbf{a}}} \in \mathcal{R}_q^{(m-k)\times k}$ from a discrete Gaussian distribution with variance τ. It computes a vector of polynomials $\hat{\mathbf{a}} := (\hat{\mathbf{a}}'^T, \mathbf{h}\hat{\mathbf{g}} - \hat{\mathbf{a}}'^T\hat{\mathbf{T}}_{\hat{\mathbf{a}}})^T$. Let $\mathsf{F}_{\hat{\mathbf{a}}} : \mathcal{R}_q^m \rightarrow \mathcal{R}_q$ be the SIS function defined by $\hat{\mathbf{a}}$

(i.e., $\mathsf{F}_{\hat{\mathbf{a}}}(\hat{\mathbf{x}}) := \hat{\mathbf{a}} \cdot \hat{\mathbf{x}}$). The algorithm computes $\mathbf{S} := \mathsf{F}_{\hat{\mathbf{a}}}(\hat{\mathbf{s}})$, it sets $\mathsf{sk} := (\hat{\mathbf{s}}, \hat{\mathbf{T}}_{\hat{\mathbf{a}}})$, $\mathsf{pk} := (\mathbf{S}, \hat{\mathbf{a}})$, and outputs $(\mathsf{sk}, \mathsf{pk})$.

Signing. The interactive signing protocol $\langle \mathsf{S}(\mathsf{sk}), \mathsf{U}(\mathsf{pk}, msg) \rangle$ works as follows:

1. **Signer:** The signer samples a masking vector $\hat{\mathbf{r}} \leftarrow_{\$} D^m_{\mathcal{R},\sigma}$ and computes a commitment $\mathbf{R} := \mathsf{F}_{\hat{\mathbf{a}}}(\hat{\mathbf{r}})$. It sends $\mathbf{R}$ to the user.

2. **User:** The user receives $\mathbf{R}$ and samples masking parameters $\mathbf{t}_1 \leftarrow_{\$} D_{\mathcal{R},\alpha}$, $\hat{\mathbf{t}}_2 \leftarrow_{\$} D^m_{\mathcal{R},\beta}$. If $\|\hat{\mathbf{t}}_2\| > t\beta\sqrt{mn}$, it resamples $\hat{\mathbf{t}}_2$. It computes its challenge via $\mathbf{c} := \mathsf{H}(\mathbf{R} - \mathbf{t}_1 \cdot \mathbf{S} - \mathsf{F}_{\hat{\mathbf{a}}}(\mathbf{t}_2), msg)$ and blinds it by computing $\mathbf{c}^* := \mathbf{c} - \mathbf{t}_1$. With probability $\min(1, \frac{D_{\mathcal{R},\alpha}}{G_1 \cdot D_{\mathcal{R},\alpha,\mathbf{c}}})$, the user sends $\mathbf{c}^*$ to the signer. Otherwise, it repeats the entire step from scratch.

3. **Signer:** The signer receives $\mathbf{c}^*$ and computes its response $\hat{\mathbf{z}}^* := \mathbf{c}^* \cdot \hat{\mathbf{s}} + \hat{\mathbf{r}}$. With probability $\min(1, \frac{D^m_{\mathcal{R},\sigma}}{G_2 \cdot D^m_{\mathcal{R},\sigma,\mathbf{c}^* \cdot \hat{\mathbf{s}}}})$, $\hat{\mathbf{z}}^*$ will be used by the signer. With probability $1 - \min(1, \frac{D^m_{\mathcal{R},\sigma}}{G_2 \cdot D^m_{\mathcal{R},\sigma,\mathbf{c}^* \cdot \hat{\mathbf{s}}}})$, the signer uses $\hat{\mathbf{T}}_{\hat{\mathbf{a}}}$ to sample a preimage of $\mathbf{c}^* \cdot \mathbf{S} + \mathbf{R}$ via $\hat{\mathbf{z}}^* \leftarrow \mathsf{PreSample}(\hat{\mathbf{T}}_{\hat{\mathbf{a}}}, \mathbf{c}^* \cdot \mathbf{S} + \mathbf{R}, \sigma)$. If $\|\hat{\mathbf{z}}^*\| > t\sigma\sqrt{mn}$, it samples a fresh preimage for $\mathbf{c}^* \cdot \mathbf{S} + \mathbf{R}$. Once this condition is satisfied, it sends $\hat{\mathbf{z}}^*$ to the user.

4. **User:** The user unblinds the signer's response by simply computing $\hat{\mathbf{z}} := \hat{\mathbf{z}}^* - \hat{\mathbf{t}}_2$. It outputs $(\mathbf{c}, \hat{\mathbf{z}})$ as its blind signature.

Verification. On input public key pk, a purported signature $(\hat{\mathbf{z}}, \mathbf{c})$ and message $msg \in \{0,1\}^*$, algorithm $\mathsf{Ver}(\mathsf{pk}, (\hat{\mathbf{z}}, \mathbf{c}), msg)$ outputs 1 iff $\|\hat{\mathbf{z}}\| \leq t(\beta + \sigma)\sqrt{mn}$ and $\mathsf{H}(\mathsf{F}_{\hat{\mathbf{a}}}(\hat{\mathbf{z}}) - \mathbf{S} \cdot \mathbf{c}, msg) = \mathbf{c}$. Otherwise, it outputs 0.

Claimed Security Results

Thanks to the use of a trapdoor, the signer is always able to respond to the signer, without having to abort the interactive protocol. This also results in freeing the user from having to rejection sample when unblinding:

Lemma 3.3.10. *(Theorem 2 in [42]) The construction of [42] has perfect correctness.*

Even though no commitment scheme is used, blindness is shown to hold in a statistical sense:

Lemma 3.3.11. *(Theorem 3 in [42]) The construction of [42] is statistically blind.*

Finally, unforgeability is based on the hardness of the (Ring) $k - \mathsf{SIS}$ problem:

Lemma 3.3.12. *(Theorem 4 in [42]) If Ring $k - \mathsf{SIS}_{q,m,D}$ is hard for the parameters set according to Table 3.4, then construction of [42] is one-more unforgeable.*

3.3.5 The Forking Lemma and Other Flawed Constructions

The recent work of [98] states a very subtle flaw that is shared by all of the constructions presented so far and which stems from an incorrect application of the general Forking Lemma [32]. The key strategy employed in their unforgeability proofs is to *rewind* the adversary with a partially changed random oracle so as to obtain two *distinct* values χ and χ' s.t. $\mathsf{F}(\chi) = \mathsf{F}(\chi')$, where F is the (Ring) SIS hash function family. The value $\chi - \chi'$ is then a non-trivial solution for the underlying hard lattice problem (i.e., $\mathsf{R} - \mathsf{SIS}$). In order to prove that $\chi - \chi'$ is non-trivial, they try to apply an argument similar to Lemma 8 from [155], and state that non-triviality is a direct consequence of the scheme's witness indistinguishability. This is incorrect because Lemma 8 of [155] only implies that there exist two distinct secret keys sk, sk', leading to identical protocol transcripts. This argument however *does not* suffice for claiming non-triviality and applying the general Forking Lemma.

Zhu et al. [188] follow a completely different approach based on the hardness of the closest vector problem (CVP) in order to design round-optimal (i.e., with only 2-moves) blind signature schemes from lattice assumptions. Their construction however has been shown to be flawed in [57] (also see [21] for an attack).

Another line of papers [119, 185, 187, 82] attempt to construct "RSA-style" blind signature schemes (and variants thereof). All of them however are vulnerable to an attack described in [21] (also see [159] for a discussion).

Remark 3. (from [159]) It is impossible to construct *secure* RSA-style [53, 54, 55] blind signatures from lattice assumptions (i.e., following the pattern: hash $\rightarrow$ blind $\rightarrow$ invert $\rightarrow$ unblind) because then the scheme becomes vulnerable to an attack described in [159].

Finally, [143] (which we present in Chapter 4) attempts to construct partially-blind signature schemes (i.e., schemes in which both the signer and the user share a common public value) from lattice assumptions. This work also has a flawed security proof due to the incorrect application of the Forking Lemma mentioned above. We are currently unaware of any provably secure partially-blind signature scheme from lattice assumptions.

3.4 Overview of Provably Secure Lattice-Based BSS

In this section, we review all provably secure proposals of blind signature schemes from lattice assumptions. There currently exist two approaches for rendering blind signatures from lattices. The first by [98] adapts the paradigm of the Okamoto-Schnorr blind signature [139] from the discrete logarithm setting to the lattice setting and expands on the techniques introduced in [21, 19] to obtain a very modular (generic) construction. The second one followed by Agrawal et al. [12] is based on the observation that homomorphic encryption can be used to achieve round-optimality for blind signatures in the lattice setting [84].

3.4.1 Hauck et al.'s Blind Signature Scheme

The recent work of [98] proposes for the first time a correct and modular treatment for rendering canonical (i.e., 3-move) blind signature schemes from *any* linear hash function family displaying some form of correctness error. Their construction is secure in the ROM even against forgers that are allowed to perform concurrent protocol executions. The key idea is to have the signer commit to multiple masking terms during its first move, then have the user also pick multiple masking parameters and to store (a function of) each such combination as a leaf of a Merkle tree. This approach guarantees that for sufficiently many sampled blinding parameters, the probability of aborting during rejection-sampling (on either side) in the signing protocol will be negligible. The user can use an authentication path in order to prove that a particular trial under the Merkle tree's root was indeed used for producing its signature. While

Table 3.5: Parameter definitions for the Hauck et al. blind signature scheme.

Parameter	Definition and Constraints
λ	main security parameter
n	integer power of 2
m	number of polynomials in the secret key s.t. $m > \log(q)/\log(2*d)$
q	prime modulus s.t. $q \geq 4dmn\sqrt{n}\log(n)$
ι	number of irreducible factors of x^n+1 modulo q s.t. $q \equiv 2\iota+1 \pmod{4\iota}$
δ	bound for the infinity norm of torsion-free elements from the kernel of F s.t. $(\delta+1)^{mn} > q^n$
d_{sk}	bound for the infinity norm of secret keys
$\mathcal{D}_{sk}$	$\{\mathbf{v} \in \mathcal{R}_q : \|\mathbf{v}\|_\infty \leq d_{sk}\}$
$d_{c'}$	bound for the infinity norm of the outputs of H
$S_{c'}$	the challenge space $\{\mathbf{v} \in \mathcal{R}_q : \|\mathbf{v}\|_\infty \leq d_{c'}\}$
u, v, w	positive integer constants for controlling the accept-reject ratio of rejection sampling
μ	number of blinding parameters $\mathbf{b}_j$ picked by the user
η	number of commitments $\mathbf{R}_i$ sent by the signer
ν	number of blinding parameters $\hat{\mathbf{a}}_k$ picked by the user
d_b	bound for the infinity norm of blinding parameters $\mathbf{b}_j$ s.t. $d_b := ud_{c'}n$
S_b	$\{\mathbf{v} \in \mathcal{R}_q : \|\mathbf{v}\|_\infty \leq d_b\}$
d_c	bound for the infinity norm of blinded challenges $\mathbf{c}$ s.t. $d_c < \frac{1}{2\sqrt{\iota}}q^{1/\iota}$
S_c	$\{\mathbf{v} \in \mathcal{R}_q : \|\mathbf{v}\|_\infty \leq d_c\}$
d_r	bound for the infinity norm of blinding parameters $\hat{\mathbf{r}}_i$ s.t. $d_r \geq vmn^2 d_{sk} d_c$
$\mathcal{D}_r$	$\{\mathbf{v} \in \mathcal{R}_q : \|\mathbf{v}\|_\infty \leq d_r\}$
d_s	bound for the infinity norm of signer responses $\hat{\mathbf{s}}$ s.t. $d_s := d_r n d_{sk} d_c$
$\mathcal{D}_s$	$\{\mathbf{v} \in \mathcal{R}_q : \|\mathbf{v}\|_\infty \leq d_s\}$
d_a	bound for the infinity norm of blinding parameters $\hat{\mathbf{a}}_k$ s.t. $d_a := wd_s nm$
$\mathcal{D}_a$	$\{\mathbf{v} \in \mathcal{R}_q : \|\mathbf{v}\|_\infty \leq d_a\}$
$d_{s'}$	bound for the infinity norm of unblinded responses $\hat{\mathbf{s}}'$ s.t. $d_{s'} := d_a - d_s$
$\mathcal{D}_{s'}$	$\{\mathbf{v} \in \mathcal{R}_q : \|\mathbf{v}\|_\infty \leq d_{s'}\}$
d	$d := 2d_{s'}$, $d < \frac{1}{2}\min\{q, 2^{2\sqrt{n\log(q)\log(\delta)}}(n\log(q)/\log(\delta))^{-1/4}\}$
$\mathcal{D}$	$\{\mathbf{v} \in \mathcal{R}_q : \|\mathbf{v}\|_\infty \leq d\}$

the construction in [98] is generic, below we focus on its instantiation from the Ring SIS linear hash function family.

Construction

Let $2\mathsf{Int} : [\eta] \times [\nu] \times [\mu] \to [\eta\nu\mu]$ be the mapping $(i, j, k) \mapsto i + \eta \cdot (j-1) + \eta\nu \cdot (k-1)$, s.t. $2\mathsf{Int}(1,1,1) = 1$ and $2\mathsf{Int}(\eta, \nu, \mu) = \eta\nu\mu$. The main building blocks for the construction of [98] are:

- a hash function $\mathsf{H} : \{0,1\}^* \to S_{c'} := \{\mathbf{c}' \in \mathcal{R}_q : \|\mathbf{c}'\|_\infty \leq d_{c'}\}$, modeled as a random oracle,
- a collision-free and chain-free hash function $\mathsf{G} : \{0,1\}^* \to \{0,1\}^{2\lambda}$.

Parameter Generation. On input the security parameter λ, algorithm $\mathsf{PG}(1^\lambda)$ selects the scheme's public parameters par according to Table 3.5 and it outputs par.

Key Generation. On input the scheme's parameters par, algorithm $\mathsf{KG}(\mathsf{par})$ samples a secret key $\hat{\mathbf{z}} \leftarrow_\$ \mathcal{D}_{sk}^m$. It sets $\mathsf{sk} := \hat{\mathbf{z}}$ and computes the public key via $\mathsf{pk} := \mathsf{F}(\mathsf{sk})$ and outputs $(\mathsf{sk}, \mathsf{pk})$.

Signing. The interactive signing protocol $\langle \mathsf{S}(\mathsf{sk}), \mathsf{U}(\mathsf{pk}, msg) \rangle$ works as follows:

1. **Signer:** The signer picks masking parameters $\hat{\mathbf{r}}_i \leftarrow_\$ \mathcal{D}_r^m, \forall i \in [\eta]$ and computes $\mathbf{R}_i := \mathsf{F}(\hat{\mathbf{r}}_i), \forall i \in [\eta]$. It sends commitment $\hat{\mathbf{R}} := (\mathbf{R}_1, \ldots, \mathbf{R}_\eta)$ to the user.

2. **User:** The user receives $\hat{\mathbf{R}} = (\mathbf{R}_1, \ldots, \mathbf{R}_\eta)$ and picks its own masking parameters $\hat{\mathbf{a}}_1, \ldots, \hat{\mathbf{a}}_\nu \leftarrow_\$ \mathcal{D}_a^m$, $\mathbf{b}_1, \ldots, \mathbf{b}_\mu \leftarrow_\$ S_b$, and $\gamma \leftarrow_\$ \mathbb{Z}_\eta$. It computes all possible combinations of masked commitments $\mathbf{R}'_{i\oplus\gamma,j,k} := \mathbf{R}_i + \mathsf{F}(\hat{\mathbf{a}}_k) + \mathbf{b}_j \cdot \mathsf{pk}$ for all $(i, j, k) \in [\eta] \times [\mu] \times [\nu]$. It constructs a Merkle tree with the masked commitments as leaves via $(\mathsf{tree}, \mathsf{root}) := \mathsf{HashTree}(\mathbf{R}'_{1,1,1}, \ldots, \mathbf{R}'_{\eta,\mu,\nu})$. It then computes its challenge as $\mathbf{c}' := \mathsf{H}(\mathsf{root}, msg)$ and proceeds to locate the first masking polynomial $\mathbf{b}_j$ for which rejection-sampling succeeds. If there exists an index $j \in [\mu]$ s.t. $\mathbf{c}' + \mathbf{b}_j \in S_c$, it sets $\mathbf{c} := \mathbf{c}' + \mathbf{b}_j$ and sends $\mathbf{c}$ to the user. Otherwise, (i.e., if no such index is found), it sends $\perp$ to the signer, indicating failure.

3. **Signer:** The signer receives $\mathbf{c} \in S_c$ and computes its response with the help of the secret key. In particular, it locates the first $i \in [\eta]$ s.t. $\hat{\mathbf{s}} := \hat{\mathbf{r}}_i + \mathbf{c} \cdot \mathsf{sk}$ falls within the set $\mathcal{D}_s^m$. The signer sends $\hat{\mathbf{s}}$ to the user. Otherwise, if no masking term $\hat{\mathbf{r}}_i$ satisfies this criterion, it sends $\perp$ to the user.

4. **User:** The user receives $\hat{\mathbf{s}} \in \mathcal{D}_s^m$ and locates the first index $i \in [\eta]$ s.t. $\mathsf{F}(\hat{\mathbf{s}}) = \mathbf{c} \cdot \mathsf{pk} + \mathbf{R}_i$ (if no such index is found, it outputs $\perp$). It then finds the first index $k \in [\nu]$ for which the corresponding blinding term $\hat{\mathbf{a}}_k$ causes $\hat{\mathbf{s}}' := \hat{\mathbf{s}} + \hat{\mathbf{a}}$ to fall within $\mathcal{D}_{s'}^m$ (if no such index is found, it outputs $\perp$). The user recalls the index $j \in [\mu]$ that it used during Step 2 and computes an authentication path $\mathsf{auth} := \mathsf{BuildAuth}(2\mathsf{Int}(i \oplus \gamma, j, k), \mathsf{tree})$ for the specific combination under root that was used. It outputs $(\hat{\mathbf{s}}', \mathbf{c}', \mathsf{auth})$ as its blind signature.

Verification. On input the public key pk, a message $msg \in \{0,1\}^*$ and a purported signature of the form $(\mathbf{c}', \hat{\mathbf{s}}', \mathsf{auth})$, the verifier computes $\mathbf{R}' := \mathsf{F}(\hat{\mathbf{s}}') - \mathbf{c}' \cdot \mathsf{pk}$ and $\mathsf{root} := \mathsf{RootCalc}(\mathbf{R}', \mathsf{auth})$. If $\mathbf{c}' = \mathsf{H}(\mathsf{root}, msg)$ and $\hat{\mathbf{s}}' \in \mathcal{D}_{s'}^m$, it outputs 1. Otherwise, it outputs 0.

Proved Security Results

The main advantage of using a Merkle tree is that both parties can sample multiple masking parameters but only use the combination which causes all rejection samplings on their respective side of the protocol to succeed. This leads to the following result about correctness:

Lemma 3.4.1. *(Adapted from Lemma 3 of [98]) The construction of [98] has a correctness error of* $(1 - e^{-1/u} + o(1))(1 - e^{-1/v} + o(1))(1 - e^{-1/w} + o(1))$.

By avoiding the use of a commitment scheme, the construction of [98] achieves statistical blindness. If the underlying hash function family F has sufficient min-entropy, the construction also seems to attain perfect blindness.

Lemma 3.4.2. *(Adapted from Theorem 2 of [98]) Let* $\mathsf{LHF} = (\mathsf{PGen}, \mathsf{F})$ *denote the* $\mathsf{R-SIS}$ *hash function family. For parameters set according to Table 3.5, the construction of [98] is perfectly blind relative to all* $\mathsf{par} \in \mathsf{PGen}(1^\kappa)$.

Finally, unforgeability is proven in two steps: (i) by reducing collision-resistance of the underlying LHF to one-more-man-in-the-middle security (OMMIM) [97] for an intermediate identification scheme (cf. Section 5.2 of [98]), and (ii) reducing OMMIM security of the identification scheme to OMUF security of the blind signature scheme. This leads to the following (informal) result:

Lemma 3.4.3. *(Adapted from Theorem 1 of [98]) Let* $\mathsf{LHF} = (\mathsf{PGen}, \mathsf{F})$ *denote the* $\mathsf{R-SIS}$ *hash function family. If* LHF *is collision-resistant relative to* $\mathsf{par} \in \mathsf{PGen}(1^\kappa)$*, then the construction of [98] is one-more unforgeable relative to* par *in the ROM.*

Remark 4. We observe that although blindness and unforgeability are proved relative to *fixed* $\mathsf{par} \in \mathsf{PGen}(1^\kappa)$, it is possible to extend the proof for $\mathsf{par} \leftarrow_\$ \mathsf{PGen}(1^\kappa)$ using techniques from [97].

Discussion on Hauck et al.'s Blind Signature Scheme

The scheme by Hauck et al. is an adaptation of the Okamoto-Schnorr blind signature [139] from the discrete logarithm regime to the lattice regime. This unfortunately incurs a significant loss in advantage larger than $2^{Q_S}/|S_{c'}|$ for the reduction, where Q_S denotes the maximum number of signatures that the signer can issue before needing to replace its key. This loss is due to the forger's ability to perform concurrent protocol executions with the signer which significantly limits Q_S in practice to a polylogarithmic amount. For example, for a security level of 128 bits, the number of signatures that can be issued before needing to change the public key is $\log(128) = 7$. Furthermore, blindness is only proven in the weaker honest signer model.

An important remark about the unforgeability proof of [98] is that due to the multiple trials involved in each rejection sampling, the protocol will, with overwhelming probability not abort. This allows the construction of [98] to remain consistent with the standard notion of unforgeability which is only meaningful for blind signature schemes with (at most) negligible correctness error. If the scheme can abort with noticeable probability, then even honest adversaries may not be able to produce even l valid signatures after l signing sessions. However, it still has to come up with $l + 1$ signatures in order to win in the unforgeability game. This leads to a significant weakening of the definition. In Chapter 5, we revisit the notion of unforgeability and propose a more general definition which allows the user to revoke sessions that did not yield a valid signature for him.

3.4.2 Agrawal et al.'s Blind Signature Scheme

The state-of-the-art lattice-based proposal by Agrawal et al. [12] follows a completely different design approach for rendering blind signature schemes in the ROM. This is accomplished by simplifying the standard model, 2-move construction by [84] which results in a round optimal, very efficient and at the same time simple scheme. Furthermore, unlike [98], their constructions do not limit the signer to issuing a polylogarithmic amount of signatures.

A Rejection-free variant of Lyubashevsky's Digital Signature Scheme

A crucial component of [12] is a rejection-free variant of Lyubashevsky's digital signature scheme [123], which we recall below. Removing the rejection sampling step is important in order to be able to express the signing algorithm as a relatively simple circuit. The main building blocks are:

- a hash function $\mathsf{H} : \{0,1\}^* \to \{\mathbf{v} \in \{0,\pm 1\}^k : \|\mathbf{v}\|_1 \leq \alpha\}$, modeled as a random oracle,
- a pseudorandom function family (PRF) $F : \{0,1\}^r \times \{0,1\}^* \to \{0,1\}^r$ (used for derandomizing the signing algorithm).

Key Generation. On input the security parameter 1^λ, algorithm $\mathsf{Sig.KeyGen}(1^\lambda)$ samples the key of PRF F as $\mathsf{k}_{\mathsf{prf}} \leftarrow_\$ \{0,1\}^r$ as well as matrices $\mathbf{A} \leftarrow_\$ \mathbb{Z}_q^{n \times m}$ and $\mathbf{S} \leftarrow_\$ \{-d, \ldots, 0, \ldots, d\}^{m \times k}$. It computes $\mathbf{T} := \mathbf{AS}$, and sets $\mathsf{vk} := (\mathbf{A}, \mathbf{T})$ as the (public) verification key and $\mathsf{sk} := (\mathsf{k}_{\mathsf{prf}}, \mathbf{S})$ as the (private) signing key. It outputs $(\mathsf{vk}, \mathsf{sk})$.

Signing. On input the signing key sk and a message $msg \in \{0,1\}^*$, algorithm $\mathsf{Sig.Sign}(\mathsf{sk}, msg)$ generates message-specific randomness $rnd = F(\mathsf{k}_{\mathsf{prf}}, msg)$. It samples $\mathbf{y} \leftarrow_{rnd} D_\sigma^m$ using fixed randomness rnd. It sets $\mathbf{c} := \mathsf{H}(\mathbf{Ay}, msg)$ and $\mathbf{z} := \mathbf{y} + \mathbf{Sc}$ and it outputs $(\mathbf{z}, \mathbf{c})$ as the signature on message msg.

Verification. On input the verification key vk, message $msg \in \{0,1\}^*$, and a purported signature $(\mathbf{z}, \mathbf{c})$, algorithm $\mathsf{Sig.Verify}(\mathsf{vk}, msg, (\mathbf{z}, \mathbf{c}))$ checks if $\|\mathbf{z}\| \leq (\sigma + \alpha d)\sqrt{m}$ and $\mathsf{H}(\mathbf{Az} - \mathbf{Tc}, msg) = \mathbf{c}$. If both conditions are true, it outputs 1 (accept). Otherwise, it outputs 0 (reject).

Construction

We now describe the blind signature construction of [12]. In the following, let $\mathbf{G}$ denote the *gadget matrix*, i.e., $\mathbf{G} := (1, 2, 2^2, \ldots, 2^{\lceil \log(q) \rceil - 1})^T \otimes \mathbf{I}_n$ for given parameters q, n, where $\mathbf{I}_n$ denotes the $n \times n$ identity matrix and $\otimes$ denotes the tensor product of matrices. The blind signature scheme requires the following cryptographic blocks:

- a hash function $\mathsf{H} : \{0,1\}^* \to C$ modeled as a random oracle,

- a *circuit-private* homomorphic encryption (HE) scheme $\mathsf{HE} = (\mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{He.Dec}, \mathsf{He.Eval})$,
- a signature scheme $\mathsf{Sig} = (\mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$ that can be used within an HE scheme (instantiated with the rejection-free variant presented above).
- zero-knowledge proofs of knowledge (ZKPoK) for *exact* linear relations with small coefficients, i.e., for the existence of a vector $\mathbf{v}$ such that $\mathbf{v}$ has low Euclidean norm and $\mathbf{Av} = \mathbf{b} \pmod q$ for some public $(\mathbf{A}, \mathbf{b}, q)$.

Key Generation. On input the security parameter λ, algorithm $\mathsf{KG}(1^\lambda)$ generates the keys by invoking the digital signature's key generation algorithm $(\mathsf{Sig.sk}, \mathsf{Sig.vk}) \leftarrow_\$ \mathsf{Sig.KeyGen}(\mathsf{Sig.sk}, \mathsf{Sig.vk})$. It outputs $(\mathsf{sk}, \mathsf{pk}) := (\mathsf{Sig.sk}, \mathsf{Sig.vk})$.

Signing. The interactive signing protocol $\langle \mathsf{S}(\mathsf{sk}), \mathsf{U}(\mathsf{pk}, msg) \rangle$, where $msg \in \{0,1\}$, works as follows:

1. **User:**
   - The user samples $\bar{\mathbf{s}} \leftarrow_\$ \mathbb{Z}_q^{n-1}, \mathbf{e} \leftarrow_\$ D_{\mathbb{Z}^m,\alpha}$ and computes $\mathbf{A} = \mathsf{H}(\mathsf{pk}, \mathsf{id})$ using a user identifier id, $\mathbf{s} := (-\bar{\mathbf{s}}, 1) \in \mathbb{Z}_q^n$ and $\hat{\mathbf{A}} := \begin{pmatrix} \mathbf{A} \\ \bar{\mathbf{s}}^T\mathbf{A} + \mathbf{e}^T \end{pmatrix}$. It sets $\mathsf{HE.SK} := \mathbf{s}$ and $\mathsf{HE.PK} := \hat{\mathbf{A}}$.
   - The user encrypts msg using $\mathsf{HE.PK}$. This is done by sampling a matrix $\mathbf{R} \leftarrow_\$ \{0, \pm 1\}^{m \times m}$, and then computing $\mathbf{C} := \hat{\mathbf{A}}\mathbf{R} + b\mathbf{G} \in \mathbb{Z}_q^{m \times n}$. It sets $\mathsf{CT}_{msg} := \mathbf{C}$ as the ciphertext. Notice that the last column of $\mathbf{s}^T\mathbf{C} = \mathbf{e}^T\mathbf{R} + b\mathbf{s}^T\mathbf{G}$ is close to $b\frac{q}{2}$.
   - The user generates ZKPoK π_{SK} and π_{CT} proving that its public key and ciphertext are well-formed. In particular, π_{SK} proves knowledge of a short vector $(\mathbf{x}, \mathbf{y})$ s.t. the last row of $\hat{\mathbf{A}}$ has the form $\mathbf{x}^T\mathbf{A} + \mathbf{y}$, while π_{CT} proves that $\mathsf{CT}_{msg} = \hat{\mathbf{A}}\mathbf{R} + b\mathbf{G}$, for a low-norm matrix $\mathbf{R}$ and $b \in \{0,1\}$.
   - It sends $(\mathsf{HE.PK}, \pi_{\mathsf{SK}}, \mathsf{CT}_{msg}, \pi_{\mathsf{CT}})$ to the signer.
2. **Signer:**

- The signer receives $(\mathsf{HE.PK}, \pi_{\mathsf{SK}}, \mathsf{CT}_{msg}, \pi_{\mathsf{CT}})$ and it verifies the validity of both π_{SK} and π_{CT}. If either proof is invalid, it outputs $\perp$.
- It homomorphically evaluates the digital signature's signing algorithm (viewed as a circuit) on the encrypted message by computing $\mathsf{CT}_\sigma := \mathsf{HE.Eval}(\mathsf{Sig.Sign}_{\mathsf{sk}}, \mathsf{CT}_{msg})$.
- It sends CT_σ back to the user.

3. **User:** The user decrypts ciphertext CT_σ using its secret key HE.SK. This is done by computing the inner product of $\mathbf{s}^T$ and the last column of CT_σ. If the norm of the result is smaller than $q/4$, it outputs 0. Otherwise, it outputs 1.

Verification. Verification is performed identically to Sig.Verify.

Proved Security Results

Under the assumption that the ZKPoK systems underlying the scheme's instantiation are correct, the scheme of [12] is also correct:

Lemma 3.4.4. *([12]) If the ZKPoK proof systems π_{SK} and π_{CT} are correct,* HE.Eval *and* HE.Dec *are correct, and* Sig.Verify *are correct, then the construction of [12] is also correct.*

Furthermore, the zero-knowledge property implies blindness for the scheme:

Lemma 3.4.5. *(Theorem 6.3 in [12]) If the underlying proof systems are ZKPoK, then the construction of [12] is blind against honest-signers.*

Remark 5. The construction of [12] can be upgraded to remain secure even against malicious signers. This is done by:

1. Having the signer append a proof π_{vk} (constructed with ZKPoK) to its verification key, proving that Sig.vk is well-formed.
2. Using a homomorphic signature scheme with context hiding security to authenticate that algorithm Sig.Sign was homomorphically evaluated.

Finally, unforgeability is derived from the UF-CMA security of the rejection-free variant of Lyubashevsky's signature.

Lemma 3.4.6. *(Theorem 6.4 in [12]) If the underlying digital signature scheme is secure against universal forgery under chosen-message attack, then the construction of [12] is one-more unforgeable.*

Discussion on Agrawal et al.'s Blind Signature Scheme

What is notable about the construction of [12] is its reliance on very heavy machinery such as Non-Interactive Zero-Knowledge (NIZK) proofs and homomorphic encryption schemes that are capable of evaluating a random oracle homomorphically for the communication between the two parties. In particular, while this construction shows great promise for practical use (cf. Section 3.6), there are several serious roadblocks barring implementation. First, by design, the signing algorithm $\mathsf{Sig.Sign}_{\mathsf{Sig.sk}}$ acts as a circuit on which the ciphertext must be homomorphically evaluated by the signer. However, $\mathsf{Sig.Sign}_{\mathsf{Sig.sk}}$ internally uses a hash function modeled as a random oracle and as such, this hash function must also be evaluated *homomorphically*. Unfortunately, choosing such a hash function is a highly non-trivial matter which to the best of our knowledge is still a major open problem. Second, the homomorphic encryption scheme itself needs to be chosen carefully and in a way capable of handling the diverse formats involved during the homomorphic signing. Ensuring compatibility between the homomorphic encryption format and the format needed for evaluating the hash function seems particularly tricky to ensure when instantiating Sig.Sign with their proposed abort-free variant of Dilithium-G's signing algorithm[5]. Finally, Dilithium-G compresses the signature elements with Huffman codes. However, in [12] this will need to be done under the homomorphic encryption layer, which could be expensive to implement. Addressing all of these issues simultaneously is particularly tricky and could lead to insecure implementations. These factors suggest that the scheme of [12] is mostly of theoretical interest.

[5]A Fisher-Yates shuffle no longer works for mapping to Dilithium-G's challenge space because this task must be done *homomorphically*.

3.5 Relations to Impossibility Results

An important consideration to take into account when designing blind signature schemes are impossibility results. These typically state that under certain conditions, reductions to underlying cryptographic problems do not provide a meaningful security statement. In other words, if such a reduction exists, then the underlying problem is already easy. In this section, we review these results and how they impact the lattice-based constructions covered in this chapter.

The first result that we examine is proven by Katz et al. [108]. The authors prove that blind signature schemes are impossible to construct from one-way permutations, even in the ROM.

Theorem 3.5.1. *(Theorem 1 from [108]) There is no black-box construction of blind signature schemes from one-way functions.*

While it is currently unknown whether one-way permutations can even be constructed from the LWE problem, *all* of the constructions considered in this paper circumvent this result by relying on the (stronger) collision-resistance property of R – SIS (or Ring k-SIS in the case of [42]).

The second result by Baldimtsi et al. [27] states that Schnorr-type blind signatures are impossible to construct when unforgeability is based on a one-witness hard problem. In the context of lattice-based schemes, this rules out any hopes of designing a blind signature scheme based *solely* on the LWE problem. No constructions exist whose unforgeability relies solely on LWE. All of the constructions considered in this survey avoid this kind of impossibility result by relying on a multi-witness problem (namely, SIS or Ring k – SIS). Interestingly, an earlier version of BLAZE [18] managed to avoid this kind of impossibility result by taking a "hybrid" approach akin to [69]. In particular, it bases key-secrecy on the hardness of LWE (i.e., a one-witness problem), and unforgeability on SIS (i.e., a multi-witness problem). Unfortunately, [18] then attempts to simulate the signer without the secret key which exposes it to universal forgeability [155].

The final result by Fischlin and Schröder [78] provides a resetting meta-reduction (i.e., a reduction against another reduction) which rules out the possibility of constructing secure schemes with at most three moves, statistical blindness, and which have statistical signature-derivation checks (i.e., one can verify from the protocol transcript between a signer and an honest user if the user was able to obtain a valid signature from the interaction). While this immediately rules out any blind signature schemes in the standard model with at most 3 moves, it does not capture direct constructions of lattice-based blind signatures proven secure in the ROM. In general, the impossibility results of [78] are typically circumvented by using complexity leveraging [168, 70] or by using a common reference string [75].

Theorem 3.5.2. *(Theorem 2 from [78]) Let* BSS *be a three-move blind signature scheme, which is statistically blind and has statistical signature-derivation checks. Then there is no resetting (with restricted cross-resets) black-box reduction from unforgeability of the blind signature scheme* BSS *to a hard non-interactive problem.*

All of the constructions considered in this paper circumvent this result by using a programmable random oracle. Table 3.6 summarizes the above observations.

Table 3.6: Summary of all lattice-based blind signature schemes in the literature and their adherence to impossibility results.

Proposal	Baldimtsi and Lysyanskaya [27]	Katz et al. [108]	Fischlin and Schröder [78]	Incorrect use of the Forking Lemma
Rückert [159]	-	-	-	✓
BLAZE [21, 19]	-	-	-	✓
3-move BLAZE+ [19]	-	-	-	✓
4-move BLAZE+ [19]	-	-	-	✓
Ermann et al. [42]	-	-	-	✓
Hauck et al. [98]	-	-	-	-
Agrawal et al. [12]	-	-	-	-

3.6 Comparison With Other Post-Quantum Proposals

In this section, we review the concrete sizes of keys, produced signatures and total communication for each of the schemes discussed in Sections 3.3 and 3.4. In addition, we compare them to other post-quantum proposals in the literature. Table 3.7

summarizes the aforementioned sizes per scheme, also listing the underlying (post-quantum) hardness assumption per scheme, as well as the estimated bit security level.

Table 3.7: Summary of post-quantum blind signature schemes in the literature. We denote unspecified sizes with a dash.

Proposal	Hardness assumption(s)	Bit security	sk size	pk size	Signature size	Total communication
Flawed Lattice-Based Blind Signature Schemes						
Rückert [159]	R – SIS	102	23.6 KB	23.6 KB	89.4 KB	119.1 KB
BLAZE [21, 19]	R – SIS	$\approx$ 128	0.8 KB	3.9 KB	6.6 KB	351.6 KB
3-move BLAZE+ [19]	R – SIS	$\approx$ 128	0.75 KB	3.9 KB	6.7 KB	177.8 KB
4-move BLAZE+ [19]	R – SIS	$\approx$ 128	0.75 KB	3.9 KB	6.7 KB	$\approx$ 265 KB
Ermann et al. [42]	Ring k – SIS	100	5.86 MB	852 KB	868 KB	-
Provably secure Lattice-Based Blind Signature Schemes						
Hauck et al. [98]	R – SIS	128	4.15 MB	0.4 MB	7.73 MB	33.3 MB
Agrawal et al. [12]	R – LWE & R – SIS	**128**	-	$\leq$ **2 KB**	$\leq$ **3 KB**	-
Other Post-Quantum Blind Signature Schemes						
Petzoldt et al. [151]	Rainbow	128	70.2 KB	106.8 KB	28.5 KB	-
Blazy et al. [33]	CFS & Syndrome Decoding	100	-	15 KB	200 KB	-

In terms of feasibility, the construction of Agrawal et al. [12] seems to have a clear advantage across the board over all other constructions (both lattice-based and non-lattice-based), while at the same time being practical. Unfortunately, as discussed in Section 3.4.2 there are several serious issues that need to be addressed before reaching an implementation. On the other hand, the proposal of Hauck et al. [98] modularizes the methodology that its predecessors [159, 21, 19] unsuccessfully tried to apply, and relies on *very simple* cryptographic components to build. Unfortunately, it is ill-suited for practical use due to its key, signature and communication sizes being in the order of megabytes. A major bottleneck in the communication traffic of [98] is caused by the fact that the signer has to send $\eta = 60$ commitments during its first move. For the concrete parameters proposed in [98], this corresponds to 26 MB which is approximately 78% of the overall traffic exchanged between signer and user. Moreover, each signer is only limited to issuing up to 7 signatures per public key in order to prevent potential ROS attacks [164], which *severely* limits its practical use. While this latter issue could potentially be lifted through a transform proposed in [107], the resulting scheme's relevant sizes would still be in the order of megabytes. Adapting the transform of [107] for lattice-based schemes may however

be of independent interest.

3.7 Conclusions, Open Problems and Future Work

While blind signatures from standard number-theoretic assumptions have been under scrutiny for over three decades, lattice-based schemes have received attention only during the past decade. In addition, constructing BSS from lattice assumptions has proven to be an incredibly arduous task, as evidenced by the fact that all of the proposed constructions in the literature either have an incorrect proof of security, are too inefficient for practical use, or need to overcome major instantiation and/or implementation issues. Another important thing to note is that all constructions are proven secure in the ROM. However, in a true quantum setting, the adversarial forger can query the random oracle with quantum state (i.e., in superposition), which may result in him getting some information about all (exponentially many) values right at the start [38]. Hence, an interesting open question is if it is possible to construct BSS from lattice assumptions in the QROM, or even to completely do away with the RO.

As we mentioned in Section 3.2.2, the existing notion of unforgeability is only meaningful for BSS with at most negligible correctness error. This creates a definitional gap between unforgeability and schemes like [159] which display a noticeable correctness error. In addition, schemes like [159] provide the user with a means of "revoking" a session if he was unable to obtain a valid signature from a session. It is therefore interesting to design a framework for BSS (akin to the one from [97]) which encompasses four-move constructions like [159] and [21]. This direction is explored in Chapter 5.

Finally, the construction of [98] seems like a good starting point for investigating the possibility of rendering variants of BSS from lattice assumptions. In particular, partially-blind signature schemes [6, 7] and fair blind signature schemes [173] seem to be within reach.

Chapter 4

Leakage–Resilient Partially–Blind Signatures from Lattices

4.1 Introduction

Blind signature schemes separate the owner of a message from the signer by allowing the owner of the message to interact with the signer and obtain a signature on it that remains unintelligible from the signer's view. The resulting signature can still be verified against the signer's public key, just like with typical digital signatures. However, nobody – including the signer himself – can link a message-signature pair to a signing transcript. As one would suspect though, such a high level of privacy has some grave drawbacks. First, by design, blind signatures provide perfect confidentiality for the receiving user with regards to the message being signed. As a result, blind signatures can potentially provide a gateway for committing "perfect" crimes [181] such as money laundering, blackmailing, etc. Second, blind signing provides no guarantees to the signer that the blinded message he signed, is of the right "format" or contains some valid information that should be included in the message (e.g.: the denomination of a digital coin, the date a voucher was issued, etc.). Moreover, given that the only attributes over which the signer has control are those bound to his public key, we might end up in a case where multiple keys need to be managed, resulting to an increased complexity for both the signer and verifiers (which is even more prob-

lematic if devices with constrained memory (e.g., smart-cards) are being used [6]). Consider for example a signer that issues blind signatures which expire at the end of the week, then the signer's public key needs to be updated every week, or consider the case of e-cash with multiple denominations: the signer/bank will need to use a different public key for each allowable coin denomination. These major shortcomings of blind signatures spurred the research community to invent primitives with features that could bypass these issues.

The two major models that have been proposed, in an effort to overcome these issues are: *fair blind signature schemes (FBSS)* [173, 80] and *partially-blind signature schemes (PBSS)* [6, 7]. Fair blind signatures allow a trusted third party to revoke blindness in order to identify either the session during which a given signature was issued (session tracing), or a signature, given a signer's view of a specific session (signature tracing). On the other hand, partially-blind signatures allow a signer and a user to include a commonly agreed upon piece of information (denoted info) to the signature. The key idea for achieving this in [7] was to adapt a method proposed in [60] by letting the signer use a secret key, along with two public keys, one of which includes info, with the help of a public hash function. As a result, the final signature is bound to these public keys and thus, to info as well. This approach has the benefit of greatly simplifying key management, because the signer only needs a single key in order to be able to include any auxiliary information (i.e. expiration date or denomination value). Note that PBSS do not immediately solve the problem of whether the blinded message to be signed is of the right format (this problem would be solved generically by including a zero-knowledge proof of knowledge on the format of the message), however, they provide an efficient way to make sure that the info part of the message included the necessary to application information and is of the right format. We would also like to mention that the more recent work of [161] proposed a unified security model called *fair partially blind signatures* (FPBSS), which combines the security models of both aforementioned primitives into a single. Building a construction in that model would be ideal for real-world applications, balancing the individual needs of customers (blindness), service providers (partial control), and

authorities (fairness), and is currently an open problem.

4.1.1 Contributions and Related Work

A previous attempt to construct partially blind signatures from lattices was made in [177]. However, the construction of [177] does not prove partial blindness concretely and in fact seems to prove something weaker than the required notion as it relies on qualitative (if not ambiguous) properties of the signer that *cannot* be captured by the security model of PBSSs. Furthermore, its scope is more limited compared to our proposal because it allows disclosures of the signed message which are acceptable in some applications (e-cash) but unacceptable in others (e-voting, e-auctions). Finally, the scheme of [177] is vulnerable to side-channel attacks, because of the use of discrete Gaussian sampling for the blind signing step [95, 149, 72].

We propose the first leakage-resilient, lattice-based partially-blind signature scheme in the literature. Our construction is inspired by the work of [159] which at the time of publication was the best known leakage-resilient BSS based on lattices. However, being a regular BSS, it is subject to the limitations discussed above. Our approach represents a significant step forward for partially blind signature schemes because:

- First, because the vast majority of previous PBSS proposals, are based on number theoretic assumptions, such as the hardness of large integer factorization, or the computation of discrete logarithms. These include early work by [6, 7, 5] and later advances [58, 140, 118, 35, 34, 87]. Unfortunately, the security of these schemes would be in jeopardy should a reasonable scale quantum computer be constructed, thanks to Shor's algorithm [169]. Consequently, all of these constructions are ill-suited for the post-quantum era.

- Second, although a tremendous amount of progress has been made in the design of conventional digital signatures from lattices over the past decade [86, 121, 122, 123, 96, 66, 22, 124, 69, 111], there is a serious relative dearth when it comes to lattice-based *blind* signatures [159, 188] (the latter of which has recently been shown to be problematic [57]) despite their importance for privacy-preserving applications.

- Regarding efficiency, our construction is as efficient as the lattice-based blind signature scheme in [159], both in terms of key sizes (ours are slightly smaller) and in computational complexity. However, our construction is not only one step closer to practical applications by allowing the inclusion of a commonly agreed piece of information in the final signature, but also relies on a milder - by a factor of n (the security parameter) - hardness assumption for the underlying worst-case lattice problem. This is important because one has to rely on as mild assumptions as possible in anticipation of attacks arising from emerging technologies. We show that all of the extensions considered in [159] are also satisfied by our scheme, along with an additional extension discussed in Section 4.5.3. In that case, we show that the efficient transformation that was proposed in [167] can also be used for PBSS, which we believe might be a result of interest on its own when designing such schemes.

4.1.2 Our technique and main challenges

Extending [159] to a PBSS was conjectured to be possible in [160]. However, no suggestions as to how this could be realized were given, and the problem was not formally addressed until now as it apparently involved several technical challenges. As per the security model of PBSS [7], we need to show that our scheme is correct, partially blind, and unforgeable. Unfortunately, lattices lack the algebraic structure that is present in (finite) cyclic groups, and which very naturally allows one to achieve partial blindness by simply computing the product/sum of any group element with a *random* group element. This problem can be rectified through rejection sampling [159, 122], which allows us to make the distributions of exchanged messages, independent of the respective messages that they "hide". However, this comes at the price of added complexity. Reducing this complexity is by no means trivial: being able to avoid/simplify rejection sampling would in turn impact many other lattice-based constructions such as [122, 66, 69, 159]). The complexity introduced by rejection sampling makes all of the aforementioned security properties (as well as the extensions that we consider) non-trivial to achieve *simultaneously* because they are interconnected to one another.

In particular:

- Correctness is hindered, meaning that even if both parties involved in the signature issuing protocol are honest, the protocol may need to be restarted. We address this issue in the same way as [159]. However, since it is possible for the signature issuing protocol to restart, it is important to make sure that both partial blindness and unforgeability hold, even across restarts.

- Regarding partial blindness, PBSS are built by combining the framework of [7] with witness-indistinguishable identification protocols. For this work, we will use (a slight variant of) the witness-indistinguishable identification scheme of [122] as a basis. However, due to the aforementioned rejection sampling strategy, it is not possible to apply the transformation of [7] in a straightforward manner. This is due to the fact that rejection sampling causes the coefficients of a blinded message to come from a larger set (roughly by a factor of at least n) than the original message's, whenever applied. This turns out to be problematic when we want to "unblind" to produce the final signature. We address this issue by having the user send a "shrinked" version of the blinded challenge to the signer (i.e., reducing it modulo the range of the challenge space's coefficients - typically, modulo 3), by carefully setting our scheme's multiple interconnected parameters, and analyzing the distributions of messages exchanged between the two parties. Our scheme is shown to be partially blind and an important implication of our approach is obtaining a milder by n hardness assumption for our scheme's unforgeability property. In addition, we employ a statistically hiding commitment scheme to make sure that partial blindness is preserved across protocol restarts.

- Proving unforgeability is also non-trivial because a malicious user might falsely claim that he failed to obtain a valid signature out of a protocol execution, thus causing the protocol to abort and potentially "buying" himself multiple valid signatures (this scenario would obviously be catastrophic for applications like e-cash or e-voting). We address this issue by introducing a fourth move to our

signature issuing protocol, which serves as a special proof of failure in case the protocol has to be restarted and is akin to [159]. As in [159], we need to show that a malicious user cannot obtain a valid signature out of an aborted protocol execution, unless he is able to solve a computationally hard lattice problem. However, as we will see in Section 4.4.3, this is considerably trickier to achieve compared to [159] because in our PBSS setting there are multiple scenarios which may cause the protocol's restart (in [159] there is only one). Nevertheless, our construction's security will be formally proven in the ROM [31] under standard worst-case lattice problems pertaining to ideals [125].

- Finally, with respect to leakage resilience, we will show that if we impose an additional requirement on the size of one of our scheme's parameters, then it is also resistant against key-leakage via arbitrary side-channels.

4.1.3 Relationship between the present work and impossibility results for blind signature schemes

In [78], the authors give an impossibility result for 3-move BSS with the help of a meta-reduction (i.e., a reduction between reductions). Their approach plays the two security requirements of BSS (blindness and unforgeability) against each other, resulting in a proof that finding black-box reductions from unforgeability to non-interactive problems (like RSA, or discrete logarithm) is hard, unless the problems involved were already easy. Their work covers a broad class of BSS in the literature [5, 53, 155] and subsumes many prior impossibility results for BSS [47, 120, 48]. However, the main result of [78] does not apply to our construction. First, the results of [78] are given for BSS rather than PBSS which means that one would first have to show that a corresponding result also holds for PBSS. Second, [78] does not rule out reductions in the ROM [27, p. 3]. Third and most importantly, [78] only applies to BSS with *at most three* moves, that admit *statistical signature-derivation checks* (i.e., an observer can determine only from the public data and messages exchanged between a malicious signer and an honest user, whether the user successfully obtained a valid signature

or not). In our 4-move scheme however, it is impossible for one to tell whether the user truly obtained a valid signature or not within 3 moves because the user has not revealed all of the relevant information (i.e., its blinding parameters) that he uses to produce his final signature. This is important because the components of the final signature must satisfy a certain relation but also fall within certain bounded domains for the signature to be deemed valid. This originates from our rejection sampling strategy and is in sharp contrast to previous number theoretic BSS (and PBSS), where all of the final signature's components would always fall within some finite group (e.g., $\mathbb{Z}_N$ in the case of [53]), and thus checks like these would trivially be true due to finite group arithmetic rules. This is in accordance with an observation made by [78], stating that if the user sends a second message to the signer, which depends on his first message, then the resetting strategy of their meta-reduction cannot be applied (see [76] for a discussion on the limitations of their strategy). The same argument can also be used for [159]. Additionally, the fairly more recent results of [27] also do not apply to our work. The reason is that the results of that paper only concern schemes with a *unique-witness* relation between the public and secret key. While many constructions like the original Schnorr BSS fall under that category, our construction relies on a *many-to-one* witness relation between its public and secret keys (see Lemma 5 in Section 4.4.3). Finally, the impossibility result of [108] is circumvented by the fact that our underlying hash function family is collision resistant and not simply one-way.

4.1.4 Organization

Section 4.2 sets the required theoretical and notational groundwork and we describe the formal security model of leakage-resilient PBSS, along with extensions based on the BSS literature. In Section 4.3, we consider for the first time, extensions to the basic security model of PBSS, based on their BSS counterparts. Various results from the BSS literature are shown to still hold in the PBSS context. In Section 4.4, we give a detailed description of our lattice-based construction and prove that it abides by the formal security model of PBSS, and that it is leakage resilient. Once we have established the baseline security of our scheme, Section 4.5 examines additional security

properties for our proposal.

In this chapter, we reprint the main construction in [143], of which the dissertation author was the main investigator and author.

4.2 Preliminaries

Throughout this chapter, n will be used to denote the main security parameter.

4.2.1 Syntax and Security Model of Leakage–Resilient PBSS

We now define leakage-resilient partially-blind signature schemes and their security model.

Syntax and Security Model

Partially-blind Signature Schemes (PBSS) are a generalization of regular blind signature schemes (BSS) [53, 104, 155] and a simplification of fair partially-blind signature schemes (FPBSS) [161]. The security model for PBSS was formalized in [7, 140].

Definition 4.2.1. (Partially-Blind Signature Schemes) A PBSS is comprised by three algorithms $(\mathsf{KG}, \mathsf{Sign} = \langle \mathsf{S}, \mathsf{U} \rangle, \mathsf{Ver})$, where Sign is an interactive protocol executed between S and U. Their specification is the following:

- **Key Generation.** On input the security parameter λ, algorithm $\mathsf{KG}(1^\lambda)$ outputs a private signing key sk and a corresponding public verification key pk.
- **Signature Issuing Protocol.** Protocol $\mathsf{Sign}(\mathsf{sk}, \mu, \mathsf{info})$ jointly executes algorithms $\mathsf{S}(\mathsf{sk}, \mathsf{info})$ and $\mathsf{U}(\mathsf{pk}, \mu, \mathsf{info})$ in an interactive manner. The signer's private output is a view V consisting of all messages exchanged between the parties, and the user's private output is a signature σ on message μ and the common information info under sk. The common information info is agreed upon by the signer and the user prior to the protocol's execution and is assumed to be a common input to both parties. We also assume that the protocol generates a status message like "ok" or $\perp$ for the signer, denoting success or failure, respectively.

- **Signature Verification.** Algorithm $\mathsf{Ver}(\mathsf{pk}, \mu, \mathsf{info}, \sigma)$ returns 1 iff σ is a valid signature on message μ w.r.t. common information info under public key pk, and 0 otherwise.

In the interactive protocol, signer views can be interpreted as random variables and we will consider two views V_1 and V_2 "equal" if no computationally unbounded algorithm A exists that distinguishes them with non-negligible probability. Notice that by fixing info to the empty string, we obtain the usual definition of blind signature schemes [155].

A partially-blind signature scheme needs to satisfy three properties: correctness, partial blindness, and unforgeability [7, 140, 161].

Correctness for PBSS is defined as in regular digital signatures, i.e., if both the signer and the user comply with the signature issuing protocol, then the user successfully obtains a valid signature with overwhelming probability. More formally:

Definition 4.2.2. (Correctness of PBSS) A PBSS is *correct* with *correctness error* $\varepsilon \in [0,1]$, if for all uniformly picked messages $\mu \in \{0,1\}^*$, all uniformly picked common-part information $\mathsf{info} \in \{0,1\}^*$, all honestly generated keys $(\mathsf{pk}, \mathsf{sk})$, and any honestly generated signature σ through the signature-issuing protocol, σ is a valid signature with probability:

$$\Pr\left[\sigma \neq \perp \wedge \mathsf{PBSS.Ver}(\mathsf{pk}, \mu, \mathsf{info}, \sigma) = 1 \;\middle|\; \begin{array}{l} \mu, \mathsf{info} \leftarrow_{\$} \{0,1\}^*, \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{PBSS.KG}(1^\lambda), \\ \sigma \leftarrow \langle \mathsf{S}(\mathsf{sk}, \mathsf{info}), \mathsf{U}(\mathsf{pk}, \mu, \mathsf{info}) \rangle \end{array}\right] \geq 1 - \varepsilon$$

Partial blindness generalizes the notion of *blindness*[1] [104, 155], and informally requires that it is infeasible for a malicious signer to link any valid signature to the exact instance/session of the signature-issuing protocol in which it was created. A formal definition is given by means of game $\mathbf{PBlind}_{\mathsf{PBSS}}(1^\lambda)$ in Figure 4.1 [7].

Definition 4.2.3. (Partial Blindness) A PBSS is (t, θ)-*partially blind* if for any PPT algorithm S^* (working in modes find, issue, and guess), running in time at most t, we

[1] In the literature, some authors also use the term *unlinkability* to describe this property (e.g.: [166]).

Game $\mathbf{PBlind}_{\mathsf{PBSS}}(1^\lambda)$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{PBSS.KG}(1^\lambda)$

$(\mu_0, \mu_1, \mathsf{info}, \mathsf{state}_{\mathsf{find}}) \leftarrow_\$ \mathsf{S}^*(\mathsf{find}, 1^\lambda)$

$b \leftarrow_\$ \{0,1\}$

$\mathsf{state}_{\mathsf{issue}} \leftarrow_\$ \mathsf{S}^{*\langle \cdot, \mathsf{U}(\mathsf{pk},\mu_b,\mathsf{info})\rangle^1, \langle \cdot, \mathsf{U}(\mathsf{pk},\mu_{1-b},\mathsf{info})\rangle^1}(\mathsf{issue}, \mathsf{state}_{\mathsf{find}})$

$\sigma_b := \mathsf{U}(\mathsf{pk}, \mu_b, \mathsf{info}),\ \sigma_{1-b} := \mathsf{U}(\mathsf{pk}, \mu_{1-b}, \mathsf{info})$

If $(\sigma_0 = \bot \vee \sigma_1 = \bot)$

  $b' \leftarrow_\$ \mathsf{S}^*(\mathsf{guess}, \bot, \bot, \mathsf{state}_{\mathsf{issue}})$

Else

  $b' \leftarrow_\$ \mathsf{S}^*(\mathsf{guess}, \sigma_0, \sigma_1, \mathsf{state}_{\mathsf{issue}})$

return $[\![b' = b]\!]$

Figure 4.1: Security game for partial blindness.

have: $\left|\Pr\left[\mathbf{PBlind}^{\mathsf{S}^*}_{\mathsf{PBSS}}(1^\lambda) = 1\right] - 1/2\right| \leq \theta$.

Notice that the notion of partial blindness closely resembles that of blindness [104], the only difference being that now there is an additional commonly known factor, info, which corresponds to the public part of the message-to-be-signed, that also needs to be taken into account. In the game of Figure 4.1, the malicious signer, S^*, generates his public/secret keys via the scheme's key generation algorithm (we relax this requirement when we discuss dishonest-key partial blindness). He then selects messages μ_0, μ_1 and common information info on his own (mode find). He then interacts with honest users $\mathsf{U}(\mathsf{pk}, \mu_b, \mathsf{info})$ and $\mathsf{U}(\mathsf{pk}, \mu_{1-b}, \mathsf{info})$, after a secret coin flip $b \leftarrow_\$ \{0,1\}$ (mode issue). If either user session aborts before completion, the signer is merely notified of the event, but receives no signature. After seeing the unblinded signatures in the original order, the signer's task is to correctly guess b (mode guess).

We further parameterize matters in Definition 4.2.3. S^*'s advantage is defined as: $\mathbf{Adv}^{\mathbf{PBlind}}_{\mathsf{PBSS}}(\mathsf{S}^*) := \left|\Pr\left[\mathbf{PBlind}^{\mathsf{S}^*}_{\mathsf{PBSS}}(1^\lambda) = 1\right] - 1/2\right|$. We will call PBSS *statistically partially-blind* if it is (∞, θ)–partially-blind for a negligible θ, and *perfectly partially-blind* if θ is 0.

Unforgeability of PBSS is stronger than the one defined for regular blind signatures [104, 155], since "recombination" attacks should be ruled out [161]. Additionally, the adversarial user is allowed to select both the messages and the common information

Game $\mathbf{OMUF}_{\mathsf{PBSS}}(1^\lambda)$
1: $(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{PBSS.KG}(1^\lambda)$
2: $\mathsf{H} \leftarrow_\$ \mathcal{H}(1^\lambda)$
3: $(\mathsf{info}, (\mu_1, \sigma_1), \ldots, (\mu_{k_{\mathsf{info}}+1}, \sigma_{k_{\mathsf{info}}+1})) \leftarrow_\$ \mathsf{U}^{*\mathsf{H}(.), \langle \mathsf{S}(\mathsf{sk}), . \rangle^\infty}(\mathsf{pk})$
4: k_{info} := # successful, complete interactions w.r.t. info.
5: $b_1 := [\![\mu_i \neq \mu_j, \forall i, j = 1, \ldots, k_{\mathsf{info}} + 1 \text{ with } i \neq j]\!]$
6: $b_2 := [\![\mathsf{PBSS.Ver}(\mathsf{pk}, \mu_i, \mathsf{info}, \sigma_i) = 1, \forall i = 1, \ldots, k_{\mathsf{info}} + 1]\!]$
7: return $b_1 \wedge b_2$

Figure 4.2: Security game for unforgeability of PBSS.

info that he queries, in an adaptive manner. Put another way, a malicious user should be unable to generate a valid signature for a new info, instead of just for a new message [161]. The notion of unforgeability of PBSS is defined in terms of the game of Figure 4.2, which we derive from the more general game of [161], where $\mathcal{H}$ denotes a family of random oracles:

Definition 4.2.4. (Unforgeability of PBSS) An interactive partially-blind signature scheme $\mathsf{PBSS} = (\mathsf{KG}, \langle \mathsf{S}, \mathsf{U} \rangle, \mathsf{Ver})$ is $(t, q_{\mathsf{sig}}, q_{\mathsf{H}}, \theta)$–*unforgeable* if for any PPT algorithm U^* running in time t, making at most q_{sig} signature queries and q_{H} hash oracle queries, we have $\Pr\left[\mathbf{OMUF}^{\mathsf{U}^*}_{\mathsf{PBSS}}(1^\lambda) = 1\right] \leq \theta$.

Note that in the unforgeability game of Figure 4.2, the adversarial user outputs $k_{\mathsf{info}} + 1$ valid message-signature pairs that correspond to a single info, where $0 \leq k_{\mathsf{info}} \leq q_{\mathsf{sig}}$ denotes the number of successful, complete interactions that took place. The case where $k_{\mathsf{info}} = 0$ corresponds to the scenario in which the malicious user outputs a signature for a *new* info that was never queried to the signing oracle and is known as a "recombination" attack (not present for regular blind signatures). We can define *strong unforgeability* by changing the condition of line 5 in game $\mathbf{OMUF}_{\mathsf{PBSS}}(1^\lambda)$ to "$b_1 := [\![(\mu_i, \sigma_i) \neq (\mu_j, \sigma_j), \forall i, j = 1, \ldots, k_{\mathsf{info}} + 1 \text{ with } i \neq j]\!]$".

Leakage-resilient cryptographic primitives are designed to remain secure even if an arbitrary, but bounded portion of the secret key (and/or other internal state information in general) of an honest party leaks to an adversary during computation. This augmentation of the notion of unforgeability helps safeguard against various forms

of side-channel attacks, such as: timing attacks [95, 149], data remanence attacks, power-monitoring attacks [72], or implementations using poor random number generation. Unfortunately, [95, 149] provide clear evidence that cache timing attacks in particular are a practical threat to post-quantum cryptographic constructions. As a result, proving that a scheme is resistant against key leakage is a very important property if we want to consider long-term security, and constructions possessing it grant us a very high level of confidence when deploying them in practice.

To model leakage resilience in the context of unforgeable PBSS, we refer to [109], and grant the adversarial user access to a leakage oracle, $\mathsf{Leak}(.)$, in the above unforgeability experiment (our scheme satisfies the properties required by [109]). The adversary can adaptively query a series of functions $f_i, i \in \{1, \ldots, \kappa\}$ to this oracle, and receive $f_i(\mathsf{sk}) \in \{0, 1\}$. We consider the signer's secret state to consist solely of his secret key and that his secret key does not change over time. We also consider the same bounded leakage model as in [159]. More precisely, we impose the constraint $\sum_{i=1}^{n} |f_i(\mathsf{sk})| < \lambda(|\mathsf{sk}|)$, where $\lambda = \lambda(.)$ is a function of the length of the secret key, and dictates the amount of tolerable leakage. Of course, this extension only makes sense as long as $\lambda(|\mathsf{sk}|) < \min\{|\mathsf{sk}|, |\sigma|\}$, where $|.|$ denotes bit-length, and σ is a signature. The game modeling leakage resilience for the unforgeability of partially blind signature schemes is defined below:

Definition 4.2.5. (Leakage Resilience of PBSS) An interactive partially blind signature scheme $\mathsf{PBSS} = (\mathsf{KG}, \langle \mathsf{S}, \mathsf{U} \rangle, \mathsf{Ver})$ is *leakage-resilient* with parameter λ, iff for any efficient algorithm U^*, the probability that experiment $\mathbf{LR-OMUF}_{\mathsf{PBSS},\lambda-\mathsf{Leak}}(1^\lambda)$ (Figure 4.3) evaluates to 1 is negligible (as a function of λ).

4.3 Extensions

In this Section, we discuss several extensions of the classic security model of PBSS that are applicable to our construction. We consider honest-user unforgeability, selective-failure blindness, and dishonest-key blindness. To the best of our knowledge, none of these properties have previously been examined in the context of PBSS.

Game LR – OMUF$_{\mathsf{PBSS},\lambda-\mathsf{Leak}}(1^\lambda)$

1: $(\mathsf{pk},\mathsf{sk}) \leftarrow_\$ \mathsf{PBSS.KG}(1^\lambda)$
2: $\mathsf{H} \leftarrow_\$ \mathcal{H}(1^\lambda)$
3: $(\mathsf{info},(\mu_1,\sigma_1),\ldots,(\mu_{k_{\mathsf{info}}+1},\sigma_{k_{\mathsf{info}}+1})) \leftarrow_\$ \mathsf{U}^{*\mathsf{H}(.),\langle \mathsf{S}(\mathsf{sk}),.\rangle^\infty,\mathsf{Leak}(\mathsf{sk},.)}(\mathsf{pk})$
4: k_{info} := # successful, complete interactions w.r.t. info.
5: Let $f_1,\ldots,f_\kappa$ be the leakage queries of U^*, each with output length λ_i.
6: $b_1 := [\![\mu_i \neq \mu_j, \forall i,j = 1,\ldots,k_{\mathsf{info}}+1 \text{ with } i \neq j]\!]$,
7: $b_2 := [\![\mathsf{PBSS.Ver}(\mathsf{pk},\mu_i,\mathsf{info},\sigma_i) = 1, \forall i = 1,\ldots,k_{\mathsf{info}}+1]\!]$
8: $b_3 := [\![\sum_{i=1}^{\kappa} \lambda_i \leq \lambda(|\mathsf{sk}|)]\!]$
6: Return $b_1 \wedge b_2 \wedge b_3$

Figure 4.3: Security game for leakage resilience of PBSS.

4.3.1 Dishonest-key Partial Blindness

In the definition of (partial) blindness (step 1 of Figure 4.1), we implicitly assumed that the signer generates his secret and public keys through the scheme's key generation algorithm. This however may not necessarily be true. Abdalla et al. put forth the notion of *dishonest-key blindness*[2] [4] to capture scenarios in which the malicious S^* is allowed to construct his public key in a special way that would give him an edge in breaking the blindness property of a blind signature scheme. This type of attack has already been considered in [140], where it was also incorporated into the security model of partially-blind signature schemes. As such, steps 1 and 2 of $\mathbf{PBlind}_{\mathsf{PBSS}}(1^\lambda)$ in Figure 4.1 are replaced with "$(\mathsf{pk},\mu_0,\mu_1,\mathsf{info},\mathsf{state}_{\mathsf{find}}) \leftarrow_\$ \mathsf{S}^*(\mathsf{find},1^\lambda)$". Achieving partial blindness in this setting is harder but not impossible.

4.3.2 Selective-failure Partial Blindness

Another type of attack not captured by the blindness property of blind signature schemes is that in which the adversarial signer can pick messages of its own choosing from some secret distribution, in hopes of causing one of the user instances with which it interacts to abort, thereby gaining an advantage in linking the messages with the produced signatures. Motivated by this observation, the notion of *selective-failure blindness* was proposed by [49] and informally states that no malicious signer S^* can

[2] We interchangeably refer to this model as the *malicious signer model*.

cause the protocol's execution with an honest user U to abort through adversely-chosen messages μ_0, μ_1. We generalize this notion to account for the common piece of information info.

Definition 4.3.1. (Selective-Failure Partial Blindness) A $\mathsf{PBSS} = (\mathsf{KG}, \langle \mathsf{S}, \mathsf{U} \rangle, \mathsf{Ver})$ is called (t, θ)–selective-failure partially blind if it is unforgeable (as per $\mathbf{OMUF}_{\mathsf{PBSS}}(1^\lambda)$) and for any PPT algorithm S^* (working in modes find, issue and guess), running in time at most t, we have:

$$\left| \Pr\left[\mathbf{SF-PBlind}^{\mathsf{S}^*}_{\mathsf{PBSS}}(1^\lambda) = 1 \right] - 1/2 \right| \leq \theta,$$

where $\mathbf{SF-PBlind}_{\mathsf{PBSS}}(1^\lambda)$ is defined in Figure 4.4.

Game $\mathbf{SF-PBlind}_{\mathsf{PBSS}}(1^\lambda)$

1: $(\mathsf{pk}, \mu_0, \mu_1, \mathsf{info}, \mathsf{state}_{\mathsf{find}}) \leftarrow_{\$} \mathsf{S}^*(\mathsf{find}, 1^\lambda)$
2: $b \leftarrow_{\$} \{0, 1\}$
3: $\mathsf{state}_{\mathsf{issue}} \leftarrow_{\$} \mathsf{S}^{*\langle \cdot, \mathsf{U}(\mathsf{pk}, \mu_b, \mathsf{info}) \rangle^1, \langle \cdot, \mathsf{U}(\mathsf{pk}, \mu_{1-b}, \mathsf{info}) \rangle^1}(\mathsf{issue}, \mathsf{state}_{\mathsf{find}})$
4: $\sigma_b := \mathsf{U}(\mathsf{pk}, \mu_b, \mathsf{info})$, $\sigma_{1-b} := \mathsf{U}(\mathsf{pk}, \mu_{1-b}, \mathsf{info})$
5: Define answer as: first if only the first execution has failed,
second if only the second execution has failed,
both if both executions have failed,
(σ_b, σ_{1-b}) otherwise.
6: $b' \leftarrow_{\$} \mathsf{S}^*(\mathsf{guess}, \mathsf{answer}, \mathsf{state}_{\mathsf{issue}})$
7: return $[\![b' = b]\!]$

Figure 4.4: Security game for selective-failure partial blindness.

The following two results from [77] naturally generalize to the PBSS setting. They state that selective-failure partial blindness is a strictly stronger notion than partial blindness.

Lemma 4.3.1. (Generalization of Proposition 1 from [77]) *Every selective-failure partially-blind signature scheme* $\mathsf{SF-PBSS}$ *is also a secure partially blind signature scheme.*

Lemma 4.3.2. (Generalization of Proposition 2 from [77]) *If there exists a secure partially-blind signature scheme* PBSS*, then there exists a secure partially-blind signature scheme* PBSS' *which is not selective-failure partially-blind.*

Any partially blind signature scheme PBSS can be converted into a selective-failure partially blind signature scheme SF – PBSS as follows:

Construction 4.3.1. (Generalization of Construction 1 from [77]) Let $\mathsf{PBSS} = (\mathsf{KG}, \langle \mathsf{S}, \mathsf{U} \rangle, \mathsf{Ver})$ be a partially blind signature scheme and let $C(1^\lambda)$ be a commitment scheme. We define a new signature scheme $\mathsf{SF-PBSS} = (\mathsf{KG}', \langle \mathsf{S}', \mathsf{U}' \rangle, \mathsf{Ver}')$ as follows:

Key Generation. $\mathsf{KG}'(1^\lambda)$ runs $(\mathsf{sk}, \mathsf{pk}) \leftarrow_\$ \mathsf{KG}(1^\lambda)$. It also chooses a function $\mathsf{com} \leftarrow_\$ C(1^\lambda)$. The algorithm sets $\mathsf{sk}' := \mathsf{sk}$ and $\mathsf{pk}' := (\mathsf{pk}, \mathsf{com})$, and returns $(\mathsf{sk}', \mathsf{pk}')$.

Signing Protocol. Signer S' is identical to signer S. User U' commits to message μ by computing $C \leftarrow \mathsf{com}(\mu; r)$ for some $r \leftarrow_\$ \{0,1\}^\lambda$ and then invokes the original user $\mathsf{U}(\mathsf{pk}, C, \mathsf{info})$, who interacts with signer $\mathsf{S}(\mathsf{sk}, \mathsf{info})$. When U outputs a signature σ, U' outputs $(\mu, \mathsf{info}, (\sigma, r))$.

Signature Verification. The verification algorithm $\mathsf{Ver}'(\mathsf{pk}', \mu, \mathsf{info}, \sigma')$ parses σ' as (σ, r) and returns the output of $\mathsf{Ver}(\mathsf{pk}, \mu, \mathsf{info}, \mathsf{com}(\mu; r))$.

Theorem 4.3.3. *If* PBSS *is a secure partially-blind signature scheme and* $C(1^\lambda)$ *is a secure commitment scheme, then the scheme* SF – PBSS *defined in Construction 4.3.1 is a selective-failure partially blind signature scheme.*

Multi-execution selective-failure partial blindness further generalizes selective-failure blindness [77] to account for an arbitrary number of executions with user instances. We define *multi-execution selective-failure partial blindness* as a natural generalization of selective failure partial blindness.

Definition 4.3.2. (Multi-execution selective-failure partial blindness) An interactive partially-blind signature scheme $\mathsf{PBSS} = (\mathsf{KG}, \langle \mathsf{S}, \mathsf{U} \rangle, \mathsf{Ver})$ is called (t, θ)–multi-execution selective-failure partially-blind if it is unforgeable (as per $\mathbf{OMUF}_{\mathsf{PBSS}}(1^\lambda)$), and for any PPT algorithm S^* (working in modes find, issue and reveal), running in time t, we have: $\left|\Pr\left[\mathbf{MESF-PBlind}^{\mathsf{S}^*}_{\mathsf{PBSS}}(1^\lambda) = 1\right] - 1/2\right| \leq \theta$, where $\mathbf{MESF-PBlind}_{\mathsf{PBSS}}(1^\lambda)$ is defined in Figure 4.5.

Note that the malicious signer selects his public key, a common-part information info, and messages $\mu_1, \ldots, \mu_k$, where k denotes the number of user instances he will interact

Game $\mathbf{MESF-PBlind}_{\mathsf{PBSS}}(1^\lambda)$

1: $\mathsf{st}^{\mathsf{rev}} \leftarrow (\perp, \ldots, \perp)$
2: $(\mathsf{pk}, \mu_1, \ldots, \mu_k, \mathsf{info}, \mathsf{state}_{\mathsf{find}}) \leftarrow_{\$} \mathsf{S}^*(\mathsf{find}, 1^\lambda)$
3: Select a uniformly random permutation π over $\{1, \ldots, k\}$.
4: $\mathsf{state}_{\mathsf{issue}} \leftarrow_{\$} \mathsf{S}^{*\langle .,\mathsf{U}(\mathsf{pk},\mu_{\pi(1)},\mathsf{info})\rangle^1,\ldots,\langle .,\mathsf{U}(\mathsf{pk},\mu_{\pi(k)},\mathsf{info})\rangle^1,\mathsf{Reveal}(.,\pi,\mathsf{st}^{\mathsf{rev}})}(\mathsf{issue}, \mathsf{state}_{\mathsf{find}})$
$\sigma_{\pi(1)} := \mathsf{U}(\mathsf{pk}, \mu_{\pi(1)}, \mathsf{info}), \ldots, \sigma_{\pi(k)} := \mathsf{U}(\mathsf{pk}, \mu_{\pi(k)}, \mathsf{info})$,
immediately stored in $\mathsf{st}^{\mathsf{rev}}$, once an execution terminates; $\mathsf{Reveal}(., \pi, \mathsf{st}^{\mathsf{rev}})$ is an oracle, which on input an index i, returns $(\pi(i), \mathsf{st}_i^{\mathsf{rev}})$.
5: If $(\sigma_1 = \perp \vee \ldots \vee \sigma_k = \perp)$
6: $\quad \mathbf{v} \leftarrow (\llbracket \sigma_1 \neq \perp \rrbracket, \ldots, \llbracket \sigma_k \neq \perp \rrbracket) \in \{0,1\}^k$
7: Else
8: $\quad \mathbf{v} \leftarrow (\sigma_1, \ldots, \sigma_k)$
9: $(i_0, i_1) \leftarrow_{\$} \mathsf{S}^{*,\mathsf{Reveal}(.,\pi,\mathsf{st}^{\mathsf{rev}})}(\mathsf{reveal}, \mathbf{v}, \mathsf{state}_{\mathsf{issue}})$
10: return 1 if-f $\pi(i_0) < \pi(i_1)$ and S^* never queried Reveal on indices i_0, i_1.

Figure 4.5: Security game for multi-execution selective-failure partial blindness.

with (mode find). We select a random permutation over $\{1, \ldots, k\}$ and give $\mu_{\pi(i)}$ to user i. All users use the same signer public key and info. If at least one user instance aborts during mode issue, we merely notify S^* which user instances aborted and which did not through a binary vector $\mathbf{v} \in \{0,1\}^k$. Otherwise, S^* is given all message-signature pairs in the original order. In mode reveal, the signer's task is to link a message-signature pair to an execution (i.e., find two input indices i_0, i_1 s.t. $\pi(i_0) < \pi(i_1)$). Furthermore, we assume that S^* has access to a Reveal oracle (with internal state $\mathsf{st}^{\mathsf{rev}}$) during modes issue and reveal, which on input i reveals the map $\{i \mapsto (\pi(i), \sigma_i)\}$, provided that $\sigma_i \neq \perp$. The adversary wins if the indices he found were never queried to Reveal. Thus, S^* can reveal up to $k-2$ maps before guessing, which implies the following:

Lemma 4.3.4. (Generalization of Proposition 3 from [77]) *A selective-failure partially blind signature scheme is also multi-execution selective-failure partially-blind.*

4.3.3 Honest-user Unforgeability

In [167], the authors propose a strengthened notion of one-more unforgeability for blind signatures, called *unforgeability in the presence of honest users* (or *honest-user unforgeability*, for short). The idea is that an adversary could exploit the presence of an honest user (modeled as an oracle P in Figure 4.6), and use him as an intermediary to indirectly obtain signatures from the signer (it is not difficult to see that the absence

of such honest users leads to the classic notion of unforgeability of BSS [104, 155]. However, unforgeability is shown to be weaker than honest-user unforgeability [167]). That way, the adversary may be able to produce more signatures than the number of times he directly interacted with the signer. These kinds of attacks are not captured by the notion of unforgeability for regular blind signatures.

Honest-user unforgeability however is given with regular blind signature schemes in mind. Here, we adapt it for partially-blind signature schemes, thus obtaining an even stronger notion of unforgeability for PBSS. We also show that the transformation given in [167] is still relevant when it comes to PBSS, a result which we believe may be of interest in its own right. Before giving the new definition, we must fix some notation. Let $\mathsf{P}(\mathsf{sk}, \mathsf{pk}, ., .)$ be an oracle that on input μ (a message) and common information info, executes the signature issuing protocol $\langle \mathsf{S}, \mathsf{U} \rangle$, thus obtaining a signature σ. Let trans denote the transcript comprised of all messages exchanged between the parties in such an interaction. When the protocol terminates, P returns (σ, trans). The execution of $\langle \mathsf{S}(\mathsf{sk}, \mathsf{info}), \mathsf{U}(\mathsf{pk}, \mu, \mathsf{info}) \rangle$ by P is considered to be atomic, i.e., during a call to P, no other interactions occur. If the interaction aborts, P returns $(\perp, \mathsf{trans})$, where trans is the transcript up to that point of execution.

Definition 4.3.3. An interactive partially-blind signature scheme $\mathsf{PBSS} = (\mathsf{KG}, \langle \mathsf{S}, \mathsf{U} \rangle, \mathsf{Ver})$ is *honest-user unforgeable* if Ver is deterministic, and for any efficient algorithm U^*, we have $\Pr\left[\mathbf{HU-OMUF}^{\mathsf{U}^*}_{\mathsf{PBSS}}(1^\lambda) = 1\right]$ is negligible (as a function of λ), where $\mathbf{HU-OMUF}_{\mathsf{PBSS}}(1^\lambda)$ is defined in Figure 4.7.

Replacing the condition of line 3 with "$(\mu_i^*, \sigma_i^*) \neq (\mu_j, \sigma_j), \forall i = 1, \ldots, k_{\mathsf{info}}$ and $\forall j = 1, \ldots, n_{\mathsf{info}}$", and the condition of line 4 with "$(\mu_i^*, \sigma_i^*) \neq (\mu_j^*, \sigma_j^*), \forall i, j = 1, \ldots, k_{\mathsf{info}}$, with $i \neq j$" we obtain the notion of *strong honest-user unforgeability*.
Note that when counting the interactions in which S returns "ok", we do not count the interactions simulated by P. Also notice that line 3 of Figure 4.7 prevents U^* from winning in the unforgeability game by simply outputting a message-signature pair that an honest user obtained for him.

Running a probabilistic algorithm on the exact same input will most likely result in an entirely different output. We adapt the notion of probabilistic algorithms within

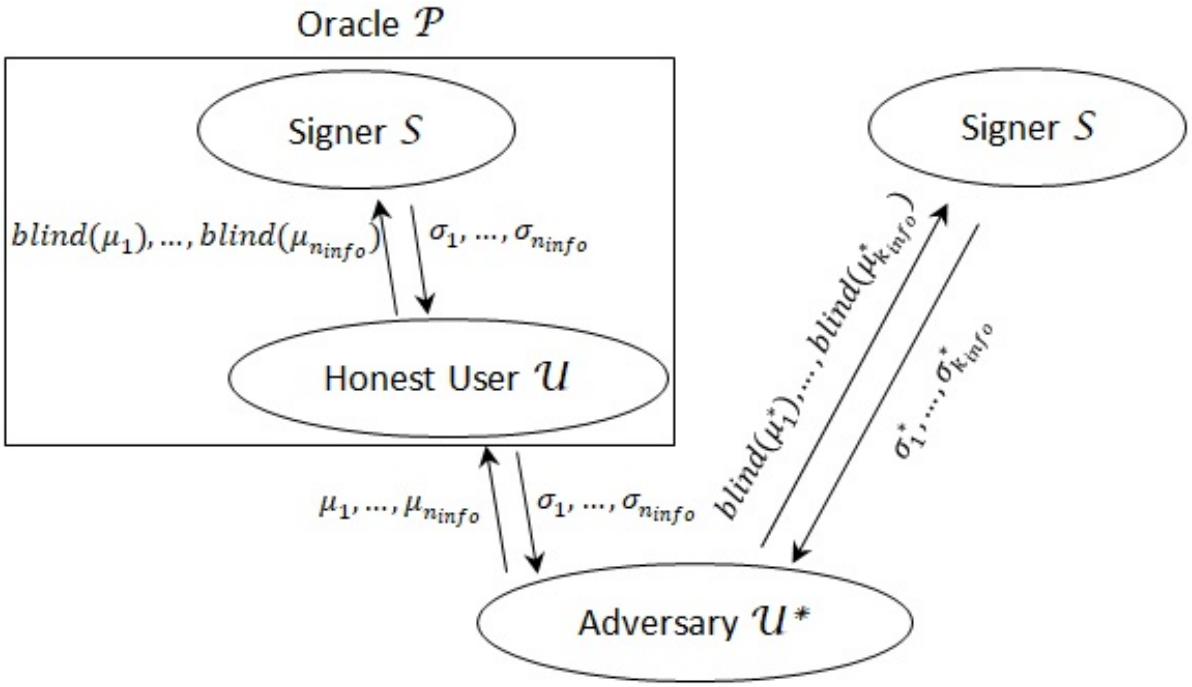

Figure 4.6: Schematic diagram of forgery in the presence of honest users. All queries shown in the figure correspond to a particular info.

Game HU – OMUF$_{\mathsf{PBSS}}(1^\lambda)$
1: $(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{PBSS.KG}(1^\lambda)$
2: $(\mathsf{info}, (\mu^*_1, \sigma^*_1), \ldots, (\mu^*_{k_{\mathsf{info}}+1}, \sigma^*_{k_{\mathsf{info}}+1})) \leftarrow_\$ \mathsf{U}^{*\langle \mathsf{S}(\mathsf{sk}),. \rangle^\infty, \mathsf{P}(\mathsf{sk},\mathsf{pk},.,.)}(\mathsf{pk})$
k_{info} := # complete, successful, direct interactions with S, using common information info.
Let $\mu_1, \ldots, \mu_{n_{\mathsf{info}}}$ be the messages pertaining to info that were queried to P(sk, pk, ., .).
3: $b_1 := [\![\mu^*_i \neq \mu_j, \forall i = 1, \ldots, k_{\mathsf{info}}, \text{ and } \forall j = 1, \ldots, n_{\mathsf{info}}]\!]$
4: $b_2 := [\![\mu^*_i \neq \mu^*_j, \forall i, j = 1, \ldots, k_{\mathsf{info}} + 1 \text{ with } i \neq j]\!]$
5: $b_3 := [\![\mathsf{PBS.Ver}(\mathsf{pk}, \mu^*_i, \mathsf{info}, \sigma^*_i) = 1, \forall i = 1, \ldots, k_{\mathsf{info}} + 1]\!]$
6: return $b_1 \wedge b_2 \wedge b_3$

Figure 4.7: Security game for honest-user unforgeability of PBSS.

the context of PBSS. More formally, we have the following definition:

Definition 4.3.4. (Probabilistic PBSS) An interactive, partially-blind signature scheme PBSS = (KG, ⟨S, U⟩, Ver) is called *probabilistic* if-f:

$$\Pr\left[\sigma_1 = \sigma_2 \;\middle|\; \begin{array}{l} \mu, \mathsf{info} \leftarrow_\$ \{0,1\}^*, (\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{PBSS.KG}(1^\lambda), \\ \sigma_1 \leftarrow_\$ \langle \mathsf{S}(\mathsf{sk}, \mathsf{info}), \mathsf{U}(\mathsf{pk}, \mu, \mathsf{info}) \rangle, \\ \sigma_2 \leftarrow_\$ \langle \mathsf{S}(\mathsf{sk}, \mathsf{info}), \mathsf{U}(\mathsf{pk}, \mu, \mathsf{info}) \rangle \end{array}\right] = negl(\lambda)$$

It is not hard to prove that the following holds:

Lemma 4.3.5. (Generalization of Lemma 10 from [167]) *Any correct, probabilistic, and strongly unforgeable PBSS is also strongly honest-user unforgeable.*

Proof. We prove this Lemma by contradiction. To this end, suppose that PBSS is not strongly honest-user unforgeable. As per the definition of strong honest-user unforgeability, there exists a PPT adversary A breaking the strong unforgeability property of PBSS with noticeable probability. Then, A successfully outputs a single info and $k_{\mathsf{info}}+1$ message-signature pairs: $(\mu_1^*, \sigma_1^*), \ldots, (\mu_{k_{\mathsf{info}}+1}^*, \sigma_{k_{\mathsf{info}}+1}^*)$ for some $k_{\mathsf{info}} \in \mathbb{N}_0$, whereas S only outputs "ok" after at most $k_{\mathsf{info}} + 1$ (direct) interactions with A. Furthermore, the following properties are satisfied:

i) $(\mu_i^*, \sigma_i^*) \neq (\mu_j^*, \sigma_j^*), \forall 1 \leq i < j \leq k_{\mathsf{info}} + 1$

ii) $\mathsf{PBSS.Ver}(\mathsf{pk}, \mu_i^*, \mathsf{info}, \sigma_i^*) = 1, \forall i = 1, \ldots, k_{\mathsf{info}} + 1$

Next, let $\mu_1, \ldots, \mu_{n_{\mathsf{info}}}$ be the messages queried through the honest user U (i.e., indirectly), and let $\sigma_1, \ldots, \sigma_{n_{\mathsf{info}}}$ be the corresponding signatures. Then we also have:

iii) $\{(\mu_i, \sigma_i)\} \cap \{(\mu_j^*, \sigma_j^*)\} = \emptyset, \forall i = 1, \ldots, n_{\mathsf{info}}$ and $\forall j = 1, \ldots, k_{\mathsf{info}} + 1$

iv) $\mathsf{PBSS.Ver}(\mathsf{pk}, \mu_i, \mathsf{info}, \sigma_i) = 1, \forall i = 1, \ldots, n_{\mathsf{info}}$ with overwhelming probability (because PBSS is correct)

v) $(\mu_i, \sigma_i) \neq (\mu_j, \sigma_j), \forall 1 \leq i < j \leq n_{\mathsf{info}}$ with overwhelming probability (because PBSS is probabilistic)

We define the sequence of message-signature pairs:

$$(\tilde{\mu}_i, \tilde{\sigma}_i) := \begin{cases} (\mu_i^*, \sigma_i^*) & \text{, if} \quad 1 \leq i \leq k_{\mathsf{info}} + 1 \\ (\mu_i, \sigma_i) & \text{, if} \quad k_{\mathsf{info}} + 1 < i \leq k_{\mathsf{info}} + n_{\mathsf{info}} + 1 \end{cases}$$

Properties ii) and iv) imply that $\mathsf{PBSS.Ver}(\mathsf{pk}, \tilde{\mu}_i, \mathsf{info}, \tilde{\sigma}_i) = 1, \forall i = 1, \ldots, k_{\mathsf{info}}+n_{\mathsf{info}}+1$. Additionally, properties i), iii) and v) imply that the elements of the above sequence are pairwise distinct.

We now construct a simulator B to attack the strong unforgeability property of PBSS. B runs the adversary A and honest-user U in a black-box manner. When A makes an (indirect) query to U, B runs U and forwards the response to A. When either A or U query the signer, B redirects their queries to an external signer S. When A outputs his forgery, B outputs $(\tilde{\mu}_i, \tilde{\sigma}_i), i = 1, \ldots, k_{\mathsf{info}} + n_{\mathsf{info}} + 1$ (as defined above) as his own forgery in the strong unforgeability game. We previously showed that these are pairwise distinct and S returned "ok" as his status message at most $k_{\mathsf{info}} + n_{\mathsf{info}}$ times. Thus, PBSS is not strongly unforgeable, which contradicts our hypothesis. □

We now present a way to turn *any* unforgeable PBSS into an honest-user unforgeable PBSS, that is analogous to the one from [167]. This transformation comes at the expense of a negligible overhead compared to the original PBSS.

Construction 4.3.2. Let $\mathsf{PBSS}' = (\mathsf{KG}', \langle \mathsf{S}', \mathsf{U}' \rangle, \mathsf{Ver}')$ be an interactive partially blind signature scheme. We define a new partially blind signature scheme $\mathsf{PBSS} = (\mathsf{KG}, \langle \mathsf{S}, \mathsf{U} \rangle, \mathsf{Ver})$ through the following algorithms:

- **Key Generation.** Algorithm $\mathsf{KG}(1^\lambda)$ runs $(\mathsf{sk}', \mathsf{pk}') \leftarrow \mathsf{KG}'(1^\lambda)$ and returns the key pair.
- **Signature Issuing Protocol.** Signer S is identical to the original signer S′. User $\mathsf{U}(\mathsf{pk}, \mu, \mathsf{info})$ chooses $r \leftarrow_\$ \{0,1\}^\lambda$, sets $\mu' \leftarrow \mu \| r$, and then invokes the original user $\mathsf{U}'(\mathsf{pk}, \mu', \mathsf{info})$, who then interacts with $\mathsf{S}'(\mathsf{sk}, \mathsf{info})$. When U′ outputs a signature σ, U computes $\sigma' \leftarrow (\sigma, r)$ and returns σ'.
- **Signature Verification.** Algorithm $\mathsf{Ver}(\mathsf{pk}, \mu, \mathsf{info}, \sigma')$ parses σ' as (σ, r) and returns the result of $\mathsf{Ver}'(\mathsf{pk}, \mu \| r, \mathsf{info}, \sigma)$.

Theorem 4.3.6. *If correct, partially blind, and unforgeable PBSS exist, then there exist PBSS which are complete, partially blind, unforgeable, and also honest-user unforgeable.*

Proof. It is trivial to see that if PBSS′ is correct and partially blind, then so is PBSS. Thus, we only need to show that PBSS is honest-user unforgeable, if PBSS′ is unforgeable. We will prove this by contradiction. Assume that PBSS′ is unforgeable but PBSS

is not honest-user unforgeable. Thus, as per Definition 4.3.3, there exists an efficient adversary U^* that wins in game $\mathbf{HU-OMUF}_{\mathsf{PBSS},\mathsf{U}^*}(1^\lambda)$ with noticeable probability. We will construct an attacker B that breaks the unforgeability of PBSS′:

Setup. Algorithm B receives a public key pk as input and runs U^* in a black-box manner, simulating the oracles as follows:

Direct Signing Queries. If U^* directly invokes the signing oracle S', B simply relays all messages exchanged between the malicious user and the signer.

Indirect Signing Queries. If U^* indirectly invokes S' through oracle P on message, $\mu \in \{0,1\}^*$, and common information $\mathsf{info} \in \{0,1\}^*$, then B chooses a random $r \leftarrow_\$ \{0,1\}^\lambda$, sets $\mu' \leftarrow \mu \| r$, and engages in an interactive PBSS with the signer S', by assuming the role of the honest user U'. When the protocol terminates, B obtains a signature σ on message μ', and common information info. He sets $\sigma' \leftarrow (\sigma, r)$, stores the tuple $(\mu', \mathsf{info}, \sigma')$ in a list L, and outputs σ', along with the corresponding transcript trans to the adversary U^*.

Forgery. Since U^* is efficient, he eventually stops and outputs a single info, and a sequence of message-signature pairs: $(\mu_1^*, \sigma_1^*), \ldots, (\mu_{k_{\mathsf{info}}+1}^*, \sigma_{k_{\mathsf{info}}+1}^*)$. In turn, B retrieves all message-signature pairs $(\mu_1', \sigma_1'), \ldots, (\mu_{n_{\mathsf{info}}}', \sigma_{n_{\mathsf{info}}}')$ pertaining to that particular info from L (and discards the rest). He then parses σ_i^* as $(\tilde{\sigma}_i, r_i^*)$, sets $\tilde{\mu}_i \leftarrow \mu_i^* \| r_i^*, \forall i = 1, \ldots, k_{\mathsf{info}}+1$, and outputs (μ_1', σ_1'), $\ldots, (\mu_{n_{\mathsf{info}}}', \sigma_{n_{\mathsf{info}}}')$, and $(\tilde{\mu}_1, \tilde{\sigma}_1), \ldots, (\tilde{\mu}_{k_{\mathsf{info}}+1}, \tilde{\sigma}_{k_{\mathsf{info}}+1})$.

Analysis. Because U^* runs in polynomial-time and all queries are handled efficiently, B runs in polynomial-time as well. Since U^* succeeds in $\mathbf{HU-OMUF}_{\mathsf{PBSS},\mathsf{U}^*}(1^\lambda)$, he outputs a single info and $k_{\mathsf{info}} + 1$ valid message-signature pairs. B simulated the honest-user algorithm U' to compute the message-signature pairs: $(\mu_1', \sigma_1'), \ldots, (\mu_{n_{\mathsf{info}}}', \sigma_{n_{\mathsf{info}}}')$, thus all these pairs are valid with overwhelming probability (due to correctness).

Observe that all messages are pairwise distinct. Indeed, consider the messages $(\mu_1', \ldots, \mu_{n_{\mathsf{info}}}')$ and $(\tilde{\mu}_1, \ldots, \tilde{\mu}_{k_{\mathsf{info}}+1})$, pertaining to common information info. These are of the form $\mu_i' = \mu_i \| r_i, \forall i = 1, \ldots, n_{\mathsf{info}}$ and $\tilde{\mu}_j = \mu_j^* \| r_j^*, \forall j = 1, \ldots, k_{\mathsf{info}} + 1$, respectively. Because the r_i are chosen uniformly at random from $\{0,1\}^\lambda$, it follows that $(\mu_1', \ldots, \mu_{n_{\mathsf{info}}}')$ are pairwise distinct with overwhelming probability. Similarly,

because U^* wins in $\mathbf{HU-OMUF}_{\mathsf{PBSS},\mathsf{U}^*}(1^\lambda)$, messages $(\mu_1^*,\ldots,\mu_{k_{\mathsf{info}}+1}^*)$ are pairwise distinct and thus, $(\tilde{\mu}_1,\ldots,\tilde{\mu}_{k_{\mathsf{info}}+1})$ are also distinct. Moreover, by definition we have $\{\mu_1,\ldots,\mu_{n_{\mathsf{info}}}\}\cap\{\mu_1^*,\ldots,\mu_{k_{\mathsf{info}}+1}^*\}=\emptyset$, and thus, $\mu_i'\neq\tilde{\mu}_j,\forall i,j$.

Next, we show that B could produce one more message-signature pair than the number of successful, complete protocol interactions with S'. Because U^* wins in experiment $\mathbf{HU-OMUF}_{\mathsf{PBSS},\mathsf{U}^*}(1^\lambda)$, it follows that in at most k_{info} of the protocol executions that B relayed between U^* and S', the signer returned "ok". Furthermore, B executed a total of n_{info} honest-user instances to simulate oracle P. Since U^* successfully outputs $k_{\mathsf{info}}+1$ message-signature pairs for pairwise distinct messages μ_i, it follows that B has asked a total of at most $k_{\mathsf{info}}+n_{\mathsf{info}}$ queries in which S' returned "ok". However, B returned a total of $n_{\mathsf{info}}+k_{\mathsf{info}}+1$ message-signature pairs for info, which contradicts our assumption that PBSS is unforgeable. □

4.4 A PBSS from Ring-SIS

We now present our lattice-based PBSS. Its time and space complexity are quasilinear, $\tilde{O}(n)$ in the security parameter, and its security will be proven in the random oracle model under the worst-case assumption that $\mathsf{Ring-SVP}_{\gamma,\infty}$ is hard to solve in the ring $\mathcal{R}_q$ for $\gamma=\tilde{O}(n^4)$. Notice that it is possible for our scheme to be instantiated with regular q-ary lattices and thus have its security based on regular SIS and SVP instead. Here we describe only the more efficient ideal lattice variant. Our scheme relies on carefully setting multiple interconnected parameters which are detailed in Table 4.1 (sorted by order of appearance in our construction). All sets are subsets of $\mathcal{R}_q=\mathbb{Z}_q[X]/\langle X^n+1\rangle$ and are defined by means of a l_∞−norm bound. The third column gives an indication of the asymptotic magnitude of the corresponding parameter/set w.r.t. the main security parameter n. The last column provides insight as to the role(s) that the corresponding parameter/set has in the interactive protocol, shown in its entirety in Figure 4.8. Some sets introduce a correctness defect which can be rectified by increasing the value of parameter ϕ, which improves performance but requires a slightly stronger hardness assumption (by some constant factor). As in

Table 4.1: Scheme parameters for main security parameter n.

Parameter	Value	Asymptotics	Purpose
n	power of 2	-	main security parameter
d_s	positive integer constant $< q/(4n)$	$\mathcal{O}(1)$	secret key size, unforgeability
$\mathcal{D}_s$	$\{\mathbf{f} \in \mathcal{R}_q : \|\mathbf{f}\|_\infty \leq d_s\}$	$\mathcal{O}(1)$	secret key space
c_m	$> 1/\log(2d_s)$	$\tilde{\mathcal{O}}(1)$	witness indistinguishability, leakage resilience
m	$\lfloor c_m \log q \rfloor + 1$	$\Omega(\log(n))$	worst-case to average-case reduction
$\mathcal{D}_\epsilon$	$\{\mathbf{f} \in \mathcal{R}_q : \|\mathbf{f}\|_\infty \leq d_\epsilon := 1\}$	$\mathcal{O}(1)$	hash output size
ϕ	positive integer constant ≥ 1	$\mathcal{O}(1)$	correctness, speed
$\mathcal{D}_a$	$\{\mathbf{f} \in \mathcal{R}_q : \|\mathbf{f}\|_\infty \leq d_a := \phi n d_\epsilon\}$	$\mathcal{O}(n)$	partial blindness
$\mathcal{D}_{a'}$	$\{\mathbf{f} \in \mathcal{R}_q : \|\mathbf{f}\|_\infty \leq d_{a'} := \phi n(d_a + d_\epsilon) + d_\epsilon\}$	$\mathcal{O}(n^2)$	partial blindness
G_ϵ	$\{\mathbf{f} \in \mathcal{R}_q : \|\mathbf{f}\|_\infty \leq d_{a'} - (d_a + d_\epsilon)\}$	$\mathcal{O}(n^2)$	partial blindness
$\mathcal{D}_y$	$\{\mathbf{f} \in \mathcal{R}_q : \|\mathbf{f}\|_\infty \leq d_y := \phi m n^2 d_s d_\epsilon\}$	$\tilde{\mathcal{O}}(n^2)$	witness indistinguishability
G_*	$\{\mathbf{f} \in \mathcal{R}_q : \|\mathbf{f}\|_\infty \leq d_{G_*} := d_y - n d_s d_\epsilon\}$	$\tilde{\mathcal{O}}(n^2)$	witness indistinguishability, correctness defect
$\mathcal{D}_\beta$	$\{\mathbf{f} \in \mathcal{R}_q : \|\mathbf{f}\|_\infty \leq d_\beta := \phi m n d_{G_*}\}$	$\tilde{\mathcal{O}}(n^3)$	partial blindness
G	$\{\mathbf{f} \in \mathcal{R}_q : \|\mathbf{f}\|_\infty \leq d_G := d_\beta - d_{G_*}\}$	$\tilde{\mathcal{O}}(n^3)$	partial blindness, correctness defect
G_ω	$\{\mathbf{f} \in \mathcal{R}_q : \|\mathbf{f}\|_\infty \leq d_\omega := d_a - d_\epsilon\}$	$\tilde{\mathcal{O}}(n)$	partial blindness, correctness defect
G_σ	$\{\mathbf{f} \in \mathcal{R}_q : \|\mathbf{f}\|_\infty \leq d_\sigma := d_\beta - d_{G_*}\}$	$\tilde{\mathcal{O}}(n^3)$	partial blindness, correctness defect
G_δ	$\{\mathbf{f} \in \mathcal{R}_q : \|\mathbf{f}\|_\infty \leq d_\delta := d_{a'} - d_\epsilon\}$	$\mathcal{O}(n^2)$	partial blindness, correctness defect
$\mathcal{D}$	$\{\mathbf{f} \in \mathcal{R}_q : \|\mathbf{f}\|_\infty \leq d_\mathcal{D} := d_{G_*} + d_\beta + n d_s d_\omega\}$	$\tilde{\mathcal{O}}(n^3)$	collisions under h
q	$\geq 4 d_\mathcal{D} m n \sqrt{n} \log(n)$	$\tilde{\Theta}(n^4\sqrt{n})$	worst-case to average-case reduction

[159], we do not unwind the parameters d_s and d_ϵ in favor of making the proofs of some lemmas that involve them, easier to understand. In particular, for our scheme d_ϵ will be the constant 1, but one can increase it in order to be able to sign hash values of bit-length $> n \log(3)$.

4.4.1 Our Construction

We go on to provide definitions for the triplet of algorithms $(\mathsf{KG}, \mathsf{Sign} = \langle \mathsf{S}, \mathsf{U} \rangle, \mathsf{Ver})$ comprising our partially-blind signature scheme. Sample parameters are given in Table 4.2.

- **Key Generation.** $\mathsf{PBSS.KG}(1^n)$ chooses a secret key $\hat{\mathbf{s}} \leftarrow_\$ \mathcal{D}_s^m$ (see Table 4.1), and a homomorphic hash function $\mathsf{h} \leftarrow_\$ \mathcal{H}(\mathcal{R}_q, m)$. Next, it selects a function $\mathsf{com} \leftarrow_\$ C(1^n)$ and a hash function $\mathsf{H} \leftarrow_\$ \mathcal{H}(1^n)$ mapping $\{0,1\}^* \rightarrow \mathcal{D}_\epsilon \subset \mathcal{D}$, where $C(1^n)$ is a family of commitment schemes, mapping $\{0,1\}^* \times \{0,1\}^n \rightarrow \{0,1\}^n$. It also selects a public hash function $\mathcal{F} : \{0,1\}^* \rightarrow \mathcal{R}_q$ that maps arbitrary strings to a random public key, whose secret key is not known by anyone [7].

The algorithm computes the public key $\mathbf{S} \leftarrow \mathsf{h}(\hat{\mathbf{s}})$ and gives the pair $(\hat{\mathbf{s}}, \mathbf{S})$ to the signer. For simplicity, we will treat $\mathsf{h}, \mathsf{com}, \mathsf{H}, \mathcal{F}$ and the rest of the parameters in Table 4.1 as globally known. Alternatively, the signer can set the parameter values and include them in the public key.

- **Signature Issuing Protocol.** The signature-issuing protocol PBSS.Sign is described by the joint execution of algorithms S and U as depicted in Figure 4.8. The signer's private input is his secret key $\hat{\mathbf{s}}$, whereas the user's private input is the message to-be-signed, μ. The common information info is assumed to be negotiated outside the signature scheme and is thus treated as common input to both parties. Eventually, the user obtains a signature $(r, \hat{\mathbf{z}}, \boldsymbol{\omega}, \hat{\boldsymbol{\sigma}}, \boldsymbol{\delta})$ for message μ and common information info. If the protocol needs to be restarted during Step 2, the user only selects new $\mathbf{a} \leftarrow_{\$} \mathcal{D}_a$ and $\mathbf{a}' \leftarrow_{\$} \mathcal{D}_{a'}$, and repeats the operations that involve those, while keeping the same $r \in \{0,1\}^n$. However, if the protocol is aborted during either Step 3 or Step 5, the user must select a new r as well, to make the protocol executions independent of one another. Finally, by means of Step 5 the signer can thwart a cheating user who has obtained a valid signature but claims the contrary. In that case, the signer simply terminates the protocol, leaving the user with what he has obtained.

- **Signature Verification.** PBSS.Ver$(\mathbf{S}, \mu, \mathsf{info}, (r, \hat{\mathbf{z}}, \boldsymbol{\omega}, \hat{\boldsymbol{\sigma}}, \boldsymbol{\delta}))$ returns 1 as output iff $\hat{\mathbf{z}} \in G^m$, $\boldsymbol{\omega} \in G_\omega$, $\hat{\boldsymbol{\sigma}} \in G_\sigma^m$, $\boldsymbol{\delta} \in G_\delta$ and $\boldsymbol{\omega} + \boldsymbol{\delta} \pmod{2d_\epsilon + 1} = \mathsf{H}\big(\mathsf{h}(\hat{\mathbf{z}}) + \boldsymbol{\omega}\mathbf{S}, \mathsf{h}(\hat{\boldsymbol{\sigma}}) + \boldsymbol{\delta}\mathcal{F}(\mathsf{info}), \mathcal{F}(\mathsf{info}), \mathsf{com}(\mu; r)\big)$, and 0 otherwise.

4.4.2 Protocol Description

Our protocol is based on the 3-move witness-indistinguishable identification protocol of [122], in which the signer proves knowledge of a secret key $\hat{\mathbf{s}} \in \mathcal{D}_s^m$ such that $\mathsf{h}(\hat{\mathbf{s}}) = \mathbf{S}$, where $\mathbf{S}$ is the corresponding public key. The signer also uses a second public key $\mathbf{Z}$ (the "tag" public key), which is generated from the common information info with the help of a hash function. These two keys are used in conjunction by the signer to sign a message in such a way that the resulting protocol is witness-

indistinguishable. We construct our protocol by combining [122] with the framework of [7].

Upon commencing, the signer selects random nonce vectors $\hat{\mathbf{y}}_1 \in \mathcal{D}_y^m$ and $\hat{\mathbf{y}}_2 \in G_*^m$ and computes commitments $\mathbf{Y}_1 = \mathsf{h}(\hat{\mathbf{y}}_1)$ and $\mathbf{Y} = \mathsf{h}(\hat{\mathbf{y}}_2) + \boldsymbol{\gamma}\mathbf{Z}$, where $\mathbf{Z} = \mathcal{F}(\mathsf{info})$, which he then sends to the user. As is the case with all constructions that rely on the Fiat-Shamir heuristic [74], the user computes the challenge $\boldsymbol{\varepsilon}$ as a function (involving H) of $\mathbf{Y}_1, \mathbf{Y}$, the "tag" public key $\mathbf{Z}$, and the message to-be-signed, μ, and then "blinds" it by computing $\boldsymbol{\varepsilon}^* = \boldsymbol{\varepsilon} - \mathbf{a} - \mathbf{a}' \pmod{2d_\epsilon + 1}$, before sending it to the signer. The signer computes $\mathbf{e} = \boldsymbol{\varepsilon}^* - \boldsymbol{\gamma} \pmod{2d_\epsilon + 1}$, and then the "blinded" signature $\hat{\mathbf{z}}^* = \hat{\mathbf{y}}_1 - \mathbf{e}\hat{\mathbf{s}}$. Because h is a homomorphism, the user can check that $\mathsf{h}(\hat{\mathbf{z}}^*) = \mathbf{S}\mathbf{e} + \mathbf{Y}_1$ using public knowledge only. Finally, the user "unblinds" the signature by computing $\hat{\mathbf{z}} = \hat{\mathbf{z}}^* + \hat{\boldsymbol{\beta}}$ and $\boldsymbol{\omega} = \mathbf{e} + \mathbf{a}$, as well as $\hat{\boldsymbol{\sigma}} = \hat{\mathbf{y}}_2 + \hat{\boldsymbol{\beta}}'$ and $\boldsymbol{\delta} = \boldsymbol{\gamma} + \mathbf{a}'$, which correspond to common information info. There are a few issues that need to be addressed at this point. First, the protocol must be complete. Second, the messages transmitted by the user must be distributed independently of the signed message μ, in order to achieve partial blindness. Finally, to prove unforgeability, we need to make sure that the messages transmitted by the signer do not leak information about his secret key to the user. All issues are addressed via rejection sampling [121, 122].

In Step 2, we need to make sure that the blinded challenge $\boldsymbol{\varepsilon}^*$ that the user computes, leaks no information about the message being signed, and that it is uniformly distributed. This is necessary because $\boldsymbol{\omega} + \boldsymbol{\delta} \pmod{2d_\epsilon + 1} = \boldsymbol{\varepsilon}$ (both $\boldsymbol{\omega}$ and $\boldsymbol{\delta}$ will be part of the final signature) and thus $\boldsymbol{\varepsilon}^*$ needs to hide $\boldsymbol{\varepsilon}$. This is done in two steps: computing the blinded challenge, and then "shrinking" it modulo the range of coefficients in $\mathcal{D}_\epsilon$. First, to hide $\boldsymbol{\varepsilon}$ we rejection-sample $\boldsymbol{\varepsilon} - \mathbf{a} - \mathbf{a}'$ to make sure that it falls within G_ϵ. For that purpose, $\mathbf{a}'$ will need to be picked from a relatively larger set than $\boldsymbol{\varepsilon} - \mathbf{a}$ to "mask" the difference (and thus $\boldsymbol{\varepsilon}$ too). Otherwise, the user performs a "local restart" by picking fresh $\mathbf{a}$ and $\mathbf{a}'$. The correctness defect introduced here can effectively be lowered to 0 because the user can repeat it locally. Second, provided that $\boldsymbol{\varepsilon} - \mathbf{a} - \mathbf{a}' \in G_\epsilon$, we have to ensure that $\boldsymbol{\varepsilon}^* := \boldsymbol{\varepsilon} - \mathbf{a} - \mathbf{a}' \pmod{2d_\epsilon + 1}$ is also distributed uniformly over $\mathcal{D}_\epsilon$ before sending it to the signer. We achieve this by imposing a re-

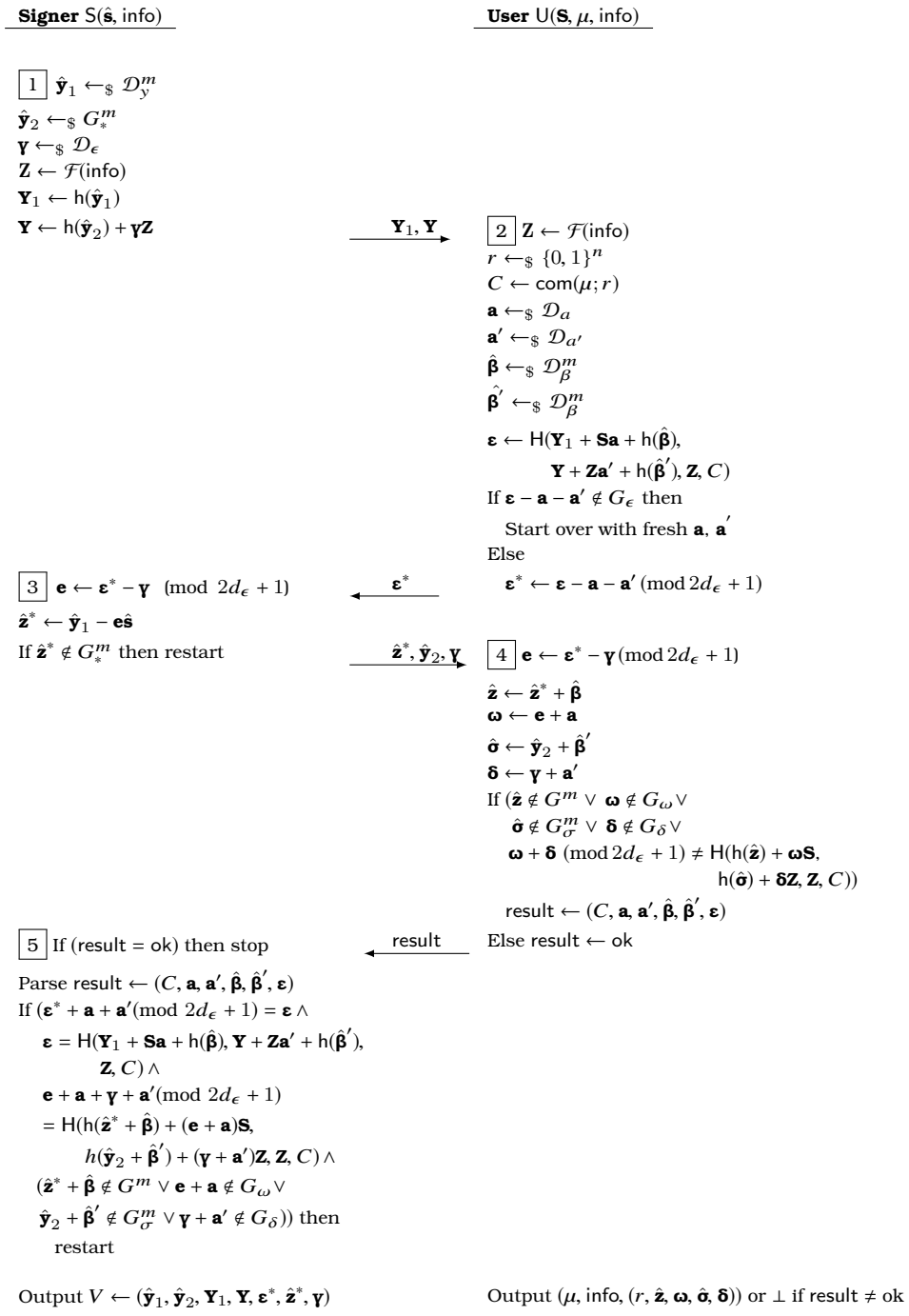

Figure 4.8: The five-step, four-move signature issuing protocol (steps shown in boxed numbers) for the proposed PBSS. All parameter and set definitions are given in Table 4.1. For brevity, we omit any verifications performed by the two parties w.r.t. the domains from which the protocol messages come from.

striction on the "shape" of G_ϵ. For our case of $d_\epsilon = 1$, this can be achieved by requiring that the range of coefficients in G_ϵ is a multiple of $2d_\epsilon + 1 = 3$. However, notice that if we require that $2[d_{a'} - (d_a + d_\epsilon)] + 1 = 2(\phi^2 n^2 - 1) + 1 \equiv 0 \pmod 3$, this is equivalent to $\phi^2 \equiv 2 \pmod 3$, which has no solutions. To fix this, we set the upper bound for the coefficients in $\mathcal{D}_{a'}$ to be slightly higher, i.e., $d_{a'} := \phi n(d_a + 1) + 1$ (or $d_{a'} := \phi n(d_a + d_\epsilon) + d_\epsilon$ in general). By following the same rationale as above, for the case of $d_\epsilon = 1$, we obtain the congruence $\phi^2 \equiv 1 \pmod 3$, which is satisfied by all natural numbers that are not a multiple of 3. Thus, we need to select ϕ to be non-congruent to 0 modulo 3, which is not a steep requirement at all, given the natural density of such numbers. All of the parameter sets proposed in Table 4.2 satisfy this condition.

Upon receiving the "shrinked" blinded challenge $\boldsymbol{\varepsilon}^*$, the signer computes $\mathbf{e} \leftarrow \boldsymbol{\varepsilon}^* - \boldsymbol{\gamma} \pmod{2d_\epsilon + 1}$. Notice that this computation is done modulo $2d_\epsilon + 1$ in correspondence to the computation of $\boldsymbol{\varepsilon}^*$ performed by the user during Step 2. Since both $\boldsymbol{\varepsilon}^*$ and $\boldsymbol{\gamma}$ are uniform over $\mathcal{D}_\epsilon$ (which is isomorphic to $\mathbb{Z}^n_{2d_\epsilon+1}$), $\mathbf{e}$ is also uniform over $\mathcal{D}_\epsilon$. The rationale behind the reduction modulo $2d_\epsilon + 1$ is to make the masking of $\mathbf{e}$ possible during the next step of the protocol (it is otherwise impossible to apply Lemmas 2 and 4). Next, we use rejection sampling to hide $\mathbf{e}\hat{\mathbf{s}}$ (and thus $\hat{\mathbf{s}}$) by adding to it a vector $\hat{\mathbf{y}}_1$ from a relatively larger set, compared to $\|\mathbf{e}\hat{\mathbf{s}}\|_\infty$, and outputting the result only if it falls within G^m_*. This results in $\hat{\mathbf{z}}^* = \hat{\mathbf{y}}_1 - \mathbf{e}\hat{\mathbf{s}}$ appearing to be uniform over G^m_*, despite actually being related to secret key $\hat{\mathbf{s}}$. However, if $\hat{\mathbf{z}}^* \notin G^m_*$, the protocol must be restarted. As we show in the next Section, the number of required trials can be greatly reduced by increasing one of our scheme's parameters.

Finally, rejection sampling is used again in Step 4 when the user attempts to "unblind" the components of the final signature. More specifically, the user masks $\mathbf{e}, \boldsymbol{\gamma}, \hat{\mathbf{z}}^*$ and $\hat{\boldsymbol{\sigma}}$ with the help of $\mathbf{a}, \mathbf{a}', \hat{\boldsymbol{\beta}}$ and $\hat{\boldsymbol{\beta}}'$, respectively (which were prepared during Step 2). Unfortunately, rejection sampling needs to be applied four times in total, which considerably decreases the user's chance of obtaining a signature without having to restart the protocol (see for example the first column of Table 4.2). However, the correctness defect introduced during Step 4 can also be ameliorated by increasing one of the scheme's parameters (namely, ϕ) at the expense of a slightly stronger hardness

assumption. In particular, if any of $\hat{\mathbf{z}}^* + \hat{\boldsymbol{\beta}}, \mathbf{e} + \mathbf{a}, \hat{\mathbf{y}}_2 + \hat{\boldsymbol{\beta}}'$ or $\boldsymbol{\gamma} + \mathbf{a}'$ does not fall within $G^m, G_\omega, G_\sigma^m$ or G_δ, respectively, the user sends $(C, \mathbf{a}, \mathbf{a}', \hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\beta}}')$ to the signer, who then verifies whether the user has indeed failed to obtain a valid signature, or not. The signer does so by tracing the computations performed on the user's side. We stress that without this fifth final step, it is impossible for the signer to know whether the user successfully produced a valid signature during Step 4, or not. Indeed, the signer does not know if $\hat{\mathbf{z}} \in G^m \wedge \boldsymbol{\omega} \in G_\omega \wedge \hat{\boldsymbol{\sigma}} \in G_\sigma^m \wedge \boldsymbol{\delta} \in G_\delta$, because he has never seen any of the masking terms $\hat{\boldsymbol{\beta}}, \mathbf{a}, \hat{\boldsymbol{\beta}}', \mathbf{a}'$ that were used to compute $\hat{\mathbf{z}}, \boldsymbol{\omega}, \hat{\boldsymbol{\sigma}}$, and $\boldsymbol{\delta}$, respectively. However, as we will prove in Section 4.4.3, the signer cannot be tricked into restarting the protocol by a malicious user, unless the latter is able to find collisions for h in $\mathcal{D} \times \mathcal{D}$. Additionally, for proving unforgeability we will require that com is binding. Finally, to prevent the signer from learning information about the signed message, μ, across restarts, we will require that com is also hiding.

Table 4.2: Sample parameter instantiations for our PBSS. Parameters are set so that the collision problem is hard to solve [122, 162]. The parameters in the first column use the mildest hardness assumption, the set of the second column aims to reduce the number of required repetitions, and the third set aims to decrease the signature size, while keeping the number of required repetitions small (other trade-offs are also possible). For the second and third column, the optimisation goal is denoted in bold face. In all cases, the Hermite factor is taken to be 1.007, and the estimated security level is 92 bits [81, 135]. To decrease the expected number of repetitions ($e^{5/\phi}$ as we prove in Theorem 4.4.3), we need to increase the value of the parameter ϕ, thus sampling our masking vectors throughout the protocol from larger sets. Finally, as we discuss in Section 5.2, ϕ must not be a multiple of 3 (in case $d_\epsilon = 1$).

Parameter	Sample Instantiations		
n (power of 2)	2048	2048	2048
q (prime $\approx n^7$)	$\approx 2^{77}$	$\approx 2^{77}$	$\approx 2^{77}$
ϕ	1	29	16
d_s	1	1	21619
m	78	78	5
Repetitions	148	**1.19**	1.37
Secret key size	31.65 kB	31.65 kB	19.71 kB
Public key size	19.71 kB	19.71 kB	19.71 kB
Signature size	1868.8 kB	2260.6 kB	**168.3** kB
Communication	3078.84 kB	3664.6 kB	320.72 kB

4.4.3 Analysis and Security

We now provide theorems and supporting lemmas showing that our proposed scheme satisfies the basic security requirements of leakage-resilient PBSS, namely: correctness, partial blindness, unforgeability, and leakage resilience. Once we have established the baseline security of our scheme, we consider further extensions of the security model.

Correctness

The next lemma is also required for our analysis, as it provides a bound (w.r.t. the infinity norm) for the product of any pair of polynomials in $\mathcal{R}_q$, when they are reduced modulo $X^n + 1$.

Lemma 4.4.1. *(Lemma 3.2 in [159, p. 28]) Let $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{R}_q$ be arbitrary polynomials. Then $\|\boldsymbol{ab} \bmod (X^n + 1)\|_\infty \leq n\|\boldsymbol{a}\|_\infty\|\boldsymbol{b}\|_\infty$.*

Theorem 4.4.2. *(Correctness) Let $g(n) = \omega(\log^5(n))$. Our PBSS is correct after at most $g(n)$ (or, an expected number of $e^{5/\phi}$) repetitions.*

Proof. First, note that if no restarts occur, the protocol produces a valid signature. That is, for all honestly generated key pairs $(\hat{\mathbf{s}}, \mathbf{S})$, all messages $\mu \in \{0,1\}^*$, all common information $\mathsf{info} \in \{0,1\}^*$, and all signatures $(r, \hat{\mathbf{z}}, \boldsymbol{\omega}, \hat{\boldsymbol{\sigma}}, \boldsymbol{\delta})$ we have: $\hat{\mathbf{z}} \in G^m, \boldsymbol{\omega} \in G_\omega, \hat{\boldsymbol{\sigma}} \in G_\sigma^m, \boldsymbol{\delta} \in G_\delta$, and $\mathsf{h}(\hat{\mathbf{z}}) + \boldsymbol{\omega}\mathbf{S} = \mathsf{h}(\hat{\mathbf{z}}^* + \hat{\boldsymbol{\beta}}) + (\mathbf{e}+\mathbf{a})\mathbf{S} = \mathsf{h}(\hat{\mathbf{y}_1} - \mathbf{e}\hat{\mathbf{s}} + \hat{\boldsymbol{\beta}}) + (\mathbf{e}+\mathbf{a})\mathbf{S} = \mathbf{Y}_1 + \mathbf{a}\mathbf{S} + \mathsf{h}(\hat{\boldsymbol{\beta}})$. Additionally, we have: $\boldsymbol{\omega} + \boldsymbol{\delta} = (\mathbf{e} + \mathbf{a}) + (\boldsymbol{\gamma} + \mathbf{a}') = (\mathbf{e} + \boldsymbol{\gamma}) + (\mathbf{a} + \mathbf{a}')$. Therefore, by reducing modulo $2d_\epsilon + 1$, we obtain: $\boldsymbol{\omega} + \boldsymbol{\delta} (\mathrm{mod}\, 2d_\epsilon + 1) = (\mathbf{e} + \boldsymbol{\gamma}) + (\mathbf{a} + \mathbf{a}') (\mathrm{mod}\, 2d_\epsilon + 1) = \boldsymbol{\varepsilon}^* + \mathbf{a} + \mathbf{a}' (\mathrm{mod}\, 2d_\epsilon + 1) = \boldsymbol{\varepsilon}$.
Thus, we have shown that: $\boldsymbol{\omega} + \boldsymbol{\delta} (\mathrm{mod}\, 2d_\epsilon + 1) = \mathsf{H}(\mathsf{h}(\hat{\mathbf{z}}) + \boldsymbol{\omega}\mathbf{S}, \mathsf{h}(\hat{\boldsymbol{\sigma}}) + \boldsymbol{\delta}\mathcal{F}(\mathsf{info}), \mathcal{F}(\mathsf{info}), \mathsf{com}(\mu, r))$, and $\mathsf{PBSS.Ver}(\mathbf{S}, \mu, \mathsf{info}, (r, \hat{\mathbf{z}}, \boldsymbol{\omega}, \hat{\boldsymbol{\sigma}}, \boldsymbol{\delta}))$ returns 1 as its output.

Next, we consider all possible restart cases and address the introduced correctness error in each one of them:

Restarts occurring at Step 2. Restarts during this step do not affect correctness at all, because the user just performs them locally. By applying Lemma 2.5.8, with $k = n, A = d_a + d_\epsilon$ and $B = d_{a'} = \phi n(d_a + d_\epsilon) + d_\epsilon$ to ensure that $\boldsymbol{\varepsilon} - \mathbf{a} - \mathbf{a}' \in G_\epsilon$, we

obtain an expected number of trials which is constant ($e^{1/\phi}$), and which decreases as ϕ increases.

Restarts occurring at Step 3. In Step 3, the signer rejection-samples $\hat{\mathbf{z}}^* = \hat{\mathbf{y}}_1 - \mathbf{e}\hat{\mathbf{s}}$ to ensure that it lies in G_*^m. According to Lemma 4.4.1, $\|\mathbf{e}\hat{\mathbf{s}} \bmod (X^n + 1)\|_\infty \leq nd_sd_\epsilon$. Therefore, if we apply Lemma 2.5.8 with $k = mn$, $A = nd_sd_\epsilon$ and $B = d_y$, we conclude that the probability of success is $e^{-1/\phi}$ and the maximum number of trials is $\omega(\log(n))$ during this step. Thus, after an expected number of $e^{1/\phi}$ trials, the protocol successfully proceeds to Step 4.

Restarts occurring after Step 4. During the "Unblind Phase" of Step 4, the user requires that $\hat{\mathbf{z}}^* + \hat{\boldsymbol{\beta}} \in G^m, \hat{\mathbf{y}_2} + \hat{\boldsymbol{\beta}}' \in G_\sigma^m, \mathbf{e} + \mathbf{a} \in G_\omega$, and $\boldsymbol{\gamma} + \mathbf{a}' \in G_\delta$. Otherwise, he requests a protocol restart from the signer. By applying Lemma 2.5.8 with $k = mn, A = d_{G_*}, B = d_\beta$ to $\hat{\mathbf{z}}^* + \hat{\boldsymbol{\beta}}$, we obtain a success probability $e^{-1/\phi}$ and a maximum number of trials of $\omega(\log(n))$. Similarly, for $\hat{\mathbf{y}_2} + \hat{\boldsymbol{\beta}}'$ with $k = mn, A = d_{G_*}, B = d_\beta = \phi mnd_{G_*}$, Lemma 2.5.8 yields a success probability $e^{-1/\phi}$ and a maximum number of trials of $\omega(\log(n))$. For $\mathbf{e}+\mathbf{a}$, Lemma 2.5.8 with $k = n, A = d_\epsilon$, and $B = d_a = \phi nd_\epsilon$ yields a success probability of $e^{-1/\phi}$. Finally, for $\boldsymbol{\gamma} + \mathbf{a}'$, if we apply Lemma 2.5.8 with $k = n, A = d_a + d_\epsilon$, and $B = \phi n(d_a + d_\epsilon) + d_\epsilon$ yields a success probability of $e^{-1/\phi}$.

In total, after at most $g(n) = \omega(\log^5(n))$, or an expected number of $e^{5/\phi}$ restarts, the protocol is indeed complete. □

Remark 6. Note that all operations involved in our scheme (including restarts), as well as sizes of private keys, public keys and signatures are of quasilinear complexity.

Remark 7. Also note that the parameter ϕ controls the number of trials. Increasing its value, decreases the expected number of protocol restarts, and vice-versa.

Partial Blindness

In proving that our construction is partially blind, we follow an approach similar to [159, p. 14] and show that all protocol messages exchanged between the user and the signer, along with the final output, are distributed independently from the signed message. For our analysis, we treat each of the exchanged messages and the output signature as random variables.

Theorem 4.4.3. *(Partial Blindness) If com is $\theta_{com}^{(h)}$ - hiding, then our* PBSS *is $(\infty, \theta_{com}^{(h)})$ - partially-blind.*

Proof. As per $\mathbf{PBlind}_{\mathsf{PBSS}}(1^n)$ (see Section 4.2.1), the malicious signer chooses common information info, and two messages μ_0, μ_1, and then interacts with two honest users, $\mathsf{U}(\mathbf{S}, \mu_b, \mathsf{info})$ and $\mathsf{U}(\mathbf{S}, \mu_{1-b}, \mathsf{info})$, after a secret coin flip $b \leftarrow_{\$} \{0,1\}$.

Distribution of $\boldsymbol{\varepsilon}^*$. Let $\boldsymbol{\varepsilon}_b^*, \boldsymbol{\varepsilon}_{1-b}^*$ be the first protocol messages of users $\mathsf{U}(\mathbf{S}, \mu_0, \mathsf{info})$ and $\mathsf{U}(\mathbf{S}, \mu_1, \mathsf{info})$, respectively. Both are of the form $\boldsymbol{\varepsilon} - \mathbf{a} - \mathbf{a}'$ (mod $2d_\epsilon + 1$), with $\boldsymbol{\varepsilon} - \mathbf{a} \in \{\mathbf{f} \in \mathcal{R}_q : \|\mathbf{f}\|_\infty \leq d_a + d_\epsilon\}$ and $\mathbf{a}'$ is distributed uniformly over $\mathcal{D}_{a'}$. First, notice that by Lemma 2.5.7 with $k = n, A = d_a + d_\epsilon$ and $B = d_{a'}$, it follows that $\Delta(\boldsymbol{\varepsilon}_b - \mathbf{a}_b - \mathbf{a}'_b, \boldsymbol{\varepsilon}_{1-b} - \mathbf{a}_{1-b} - \mathbf{a}'_{1-b}) = 0$. By applying Lemma 2.3.1 to random variables $\boldsymbol{\varepsilon}_b - \mathbf{a}_b - \mathbf{a}'_b$ and $\boldsymbol{\varepsilon}_{1-b} - \mathbf{a}_{1-b} - \mathbf{a}'_{1-b}$, with $f(X) = X$ (mod $2d_\epsilon + 1$), we have $\Delta(\boldsymbol{\varepsilon}_b^*, \boldsymbol{\varepsilon}_{1-b}^*) = 0$.

Distribution of $\hat{\mathbf{z}}$. Let $\hat{\mathbf{z}}_0, \hat{\mathbf{z}}_1$ be part of the final output of $\mathsf{U}(\mathbf{S}, \mu_0, \mathsf{info})$ and $\mathsf{U}(\mathbf{S}, \mu_1, \mathsf{info})$ respectively; Note that both are of the form $\hat{\mathbf{z}}^* + \hat{\boldsymbol{\beta}}$, for $\hat{\mathbf{z}}^* \in G_*^m$ and $\hat{\boldsymbol{\beta}} \leftarrow_{\$} \mathcal{D}_\beta^m$. Additionally, both $\hat{\mathbf{z}}_0$ and $\hat{\mathbf{z}}_1$ lie in G^m because the users perform rejection sampling (Step 4) on these random variables. Therefore, their coefficients are bounded in absolute value by $d_\beta - d_{G_*}$. From Lemma 2.5.7 with $k = mn, A = d_{G_*}$ and $B = d_\beta$, we infer that $\Delta(\hat{\mathbf{z}}_0, \hat{\mathbf{z}}_1) = 0$.

Distribution of $\boldsymbol{\omega}$. Let $\boldsymbol{\omega}_0, \boldsymbol{\omega}_1$ be part of the final output of $\mathsf{U}(\mathbf{S}, \mu_0, \mathsf{info})$ and $\mathsf{U}(\mathbf{S}, \mu_1, \mathsf{info})$ respectively. Both are of the form $\mathbf{e} + \mathbf{a}$, for $\mathbf{e} \in \mathcal{D}_\epsilon$ and $\mathbf{a} \leftarrow_{\$} \mathcal{D}_a$. Additionally, both $\boldsymbol{\omega}_0$ and $\boldsymbol{\omega}_1$ lie in G_ω because the users perform rejection sampling (during Step 4) on these random variables. Therefore, their coefficients are bounded in absolute value by $d_a - d_\epsilon$. By applying Lemma 2.5.7 with $k = n, A = d_\epsilon$ and $B = d_a = \phi n d_\epsilon$, we infer that $\Delta(\boldsymbol{\omega}_0, \boldsymbol{\omega}_1) = 0$.

Distribution of $\hat{\boldsymbol{\sigma}}$. Let $\hat{\boldsymbol{\sigma}}_0, \hat{\boldsymbol{\sigma}}_1$ be part of the final output of $\mathsf{U}(\mathbf{S}, \mu_0, \mathsf{info})$ and $\mathsf{U}(\mathbf{S}, \mu_1, \mathsf{info})$ respectively. Both are of the form $\hat{\mathbf{y}}_2 + \hat{\boldsymbol{\beta}}'$, for $\hat{\mathbf{y}}_2 \in G_*^m$ and $\hat{\boldsymbol{\beta}}' \leftarrow_{\$} D_\beta^m$. Additionally, both $\hat{\boldsymbol{\sigma}}_0$ and $\hat{\boldsymbol{\sigma}}_1$ lie in G_σ^m because the users perform rejection sampling (during Step 4) on these random variables. Therefore, their coefficients are bounded in absolute value by $d_\beta - d_{G_*}$. By applying Lemma 2.5.7 with $k = mn, A = d_{G_*}$ and $B = d_\beta$, we infer that $\Delta(\hat{\boldsymbol{\sigma}}_0, \hat{\boldsymbol{\sigma}}_1) = 0$.

Distribution of $\boldsymbol{\delta}$. Let $\boldsymbol{\delta}_0, \boldsymbol{\delta}_1$ be part of the final output of $\mathsf{U}(\mathbf{S}, \mu_0, \mathsf{info})$ and $\mathsf{U}(\mathbf{S}, \mu_1, \mathsf{info})$

respectively. Both are of the form $\boldsymbol{\gamma}+\mathbf{a}'$, for $\boldsymbol{\gamma} \in \mathcal{D}_\epsilon$ and $\mathbf{a}' \leftarrow_{\$} \mathcal{D}_{a'}$. Additionally, both $\boldsymbol{\delta}_0$ and $\boldsymbol{\delta}_1$ lie in G_δ because the users perform rejection sampling (Step 4) on these random variables. Therefore, their coefficients are bounded in absolute value by $d_{a'} - d_\epsilon$. From Lemma 2.5.7 with $k = n, A = d_\epsilon$ and $B = d_{a'} = \phi n(d_a + d_\epsilon) + d_\epsilon > d_\epsilon$, we infer that $\Delta(\boldsymbol{\delta}_0, \boldsymbol{\delta}_1) = 0$.

Distribution of $\mathbf{Y}_1, \mathbf{Y}, \hat{\mathbf{y}}_2, \boldsymbol{\gamma}$ and r. These random variables are all either sampled uniformly at random from some domain, or distributed independently from the signed message μ. We note that $\mathbf{e}$ (which can be computed from $\boldsymbol{\varepsilon}^*$ and $\boldsymbol{\gamma}$) is also uniform over $\mathcal{D}_\epsilon$, since its computation is done within $\mathcal{D}_\epsilon$.

Restarts. Restarts are distinguished into two types: those that occur during Step 2 and can be handled locally by the user, and those that occur after Step 4 and cause the protocol to start over. Notice that we do not need to deal with restarts occurring in Step 3, because they do not affect partial blindness as per game $\mathbf{PBlind}_{\mathsf{PBSS}}(1^n)$.

- Restarts during Step 2: Because com is statistically hiding and the user selects a new set of $r, \mathbf{a}, \mathbf{a}', \hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\beta}}'$ every time he performs a restart during Step 2 of the signature-issuing protocol, each protocol execution is statistically independent from any preceding execution. Therefore our scheme is $(\infty, \theta^{(\mathsf{h})}_{\mathsf{com}})$ - partially blind, since com is statistically $\theta^{(\mathsf{h})}_{\mathsf{com}}$ - hiding.

- Restarts caused after Step 4: The user submits $(C, \mathbf{a}, \mathbf{a}', \hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\beta}}', \boldsymbol{\varepsilon})$ to the signer. The signer is then able to trace the computations performed on the user's side and can determine whether a restart is truly necessary. Note that the signer works with the commitment, C, instead of the original message, μ. Again, due to com's statistical hiding property, μ remains statistically hidden from the signer, since he does not possess the corresponding decommitment parameter r which would allow him recovery of μ. Thus, our scheme achieves statistical instead of perfect partial blindness.

□

Remark 8. Based on the previous discussion, if com is perfectly hiding (i.e., $\theta_{\mathsf{com}} = 0$), then PBSS is partially blind in a perfect sense, whereas if com is statistically hiding,

PBSS is partially blind in a statistical sense. In either case, a malicious signer only gains a negligible amount of information from protocol restarts, at best.

Unforgeability

The generalized Forking Lemma from [32] is a probabilistic result that lies at the core of proving the unforgeability of our scheme. Additionally, to simulate the signing oracle in the unforgeability game of Section 4.2.1, we will also need two supporting lemmas. The first states that for each public key $\mathbf{S}$ in our protocol, there exist (with overwhelming probability) at least two distinct corresponding secret keys $\hat{\mathbf{s}}, \hat{\mathbf{s}}'$.

Lemma 4.4.4. *(Lemma 3.6 in [159, p. 29]) Let* $\mathsf{h} \in \mathcal{H}(\mathcal{R}_q, m)$. *For every secret key* $\hat{\boldsymbol{s}} \leftarrow_{\$} \mathcal{D}_s^m$, *there exists (with overwhelming probability) a second* $\hat{\boldsymbol{s}}' \in \mathcal{D}_s^m \setminus \{\hat{\boldsymbol{s}}\}$ *with* $\mathsf{h}(\hat{\boldsymbol{s}}) = \mathsf{h}(\hat{\boldsymbol{s}}')$.

The next lemma is based on Lemma 3.7 from [159], suitably adapted for our construction. Informally, it states that if we interpret the components of a (malicious) user's view as random variables, then the user is unable to tell which of (at least) two possible keys $\hat{\mathbf{s}}, \hat{\mathbf{s}}' \in \mathsf{h}^{-1}(\mathbf{S}) \cap \mathcal{D}_s^m$ was used during the signature-issuing protocol, except with negligible advantage.

Lemma 4.4.5. *Let* $\mathsf{h} \in \mathcal{H}(\mathcal{R}_q, m)$ *and* $\mathbf{S} \in \mathcal{R}_q$. *For any message* $\mu \in \{0,1\}^*$ *and any two distinct secret keys* $\hat{\mathbf{s}}, \hat{\mathbf{s}}' \in \mathcal{D}_s^m$ *with* $\mathsf{h}(\hat{\boldsymbol{s}}) = \mathsf{h}(\hat{\boldsymbol{s}}')$, *the resulting protocol views* $(\boldsymbol{Y}_1, \boldsymbol{Y}, \boldsymbol{\varepsilon}^*, \hat{\boldsymbol{z}}^*, \hat{\boldsymbol{y}}_2, \boldsymbol{\gamma})$ *and* $(\boldsymbol{Y}_1', \boldsymbol{Y}', \boldsymbol{\varepsilon}^{*'}, \hat{\boldsymbol{z}}^{*'}, \hat{\boldsymbol{y}}_2', \boldsymbol{\gamma}')$ *are witness-indistinguishable.*

Proof. Initially, observe that $\mathbf{Y}_1$ and $\mathbf{Y}_1'$ do not depend on the choice of secret key. The same holds for $\mathbf{Y}$ and $\mathbf{Y}'$. Furthermore, $\boldsymbol{\varepsilon}^*$ and $\boldsymbol{\varepsilon}^{*'}$ are independent of any particular $\hat{\mathbf{y}}_1 \in \mathsf{h}^{-1}(\mathbf{Y}_1) \cap \mathcal{D}_y^m$ because $\mathbf{Y}_1$ statistically hides $\hat{\mathbf{y}}_1$ through h. Moreover, $\hat{\mathbf{y}}_2$ and $\hat{\mathbf{y}}_2'$, as well as $\boldsymbol{\gamma}$ and $\boldsymbol{\gamma}'$ are all sampled independently of the secret key. Finally, we have to show that $\hat{\boldsymbol{z}}^*$ and $\hat{\boldsymbol{z}}^{*'}$ are also distributed independently of the secret key. For that, let $\mathbf{e}$ be any factor used by the signer during Step 3 of our protocol, to compute $\hat{\boldsymbol{z}}^*$, i.e.: $\hat{\boldsymbol{z}}^* = \hat{\mathbf{y}}_1 - \mathbf{e}\hat{\mathbf{s}} \in G_*^m$. Next, we set $\hat{\mathbf{y}}_1' := \hat{\mathbf{y}}_1 - \hat{\mathbf{s}}\mathbf{e} + \hat{\mathbf{s}}'\mathbf{e}$, which implies that $\hat{\boldsymbol{z}}^* = \hat{\mathbf{y}}_1' - \hat{\mathbf{s}}'\mathbf{e}$. We then easily see that $\hat{\mathbf{y}}_1' \in \mathsf{h}^{-1}(\mathbf{Y}_1) \cap \mathcal{D}_y^m$. Indeed, $\hat{\mathbf{y}}_1' \in \mathsf{h}^{-1}(\mathbf{Y}_1)$ because $\mathsf{h}(\hat{\mathbf{y}}_1') = \mathsf{h}(\hat{\mathbf{y}}_1 - \hat{\mathbf{s}}\mathbf{e} + \hat{\mathbf{s}}'\mathbf{e}) = \mathbf{Y}_1 - \mathbf{e}\mathbf{S} + \mathbf{e}\mathbf{S} = \mathbf{Y}_1$. Additionally, $\hat{\mathbf{y}}_1 \in \mathcal{D}_y^m$ since:

$\|\hat{\mathbf{y}}_1'\|_\infty = \|\hat{\mathbf{z}}^* + \hat{\mathbf{s}}'\mathbf{e}\|_\infty \leq \|\hat{\mathbf{z}}^*\|_\infty + \|\hat{\mathbf{s}}'\mathbf{e}\|_\infty \leq d_y - nd_sd_\epsilon + nd_sd_\epsilon = d_y$, where the last inequality follows from Lemma 4.4.1. In conclusion, no malicious user can distinguish whether the honest signer is using secret key $\hat{\mathbf{s}}$ with a masking term $\hat{\mathbf{y}}_1$ or $\hat{\mathbf{s}}'$ with a masking term $\hat{\mathbf{y}}_1'$, both of which yield the same output. □

We now prove that our construction is unforgeable, provided that the commitment scheme is binding, and the collision problem $\mathsf{Col}(\mathcal{H}(\mathcal{R}_q, m, \mathcal{D}))$ being hard.

Theorem 4.4.6. *(Unforgeability) Let* Sig *denote the signature issuing oracle and* H *the hashing oracle. Let* T_{Sig} *and* T_{H} *denote the cost functions for simulating the oracles* Sig *and* H *respectively, and let* $0 \leq c < 1$ *be the probability of restarting the protocol. Our* PBSS *is* $(t, q_{\mathsf{sig}}, q_{\mathsf{H}}, \theta)$*-unforgeable if* com *is* $(t', \theta/2)$*-binding, and* $\mathsf{Col}(\mathcal{H}(\mathcal{R}_q, m, \mathcal{D}))$ *is* $(t', \theta_{\mathsf{overall}}/2)$*-hard, where* $t' = t + q_{\mathsf{H}}^{q_{\mathsf{sig}}}(q_{\mathsf{sig}}T_{\mathsf{sig}} + q_{\mathsf{H}}T_{\mathsf{H}})$ *and* $\theta_{\mathsf{overall}}$ *is noticeable if* θ *is noticeable.*

Proof. Let A be an efficient forger who successfully breaks unforgeability within time t and with noticeable probability, θ. By exploiting A's capability of forging signatures in a black-box manner, we will construct a simulator B, such that B either breaks the binding property of com, or solves the collision problem.

Setup. Simulator B flips a coin $b \leftarrow_\$ \{0,1\}$. If $b = 0$, B selects $\mathsf{h} \leftarrow_\$ \mathcal{H}(\mathcal{R}_q, m)$. Otherwise, it is given the description of h as input. B initializes a list $L_{\mathsf{H}} \leftarrow \emptyset$ of query-hash pairs of the form $(\mathcal{R}_q \times \mathcal{R}_q \times \mathcal{R}_q \times \{0,1\}^*, \mathcal{D}_\epsilon)$, a list $L_F \leftarrow \emptyset$ for queries to $\mathcal{F}$ which are of the form $(\{0,1\}^*, \mathcal{R}_q)$, and a list $L_{\mathsf{Sig}} \leftarrow \emptyset$ of message-signature pairs of the form $(\{0,1\}^* \times \{0,1\}^*, G^m \times G_\omega \times G_\sigma^m \times G_\delta)$. It then picks $\hat{\mathbf{s}} \leftarrow_\$ \mathcal{D}_s^m$ and computes $\mathbf{S} \leftarrow \mathsf{h}(\hat{\mathbf{s}})$. Moreover, B randomly pre-selects random oracle answers $\mathbf{h}_1, \ldots, \mathbf{h}_{q_{\mathsf{H}}} \leftarrow_\$ \mathcal{D}_\epsilon$, a random tape ρ, and runs $\mathsf{A}(\mathbf{S}; \mathbf{h}_1, \ldots, \mathbf{h}_{q_{\mathsf{H}}}; \rho)$ in a black-box way.

RO Queries. On input $(\mathbf{u}, \mathbf{v}, \mathbf{Z}, C)$, B determines if $(\mathbf{u}, \mathbf{v}, \mathbf{Z}, C)$ has previously been queried to H by checking whether $(\mathbf{u}, \mathbf{v}, \mathbf{Z}, C) \in L_{\mathsf{H}}$. If the answer is affirmative, B returns the same output $\boldsymbol{\epsilon}$ as before, to remain consistent. Otherwise, B returns the first unused $\mathbf{h}_i$ and stores $\big((\mathbf{u}, \mathbf{v}, \mathbf{Z}, C), \mathbf{h}_i\big)$ in L_{H}.

PBS Queries. B acts as the signer according to the protocol in Figure 4.8 and fills in L_{Sig} after A produces his output.

Forgery. Since adversary A is efficient, he eventually stops, outputting: $\big(\mu_1, \mathsf{info}, (r_1, \hat{\mathbf{z}}_1, \boldsymbol{\omega}_1, \hat{\boldsymbol{\sigma}}_1, \boldsymbol{\delta}_1)\big), \ldots, \big(\mu_J, \mathsf{info}, (r_J, \hat{\mathbf{z}}_J, \boldsymbol{\omega}_J, \hat{\boldsymbol{\sigma}}_J, \boldsymbol{\delta}_J)\big)$, where $J = k_{\mathsf{info}} + 1$ for pairwise distinct messages. If $b = 0$, the reduction tries to find two pairs $(\mu_1^*, \mathsf{info}, (r_1^*, \hat{\mathbf{z}}^*, \boldsymbol{\omega}^*, \hat{\boldsymbol{\sigma}}^*, \boldsymbol{\delta}^*))$ and $(\mu_2^*, \mathsf{info}, (r_2^*, \hat{\mathbf{z}}^*, \boldsymbol{\omega}^*, \hat{\boldsymbol{\sigma}}^*, \boldsymbol{\delta}^*))$ with $\mu_1^* \neq \mu_2^*$, and returns $(\mu_1^*, r_1^*), (\mu_2^*, r_2^*)$ to break com's binding property. If no such pair is found, it simply aborts. If $b = 1$, the simulator locates a message-signature pair $((\mu^\dagger, \mathsf{info}), (r^\dagger, \hat{\mathbf{z}}^\dagger, \boldsymbol{\omega}^\dagger, \hat{\boldsymbol{\sigma}}^\dagger, \boldsymbol{\delta}^\dagger))$, where $(\mu^\dagger, \mathsf{info})$ has never been queried to the signing oracle. The algorithm computes $\mathbf{u}^\dagger = \mathsf{h}(\hat{\mathbf{z}}^\dagger) + \mathbf{S}\boldsymbol{\omega}^\dagger$ and $\mathbf{v}^\dagger = \mathsf{h}(\hat{\boldsymbol{\sigma}}^\dagger) + \mathcal{F}(\mathsf{info})\boldsymbol{\delta}^\dagger$ and rewinds the adversary to the point where $(\mathbf{u}^\dagger, \mathbf{v}^\dagger, \mathcal{F}(\mathsf{info}), \mathsf{com}(\mu^\dagger, r^\dagger))$ was queried to the hashing oracle H. Let $1 \leq I \leq q_{\mathsf{H}}$ be the index of that query. B then re-runs $\mathsf{A}(\mathbf{S}; \mathbf{h}_1, \ldots, \mathbf{h}_{I-1}, \mathbf{h}'_I, \ldots, \mathbf{h}'_{q_{\mathsf{H}}}; \rho)$ with new random responses to queries with index $\geq I$, but using the same random tape ρ. Eventually, A will output a new forgery $((\mu^\ddagger, \mathsf{info}), (r^\ddagger, \hat{\mathbf{z}}^\ddagger, \boldsymbol{\omega}^\ddagger, \hat{\boldsymbol{\sigma}}^\ddagger, \boldsymbol{\delta}^\ddagger))$ using the same random oracle query as in the first run (after polynomially bounded time because A is efficient and all of his queries are handled efficiently). B then returns $(\hat{\mathbf{z}}^\dagger + \hat{\mathbf{s}}\boldsymbol{\omega}^\dagger, \hat{\mathbf{z}}^\ddagger + \hat{\mathbf{s}}\boldsymbol{\omega}^\ddagger)$, if $\boldsymbol{\omega}^\dagger \neq \boldsymbol{\omega}^\ddagger$, as a solution to the Collision Problem and aborts otherwise (an event that as we will explain, occurs with negligible probability).

Analysis. A's environment is perfectly simulated and restarts occur with the same probability as in the original protocol. Therefore, A has no advantage whatsoever in distinguishing the simulation.

For $b = 0$, B $(t', \theta/2)$-breaks com's binding property, if A successfully attacks com's binding property to break unforgeability.

For $b = 1$, we assume that A breaks unforgeability without attacking com. Since at least one of the produced signatures was not obtained via an interaction, the probability that B correctly guesses its index is at least $\frac{1}{k_{\mathsf{info}}+1}$. Next, notice that A can successfully predict the output of the random oracle H with probability $1/|\mathcal{D}_\epsilon|$. By applying the general Forking Lemma of [32], we can determine that after rewinding, A is again successful in producing a forgery, using the same random oracle query as in the first run with probability $\theta_{\mathsf{frk}} \geq (1-c)(\theta - \frac{1}{|\mathcal{D}_\epsilon|})(\frac{\theta - 1/|\mathcal{D}_\epsilon|}{q_{\mathsf{H}}} - \frac{1}{|\mathcal{D}_\epsilon|})$, where the additional $(1-c)$ factor accounts for a potential restart during the second run. Therefore, with probability at least θ_{frk}, the following relation holds: $\mathsf{h}(\hat{\mathbf{z}}^\dagger) + \mathbf{S}\boldsymbol{\omega}^\dagger = \mathsf{h}(\hat{\mathbf{z}}^\ddagger) + \mathbf{S}\boldsymbol{\omega}^\ddagger$. This can equivalently

be written as: $\mathsf{h}(\hat{\mathbf{z}}^{\dagger}-\hat{\mathbf{z}}^{\ddagger}+\hat{\mathbf{s}}(\boldsymbol{\omega}^{\dagger}-\boldsymbol{\omega}^{\ddagger}))=\mathbf{0}$. We observe that with overwhelming probability, $\boldsymbol{\omega}^{\dagger}\neq\boldsymbol{\omega}^{\ddagger}$. Indeed: $\boldsymbol{\omega}^{\dagger}=((\boldsymbol{\varepsilon}^{*\dagger}-\boldsymbol{\gamma}^{\dagger})\bmod(2d_{\epsilon}+1))+\mathbf{a}^{\dagger}=((\boldsymbol{\varepsilon}^{\dagger}-\mathbf{a}^{\dagger}-\mathbf{a}'^{\dagger}-\boldsymbol{\gamma}^{\dagger})\bmod(2d_{\epsilon}+1))+\mathbf{a}^{\dagger}$. Similarly, we have: $\boldsymbol{\omega}^{\ddagger}=\mathbf{e}^{\ddagger}+\mathbf{a}^{\ddagger}=((\boldsymbol{\varepsilon}^{\ddagger}-\mathbf{a}^{\ddagger}-\mathbf{a}'^{\ddagger}-\boldsymbol{\gamma}^{\ddagger})\bmod(2d_{\epsilon}+1))+\mathbf{a}^{\ddagger}$. By subtracting, we get: $\boldsymbol{\omega}^{\dagger}-\boldsymbol{\omega}^{\ddagger}=((\boldsymbol{\varepsilon}^{\dagger}-\mathbf{a}^{\dagger}-\mathbf{a}'^{\dagger}-\boldsymbol{\gamma}^{\dagger}-\boldsymbol{\varepsilon}^{\ddagger}+\mathbf{a}^{\ddagger}+\mathbf{a}'^{\ddagger}+\boldsymbol{\gamma}^{\ddagger})\bmod(2d_{\epsilon}+1))+\mathbf{a}^{\dagger}-\mathbf{a}^{\ddagger}$. If $\boldsymbol{\omega}^{\dagger}-\boldsymbol{\omega}^{\ddagger}=\mathbf{0}$, then $\boldsymbol{\varepsilon}^{\ddagger}-\boldsymbol{\gamma}^{\ddagger}\pmod{2d_{\epsilon}+1}$ is determined by polynomials selected by A and polynomials determined by B before rewinding. However, both $\boldsymbol{\varepsilon}^{\ddagger}$ and $\boldsymbol{\gamma}^{\ddagger}$ are randomly selected by B after rewinding. Therefore, the probability that $\boldsymbol{\omega}^{\dagger}=\boldsymbol{\omega}^{\ddagger}$ is $\frac{1}{|\mathcal{D}_{\epsilon}|}=\frac{1}{(2d_{\epsilon}+1)^{n}}$ which is negligible in n. Thus, $\boldsymbol{\omega}^{\dagger}\neq\boldsymbol{\omega}^{\ddagger}$ with overwhelming probability $1-1/|\mathcal{D}_{\epsilon}|$.

Next, if $\boldsymbol{\omega}^{\dagger}\neq\boldsymbol{\omega}^{\ddagger}$ then with a probability of at least $1/4$, we have $\hat{\mathbf{z}}^{\dagger}-\hat{\mathbf{z}}^{\ddagger}+\hat{\mathbf{s}}(\boldsymbol{\omega}^{\dagger}-\boldsymbol{\omega}^{\ddagger})\neq\mathbf{0}$. Indeed, by Lemma 4.4.4, there exists another $\hat{\mathbf{s}}'\neq\hat{\mathbf{s}}$ (with overwhelming probability). Furthermore, because of Lemma 4.4.5, the signing protocol is witness-indistinguishable and therefore there is a probability of at least $1/2$ that the signer's output corresponds to $\hat{\mathbf{s}}'$. Because the signer possesses the secret key while the user does not, and because of Lemma 4.4.5, all protocol messages are distributed independently of the secret key, even if $\hat{\mathbf{z}}^{\dagger}-\hat{\mathbf{z}}^{\ddagger}+\hat{\mathbf{s}}'(\boldsymbol{\omega}^{\dagger}-\omega^{\ddagger})=\mathbf{0}$, B has at least $1/2$ chance of claiming that $\hat{\mathbf{z}}^{\dagger}-\hat{\mathbf{z}}^{\ddagger}+\hat{\mathbf{s}}(\boldsymbol{\omega}^{\dagger}-\boldsymbol{\omega}^{\ddagger})\neq\mathbf{0}$. Since $\hat{\mathbf{z}}^{\dagger}-\hat{\mathbf{z}}^{\ddagger}+\hat{\mathbf{s}}(\boldsymbol{\omega}^{\dagger}-\boldsymbol{\omega}^{\ddagger})\neq\mathbf{0}$, we deduce that $\hat{\mathbf{z}}^{\dagger}+\hat{\mathbf{s}}\boldsymbol{\omega}^{\dagger}\neq\hat{\mathbf{z}}^{\ddagger}+\hat{\mathbf{s}}\boldsymbol{\omega}^{\ddagger}$. Furthermore, since $\|\hat{\mathbf{z}}^{\dagger}+\hat{\mathbf{s}}\boldsymbol{\omega}^{\dagger}\|_{\infty},\|\hat{\mathbf{z}}^{\ddagger}+\hat{\mathbf{s}}\boldsymbol{\omega}^{\ddagger}\|_{\infty}\leq d_{G}+nd_{s}d_{\omega}<d_{\mathcal{D}}$, we obtain $(\hat{\mathbf{z}}^{\dagger}+\hat{\mathbf{s}}\boldsymbol{\omega}^{\dagger},\hat{\mathbf{z}}^{\ddagger}+\hat{\mathbf{s}}\boldsymbol{\omega}^{\ddagger})$ as a collision of h in $\mathcal{D}\times\mathcal{D}$, with probability: $\theta_{\mathsf{col}}\geq\frac{1}{4(k_{\mathsf{info}}+1)}(1-\frac{1}{|\mathcal{D}_{\epsilon}|})\theta_{\mathsf{frk}}$, which is noticeable due to θ.

Restarts. Finally, we argue that the only way for a user to obtain a valid signature from an aborted interaction, is if he can solve the collision problem for h in $\mathcal{D}$. Indeed, for an abort to occur in Step 5, the user needs to "convince" the honest signer by sending him $\mathsf{result}=(C,\mathbf{a},\mathbf{a}',\hat{\boldsymbol{\beta}},\hat{\boldsymbol{\beta}}',\boldsymbol{\varepsilon})$, which together with his view of the interaction $(\mathbf{Y}_1,\mathbf{Y},\boldsymbol{\varepsilon}^{*},\hat{\mathbf{z}}^{*},\hat{\mathbf{y}_2},\boldsymbol{\gamma},\mathbf{e})$, satisfy the abort criteria:

$$\boldsymbol{\varepsilon}^{*}+\mathbf{a}+\mathbf{a}'\pmod{2d_{\epsilon}+1}=\boldsymbol{\varepsilon} \tag{4.1}$$

$$\mathsf{H}(\mathbf{Y}_1+\mathbf{S}\mathbf{a}+\mathsf{h}(\hat{\boldsymbol{\beta}}),\mathbf{Y}+\mathbf{Z}\mathbf{a}'+\mathsf{h}(\hat{\boldsymbol{\beta}}'),\mathbf{Z},C)=\boldsymbol{\varepsilon} \tag{4.2}$$

$$\mathbf{e}+\mathbf{a}+\boldsymbol{\gamma}+\mathbf{a}' \pmod{2d_\epsilon+1} = \mathsf{H}(\mathsf{h}(\hat{\mathbf{z}}^*+\hat{\boldsymbol{\beta}})+\mathbf{S}(\mathbf{e}+\mathbf{a}), \mathsf{h}(\hat{\mathbf{y}}_2+\hat{\boldsymbol{\beta}}')+\mathbf{Z}(\boldsymbol{\gamma}+\mathbf{a}'), \mathbf{Z}, C) \tag{4.3}$$

$$\hat{\mathbf{z}}^*+\hat{\boldsymbol{\beta}} \notin G^m \quad \vee \quad \mathbf{e}+\mathbf{a} \notin G_\omega \quad \vee \quad \hat{\mathbf{y}}_2+\hat{\boldsymbol{\beta}}' \notin G_\sigma^m \vee \quad \boldsymbol{\gamma}+\mathbf{a}' \notin G_\delta \tag{4.4}$$

Suppose that the malicious user successfully obtains a forged signature $(r_0, \hat{\mathbf{z}}_0, \boldsymbol{\omega}_0, \hat{\boldsymbol{\sigma}}_0, \boldsymbol{\delta}_0)$ from an aborted interaction. Thus, we may assume that $(r_0, \hat{\mathbf{z}}_0, \boldsymbol{\omega}_0, \hat{\boldsymbol{\sigma}}_0, \boldsymbol{\delta}_0)$ satisfies all of the verification criteria from Section 5.3. First, observe that the adversarial user may succeed by hiding $\boldsymbol{\varepsilon}' \neq \boldsymbol{\varepsilon}$ in the computation of $\boldsymbol{\varepsilon}^*$. However, to achieve this he would need to predict the output of H, which happens with a negligible probability of $\frac{1}{|\mathcal{D}_\epsilon|}$. Thus, we have $\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}'$ with an overwhelming probability of $1 - \frac{1}{|\mathcal{D}_\epsilon|}$. Because $\boldsymbol{\varepsilon} = \boldsymbol{\omega}_0 + \boldsymbol{\delta}_0 \pmod{2d_\epsilon+1} = \boldsymbol{\omega} + \boldsymbol{\delta} \pmod{2d_\epsilon+1}$, it follows from (4.3) that $\mathsf{h}(\hat{\mathbf{z}}^*+\hat{\boldsymbol{\beta}}) + \mathbf{S}(\mathbf{e}+\mathbf{a}) = \mathsf{h}(\hat{\mathbf{z}}_0 + \boldsymbol{\omega}_0\hat{\mathbf{s}})$. Equivalently, this can be written as:

$$\mathsf{h}(\hat{\mathbf{z}}^*+\hat{\boldsymbol{\beta}}+\hat{\mathbf{s}}(\mathbf{e}+\mathbf{a})) = \mathsf{h}(\hat{\mathbf{z}}_0+\boldsymbol{\omega}_0\hat{\mathbf{s}}). \tag{4.5}$$

Next, notice that with an overwhelming probability of at least $1-\frac{1}{|\mathcal{D}_\epsilon|}$, we have $\boldsymbol{\omega}_0 = \mathbf{e}+\mathbf{a}$ (unless $\mathbf{e}+\mathbf{a} \notin G_\omega$, in which case we have a contradiction because we know that $\boldsymbol{\omega}_0 \in G_\omega$). Indeed, the only way for the malicious user to obtain a $\boldsymbol{\omega}_0 \neq \mathbf{e}+\mathbf{a}$, is if during Step 2 he used an $\mathbf{a}_0 = \boldsymbol{\omega}_0 - \boldsymbol{\omega} + \mathbf{a}$, which implies that he would have to successfully guess $\boldsymbol{\omega}$, which he can do only with a negligible probability of $\frac{1}{|G_\omega|} \leq \frac{1}{|\mathcal{D}_\epsilon|} = \frac{1}{(2d_\epsilon+1)^n}$. From Bayes' rule, we can determine that the probability that $\mathbf{e}+\mathbf{a} \in G_\omega$, given that (4.4) holds is $\frac{e^{-1/\phi}-e^{-4/\phi}}{1-e^{-4/\phi}}$, a constant. Similarly, with an overwhelming probability of at least $1-\frac{1}{|\mathcal{D}_\epsilon|}$, we have $\boldsymbol{\delta}_0 = \boldsymbol{\gamma}+\mathbf{a}'$ (unless $\boldsymbol{\gamma}+\mathbf{a}' \notin G_\delta$, in which case we have a contradiction because $\boldsymbol{\delta}_0 \in G_\delta$). Finally, with an overwhelming probability of at least $1-\frac{1}{|D_\epsilon|}$, we have $\hat{\boldsymbol{\sigma}}_0 = \hat{\mathbf{y}}_2+\hat{\boldsymbol{\beta}}'$ (unless $\hat{\mathbf{y}}_2+\hat{\boldsymbol{\beta}}' \notin G_\sigma^m$, in which case we have a contradiction because we know that $\hat{\boldsymbol{\sigma}}_0 \in G_\sigma^m$). Thus, the only possible case for condition (4.4) to hold, is if $\hat{\mathbf{z}}^*+\hat{\boldsymbol{\beta}} \notin G^m$. Observe that in that case, the arguments of h in (4.5) cannot be equal because then $\hat{\mathbf{z}}^*+\hat{\boldsymbol{\beta}} = \hat{\mathbf{z}}_0$, which contradicts the hypothesis that $\hat{\mathbf{z}}_0 \in G^m$. Therefore, we have $\hat{\mathbf{z}}^*+\hat{\boldsymbol{\beta}} \neq \hat{\mathbf{z}}_0$, and since $\|\hat{\mathbf{z}}_0+\boldsymbol{\omega}_0\hat{\mathbf{s}}\|_\infty < d_{\mathcal{D}}$ and $\|\hat{\mathbf{z}}^*+\hat{\boldsymbol{\beta}}+\hat{\mathbf{s}}(\mathbf{e}+\mathbf{a})\|_\infty \leq d_{G_*}+d_\beta+nd_sd_\omega = d_{\mathcal{D}}$, we have a collision in $\mathcal{D}$. Thus, by applying the law of total probability, we can deduce

that the overall probability of obtaining valid signatures out of aborted interactions is: $\theta_{\mathsf{abort}} \geq (1 - \frac{1}{|\mathcal{D}_\epsilon|})^4 (\frac{e^{-1/\phi} - e^{-4/\phi}}{1 - e^{-4/\phi}})^3 \theta$, which is noticeable if θ is noticeable. In conclusion, if $b = 1$, A's overall success probability is $\theta_{\mathsf{overall}} \geq \min(\theta_{\mathsf{col}}, \theta_{\mathsf{abort}})$, which is noticeable if θ is noticeable. □

By combining Theorem 4.4.6 with Theorem 2.5.5, we obtain the following:

Corollary 4.4.1. *The proposed* PBSS *is unforgeable if solving* $\mathsf{Ring} - \mathsf{SVP}_{\gamma,\infty}$ *is hard in the worst case, for approximation factors* $\gamma \geq 16 d_{\mathcal{D}} mn \log^2(n) = \tilde{O}(n^4)$*, in ideal lattices of* $\mathcal{R}_q$.

Remark 9. As a consequence of Theorem 4.4.6, if we require that $q_{\mathsf{sig}} = o(n)$, our construction benefits from the subexponential hardness of ideal lattice problems.

Remark 10. Because **e** is reduced modulo $2d_\epsilon + 1$ in Step 3 of our signing protocol, we have a milder worst-case hardness assumption of $\tilde{O}(n^4)$, compared with the BSS from [159], which is based on the worst-case hardness of $\mathsf{Ring} - \mathsf{SVP}$ for approximation factors in $\tilde{O}(n^5)$. We believe that this "trick" could also be used on [159] to improve the hardness assumption therein.

Remark 11. Notice that our proof also covers the case in which the forger U^* attempts to forge the common part information info (i.e., $k_{\mathsf{info}} = 0$). Alternatively, one could cover this case by resorting to the modular technique of "ID-reduction" [138], and by exploiting the witness-indistinguishability of our scheme.

Leakage Resilience

In proving leakage-resilience for our scheme, we rely on the core observation of [109], which states that any collision-resistant hash function (our underlying hash-function is proven collision-resistant in [125]) is a leakage-resilient one-way hash function when certain conditions are imposed on the leakage oracle (these conditions are necessary because the recent work of [113] shows that for some leakage scenarios, leakage-resilience is impossible to achieve). This observation is also used by other works to construct leakage-resilient primitives [109, 159].

In the next theorem, we establish leakage resilience for our construction. In proving leakage resilience, we will show that the secret key's conditional min-entropy: $H_\infty(\mathsf{sk} \mid \mathsf{Leak}(\mathsf{sk})) = \min_{\mathsf{sk}'}\{-\log(\Pr[\mathsf{sk} = \mathsf{sk}' \mid \mathsf{Leak}(\mathsf{sk})])\}$ is large enough for the scheme to be secure. The proof closely follows the corresponding proof of [159], with the additional observation that $\mathbf{Z} = \mathcal{F}(\mathsf{info})$ is not related to the signer's secret key, and thus does not leak information about $\hat{\mathbf{s}}$.

Theorem 4.4.7. *(Leakage Resilience) Let $c_m = \omega(1)$ and let $L := \log(|\mathcal{D}_s|^m) = mn\log(2d_s+1)$ denote the length of the signer's secret key in the proposed* PBSS. *Given $\mathbf{S} = \mathsf{h}(\hat{\mathbf{s}})$ and a total secret-key leakage $f(\hat{\mathbf{s}})$ of $\lambda = \delta L = (1 - o(1))L$ bits, the conditional min-entropy H_∞ of $\hat{\mathbf{s}}$, is positive with overwhelming probability.*

Proof. We follow the same conservative approach as in [159] and treat the public key $\mathbf{S}$ as additional leakage. Notice that $\mathbf{Z} = \mathcal{F}(\mathsf{info})$ is not related to the signer's secret key, and thus we do not treat it as a source of additional leakage for $\hat{\mathbf{s}}$. We define the function $g(\hat{\mathbf{s}}) := f(\hat{\mathbf{s}})\|\mathbf{S}$ with a total tolerated leakage of at most $\lambda' = \lambda + n\log(q)$ bits. Next, apply Lemma 1 from [109] to g, λ', and $H' = 1$, with $\hat{\mathbf{s}}$ being the random variable. Because $H = L = mn\log(2d_s + 1)$, we have:

$$\Pr[g(\hat{\mathbf{s}}) \in Y] \geq 1 - 2^{\lambda' - H + H'} = 1 - 2^{\lambda + n\log(q) - L + 1}, \tag{4.6}$$

which we want to be $\geq 1 - 2^{-p(n)}$. For any function $p(n)$ such that $\omega(\log(n)) \leq p(n) \leq O(n\log(n))$, we bound the relative leakage from above by $\delta \leq 1 - \frac{p(n)+n\log(q)+1}{L} = 1 - \frac{\Theta(n\log(n))}{c_m\Theta(n\log(n))} = 1 - \frac{1}{\omega(1)} = 1 - \frac{1}{o(1)}$. As a result, (4.6) becomes: $\Pr[g(\hat{\mathbf{s}}) \in Y] \geq 1 - 2^{(1-\frac{p(n)+n\log(q)+1}{L})L + n\log(q) - L + 1} = 1 - 2^{p(n)}$. Thus, $\delta L = (1 - o(1))L$ leakage bits yield a non-zero conditional min-entropy with overwhelming probability $1 - 2^{-p(n)} \geq 1 - 2^{-\omega(\log(n))}$. □

Remark 12. From Theorem 6, we see that if we additionally require that $c_m = \omega(1)$ (e.g., by choosing $c_m = \log(n)$) for $m := \lfloor c_m \log(q) \rfloor + 1$, then our PBSS retains its quasi-optimal performance and is also leakage-resilient.

4.5 Additional Security Properties

In this Section, we discuss several extensions of the classic security model of PBSS that are applicable to our construction. We consider honest-user unforgeability, selective-failure blindness, and dishonest-key blindness. To the best of our knowledge, none of these properties have previously been examined in the context of PBSS.

4.5.1 Dishonest-key Partial Blindness

In the definition of (partial) blindness, we implicitly assumed that the signer generates its secret and public keys through the scheme's key generation algorithm. However, we would like to ensure that partial blindness still holds even when the signer maliciously selects its keys. Our scheme's partial blindness proof does not rely on any specific properties of the public key and thus satisfies this strengthened notion of partial-blindness as well.

4.5.2 Selective-failure Partial Blindness

The notion of partial blindness does not cover cases in which the protocol has to be aborted prematurely. However, we would like to ensure that partial blindness also holds in cases where the signer is able to cause one of the protocol executions to abort by choosing one of the messages μ_0 or μ_1 from some secret distribution. Our scheme is selective-failure partially-blind because it makes use of the transformation of Construction 4.3.1. Indeed, the signer's view is limited to commitments on the messages he signs, and uncovering them would require it to break com's hiding property. This modification comes at the expense of only a negligible computational overhead. By extension, our scheme is also multi-execution selective-failure partially-blind.

4.5.3 Honest-user Unforgeability

Our scheme can easily be modified to use the transformation of Construction 4.3.2. The user commits to $\mu \| r'$ for some $r' \leftarrow_{\$} \{0,1\}^n$ instead of μ during Step 2. If any restarts occur during Step 2, r' needs to be resampled as well. Finally, r' will be

included in the final signature and the verification condition becomes:

$$\boldsymbol{\omega} + \boldsymbol{\delta} \pmod{2d_{\epsilon} + 1} = \mathsf{H}\big(\mathsf{h}(\hat{\mathbf{z}}) + \boldsymbol{\omega}\mathbf{S}, \mathsf{h}(\hat{\boldsymbol{\sigma}}) + \boldsymbol{\delta}\mathcal{F}(\mathsf{info}), \mathcal{F}(\mathsf{info}), \mathsf{com}(\mu \| r'; r)\big).$$

4.6 Conclusions, Open Problems and Future Work

In this chapter, we presented the first leakage-resilient, lattice-based PBSS in the literature. Our construction has the same 4-move structure and uses a commitment scheme like the scheme from [159]. Its performance is quasi-optimal and its security is proven in the ROM under milder worst-case ideal lattice assumptions compared to [159]. Besides being quantum-resistant, our construction is also honest-user unforgeable, selective-failure blind, dishonest-key blind, and can withstand sub-exponential-time attacks, and limited side-channel attacks against the signer's secret key thanks to its leakage resilience.

Following the publication of this chapter's construction, the work of [98] pointed out a very subtle but essential flaw in the proof strategy employed by a wide range of proposals [159, 21, 20, 42], including the present construction. In particular, it states that the scheme's witness indistinguishability alone is insufficient to infer that the pair $(\hat{\mathbf{x}}, \hat{\mathbf{x}}')$ obtained by rewinding the adversary with partially different random oracles [32] is a non-trivial collision for h. Witness indistinguishability merely implies that there exist two distinct secret keys sk, sk', leading to identical protocol transcripts (Lemma 8 from [155]). Hence, constructing partially-blind signature schemes from lattice assumptions remains an open problem. Here, an interesting research direction would be to investigate whether the modular framework of [97, 98] can be further extended to render partially-blind signature schemes.

Chapter 5

A Framework for Blind Signatures with Revocable Sessions

5.1 Introduction

Blind signature Schemes (BSS) were pioneered by D. Chaum [53] and have since become a focal cryptographic primitive for many privacy-preserving applications such as e-cash [53], e-voting [158], wireless sensor networks [186], anonymous credentials [27] (used e.g.: in Microsoft's U-Prove technology [146]), and cryptocurrencies [184]. Informally, a BSS is an interactive protocol between a *signer* S (holding some secret key sk) and a *user* U (holding the corresponding public key pk and a message m). After the interaction is complete, U should obtain a valid signature σ on m. The protocol needs to be correct (i.e., the output signature can be verified using pk and m) and additionally, it needs to satisfy two properties: blindness (i.e., S does not obtain any information w.r.t. m) and one-more unforgeability (i.e., no more valid signatures can be created beyond the ones received after interacting with S). Since their conception, a plethora of BS constructions has been proposed under various hardness assumptions like RSA [53, 54, 55], DLP [50], bilinear pairings [87], etc. BS schemes can be rendered based on general complexity assumptions [75, 65, 104] but these result in schemes that are *very inefficient*. On the contrary, schemes proven secure in the ROM generally lead to much more feasible solutions. [97] presents a

general framework for rendering BS schemes, secure in the ROM, from *any* (perfect) linear hash function family. Unfortunately, the aforementioned works are based on hardness assumptions that are vulnerable to Shor's algorithm [169]. Given the rapid development in quantum computers, there is an increased interest for constructions secure under quantum-resistant assumptions.

Lattice-based cryptography is a prime candidate for the post-quantum era as it boasts a plethora of attractive features such as: worst-case to average-case reductions [14], simplicity, efficiency, parallelizability, and is suitable for a wide scope of (advanced) applications. A common phenomenon in lattice-based cryptography is that cryptographic constructions usually exhibit some form of a *noticeable correctness error*. This makes designing blind signature schemes from lattice assumptions a very challenging task because security arguments from the number-theoretic world do not trivially carry over (if at all), making security analysis quite arduous. The first attempt towards a lattice-based BSS in the literature was made in [159] in an effort to adapt the Okamoto-Schnorr framework [139] to the lattice setting, using Lyubashevsky's identification scheme [122] as its basis. The proposed scheme addresses the correctness error with an additional move which serves as a means of allowing the user to prove that it did not obtain a valid signature from a session, thus triggering a session restart. However, as [98] showed, the unforgeability proof is flawed due to an incomplete argument when applying the Forking Lemma [30]. Later works [21, 116, 42, 19, 143] also share this flaw in their unforgeability proofs. More recently, [98] proposed the first provably secure BSS in the ROM as an instantiation of a generic framework for designing blind signature schemes from linear hash function families with noticeable correctness error. However, in order to reduce the correctness error of the resulting BS scheme, [98] relies on trees of commitments from [19] which results in a blow-up in the number of bits that need to be transmitted per session. The more recent work of [12] also deviates from the idea of correctness error by following a completely different paradigm from [98] in order to construct a round-optimal BS scheme, secure in the ROM.

We therefore ask the following question: Can we do something *simpler* and that is

more *natural* to the lattice setting (which is more in line with classical, but incorrect works in this area [159, 21])?

Result 1: We propose for the first time, a general framework that is capable of capturing schemes like [159, 21]. In doing so, we first revisit the definitions for blindness and one-more unforgeability of BSS to take into account that a signing session may fail to produce a signature. In addition, like in [159], we endow the user with the ability to prove to the signer that he failed to obtain a signature from a signing session through a *proof-of-revocation* (PoR) which if deemed valid, results in a session being *revoked* (i.e., counts as if it did not take place).

Result 2: We describe a generic 4-move construction of BSRS schemes from *any* linear hash function family with *noticeable correctness error* and prove its security in the ROM. We aim for our construction to be modular, straightforward to implement and to avoid the use of any heavy machinery like NIZK proofs.

Result 3: We demonstrate the efficacy of our novel framework, by providing a concrete instantiation of our generic scheme from lattice assumptions (namely, $\mathsf{R-SIS}$). Surprisingly, despite having an additional move, our scheme only needs to transmit a fraction (of about 22.2) of the amount of bits that [98] does for the same level of 128-bit security. Furthermore, our scheme achieves signatures that are $\approx 12\%$ shorter for the same level of bit security.

Unfortunately, we can only prove blindness in the weaker *honest signer model* [104]. It is interesting to see if we can construct a scheme that is secure in the presence of *malicious signers* [75]. Moreover, our construction comes with an exponential security loss in the reduction from the underlying $\mathsf{R-SIS}$ assumption, which is inherent from the proof technique of Pointcheval and Stern [155]. This severely restricts the number of signatures that can be issued per public key to a poly-logarithmic amount. While our scheme might not be practical by itself (our instantiation has signature sizes in the order of megabytes), we still believe that our modular framework is of theoretical interest for designing and analyzing blind signature schemes from lattice assumptions and could potentially lead to more practical solutions. Finally, another restriction inherited from the Okamoto-Schnorr paradigm is that in order to achieve *concurrent*

unforgeability under revokes, the signer can only issue a poly-logarithmic (in the security parameter) number of signatures per public key [98]. However, increasing the amount to polynomial seems within reach [152, 107].

5.1.1 Technical Overview

Definitional Model. Our starting point is a remark from [98] which states that for BS schemes with noticeable correctness error [159, 21, 19], the standard definition of unforgeability [104] is significantly *weakened*. The reason behind this is that an interaction with the signer may fail to yield a valid signature (with noticeable probability), *even for honest adversaries*. For example, if N interactions take place but the forger only obtains $K < N$ valid signatures, it still has to come up with *at least* $N + 1$ valid signatures in order to win (i.e., it has to make up for the $N - K$ failed sessions on its own before even producing any forgeries). The adversary's position can thus become needlessly difficult within this context. This motivates the question: what if the adversary could somehow prove to the signer that it did not obtain a signature?

We define *blind signatures with revocable sessions* (BSRS) as a natural extension of canonical (i.e., three-move) BS schemes [97, 98] with a fourth move. The user is capable of producing a PoR through which a session can be *revoked*, allowing for a new interaction to take place. This PoR should only work iff the user truly failed to obtain a signature. The fourth move helps the signer ascertain the truth of the user's claim that it did not obtain a valid signature despite following the protocol. Since a session can be revoked, the definitions for blindness and unforgeability need to take this into account.

First, recall the definition of *blindness* in the (honest-signer) model [104]. The adversarial signer picks messages msg_0, msg_1 of its own choosing and it generates a secret key/public key pair using the BS scheme's key generation algorithm. It then interacts with two user sessions, holding m_b and msg_{1-b}, respectively, after a secret coin toss b. After both interactions are completed, the signer obtains a pair of signatures (σ_0, σ_1), where possibly $(\sigma_0, \sigma_1) = (\perp, \perp)$ if either session failed to produce a signature and it has to correctly guess b. This setting is limited in that it cannot

capture our setting where the user can output a PoR π along with a signature σ, potentially triggering a restart.[1] We extend the definition as follows: if a session outputs $\sigma = \perp$, the signer learns which session failed and is given the corresponding PoR π. If π is valid, the session is reinitialized. This process keeps repeating until *both* sessions have obtained valid signatures, and *only then* is the signer given (σ_0, σ_1). The signer wins if it can guess b. We call this property *blindness under revokes*.

Next, recall the definition of *unforgeability* [104]. The signer generates its keys via the key generation algorithm and then interacts with an honest signer. Once the interaction is complete, the user wins if it can output at least one more valid signature than there were interactions. This definition only has meaning if a session *always* outputs a valid signature. However, this is not the case in our setting because a session may have been revoked. Therefore, the "winning condition" must *exclude* any sessions that have been revoked. We call this property *unforgeability under revokes*. Notice that this definition is more flexible in the sense that it does not force an adversary to make up for any failed sessions. Instead, the adversary can simply prove that it did not obtain a signature, and the challenger will exclude these sessions. Most importantly, this eliminates the need to reduce the blind signature scheme's correctness error to negligible (like [98] does).

Generic Constructions. At a high-level, our generic constructions (see Section 5.3 for details) follow the typical three-move structure of identification schemes to provide a proof of knowledge of a secret key $\mathsf{sk} \in \mathcal{D}_{sk}$ s.t. $\mathsf{F}(\mathsf{sk}) = \mathsf{pk}$, where pk is the public key.

At the outset, the signer transmits a commitment $\mathbf{R} := \mathsf{F}(\mathbf{r})$ to a random vector $\mathbf{r} \in \mathcal{D}_r$. Upon receiving $\mathbf{R}$, the user samples masking parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}$ and computes a challenge $\mathbf{c}$ as a function (involving H) of $\mathbf{R}, \boldsymbol{\alpha}, \boldsymbol{\beta}$, and the message-to-be-signed *msg*, which it sends to the signer. The signer receives $\mathbf{c}$ and responds with $\mathbf{s} := \mathbf{c} \cdot \mathsf{sk} + \mathbf{r}$. By the linearity of F, $\mathsf{F}(\mathbf{s}) = \mathsf{pk} \cdot \mathbf{c} + \mathbf{R}$ should hold. Furthermore, the user "unblinds" $(\mathbf{s}, \mathbf{c})$ with the help of the blinding parameters to obtain the final signature $(\mathbf{s}', \mathbf{c}')$ on *msg*. Because of the correctness error of F, even if both parties behave honestly, there is no

[1] Indeed, the signer interacts with each user session *at most* once.

guarantee that the user will obtain a valid signature when unblinding.[2] This is the intuition behind having the additional move serving as a means of allowing the user to *prove* through a PoR that a protocol restart is necessary. Here, it is important to ensure that the PoR does not leak any information about the message, that the signer can detect a cheating user, and that it is infeasible for the user to come up with a fake PoR for a session from which it obtained a valid signature. We address these issues by having the user commit to the actual message with the help of a *collision-free* hash function during Step 2. Moreover, a PoR comprises of only the necessary random values that the user sampled during Step 2, along with its commitment to m. This allows the signer to trace the computations performed on the user's side and to revoke a session if necessary (if a PoR is invalid, the signer does nothing).

We now give some basic intuition behind the proof of Theorem 5.3.5. In brief, we reduce our scheme's unforgeability under revokes from the collision-resistance of the underlying LHF. At a high level, we follow the steps of [98] by running an adversary M against unforgeability under revokes, twice. On the second run, we use the same input I and user randomness ω, but partially change the vector from which random oracle responses come from $\mathbf{h}$ to $\mathbf{h}'$. The intuition here is to obtain two distinct (and short) preimages χ, χ' for F, forming a collision. Both χ, χ' depend on M's internal state and our main tool for showing that $\chi \neq \chi'$ with high probability is the Subset Forking Lemma from [98]. Because we distinguish between closed or revoked sessions, applying the probabilistic method from [98] is not straightforward and requires a very careful analysis. For example, this comes up when we need to bound the event of obtaining the most likely short preimage of F on the first execution of M. For revoked sessions in particular, the user will need to separately commit on the blinding parameters using a *collision-free* hash function G for our argument to go through. Of course, this also has implications on our proof of blindness under revokes when we need to show that each signature could have resulted from either session transcript (for some appropriately distributed user-side randomness).

[2] This is due to applying the well-known rejection sampling technique.

5.1.2 Related work, problems and limitations

The first provably secure BS scheme from lattice assumptions was proposed in [98] along with a modular framework for constructing canonical BS schemes from any linear hash function family with noticeable correctness error. As we discussed earlier, the authors reduce the correctness error induced by the underlying LHF with an aborting technique from [19] which involves trees of commitments. However, this leads to a more complex and less natural solution in the lattice setting.

A very recent work [12] follows a different paradigm from [139] and proposes a very efficient and round-optimal construction of blind signature schemes in the ROM. In brief, their scheme works as follows: on input the verification key $\mathsf{Sig.vk}$ of some signature scheme Sig, and a message m, the user samples a homomorphic encryption key $\mathsf{HE.sk} := (\bar{\mathbf{s}}, \mathbf{e})$ from the LWE error distribution. The corresponding decryption key $\mathsf{HE.pk}$ is computed via a hash function H (modeled as a random oracle) on input $\mathsf{Sig.vk}$ and a session identifier sid, and using a linear operation involving $(\bar{\mathbf{s}}, \mathbf{e})$. The user then homomorphically encrypts the message $\mathsf{CT}_m := \mathsf{HE.Enc}(\mathsf{HE.pk}, m)$ just as in [43] and sends $(\mathsf{HE.pk}, \mathsf{CT}_m)$ to the signer, along with respective NIZK proofs of well-formedness. On input $\mathsf{Sig.sk}$, the signer verifies the two NIZK proofs, and then homomorphically evaluates Sig's signing algorithm on CT_m to produce and output an encrypted signature CT_σ. Upon receiving CT_σ, the user homomorphically decrypts it to obtain the final blind signature σ. Concretely, if Sig is instantiated with a (abort-free) variant of Dilithium-G [68], the estimated signature size is ≈ 3 KB for ≈ 128 bits of security in the ROM. Furthermore, their adversaries are limited to obtaining 256 signatures.

While their construction shows great promise for practical use, there are several serious roadblocks barring implementation. First, by design, the signing algorithm $\mathsf{Sig.Sign}_{\mathsf{Sig.sk}}$ acts as a circuit on which the ciphertext must be homomorphically evaluated by the signer. However, $\mathsf{Sig.Sign}_{\mathsf{Sig.sk}}$ internally uses a hash function modeled as a random oracle and as such, this hash function must also be evaluated *homomorphically*. Unfortunately, choosing such a hash function is a highly non-trivial matter. While the authors provide some basic intuition on how to approach the prob-

lem, this still remains a major open problem. Second, the homomorphic encryption scheme itself needs to be chosen carefully and in a way capable of handling the diverse formats involved during the homomorphic signing. Ensuring compatibility between the homomorphic encryption format and the format needed for evaluating the hash function seems particularly tricky to ensure when instantiating Sig.Sign with their proposed abort-free variant of Dilithium-G's signing algorithm[3]. Finally, Dilithium-G compresses the signature elements with Huffman codes. However, in [12] this will need to be done under the homomorphic encryption layer, which could be expensive to implement. Addressing all of these problems simultaneously will likely require a very masterful implementation.

5.1.3 Relation to impossibility results for blind signatures.

In [78], the authors give an impossibility result for 3-move BS. Their main result states that finding black-box reductions from unforgeability to non-interactive problems is hard, unless the problems involved were already easy, and it captures many prior BS proposals [5, 53, 155]. However, it does not apply to instantiations of our 4-move framework because it is *impossible* for one to tell if the user truly obtained a valid signature within 3 moves since the user has not revealed the masking parameters that he uses to produce his final signature, and may need to restart (to retain its anonymity). This is important because under our scheme, the components of the final signature must satisfy a "shortness" condition for the signature to be deemed valid. In [108], the authors give an impossibility that rules out black-box constructions of blind signatures from one-way functions. This impossibility result also does not apply to our construction because the underlying hash function is collision resistant [125]. Finally, the meta-reduction of [27] also does not apply to our work because the hard computational problem underlying our scheme's unforgeability (R – SIS) does not rely on a unique-witness relation between public and secret keys.

[3]Fisher-Yates shuffle no longer works for mapping to Dilithium-G's challenge space because this task must be done *homomorphically*.

5.1.4 Organization

In Section 5.2 we recall the notion of linear hash function families with correctness error and introduce for the first time the notion of blind signatures with revocable sessions (BSRS) along with their syntax and security model. Section 5.3 provides a detailed description of our generic construction of BSRS schemes from any linear hash function family with correctness error. Our generic constructions are formally proven secure in the ROM. Finally, in Section 5.4 we provide a concrete instantiation of our generic construction from lattice assumptions.

In this chapter, we reprint the main construction in [144], of which the dissertation author was the main investigator and author.

5.2 Preliminaries

To facilitate the presentation of our proofs, in this chapter we adopt the standard notion of prose-based security games [32, 170]. We denote the binary output of a game $\mathbf{G}$ with an adversary A as $\mathbf{G}^{\mathsf{A}}$ and say that A wins in $\mathbf{G}$ iff $\mathbf{G}^{\mathsf{A}} = 1$.

5.2.1 Linear Hash Function Families with Correctness Error

We recall the definition of linear hash function families with correctness error which was recently defined in [98], and is a generalization of linear hash function families with perfect correctness [97].

Definition 5.2.1. (Linear Hash Function Family with Correctness Error) A linear hash function family LHF is a tuple of algorithms $(\mathsf{PGen}, \mathsf{F})$. On input the main security parameter, algorithm PGen returns parameters par, implicitly defining sets

$$\mathcal{S} = \mathcal{S}(\mathsf{par}), \quad \mathcal{D} = \mathcal{D}(\mathsf{par}), \quad \text{and} \quad \mathcal{R} = \mathcal{R}(\mathsf{par}),$$

where $\mathcal{S}$ is a ring such that $\mathcal{D}$ and $\mathcal{R}$ form modules over $\mathcal{S}$. Parameters par also

implicitly define 9 filter sets

$$\mathcal{S}_{\mathsf{xxx}} \subseteq \mathcal{S}, \mathsf{xxx} \in \{c', c, \beta\} \quad \text{and} \quad \mathcal{D}_{\mathsf{yyy}} \subseteq \mathcal{D}, \mathsf{yyy} \in \{sk, s, r, s', \alpha\}$$

For the rest of the write-up, we will assume that par is fixed and implicitly given to all algorithms. Note that, for linear hash function families with *no* correctness error, we have $\mathcal{S}_{\mathsf{xxx}} = \mathcal{S}$ and $\mathcal{D}_{\mathsf{yyy}} = \mathcal{D}$.

We assume an algorithm F that implements a module homomorphism from $\mathcal{D}$ to $\mathcal{R}$ i.e.,

$$\mathsf{F}(s_1 \cdot x + s_2 \cdot y) = s_1 \cdot \mathsf{F}(x) + s_2 \cdot \mathsf{F}(y), \forall s_1, s_2 \in \mathcal{S} \quad \text{and} \quad \forall x, y \in \mathcal{D}.$$

Moreover, we assume that F has $\lambda+1$ bits of min-entropy; i.e., for all (even unbounded) algorithms A, we have: $\Pr_{x \leftarrow_\$ \mathcal{D}, y \leftarrow_\$ \mathsf{A}(\mathsf{par})}[\mathsf{F}(x) = y] \leq 2^{-(\lambda+1)}$.

We also recall certain properties of linear hash functions that will be essential for the security analysis of our blind signature. We start by defining torsion-freeness and regularity which we will use when proving one-more unforgeability for our scheme.

Definition 5.2.2. (Torsion-free Elements from the Kernel) We say that LHF has a torsion-free element from the kernel if for all par generated with PGen, there exist $z^* \in \mathcal{D} \setminus \{0\}$ s.t. (i) $\mathsf{F}(z^*) = 0$; and (ii) for all $c_1, c_2 \in S_c$ satisfying $(c_1 - c_2) \cdot z^* = 0$ we have $c_1 = c_2$.

Note that the existence of such an element implies that F is not a 1-1 mapping.

Definition 5.2.3. (Regularity) We call LHF (ε, Q')-regular, if for all par generated with PGen, there exist sets $\mathcal{D}'_{sk}, \mathcal{D}'_r$ and a torsion-free element from the kernel z^* s.t.

$$\frac{|\mathcal{D}'_{sk}|}{|\mathcal{D}_{sk}|} \cdot \left(\frac{|\mathcal{D}'_r|}{|\mathcal{D}_r|}\right)^{Q'} \geq 1 - \varepsilon/4,$$

where

$$\mathcal{D}'_{sk} := \{sk \in \mathcal{D}_{sk} : sk + z^* \in \mathcal{D}_{sk}\}$$

and

$$\mathcal{D}'_r := \{r \in \mathcal{D}_r : r + cz^* \in \mathcal{D}_r, \forall c \in S_c\}.$$

We then define *enclosedness error* for LHF, which is directly linked to the correctness error of blind signature schemes like ours and [98]. Setting a small enclosedness error makes it easier to construct a blind signing protocol with a small correctness error.

Definition 5.2.4. (Enclosedness Errors) We say that LHF has enclosedness errors $(\delta_1, \delta_2, \delta_3)$ if for all $\mathsf{par} \in \mathsf{PGen}(1^\lambda), c' \in S_{c'}, c \in S_c, s \in \mathcal{D}_s, sk \in \mathcal{D}_{sk}$,

$$\Pr_{\beta \leftarrow_\$ S_\beta}[\beta + c' \notin S_c] < \delta_1, \Pr_{r \leftarrow_\$ \mathcal{D}_r}[c \cdot sk + r \notin \mathcal{D}_s] < \delta_2, \text{and } \Pr_{\alpha \leftarrow_\$ \mathcal{D}_\alpha}[\alpha + s \notin \mathcal{D}_{s'}] < \delta_3.$$

For our blindness proof, we need to define an additional property for LHF called *smoothness*.

Definition 5.2.5. (Smoothness) We say that LHF is smooth if the following conditions hold for all $\mathsf{par} \in \mathsf{PGen}(1^\lambda)$:

(S1) For all $s \in \mathcal{D}_s$ and $s' \in \mathcal{D}_{s'}$, we have $\|s' - s\|_\infty \in \mathcal{D}_\alpha$.

(S2) For all $s_1, s_2 \in \mathcal{D}_s$ and random variables $\alpha^* \leftarrow_\$ \{\alpha \in \mathcal{D}_\alpha | \alpha + s_1 \in \mathcal{D}_{s'}\}, \hat{\alpha} \leftarrow_\$ \{\alpha \in \mathcal{D}_\alpha | \alpha + s_2 \in \mathcal{D}_{s'}\}$ we have that $\alpha^* + s_1$ and $\hat{\alpha} + s_2$ are identically distributed.

(S3) For all $s_1, s_2 \in \mathcal{D}_s$ and random variables $\underline{\alpha^*} \leftarrow_\$ \{\alpha \in \mathcal{D}_\alpha | \alpha + s_1 \notin \mathcal{D}_{s'}\}, \underline{\hat{\alpha}} \leftarrow_\$ \{\alpha \in \mathcal{D}_\alpha | \alpha + s_2 \notin \mathcal{D}_{s'}\}$ we have that $\underline{\alpha^*} + s_1$ and $\underline{\hat{\alpha}} + s_2$ are identically distributed.

(S4) For all $c' \in S_{c'}$ and $c \in S_c$, we have $\|c - c'\|_\infty \in S_\beta$.

(S5) For all $c'_1, c'_2 \in S_{c'}$ and random variables $\beta^* \leftarrow_\$ \{\beta \in S_\beta | \beta + c'_1 \in S_c\}, \hat{\beta} \leftarrow_\$ \{\beta \in S_\beta | \beta + c'_2 \in S_c\}$ we have that $\beta^* + c'_1$ and $\hat{\beta} + c'_2$ are identically distributed.

(S6) For all $c'_1, c'_2 \in S_{c'}$ and random variables $\underline{\beta^*} \leftarrow_\$ \{\beta \in S_\beta | \beta + c'_1 \notin S_c\}, \underline{\hat{\beta}} \leftarrow_\$ \{\beta \in S_\beta | \beta + c'_2 \notin S_c\}$ we have that $\underline{\beta^*} + c'_1$ and $\underline{\hat{\beta}} + c'_2$ are identically distributed.

We finally define collision-resistance for LHF. Unlike standard collision-resistance as defined in [97], we additionally require the candidate solution to be short.

Definition 5.2.6. (Collision Resistance) We define the collision resistance game for a linear hash function family LHF as follows:

Game $\mathbf{CR}_{\mathsf{LHF}}$:

- **Setup.** $\mathbf{CR}_{\mathsf{LHF}}$ samples parameters via $\mathsf{par} \leftarrow_{\$} \mathsf{LHF.PGen}(1^{\lambda})$ and runs adversary A on input par.
- **Output Determination.** When A outputs (x_1, x_2), $\mathbf{CR}_{\mathsf{LHF}}$ returns 1 iff: (i) $\mathsf{F}(x_1) = \mathsf{F}(x_2)$, (ii) $x_1 \neq x_2$, and (iii) $x_1, x_2 \in \mathcal{D}'$, where

$$\mathcal{D}' := \{s' - c' \cdot sk : s' \in \mathcal{D}_{s'}, c' \in S_{c'}, sk \in \mathcal{D}_{sk}\} \subseteq \mathcal{D}. \tag{5.1}$$

  Otherwise, it returns 0.

We define the advantage of A in game $\mathbf{CR}_{\mathsf{LHF}}$ as $\mathbf{Adv}^{\mathbf{CR}}_{\mathsf{LHF}}(\mathsf{A}) := \Pr[\mathbf{CR}^{\mathsf{A}}_{\mathsf{LHF}} = 1]$ and denote its running time by $\mathbf{Time}^{\mathbf{CR}}_{\mathsf{LHF}}(\mathsf{A})$. We say that LHF is (ε, t)-collision resistant if for all adversaries A, we have that

$$\mathbf{Time}^{\mathbf{CR}}_{\mathsf{LHF}}(\mathsf{A}) \leq t \quad \text{and} \quad \mathbf{Adv}^{\mathbf{CR}}_{\mathsf{LHF}}(\mathsf{A}) \leq \varepsilon$$

5.2.2 Blind Signature Schemes with Revocable Sessions

The standard security properties of BS schemes are *blindness* and *one-more unforgeability* (OMUF) [104, 155]. Blindness requires that the adversarial signer must not obtain any information on the signed messages, while one-more unforgeability states that an adversarial user cannot obtain more signatures than the number of interactions with the signer.

Observe that the notions of one-more unforgeability and blindness are not suitable to describe the security of BS schemes with noticeable correctness error (e.g.: lattice-based). Indeed, even if a user behaves honestly, it may be unable to obtain a valid signature with noticeable probability. If l interactions are completed, the user still has to come up with at least $l + 1$ signatures in order to win. We relax the definition by giving the user the ability to *prove* that a session did not result in obtaining a

valid signature. These sessions are said to be *revoked* and are excluded from the final count. If a session is revoked, the signer and the user can restart a session, hopefully leading to the generation of a valid signature. This however is not compatible with the definition of blindness where the signer only gets to interact with each session *at most once* before attempting to link a pair of signatures to their respective transcripts. Motivated by these concerns, we define for the first time, blind signatures with revocable sessions (BSRS) which consider the possibility of revoking a session through a proof and restarting a session. We further elaborate on the differences of our definitional model below.

We now define the syntax of blind signature schemes with revocable sessions.

Definition 5.2.7. (Blind Signature Schemes with Revocable Sessions) A blind signature scheme with revocable sessions is a four-move blind signature scheme BSRS and is comprised by six algorithms BSRS = (PG, KG, S, U, CheckProof, Ver).

- BSRS.PG(1^λ) : On input the main security parameter λ, the randomized *parameter generation algorithm* generates and returns scheme parameters par.
- BSRS.KG(par) : On input scheme parameters par, the randomized *key generation algorithm* generates and outputs a secret/public key pair (sk, pk). Public key pk implicitly defines the scheme's *challenge space* $C := C(\mathsf{pk})$ and is known to all parties.
- BSRS.S : The *signer algorithm* is split into two sub-algorithms BSRS.S = (S_1, S_2), where:
  1. BSRS.S_1(sk) : On input the secret key sk, the randomized algorithm BSRS.S_1 returns a commitment R and a signer state stS.
  2. BSRS.S_2(sk, R, c, stS) : On input the secret key sk, a commitment R, a user challenge c, and a signer state stS, the deterministic algorithm BSRS.S_2 returns a response s, where possibly $s = \perp$.
- BSRS.U : The *user algorithm* is split into two algorithms BSRS.U = (BSRS.U_1, BSRS.U_2), where:

1. $\mathsf{BSRS.U}_1(\mathsf{pk}, R, msg)$: On input the public key pk, a commitment R, and a message msg from message space $\mathcal{M}$, the randomized algorithm $\mathsf{BSRS.U}_1$ returns a challenge c from challenge space C, and a user state stU.
2. $\mathsf{BSRS.U}_2(\mathsf{pk}, R, c, s, msg, stU)$: On input the public key pk, a commitment R, a challenge c, a response s, a message msg, and the user state stU, the deterministic algorithm $\mathsf{BSRS.U}_2$ outputs a signature σ and a proof-of-revocation (PoR) π, where either $\sigma = \perp$ or $\pi = \perp$ (but not both).

- $\mathsf{BSRS.Ver}(\mathsf{pk}, \sigma, msg)$: On input the public key pk, signature σ, and message $msg \in \mathcal{M}$, the deterministic *verification algorithm* returns 1 if the signature is valid, and 0 otherwise. We assume that BSRS.Ver returns 0 if $\sigma = \perp$.
- $\mathsf{BSRS.CheckProof}(\mathsf{pk}, \mathsf{tx}, \pi, stS)$: Is a deterministic auxiliary algorithm for checking the validity of a given PoR w.r.t. a public key and a session transcript. On input the public key pk, a session transcript tx[4], a PoR π, and the signer's state stS, it either returns 1 (valid) or 0 (invalid).

Algorithms $\mathsf{BSRS.S}_2$, BSRS.CheckProof, and $\mathsf{BSRS.U}_2$ are considered deterministic w.l.o.g. because randomness can be transmitted through the states of the signer and user, respectively. Figure 5.1 depicts the flow of interaction between a signer and a user during a signing session.

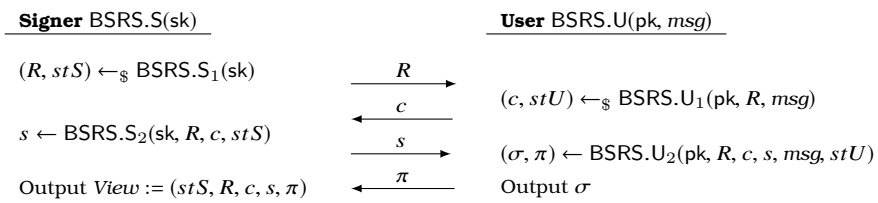

Figure 5.1: Interaction flow between signer BSRS.S and user BSRS.U.

Definition 5.2.8. (Correctness) We say that BSRS = (PG, KG, S, U, Ver, CheckProof) has *correctness error* δ, if for all messages msg from message space $\mathcal{M}$, $\mathsf{par} \in \mathsf{PG}(1^\lambda)$,

[4]We use (R, c, s) and tx interchangeably to denote the session transcript throughout this chapter.

$(\mathsf{sk}, \mathsf{pk}) \in \mathsf{KG}(\mathsf{par})$, we have

$$\Pr_{(View,\sigma)\leftarrow_{\$}\langle \mathsf{BSRS.S}(\mathsf{sk}),\mathsf{BSRS.U}(\mathsf{pk},msg)\rangle}[\mathsf{BSRS.Ver}(\mathsf{pk}, msg, \sigma) = 1] \leq \delta$$

We now provide the security model for BSRS. In particular, we give definitions for blindness under revokes and one-more unforgeability under revokes.

Recall the definition of *blindness* in the (honest-signer) model [104]. The adversarial signer S^* picks messages msg_0, msg_1 from message space $\mathcal{M}$ of his own choosing and it generates a secret key/public key pair using the BS scheme's key generation algorithm. It then interacts with two user sessions, holding msg_b and msg_{1-b}, respectively, after a secret coin toss $b \leftarrow_{\$} \{0, 1\}$. After both interactions are completed, the signer obtains a pair of signatures (σ_0, σ_1), where possibly $(\sigma_0, \sigma_1) = (\perp, \perp)$ if either session failed to produce a signature. S^* wins if it can correctly guess b. We modify the definition to allow each user to output a PoR π along with a signature σ. Furthermore, if a session outputs $\sigma = \perp$, the signer learns which session failed and is given the corresponding PoR π. If π is valid, the session is reinitialized and a new interaction can take place with that particular user (this is done by invoking oracle Relnit). This process keeps repeating until *both* sessions have obtained valid signatures, and *only then* is the signer given (σ_0, σ_1). S^* wins if it can guess b. At the end of the game, S^*'s view consists of a list of tuples of the form (tx, π) of transcripts and PoRs for each session.[5] We now give the formal definition for *blindness under revokes*.

Definition 5.2.9. (Blindness under Revokes) We define blindness under revokes of a four-move BSRS against an adversary A via the following game:
Game $\mathbf{RBlind}_{\mathsf{BSRS}}(1^\lambda)$:

- **Setup.** $\mathbf{RBlind}_{\mathsf{BSRS}}(1^\lambda)$ generates parameters via $\mathsf{par} \leftarrow_{\$} \mathsf{BSRS.PG}(1^\lambda)$, and sk, pk via $(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{BSRS.KG}(\mathsf{par})$. It samples a random bit via $b \leftarrow_{\$} \{0, 1\}$ and sets $\boldsymbol{b}_1 := b$ and $\boldsymbol{b}_2 := 1 - b$. It then runs A on input $\mathsf{par}, \mathsf{sk}, \mathsf{pk}$.

- **Online Phase.** When A outputs messages $\tilde{msg}_0, \tilde{msg}_1$, $\mathbf{RBlind}_{\mathsf{BSRS}}(1^\lambda)$ assigns

[5] Notice that for the last tuple, $\boldsymbol{\pi} = \perp$.

$msg_0 := \tilde{msg}_b$ and $msg_1 := \tilde{msg}_{1-b}$ and sets $\mathbf{sess}_1 := \mathsf{init}$ and $\mathbf{sess}_2 := \mathsf{init}$. A has access to the following oracles:

1. Oracle $\mathsf{U}_1(sid, R)$: On input a session identifier sid and a commitment R, if $sid \notin \{1, 2\}$ or $\mathbf{sess}_{sid} \neq \mathsf{init}$, the oracle returns $\perp$. Otherwise, it marks the session as open via $\mathbf{sess}_{sid} := \mathsf{open}$, it sets $\mathbf{R}_{sid} := R$ and generates a state and a challenge as $(\mathbf{st}_{sid}, \mathbf{c}_{sid}) \leftarrow_{\$} \mathsf{BSRS.U}_1(\mathsf{pk}, \mathbf{R}_{sid}, msg_{\mathbf{b}_{sid}})$. It returns $(sid, \mathbf{c}_{sid})$ to the adversary.
2. Oracle $\mathsf{U}_2(sid, s)$: On input a session identifier sid and a response s, if the session is open, the game marks the session as closed and sets $\mathbf{s}_{sid} := s$. It then generates a signature/PoR pair as $(\boldsymbol{\sigma}_{\mathbf{b}_{sid}}, \boldsymbol{\pi}_{\mathbf{b}_{sid}}) \leftarrow_{\$} \mathsf{BSRS.U}_2(\mathsf{pk}, \mathbf{R}_{sid}, \mathbf{c}_{sid}, \mathbf{s}_{sid}, \mathbf{st}_{sid})$. If $\boldsymbol{\sigma}_{\mathbf{b}_{sid}} = \perp$, the oracle outputs $(sid, \mathsf{closed}, \boldsymbol{\pi}_{\mathbf{b}_{sid}})$. Otherwise, if both sessions are closed and have yielded valid signatures (i.e., $\boldsymbol{\sigma}_0, \boldsymbol{\sigma}_1 \neq \perp$), the oracle returns $(\boldsymbol{\sigma}_0, \boldsymbol{\sigma}_1)$. Otherwise, it returns (sid, closed).
3. Oracle $\mathsf{ReInit}(sid, R, c, s, \boldsymbol{\pi})$: On input a session identifier sid, a commitment R, a challenge c, a response s, and a PoR $\boldsymbol{\pi}$, it first checks that $\mathbf{sess}_{sid} = \mathsf{closed}$ (if not, the oracle simply terminates). It sets $\mathsf{tx} := (R, c, s)$ and runs $\mathsf{BSRS.CheckProof}(\mathsf{pk}, \mathsf{tx}, \boldsymbol{\pi})$ to compute a bit b^*. It sets $\mathbf{sess}_{sid} := \mathsf{init}$ iff $b^* = 1$. Otherwise, nothing is done.

- **Output Determination.** When A outputs a bit b', $\mathbf{RBlind}_{\mathsf{BSRS}}(1^\lambda)$ returns 1 iff $b = b'$. Otherwise, it returns 0.

We define the advantage of adversary A in game $\mathbf{RBlind}_{\mathsf{BSRS}}(1^\lambda)$ as $\mathbf{Adv}^{\mathbf{RBlind}}_{\mathsf{BSRS}}(\mathsf{A}) := \left|\Pr[\mathbf{RBlind}^{\mathsf{A}}_{\mathsf{BSRS}}(1^\lambda) = 1] - 1/2\right|$. We say that BSRS is statistically blind under revokes if for all adversaries A, $\mathbf{Adv}^{\mathbf{RBlind}}_{\mathsf{BSRS}}(\mathsf{A}) \approx 0$.

One-more unforgeability [104] states that an adversarial user U should be unable to produce even a single signature more than it should be able to learn through interacting with the signer S. Our definition of *one-more unforgeability under revokes* is similar to standard unforgeability of BSS [104]. However, because of the scheme's noticeable correctness error, a session might not yield a valid signature even for an

honest adversarial user. As such, (i) we provide (through oracle Revoke) the user with a means of *proving* to the signer that a session did not yield a valid signature, thus marking it as revoked (i.e., as if it never occurred), and (ii) we modify the game's winning condition to exclude revoked sessions from the required number of signatures that the adversary has to output (i.e., $Q_{\mathsf{S}_2}(\mathsf{A}) - Q_{\mathsf{rev}}(\mathsf{A}) + 1$ signatures instead of $Q_{\mathsf{S}_2}(\mathsf{A}) + 1$). More formally, we have:

Definition 5.2.10. (One-More Unforgeability under Revokes) We define the (concurrent) one-more unforgeability under revokes for a four-move BSRS against an adversary A via the following game:
Game $\mathbf{OMUF}_{\mathsf{BSRS}}(1^\lambda)$:

- **Setup.** $\mathbf{OMUF}_{\mathsf{BSRS}}(1^\lambda)$ generates parameters via $\mathsf{par} \leftarrow_\$ \mathsf{BSRS.PG}(1^\lambda)$ and sk, pk via $(\mathsf{sk}, \mathsf{pk}) \leftarrow_\$ \mathsf{BSRS.KG}(\mathsf{par})$. It initializes $sid := 0$ and invokes A on input par and pk.

- **Online Phase.** A can access the following oracles:

  1. Oracle S_1 : takes no input. It samples a fresh session identifier by setting $sid := sid + 1$ and $\mathbf{sSess}_{sid} := \mathsf{open}$ and generates $(\mathbf{st}_{sid}, \mathbf{R}_{sid}) \leftarrow_\$ \mathsf{BSRS.S}_1(\mathsf{sk})$. Then it returns $(sid, \mathbf{R}_{sid})$.

  2. Oracle $\mathsf{S}_2(sid, c)$: takes as input a session identifier sid and a challenge c. If $\mathbf{sSess}_{sid} \neq \mathsf{open}$, it returns $\perp$. Otherwise, it sets $\mathbf{sSess}_{sid} := \mathsf{closed}$ and generates the response via $\mathbf{s}_{sid} \leftarrow_\$ \mathsf{BSRS.S}_2(\mathsf{sk}, \mathbf{st}_{sid}, \mathbf{R}_{sid}, c)$ and it returns $\mathbf{s}_{sid}$.

  3. Oracle $\mathsf{Revoke}(sid, \boldsymbol{\pi})$: takes as input a session identifier sid and a PoR $\boldsymbol{\pi}$. If $\mathbf{sSess}_{sid} = \mathsf{closed}$, it computes $b := \mathsf{BSRS.CheckProof}(\boldsymbol{\pi})$ and it sets $\mathbf{sSess}_{sid} := \mathsf{revoked}$ iff $b = 1$. Otherwise, nothing is done.

- **Output Determination.** When A outputs tuples $(msg_1, \sigma_1), \ldots, (msg_{l(\mathsf{A})}, \sigma_{l(\mathsf{A})})$, $\mathbf{OMUF}_{\mathsf{BSRS}}(1^\lambda)$ tallies the numbers of: open sessions $Q_{\mathsf{S}_1}(\mathsf{A}) := \#\{k \mid \mathbf{sSess}_k = \mathsf{open}\}$, closed sessions $Q_{\mathsf{S}_2}(\mathsf{A}) := \#\{k \mid \mathbf{sSess}_k = \mathsf{closed}\}$, and revoked sessions $Q_{\mathsf{rev}}(\mathsf{A}) := \#\{k \mid \mathbf{sSess}_k = \mathsf{revoked}\}$. It returns 1 iff (i) $msg_i \neq msg_j, \forall i \neq j$, (ii)

$\mathsf{BSRS.Ver}(\mathsf{pk}, m_i, \sigma_i) = 1, \forall i \in [l(\mathsf{A})]$, and (iii) $l(\mathsf{A}) \geq Q_{\mathsf{S}_2}(\mathsf{A}) - Q_{\mathsf{rev}}(\mathsf{A}) + 1$. Otherwise, it returns 0.

We define the *advantage* of adversary A in game $\mathbf{OMUF}^{\mathsf{A}}_{\mathsf{BSRS}}(1^\lambda)$ as $\mathbf{Adv}^{\mathbf{OMUF}}_{\mathsf{BSRS}}(\mathsf{A}) := \Pr[\mathbf{OMUF}^{\mathsf{A}}_{\mathsf{BSRS}}(1^\lambda) = 1]$. We say that BSRS is $(\varepsilon, t, Q_{\mathsf{S}_1}, Q_{\mathsf{S}_2}, Q_{\mathsf{rev}})$-OMUF-secure under revokes if for all adversaries A satisfying:

$$\mathbf{Time}^{\mathbf{OMUF}}_{\mathsf{BSRS}}(\mathsf{A}) \leq t, \quad Q_{\mathsf{S}_1}(\mathsf{A}) \leq Q_{\mathsf{S}_1}, \quad Q_{\mathsf{S}_2}(\mathsf{A}) \leq Q_{\mathsf{S}_2}, \quad Q_{\mathsf{rev}}(\mathsf{A}) \leq Q_{\mathsf{rev}} \tag{5.2}$$

we have $\mathbf{Adv}^{\mathbf{OMUF}}_{\mathsf{BSRS}}(\mathsf{A}) \leq \varepsilon$. We say that A breaks $(\varepsilon, t, Q_{\mathsf{S}_1}, Q_{\mathsf{S}_2}, Q_{\mathsf{rev}})$-OMUF security of BSRS if it satisfies (5.2) and $\mathbf{Adv}^{\mathbf{OMUF}}_{\mathsf{BSRS}}(\mathsf{A}) > \varepsilon$.

5.3 Blind Signatures from Linear Hash Functions with Noticeable Correctness Error

In this section, we present a generic compiler for blind signature schemes with revocable sessions from any linear hash function with noticeable correctness error.

5.3.1 Our Construction

We now give the formal description of our generic construction. For our construction, we use the following building blocks:

- A smooth linear hash function family $\mathsf{LHF} = (\mathsf{PGen}, \mathsf{F})$ with enclosedness errors $(\delta_1, \delta_2, \delta_3)$.
- A hash function $\mathsf{H} : \{0,1\}^* \to C := S_{c'}$ modeled as a programmable random oracle.
- A collision-free hash function $\mathsf{G} : \{0,1\}^* \to \mathcal{V} := \{0,1\}^\lambda$ modeled as a random oracle.

At a high-level, our protocol follows the three-move structure of identification schemes to prove knowledge of a secret key $\mathsf{sk} \in \mathcal{D}_{sk}$ s.t. $\mathsf{F}(\mathsf{sk}) = \mathsf{pk}$, where pk is the public key.

The signer transmits a commitment $\mathbf{R} := \mathsf{F}(\mathbf{r})$ to a random vector $\mathbf{r} \in \mathcal{D}_r$. The user receives $\mathbf{R}$ and it samples masking parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}$ and computes a challenge $\mathbf{c}$ as a function (involving H) of $\mathbf{R}, \boldsymbol{\alpha}, \boldsymbol{\beta}$, and the message *msg*, which it sends to the signer. The signer receives $\mathbf{c}$ and responds with $\mathbf{s} := \mathbf{c} \cdot \mathsf{sk} + \mathbf{r}$. Because F is linear, F, $\mathsf{F}(\mathbf{s}) = \mathsf{pk} \cdot \mathbf{c} + \mathbf{R}$ must hold. To obtain the final signature $(\mathbf{s}', \mathbf{c}')$ on *msg*, the user "unblinds" $(\mathbf{s}, \mathbf{c})$ with the help of the blinding parameters. Because of the correctness error of F, even if both parties follow the protocol, a valid signature is not guaranteed. We address this issue with an additional fourth move through which the user can *prove* that the session needs to be revoked. To avoid the PoR leaking information about *msg*, we let it comprise by random values that the signer sampled before computing the challenge. In addition, to avoid revealing *msg* to the signer, the user computes its challenge using a commitment $\mathbf{C} := \mathsf{G}(msg, \rho)$ for decommitment parameter ρ. The PoR also allows the signer to intercept users that make false claims of revocation. As we later argue, the user can only come up with a fake PoR for a session if it is able to guess the output of H.

Construction 5.3.1. We define our generic four-move blind signature scheme BSRS[LHF, H, G] = (PG, KG, S, U, CheckProof, Ver) as follows:

- $\mathsf{BSRS.PG}(1^\lambda)$: On input λ, algorithm BSRS.PG generates and returns scheme parameters par as per the specifications of column 2 of Table 5.1.
- $\mathsf{BSRS.KG}(\mathsf{par})$: On input par, algorithm BSRS.KG samples $\mathsf{sk} \leftarrow_\$ \mathcal{D}_{sk}$. It sets $\mathsf{pk} := \mathsf{F}(\mathsf{sk})$, and returns $(\mathsf{sk}, \mathsf{pk})$.
- $\mathsf{BSRS.S}_1(\mathsf{sk})$: On input a secret key sk, $\mathsf{BSRS.S}_1$ picks a random nonce vector $\mathbf{r} \leftarrow_\$ \mathcal{D}_r$ and computes $\mathbf{R} := \mathsf{F}(\mathbf{r})$. It returns the commitment $\mathbf{R}$ and a state $\mathbf{stS} := (\mathbf{r})$.
- $\mathsf{BSRS.U}_1(\mathsf{pk}, \mathbf{R}, msg)$: On input a public key pk, a commitment $\mathbf{R}$, and a message $msg \in \{0,1\}^*$, algorithm $\mathsf{BS.U}_1$ picks masking parameters $\boldsymbol{\alpha} \leftarrow_\$ \mathcal{D}_\alpha$ and $\boldsymbol{\beta} \leftarrow_\$ S_\beta$, a decommitment parameter $\rho \leftarrow_\$ \{0,1\}^\lambda$, and randomness $\rho' \leftarrow_\$ \{0,1\}^\lambda$. It commits to the message-to-be-signed as $\mathbf{C} := \mathsf{G}(msg, \rho)$, and to the masking

parameters by computing $\boldsymbol{\gamma} := \mathsf{G}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \rho')$. It sets $\mathbf{R}' := \mathbf{R} + \mathsf{F}(\boldsymbol{\alpha}) + \boldsymbol{\beta} \cdot \mathsf{pk}$ and computes the challenge $\mathbf{c}' := \mathsf{H}(\mathbf{R}', \boldsymbol{\gamma}, \mathbf{C})$. It then masks the challenge by computing $\mathbf{c} := \mathbf{c}' + \boldsymbol{\beta}$. If $\mathbf{c} \notin S_c$, then $\mathbf{c}$ "leaks" information about $\mathbf{c}'$ (which will be part of the final signature) and $\mathsf{BSRS.U}_1$ must restart, picking new masking parameters and randomnesses to maintain user anonymity. Otherwise, the user returns $\mathbf{c}$ and a state $\mathbf{stU} = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \mathbf{c}', \rho, \rho')$. It should be noted that restarts during $\mathsf{BSRS.U}_1$ do not affect correctness.

- $\mathsf{BSRS.S}_2(\mathsf{sk}, \mathbf{c}, \mathbf{stS})$: On input a secret key sk, a blinded challenge $\mathbf{c}$, and state $\mathbf{stS} = (\mathbf{r})$, $\mathsf{BSRS.S}_2$ computes a response $\mathbf{s} := \mathbf{r} + \mathbf{c} \cdot \mathsf{sk}$. Rejection sampling is necessary at this point to make $\mathbf{s}$ statistically independent of sk. If $\mathbf{s} \notin \mathcal{D}_s$, it sets $\mathbf{s} := \perp$. In any case, the algorithm returns $\mathbf{s}$ and a state $\mathbf{stS} = (\mathbf{r})$.

- $\mathsf{BSRS.U}_2(\mathsf{pk}, \mathbf{s}, \mathbf{stU})$: On input a public key pk, a response $\mathbf{s}$ and a state $\mathbf{stU} = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \mathbf{c}', \rho, \rho')$, $\mathsf{BS.U}_2$ sets $\mathbf{S} := \mathsf{F}(\mathbf{s})$. If the signer computed his response according to the protocol, then $\mathbf{S} = \mathbf{c} \cdot \mathsf{pk} + \mathbf{R}$ and $\mathbf{s} \in \mathcal{D}_s$ should hold (otherwise, $\mathsf{BS.U}_2$ sets $\boldsymbol{\sigma} := \perp$ and $\boldsymbol{\pi} := (\boldsymbol{\alpha}, \boldsymbol{\beta}, \rho', \mathbf{c}', \mathbf{C})$). $\mathsf{BS.U}_2$ then "unblinds" by computing $\mathbf{s}' := \mathbf{s} + \boldsymbol{\alpha}$ and rejection-sampling $\mathbf{s}'$. If $\mathbf{s}' \notin \mathcal{D}_{s'}$, $\mathsf{BS.U}_2$ sets $\boldsymbol{\sigma} := \perp$ and $\boldsymbol{\pi} := (\boldsymbol{\alpha}, \boldsymbol{\beta}, \rho', \mathbf{c}', \mathbf{C})$. On the contrary, if $\mathbf{s}' \in \mathcal{D}_{s'}$, $\mathsf{BS.U}_2$ sets $\boldsymbol{\sigma} := (\mathbf{c}', \mathbf{s}', \boldsymbol{\gamma}, \rho)$[6] and $\boldsymbol{\pi} := \perp$[7]. $\mathsf{BS.U}_2$ outputs $\boldsymbol{\sigma}$ and sends $\boldsymbol{\pi}$ to the signer as a notification.

- $\mathsf{BSRS.CheckProof}(\mathsf{pk}, \mathsf{tx}, \boldsymbol{\pi}, \mathbf{stS})$: On input a public key pk, a session transcript $\mathsf{tx} = (\mathbf{R}, \mathbf{c}, \mathbf{s})$, a PoR $\boldsymbol{\pi} = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \rho', \mathbf{c}', \mathbf{C})$, and a state $\mathbf{stS} = (\mathbf{r})$, auxiliary algorithm $\mathsf{CheckProof}$ sets $\boldsymbol{\gamma} := \mathsf{G}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \rho')$, $\mathbf{R}' := \mathbf{R} + \mathsf{F}(\boldsymbol{\alpha}) + \boldsymbol{\beta} \cdot \mathsf{pk}$, $\tilde{\mathbf{c}}_1 := \mathsf{H}(\mathbf{R}', \boldsymbol{\gamma}, \mathbf{C})$, and $\tilde{\mathbf{c}}_2 := \mathsf{H}(\mathsf{F}(\mathbf{s} + \boldsymbol{\alpha}) - \mathbf{c}' \cdot \mathsf{pk}, \boldsymbol{\gamma}, \mathbf{C})$. It computes $b_1 := [\![\mathbf{s} \in \mathcal{D}_s]\!]$, $b_2 := [\![\mathbf{c} - \boldsymbol{\beta} = \mathbf{c}']\!]$, $b_3 := [\![\mathbf{c}' = \tilde{\mathbf{c}}_1]\!]$, $b_4 := [\![\mathbf{c}' = \tilde{\mathbf{c}}_2]\!]$, $b_5 := [\![\mathbf{s} + \boldsymbol{\alpha} \notin \mathcal{D}_{s'}]\!]$, $b := \wedge_{i=1}^{5} b_i$ and it returns b.

- $\mathsf{BSRS.Ver}(\mathsf{pk}, \sigma, msg)$: On input a public key pk, signature $\sigma = (\mathbf{c}', \mathbf{s}', \boldsymbol{\gamma}, \rho)$, and message msg, $\mathsf{BSRS.Ver}$ sets $\mathbf{R}' := \mathsf{F}(\mathbf{s}') - \mathbf{c}' \cdot \mathsf{pk}$ and computes $b_1 := [\![\mathbf{c}' = \mathsf{H}(\mathbf{R}', \boldsymbol{\gamma}, \mathsf{G}(msg, \rho))]\!]$, $b_2 := [\![\mathbf{s}' \in \mathcal{D}_{s'}]\!]$ and $b_3 := [\![\mathbf{c}' \in S_{c'}]\!]$. It returns $b_1 \wedge b_2 \wedge b_3$.

[6] Notice that randomness ρ' is not included in the final signature, while ρ is. This is done in order to be able to prove blindness under revokes (see Theorem 5.3.2).

[7] Notice that if a session yields a valid signature, then the tuple $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \rho', \mathbf{c}', \mathbf{C})$ *cannot* be used as a PoR. This is the reason for setting $\boldsymbol{\pi} := \perp$.

5.3.2 Protocol Description

At the outset, the signer samples its masking parameter $\mathbf{r} \leftarrow_{\$} \mathcal{D}_r$ and computes a commitment $\mathbf{R} := \mathsf{F}(\mathbf{r})$ which it transmits to the user. It stores $\mathbf{r}$ in its internal state **stS** for future use.

The user receives $\mathbf{R}$ and it samples its own masking parameters $\boldsymbol{\alpha} \leftarrow_{\$} \mathcal{D}_\alpha$ and $\boldsymbol{\beta} \leftarrow_{\$} S_\beta$, as well as random bitstrings $\rho, \rho' \leftarrow_{\$} \{0,1\}^\lambda$. It commits to the message-to-be-signed *msg* with the help of random oracle G and randomness ρ by computing $\mathbf{C} := \mathsf{G}(msg, \rho)$. It also commits to its blinding parameters with the help of G and randomness ρ' by computing $\boldsymbol{\gamma} := \mathsf{G}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \rho')$. It forms the "blinded" commitment $\mathbf{R}' := \mathbf{R} + \mathsf{F}(\boldsymbol{\alpha}) + \boldsymbol{\beta} \cdot \mathsf{pk}$ and then computes a challenge $\mathbf{c}'$ as a function (involving H) of $\mathbf{R}', \boldsymbol{\gamma}$ and $\mathbf{C}$. Because $\mathbf{c}'$ will be part of the output signature, the user applies rejection sampling to it by computing the blinded challenge $\mathbf{c} := \mathbf{c}' + \boldsymbol{\beta}$ and sending it to the signer iff $\mathbf{c} \in S_c$. Otherwise, it repeats the entire step with fresh masking parameters and randomnesses. The user stores $\mathbf{c}'$, along with its masking parameters $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ and random coins (ρ, ρ') in its internal state **stU**.

The signer receives $\mathbf{c}$ and computes its response $\mathbf{s} := \mathbf{r} + \mathbf{c} \cdot \mathsf{sk}$ ($\mathbf{r}$ is transmitted through **stS**). Rejection sampling is necessary here in order to statistically decouple $\mathbf{s}$ from secret key sk. As such, it only sends $\mathbf{s}$ to the user if it falls within $\mathcal{D}_s$ (otherwise, it sends $\perp$ to indicate failure).

To obtain its signature on *msg*, the user "unblinds" response $\mathbf{s}$ with the help of blinding parameter $\boldsymbol{\alpha}$ by computing $\mathbf{s}' := \mathbf{s} + \mathbf{a}$. To make $\mathbf{s}'$ independent of $\mathbf{s}$, the user performs rejection-sampling one more time and only keeps $\mathbf{s}'$ iff it falls within $\mathcal{D}_{s'}$. If rejection-sampling fails, the user forms its PoR as $\boldsymbol{\pi} := (\boldsymbol{\alpha}, \boldsymbol{\beta}, \rho', \mathbf{c}', \mathbf{C})$ and sends it to the signer, requesting for the session to be revoked. Otherwise, the user outputs $\boldsymbol{\sigma} := (\mathbf{c}', \mathbf{s}', \boldsymbol{\gamma}, \rho)$ as the blind signature on *msg*. It is important to notice that randomness ρ is necessarily included in the signature in order to enable verifying that the signature is w.r.t. *msg*. On the contrary, ρ' is deliberately excluded from the output signature in order to be able to argue about the scheme's blindness under revokes property during our security analysis.

Complications during a signing session are handled with an additional fourth move

through which the user can request for a session to be revoked. This is done by presenting a PoR to the signer, which comprises of the random values that the signer picked before computing the challenge, and which allow the signer to trace the computations performed on the user's side of the protocol. However, to avoid revealing *msg* to the signer, the user computes its challenge using a "commitment" $\mathbf{C} := \mathsf{G}(msg, \rho)$ for decommitment parameter ρ instead. Furthermore, a fake PoR allows the signer to intercept users that make false claims of revocation. As we will later argue, a malicious user can only come up with a fake PoR for a session if it is able to guess the output of H.

5.3.3 Analysis and Security

We now provide theorems and supporting lemmas showing that our construction satisfies all security requirements, namely: correctness, blindness under revokes, and one-more unforgeability under revokes in the honest-signer model.

Theorem 5.3.1. *(Correctness) Let* $\mathsf{LHF} = (\mathsf{PGen}, \mathsf{F})$ *be a linear hash function family with enclosedness errors* $(\delta_1, \delta_2, \delta_3)$*,* $\mathsf{H} : \{0,1\}^* \to \mathcal{C}$*, and* $\mathsf{G} : \{0,1\}^* \to \mathcal{V}$ *be hash functions. Then* $\mathsf{BSRS}[\mathsf{LHF}, \mathsf{H}, \mathsf{G}]$ *has correctness error at most* $\delta_2 + \delta_3$*.*

Proof. The proof is trivial and is omitted for brevity. □

We now state that our generic constructions satisfy statistical blindness under revokes.

Theorem 5.3.2. *(Blindness Under Revokes) Let* $\mathsf{LHF} = (\mathsf{PGen}, \mathsf{F})$ *be a smooth linear hash function family and let* $\mathsf{H} : \{0,1\}^* \to \mathcal{C}$*, and* $\mathsf{G} : \{0,1\}^* \to \mathcal{V}$ *be random oracles. Then* $\mathsf{BSRS}[\mathsf{LHF}, \mathsf{H}, \mathsf{G}]$ *is statistically blind under revokes against honestly generated keys, relative to all* $\mathsf{par} \in \mathsf{PGen}(1^\lambda)$*.*

Proof. Let A be an adversary in game $\mathbf{RBlind}_{\mathsf{BSRS}[\mathsf{LHS},\mathsf{H},\mathsf{G}],\mathsf{par}}(1^\lambda)$. Let S_A denote the event that A succeeds and let $\varepsilon = \varepsilon(\lambda)$ denote its advantage s.t. $\Pr[S_\mathsf{A}] := \varepsilon + \frac{1}{2}$. A obtains as input a valid pair of keys $(\mathsf{sk}, \mathsf{pk}) \in \mathsf{BSRS.KG}(1^\lambda)$. Thus, we can write $\mathsf{pk} = \mathsf{F}(\mathsf{sk})$. After its execution, A holds $(msg_0, \boldsymbol{\sigma}_0), (msg_1, \boldsymbol{\sigma}_1)$, where $\boldsymbol{\sigma}_i$ is a signature on message

msg_i. The adversary also possesses lists of transcripts: $T_1 := [(\mathbf{R}_{1,1}, \mathbf{c}_{1,1}, \mathbf{s}_{1,1}, \boldsymbol{\pi}_{1,1}), \ldots, (\mathbf{R}_{1,k_1-1}, \mathbf{c}_{1,k_1-1}, \mathbf{s}_{1,k_1-1}, \boldsymbol{\pi}_{1,k_1-1}), (\mathbf{R}_{1,k_1}, \mathbf{c}_{1,k_1}, \mathbf{s}_{1,k_1}, \perp)]$, and $T_2 := [(\mathbf{R}_{2,1}, \mathbf{c}_{2,1}, \mathbf{s}_{2,1}, \boldsymbol{\pi}_{2,1}), \ldots, (\mathbf{R}_{2,k_2-1}, \mathbf{c}_{2,k_2-1}, \mathbf{s}_{2,k_2-1}, \boldsymbol{\pi}_{2,k_2-1}), (\mathbf{R}_{2,k_2}, \mathbf{c}_{2,k_2}, \mathbf{s}_{2,k_2}, \perp)]$ corresponding to interactions with the first and second user, respectively. A's goal is to correctly match each list of transcripts to the resulting message/signature pair.

Furthermore, let Bad denote the event in which the adversarial signer makes a query of the form:

- $\mathsf{G}(\star, \rho_{1,t})$, where $1 \leq t \leq k_1$ (resp., $\mathsf{G}(\star, \rho_{2,t})$, where $1 \leq t \leq k_2$). For $\rho_{1,k_1}, \rho_{1,k_2}$, this must occur *before* these are revealed to the adversary as part of the final signatures.

- $\mathsf{G}(\star, \star, \rho'_{1,t})$, where $1 \leq t \leq k_1$ (resp., $\mathsf{G}(\star, \star, \rho'_{2,t})$, where $1 \leq t \leq k_2$) *before* $\rho'_{1,t}$ (resp., $\rho'_{2,t}$) is revealed through an invocation of oracle Reinit. For ρ'_{1,k_1} (resp., ρ'_{2,k_2}) in particular, this can occur at any time during the game.

In case event Bad occurs, $\mathbf{RBlind}_{\mathsf{BSRS[LHS,H,G]},\mathsf{par}}(1^\lambda)$ aborts and A outputs a uniform bit. If A makes Q_G queries to oracle G, then $\Pr[S_\mathsf{A} \setminus \mathsf{Bad}] \geq \varepsilon + \frac{1}{2} - \frac{2 \cdot 2 \cdot Q_\mathsf{G}}{2^\lambda}$.

Notice that because each protocol run is statistically independent of any previous runs (since all user-side randomness is sampled anew by the respective honest users), the adversary only learns a negligible amount of information from each run. Furthermore, since each proof $\boldsymbol{\pi}_{sid,j}, j \in [k_{sid} - 1]$ contains all the necessary user-side randomness, the adversary can trivially check that such runs do not match to any of the two output message/signature pairs. Therefore, the adversary wins only if it can match the final tuples (i.e., the ones for which he does not know the randomness via some proof $\boldsymbol{\pi}_{sid,k_{sid}}$) to the output message/signature pairs. As such, in the following, we ommit the k_{sid} index above and always implicitly refer to the final tuple of a list of transcripts. We prove our theorem through the following claims:

Claim 1. *For each of the four combinations $(T_{sid}, \boldsymbol{\sigma}_i)$, where $(sid, i) \in \{1, 2\} \times \{0, 1\}$, there exists user-side randomness $\boldsymbol{rndU}_{sid,i} := (\boldsymbol{\alpha}_{sid,i}, \boldsymbol{\beta}_{sid,i}, \rho'_{sid,i})$ of the user algorithm which results in the tuple $(T_{sid}, \boldsymbol{\sigma}_i)$.*

Proof. Define the function $sess : \{0,1\} \to \{1,2\}$ such that $sess(i)$ is the signing session which resulted to the pair $(msg_i, \boldsymbol{\sigma}_i)$. Let $\boldsymbol{\alpha}_{sid,i} := \mathbf{s}'_i - \mathbf{s}_{sid}$, $\boldsymbol{\beta}_{sid,i} := \mathbf{c}_{sid} - \mathbf{c}'_i$, and let $\rho'_{sid,i} := \rho'_i$ if $sess(i) = sid$, and $\rho'_{sid,i} \leftarrow_\$ \{0,1\}^\lambda$, otherwise. Furthermore, let $\boldsymbol{\gamma}_{sid,i} := \mathsf{G}(\boldsymbol{\alpha}_{sid,i}, \boldsymbol{\beta}_{sid,i}, \rho'_{sid,i})$. Notice that because randomness ρ' is not included in the final signature, we can always use some $\rho'_{sid,i} \leftarrow_\$ \{0,1\}^\lambda$ and program G such that $\boldsymbol{\gamma}_i := \mathsf{G}(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i, \rho'_i) = \mathsf{G}(\boldsymbol{\alpha}_{sid,i}, \boldsymbol{\beta}_{sid,i}, \rho'_{sid,i}) =: \boldsymbol{\gamma}_{sid,i}$. We want to show that $\mathsf{H}(\mathbf{R}_{sid} + \mathbf{c}_{sid} \cdot \mathsf{pk}, \boldsymbol{\gamma}_{sid,i}, \mathbf{C}_i) = \mathbf{c}'_i$, where $\mathbf{C}_i = \mathsf{G}(msg_i, \rho_i)$ for some $\rho_i \leftarrow_\$ \{0,1\}^\lambda$. Since T_{sid} is a valid transcript, we have $\mathsf{F}(\mathbf{s}_{sid}) = \mathbf{R}_{sid} + \mathbf{c}_{sid} \cdot \mathsf{pk}$. Therefore,

$$\begin{aligned}
\mathbf{R}_{sid} + \boldsymbol{\beta}_{sid,i} \cdot \mathsf{pk} + \mathsf{F}(\boldsymbol{\alpha}_{sid,i}) &= \mathbf{R}_{sid} + (\mathbf{c}_{sid} - \mathbf{c}'_i) \cdot \mathsf{pk} + \mathsf{F}(\mathbf{s}'_i - \mathbf{s}_{sid}) \\
&= \mathbf{R}_{sid} + (\mathbf{c}_{sid} - \mathbf{c}'_i) \cdot \mathsf{F}(\mathsf{sk}) + \mathsf{F}(\mathbf{s}'_i - \mathbf{s}_{sid}) \\
&= \mathbf{s}'_i - \mathbf{c}'_i \cdot \mathsf{pk}
\end{aligned}$$

Since $\boldsymbol{\sigma}_i$ is a valid signature, we have $\mathbf{c}'_i = \mathsf{H}(\mathsf{F}(\mathbf{s}'_i) - \mathbf{c}'_i \cdot \mathsf{pk}, \boldsymbol{\gamma}_i, \mathsf{G}(msg_i, \rho_i))$. □

Claim 2. *The real randomness* $(\boldsymbol{rndU}_{1,b}, \boldsymbol{rndU}_{2,1-b})$ *used in* $\boldsymbol{RBlind}^{\mathsf{BSRS[LHS,H,G]},\mathsf{par}}(1^\lambda)$ *is identically distributed to the "fake" randomness* $(\boldsymbol{rndU}_{1,1-b}, \boldsymbol{rndU}_{2,b})$.

Proof. Because LHF is smooth, condition (S2) implies that $\boldsymbol{\alpha}_{1,b}$ and $\boldsymbol{\alpha}_{2,1-b}$ are identically distributed to $\boldsymbol{\alpha}_{1,1-b}$ and $\boldsymbol{\alpha}_{2,b}$, respectively. Similarly, condition (S5) implies that $\boldsymbol{\beta}_{1,b}$ and $\boldsymbol{\beta}_{2,1-b}$ are identically distributed to $\boldsymbol{\beta}_{1,1-b}$ and $\boldsymbol{\beta}_{2,b}$, respectively. Finally, $\rho'_{1,b}, \rho'_{2,1-b}$, as well as $\rho'_{1,1-b}, \rho'_{2,b}$ are all distributed uniformly over $\{0,1\}^\lambda$. □

The above two claims directly imply Theorem 5.3.2. □

We will now show that OMUF security of BSRS[LHF, H, G] is implied by the collision resistance of LHF. To this end, we will make use of the following two lemmas:

Lemma 5.3.3. *(Subset Forking Lemma [97]) Fix any integer* $Q \geq 1$ *and a set* $\mathcal{H}$ *s.t.* $|\mathcal{H}| \geq 2$ *as well as a set of side outputs* Σ*, instances* $\mathcal{I}$*, and a randomness space* Ω*. Let* C *be an algorithm that on input* $(I, \mathbf{h}) \in \mathcal{I} \times \mathcal{H}^Q$ *and randomness* $\omega \in \Omega$ *returns a tuple* (j, σ)*, where* $0 \leq j \leq Q$ *and* $\sigma \in \Sigma$*. We partition its input space* $\mathcal{I} \times \Omega \times \mathcal{H}^Q$ *into*

sets $\mathcal{W}_1, \ldots, \mathcal{W}_Q$ where for fixed $1 \leq j \leq Q$, $\mathcal{W}_j$ is the set of all $(I, \omega, \boldsymbol{h})$ that result in $(j, \sigma) \leftarrow \mathsf{C}(\boldsymbol{h}, I; \omega)$ for some arbitrary side output σ.

For any $1 \leq j \leq Q$ and $\mathcal{B} \subset \mathcal{W}_j$ define

$$\mathsf{acc}(\mathcal{B}) := \Pr_{(I,\omega,\boldsymbol{h}) \leftarrow_{\$} \mathcal{I} \times \Omega \times \mathcal{H}^Q}[(I, \omega, \boldsymbol{h}) \in \mathcal{B}]$$

$$\mathsf{frk}(\mathcal{B}, j) := \Pr_{(I,\omega,\boldsymbol{h}) \leftarrow_{\$} \mathcal{I} \times \Omega \times \mathcal{H}^Q, \boldsymbol{h}' \leftarrow_{\$} \mathcal{H}^Q | \boldsymbol{h}_{[j-1]}} \left[\begin{array}{c} (\boldsymbol{h}_j \neq \boldsymbol{h}'_j) \wedge \\ (((I, \omega, \boldsymbol{h}) \in \mathcal{B}) \wedge ((I, \omega, \boldsymbol{h}') \in \mathcal{B})) \end{array} \right].$$

Then

$$\mathsf{frk}(\mathcal{B}, j) \geq \mathsf{acc}(\mathcal{B}) \cdot \left(\frac{\mathsf{acc}(\mathcal{B})}{4} - \frac{1}{|\mathcal{H}|} \right).$$

Lemma 5.3.4. *(Generalized Splitting Lemma [98]) Let $n \in \mathbb{N}$ and $\mathcal{X}_1, \ldots, \mathcal{X}_n$ be finite sets. Let $\mathcal{B} \subset \boldsymbol{\mathcal{X}} := \mathcal{X}_1 \times \ldots \times \mathcal{X}_n$ be such that*

$$\Pr_{\boldsymbol{x} \leftarrow_{\$} \boldsymbol{\mathcal{X}}}[\boldsymbol{x} \in \mathcal{B}] := \varepsilon.$$

For any $S \subset [n], \alpha \leq \varepsilon$ and $i_1 < \ldots < i_{|\bar{S}|}$ and $\boldsymbol{\mathcal{X}}'_S := \mathcal{X}_{i_1} \times \ldots \times \mathcal{X}_{i_{|\bar{S}|}}$, define

$$\mathcal{B}_{S,\alpha} := \{(x_1, \ldots, x_n) \in \boldsymbol{\mathcal{X}} \mid \Pr_{\boldsymbol{x}' \leftarrow_{\$} \boldsymbol{\mathcal{X}}'_S}[\boldsymbol{x}_{S,\boldsymbol{x}'} \in \mathcal{B}] \geq \varepsilon - \alpha\},$$

where

$$\boldsymbol{x}_{S,\boldsymbol{x}'} := \begin{cases} \boldsymbol{x}_i, & \text{if } i \in S \\ \boldsymbol{x}'_i, & \text{otherwise} \end{cases}.$$

Then the following statements hold:

1. $\Pr_{\boldsymbol{x} \leftarrow_{\$} \boldsymbol{\mathcal{X}}}[\boldsymbol{x} \in \mathcal{B}_{S,\alpha}] \geq \alpha$
2. $\forall \boldsymbol{x} \in \mathcal{B}_{S,\alpha} : \Pr_{\boldsymbol{x}' \leftarrow_{\$} \boldsymbol{\mathcal{X}}'_S}[\boldsymbol{x}_{S,\boldsymbol{x}'} \in \mathcal{B}] \geq \varepsilon - \alpha$
3. $\Pr_{\boldsymbol{x} \leftarrow_{\$} \boldsymbol{\mathcal{X}}}[\boldsymbol{x} \in \mathcal{B}_{S,\alpha} \mid \boldsymbol{x} \in \mathcal{B}] \geq \alpha/\varepsilon$

Theorem 5.3.5. *(One-more unforgeability under revokes) If LHF is a $(\varepsilon, Q_{\mathsf{S}_1})$-regular, (ε', t')-collision resistant, linear hash function family with a torsion-free element from the*

kernel, then BSRS[LHF, H, G] *is* $(\varepsilon, t, Q_{\mathsf{S}_1}, Q_{\mathsf{S}_2}, Q_{\mathsf{rev}}, Q_{\mathsf{H}}, Q_{\mathsf{G}})$*-OMUF-secure under revokes, where*

$$\varepsilon' = O\left(\left(\varepsilon^2 - \left(\frac{|S_c|}{|C|}\right)^{Q_{\mathsf{S}_2}} \cdot \frac{Q_{\mathsf{H}}^{Q_{\mathsf{S}_2}+1} Q_{\mathsf{S}_1}^{Q_{\mathsf{S}_2}}}{|C|}\right)^3 \frac{1}{Q_{\mathsf{H}}^2 Q_{\mathsf{S}_2}^3}\right)$$

and $t' = 2t$.

Proof. Consider an adversary M that breaks $(\varepsilon, t, Q_{\mathsf{S}_1}, Q_{\mathsf{S}_2}, Q_{\mathsf{rev}}, Q_{\mathsf{H}}, Q_{\mathsf{G}})$-OMUF security under revokes of BSRS[LHF, H, G] in the random oracle model. We use M in a black-box way to construct an adversary B against the (ε', t')-collision resistance of LHF. W.l.o.g., we will assume throughout the proof that $Q_{\mathsf{S}_1}(\mathsf{M}) = Q_{\mathsf{S}_1}, Q_{\mathsf{H}}(\mathsf{M}) = Q_{\mathsf{H}}, Q_{\mathsf{G}}(\mathsf{M}) = Q_{\mathsf{G}}$, $l(\mathsf{M}) = Q_{\mathsf{S}_2}(\mathsf{M}) - Q_{\mathsf{rev}}(\mathsf{M}) + 1$, as well as $Q_{\mathsf{S}_1} \geq Q_{\mathsf{S}_2} + Q_{\mathsf{rev}}$ and $Q_{\mathsf{S}_2} \geq Q_{\mathsf{rev}}$. We will argue about our scheme's unforgeability under revokes through a sequence of game hops [170]:

Game$_0$: This corresponds to the real world. In this game, M interacts with the unforgeability game from Definition 5.2.10. The signer is simulated as in Construction 5.3.1. Thus, $\mathbf{Adv}^{\mathbf{Game}_0}(\mathsf{M}) = \mathbf{Adv}^{\mathbf{OMUF}}_{\mathsf{BSRS[LHF,H,G]}}(\mathsf{M})$.

Game$_1$: In this game, random oracle H is simulated through lazy sampling.

- **Setup:** Setup is the same as in **Game**$_0$. The challenger keeps track of queries made to oracle H with the help of a dictionary $H := \{\}$.
- **Online Phase:** M can access the following oracles:
  1. Oracle S_1 : Is the same as in **Game**$_0$.
  2. Oracle S_2 : Is the same as in **Game**$_0$.
  3. Oracle Revoke : Is the same as in **Game**$_0$.
  4. Oracle CheckProof : Is the same as in **Game**$_0$. The only difference is that now, the challenger will check if $(\mathbf{R}', \boldsymbol{\gamma}, \mathbf{C})$ and $(\mathsf{F}(\mathbf{s}_{sid} + \mathbf{a}) - \mathbf{c}' \cdot \mathsf{pk}, \boldsymbol{\gamma}, \mathbf{C})$ have

both been queried to H *before* computing $\tilde{\mathbf{c}}_1$, and $\tilde{\mathbf{c}}_2$, respectively.[8] If *both* have previously been queried, it sets $flag := 1$ (otherwise, it sets $flag := 0$) and then invokes H. Let $b := \wedge_{i=1}^{5} b_i$. If $b = 0$, the oracle returns 0. Otherwise (i.e., $b = 1$), if $flag = 0$, the game aborts. Otherwise, it returns 1.

5. Oracle H : On input $\mathbf{R}'$, commitment $\boldsymbol{\gamma}$ and some string $s \in \{0,1\}^*$, the challenger checks with dictionary H if $H[\mathbf{R}', \boldsymbol{\gamma}, s] \neq \perp$ (i.e., if it is already defined) and if so, it returns that particular value. Otherwise, it sets $H[\mathbf{R}', \boldsymbol{\gamma}, s] \leftarrow_{\$} C$. It returns the value $H[\mathbf{R}', \boldsymbol{\gamma}, s]$.

6. Oracle G : Is the same as in $\mathbf{Game}_0$.

- **Output Determination:** The challenger behaves exactly as it would in $\mathbf{Game}_0$. The only difference is that for each output message/signature pair, the challenger will ensure that the signature is not the result of M having guessed the output of oracle H without a query. This is done before verifying each individual signature. In particular, for each $(msg_t, \boldsymbol{\sigma}_t)$, where $\boldsymbol{\sigma}_t := (\mathbf{c}'_t, \mathbf{s}'_t, \boldsymbol{\gamma}_t, \rho_t), \forall t \in [l(\mathsf{M})]$, the challenger checks if $(\mathsf{F}(\mathbf{s}'_t) - \mathbf{c}'_t \cdot \mathsf{pk}, \boldsymbol{\gamma}_t, \mathsf{G}(msg_t, \rho_t))$ has previously been queried to H, and if so, it sets a $flag := 1$ (otherwise, it sets $flag := 0$). It then invokes BSRS.Ver to verify the signature. If BSRS.Ver outputs 0, the game returns 0. If BSRS.Ver outputs 1 and $flag = 0$, the game aborts. Otherwise, it outputs 1.

Claim 3. *Let E_1 denote the event in which M successfully guesses the output of oracle H, without querying it. Then,* $\Pr[E_1] \leq \frac{Q_{\mathsf{H}}}{|C|}$.

Proof. Notice that $\mathbf{Game}_1$ is identical to $\mathbf{Game}_0$, unless the game aborts either during CheckProof's execution, or before verifying the output signatures during Output Determination. This will happen iff M predicts H's output without querying the oracle. As a result, M will be able to either revoke a closed session from which he obtained a valid signature, or to forge a valid signature on his own. By the union bound, the probability of M making such a guess is at most $\frac{Q_{\mathsf{H}}}{|C|}$. Thus, the statistical difference between $\mathbf{Game}_1$ and $\mathbf{Game}_0$ is at most $\frac{Q_{\mathsf{H}}}{|C|}$. □

[8] This can be done with simple lookups in H, in amortized $O(1)$ time.

Remark 13. Notice that this game hop rules out the event in which a user successfully hides a challenge $\mathbf{c}'' \neq \mathbf{c}'$ in the computation of $\mathbf{c}$ during $\mathsf{BSRS.S}_2$. Indeed, if $\mathbf{c} = \mathbf{c}' + \boldsymbol{\beta} = \mathbf{c}'' + \boldsymbol{\beta}'$ for $\boldsymbol{\beta} \neq \boldsymbol{\beta}'$, then $\boldsymbol{\beta} = \mathbf{c}'' + \boldsymbol{\beta}' - \mathbf{c}'$. While a malicious user can control $\mathbf{c}''$ and $\boldsymbol{\beta}'$, he will need to predict $\mathbf{c}'$, which is an output of H.

Game$_2$: In this game, we simulate oracle G through lazy sampling. Simulation is done in a way that excludes collisions.

- **Setup:** Setup is the same as in game **Game**$_1$. The challenger keeps track of his responses with the help of dictionary $G := \{\}$.
- **Online Phase:** M can access the following oracles:
  1. Oracle S_1 : Is the same as in **Game**$_1$.
  2. Oracle S_2 : Is the same as in **Game**$_1$.
  3. Oracle Revoke : Is the same as in **Game**$_1$.
  4. Oracle $\mathsf{CheckProof}$: Is the same as in **Game**$_1$.
  5. Oracle H : Is the same as in **Game**$_1$.
  6. Oracle G : On input a string $s \in \{0,1\}^*$, and randomness $\rho \in \{0,1\}^\lambda$, the oracle checks with dictionary G if $G[s,\rho] \neq \perp$ and if so, it returns that particular value. Otherwise, it sets $G[s,\rho] \leftarrow_\$ \mathcal{V}$. If there already exists a different key (s^*, ρ^*) in dictionary G s.t. $G[s^*,\rho^*] = G[s,\rho]$, the game aborts. G returns the value $G[s,\rho]$.
- **Output Determination:** Is identical to **Game**$_1$.

Claim 4. *Let E_2 denote the event in which simulation of oracle G through lazy sampling fails. Then,* $\Pr[E_2] \leq \frac{2 \cdot Q_\mathsf{G}^2}{2^\lambda}$.

Proof. The game will abort if during an invocation of G, there already exists a different key in G, mapping to the same value. Discovering such a collision occurs with probability at most $\frac{Q_\mathsf{G}^2}{2^\lambda}$. Thus, by the union bound, the probability that the oracle simulation

fails is at most $\frac{Q_{\mathsf{G}}^2}{2^\lambda} + \frac{Q_{\mathsf{G}}^2}{2^\lambda}$ (since oracle G is invoked twice on the user's side). Moreover, notice that $\mathbf{Game}_2$ is identical to $\mathbf{Game}_1$, unless E_2 occurs. Therefore, the statistical difference between $\mathbf{Game}_2$ and $\mathbf{Game}_1$ is at most $\frac{2 \cdot Q_{\mathsf{G}}^2}{2^\lambda}$. □

$\mathbf{Game}_3$: We change $\mathbf{Game}_2$ to additionally require that the challenger correctly guesses the *exact* number of closed and revoked sessions that M will have during the Output Determination phase. If the guess is incorrect, the game aborts.

- **Setup:** The challenger makes a guess $(\kappa, r) \in \{0, \ldots, Q_{\mathsf{S}_2}\} \times \{0, \ldots, Q_{\mathsf{rev}}\}$ for the exact number of closed and revoked sessions with M, respectively. Setup is the same as $\mathbf{Game}_2$.
- **Online Phase:** M can access the following oracles:
  1. Oracle S_1 : Is the same as in $\mathbf{Game}_2$.
  2. Oracle S_2 : Is the same as in $\mathbf{Game}_2$.
  3. Oracle Revoke : Is the same as in $\mathbf{Game}_2$.
  4. Oracle CheckProof : Is the same as in $\mathbf{Game}_2$.
  5. Oracle H : Is the same as in $\mathbf{Game}_2$.
  6. Oracle G : Is the same as in $\mathbf{Game}_2$.
- **Output Determination:** Is identical to $\mathbf{Game}_2$. However, before checking the winning condition, the game aborts if $Q_{\mathsf{S}_2}(\mathsf{M}) \neq \kappa$ or $Q_{\mathsf{rev}}(\mathsf{M}) \neq r$.

Claim 5. *Let E_3 denote the event in which $\mathbf{Game}_3$ incorrectly guesses M's number of closed and revoked sessions. Then,* $\mathbf{Adv}^{\mathbf{Game}_3}(\mathsf{M}) = \frac{1}{(Q_{\mathsf{S}_2}+1)(Q_{\mathsf{rev}}+1)} \cdot \mathbf{Adv}^{\mathbf{Game}_2}(\mathsf{M})$.

Proof. Notice that $\mathbf{Game}_3$ is identical to $\mathbf{Game}_2$, unless the former *incorrectly* guesses the number of closed and revoked sessions. Furthermore, we observe that the winning condition for $\mathbf{Game}_2$ is independent of E_3. Thus, $\mathbf{Adv}^{\mathbf{Game}_3}(\mathsf{M}) := \Pr[\mathbf{Game}_3^{\mathsf{M}} = 1] = \Pr[\mathbf{Game}_2^{\mathsf{M}} = 1] \cdot \Pr[E_3^c] = \mathbf{Adv}^{\mathbf{Game}_2}(\mathsf{M}) \cdot \frac{1}{(Q_{\mathsf{S}_2}+1)(Q_{\mathsf{rev}}+1)}$. □

Corollary 5.3.1. $\mathbf{Adv}^{\mathbf{Game}_3}(\mathsf{M}) \leq \frac{1}{(Q_{\mathsf{S}_2}+1)(Q_{\mathsf{rev}}+1)} \cdot (\mathbf{Adv}^{\mathbf{OMUF}}_{\mathsf{BSRS[LHF,H,G]}}(\mathsf{M}) + \frac{Q_{\mathsf{H}}}{|C|} + \frac{2 \cdot Q_{\mathsf{G}}^2}{2^\lambda})$.

Wrapping Adversaries A_i**.** For each $1 \le i \le Q_{\mathsf{S}_2} - Q_{\mathsf{rev}} + 1$, we define an auxiliary wrapper algorithm A_i which perfectly simulates M's interface in $\mathbf{Game}_3$, in order to extract a short preimage of LHF. These wrappers will later be used as a core component to construct an adversary B against the collision resistance of LHF.

Throughout the proof, we denote the challenge space by $C := S_{c'} = S_{c'}(\mathsf{par})$ and by $|C| \ge 2^{2\lambda}$ its size. We denote the set from which our algorithms draw their randomness by $\Omega := \{0,1\}^N \times \mathcal{V}^{Q_\mathsf{G}} \times \mathcal{D}_r^{Q_{\mathsf{S}_1}+Q_{\mathsf{S}_2}+Q_{\mathsf{rev}}}$, where $N \in \mathbb{N}$. Note that for fixed randomness ω, A_i becomes deterministic. The description of A_i is as follows:

- **Setup:** On input an instance $I = (\mathsf{sk}, \mathsf{par})$, a vector of random oracle responses $\mathbf{h} \in C^{Q_\mathsf{H}}$, and random tape $\omega = ((\omega_\mathsf{M}, \kappa, r), \mathbf{g}, \mathbf{r}) \in \Omega$, A_i sets $\mathbf{R} := \mathsf{F}(\mathbf{r})$, $\mathsf{pk} := (\mathsf{F}(\mathsf{sk}), \mathsf{par})$, and initializes a counter $sid := 0$. It also initializes dictionaries $H := \{\}$ (used for keeping track of the responses of oracle H), $Ind := \{\}$ (used for keeping track of the index within $\mathbf{h}$, corresponding to each query made to oracle H), and $G := \{\}$, as well as counters $qid := 0$ and $gid := 0$. A_i then runs M on input pk.

- **Online Phase:** M can access the following oracles:

  1. Oracle S_1 : This oracle receives no input. It samples a fresh session identifier $sid := sid + 1$ and $\mathbf{sSess}_{sid} := \mathsf{open}$. It also intitializes internal session variables $\mathbf{c}_{sid}, \boldsymbol{\alpha}_{sid}, \boldsymbol{\beta}_{sid}, \boldsymbol{\gamma}_{sid}, \rho'_{sid}, \mathbf{s}_{sid}, \mathbf{c}'_{sid}, \mathbf{C}_{sid}, \mathbf{R}'_{sid}$ to $\perp$. Then it returns $(sid, \mathbf{R}_{sid})$.

  2. Oracle $\mathsf{S}_2(sid, \mathbf{c})$: On input a session identifier sid and a (blinded) challenge $\mathbf{c}$, if the session is not open, the oracle returns $\perp$. Otherwise, it sets $\mathbf{sSess}_{sid} := \mathsf{closed}$. Provided that $\mathbf{c} \in S_c$ (if not, it returns $\perp$), it sets $\mathbf{c}_{sid} := \mathbf{c}$ and computes its response as $\mathbf{s}_{sid} := \mathbf{c}_{sid} \cdot \mathsf{sk} + \mathbf{r}_{sid}$. If $\mathbf{s}_{sid} \notin \mathcal{D}_s$, it returns $\perp$. Otherwise, it returns $\mathbf{s}_{sid}$.

  3. Oracle $\mathsf{Revoke}(sid, \boldsymbol{\pi})$: On input a session identifier sid and PoR $\boldsymbol{\pi}$, if $\mathbf{sSess}_{sid} = \mathsf{closed}$, the oracle computes a bit $b := \mathsf{CheckProof}(sid, \boldsymbol{\pi})$. It sets $\mathbf{sSess}_{sid} := \mathsf{revoked}$ iff $b = 1$. If $\mathbf{sSess}_{sid} \neq \mathsf{closed}$, nothing is done.

4. Oracle $\mathsf{CheckProof}(sid, \boldsymbol{\pi})$: On input a session identifier sid and PoR $\boldsymbol{\pi}$, the oracle parses the PoR as $\boldsymbol{\pi} := (\boldsymbol{\alpha}, \boldsymbol{\beta}, \rho', \mathbf{c}', \mathbf{C})$ and sets $\boldsymbol{\gamma} := \mathsf{G}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \rho')$, $\mathbf{R}' := \mathbf{R}_{sid} + \mathsf{F}(\boldsymbol{\alpha}) + \boldsymbol{\beta} \cdot \mathsf{pk}$. A_i checks if $(\mathbf{R}', \boldsymbol{\gamma}, \mathbf{C})$ and $(\mathsf{F}(\mathbf{s}_{sid} + \boldsymbol{\alpha}) - \mathbf{c}' \cdot \mathsf{pk}, \boldsymbol{\gamma}, \mathbf{C})$ have both been queried to H. If *both* have previously been queried, it sets $flag := 1$ (otherwise, it sets $flag := 0$). It then computes $\tilde{\mathbf{c}}_1 := \mathsf{H}(\mathbf{R}', \boldsymbol{\gamma}, \mathbf{C})$, $\tilde{\mathbf{c}}_2 := \mathsf{H}(\mathsf{F}(\mathbf{s}_{sid} + \boldsymbol{\alpha}) - \mathbf{c}' \cdot \mathsf{pk}, \boldsymbol{\gamma}, \mathbf{C})$ as well as $b_1 := [\![\mathbf{s}_{sid} \in \mathcal{D}_s]\!], b_2 := [\![\mathbf{c}_{sid} - \boldsymbol{\beta} = \mathbf{c}']\!], b_3 := [\![\mathbf{c}' = \tilde{\mathbf{c}}_1]\!], b_4 := [\![\mathbf{c}' = \tilde{\mathbf{c}}_2]\!], b_5 := [\![\mathbf{s}_{sid} + \boldsymbol{\alpha} \notin \mathcal{D}_{s'}]\!]$, and $b := \wedge_{i=1}^{5} b_i$. If $b = 0$, the oracle returns 0. Otherwise (i.e., $b = 1$), if $flag = 0$, A_i internally aborts M.[9] Otherwise (i.e., $flag = 1$), it updates its internal variables by setting: $\boldsymbol{\alpha}_{sid} := \boldsymbol{\alpha}, \boldsymbol{\beta}_{sid} := \boldsymbol{\beta}, \mathbf{c}'_{sid} := \mathbf{c}', \rho'_{sid} := \rho, \mathbf{C}_{sid} := \mathbf{C}, \boldsymbol{\gamma}_{sid} := \boldsymbol{\gamma}, \mathbf{R}'_{sid} := \mathbf{R}'$, and it returns 1.

5. Oracle $\mathsf{H}(\mathbf{R}', \boldsymbol{\gamma}, s)$: On input $\mathbf{R}'$, commitment $\boldsymbol{\gamma}$, and string $s \in \{0,1\}^*$, the oracle checks with dictionary H if $H[\mathbf{R}', \boldsymbol{\gamma}, s] \neq \perp$ (i.e., if it is already defined) and if so, it returns that particular value. Otherwise, it samples a fresh identifier $qid := qid + 1$ and sets $H[\mathbf{R}', \boldsymbol{\gamma}, s] := \mathbf{h}_{qid}$. It also stores index qid to dictionary Ind via $Ind[\mathbf{R}', \boldsymbol{\gamma}, s] := qid$. It returns the value $H[\mathbf{R}', \boldsymbol{\gamma}, s]$.

6. Oracle $\mathsf{G}(s, \rho)$: On input a string $s \in \{0,1\}^*$, and randomness $\rho \in \{0,1\}^\lambda$, the oracle checks with dictionary G if $G[s, \rho] \neq \perp$ (i.e., if it is already defined) and if so, it returns that particular value. Otherwise, it samples a fresh identifier $gid := gid + 1$ and sets $G[s, \rho] := \boldsymbol{g}_{gid}$. If there already exists a different key (s^*, ρ^*) in dictionary G s.t. $G[s^*, \rho^*] = G[s, \rho]$, A_i internally aborts M. G returns the value $G[s, \rho]$.

- **Output Determination:** Let $(msg_1, \boldsymbol{\sigma}_1), \ldots, (msg_{l(\mathsf{M})}, \boldsymbol{\sigma}_{l(\mathsf{M})})$, where $\boldsymbol{\sigma}_t := (\mathbf{c}'_t, \mathbf{s}'_t, \boldsymbol{\gamma}_t, \rho_t), \forall t \in [l(\mathsf{M})]$ be M's output after interaction. First, A_i checks that the output message-signature pairs contain pairwise-distinct messages (if not, A_i internally aborts M). Next, for each $t \in [l(\mathsf{M})]$, A_i checks if $(\mathsf{F}(\mathbf{s}'_t) - \mathbf{c}'_t \cdot \mathsf{pk}, \boldsymbol{\gamma}_t, \mathsf{G}(msg_t, \rho_t))$ has previously been queried to H, and if so, it sets $flag := 1$ (otherwise, it sets $flag := 0$). It then invokes $\mathsf{BSRS.Ver}$ to verify $(msg_t, \boldsymbol{\sigma}_t)$. If $\mathsf{BSRS.Ver}$ outputs 1 and

[9] We implicitly assume that whenever this occurs, A_i returns the value $(0, 0)$.

flag = 0, the signature is the result of M guessing H's output and thus, A_i returns $(0,0)$. If BSRS.Ver outputs 0, the signature is not valid, and A_i returns $(0,0)$. For $t \in [l(\mathsf{M})]$, A_i sets internal variables as $\hat{\mathbf{J}}_t := Ind[\mathsf{F}(\mathbf{s}'_t) - \mathbf{c}'_t \cdot \mathsf{pk}, \boldsymbol{\gamma}_t, \mathsf{G}(msg_t, \rho_t)], \hat{\mathbf{h}}_t := \mathbf{h}_{\hat{\mathbf{J}}_t}, \hat{\mathbf{s}}'_t := \mathbf{s}'_t, \hat{\boldsymbol{\chi}}_t := \hat{\mathbf{s}}'_t - \hat{\mathbf{h}}_t \cdot \mathsf{sk}$. Moreover, it defines internal variables $\hat{\mathbf{J}}, \hat{\mathbf{h}}, \hat{\mathbf{s}}', \hat{\boldsymbol{\chi}}$ for each revoked session. This is done by setting $k := 1$ and iterating over $t \in [sid]$. If $\mathbf{sSess}_t$ = revoked, it sets $\hat{\mathbf{J}}_{l(\mathsf{M})+k} := Ind[\mathbf{R}'_t, \boldsymbol{\gamma}_t, \mathbf{C}_t], \hat{\mathbf{h}}_{l(\mathsf{M})+k} := \mathbf{h}_{\hat{\mathbf{J}}_{l(\mathsf{M})+k}}, \hat{\mathbf{s}}'_{l(\mathsf{M})+k} := \mathbf{s}_t + \boldsymbol{\alpha}_t, \hat{\boldsymbol{\chi}}_{l(\mathsf{M})+k} := \hat{\mathbf{s}}'_{l(\mathsf{M})+k} - \hat{\mathbf{h}}_{l(\mathsf{M})+k} \cdot \mathsf{sk}$ and $k := k + 1$. A_i then tallies the numbers of: open sessions $Q_{\mathsf{S}_1}(\mathsf{M}) := \#\{k \mid \mathbf{sSess}_k = \mathsf{open}\}$, closed sessions $Q_{\mathsf{S}_2}(\mathsf{M}) := \#\{k \mid \mathbf{sSess}_k = \mathsf{closed}\}$, and revoked sessions $Q_{\mathsf{rev}}(\mathsf{M}) := \#\{k \mid \mathbf{sSess}_k = \mathsf{revoked}\}$. If $Q_{\mathsf{S}_2}(\mathsf{M}) \neq \kappa$ or $Q_{\mathsf{rev}}(\mathsf{M}) \neq r$, A_i returns $(0,0)$. If the winning condition $l(\mathsf{M}) = Q_{\mathsf{S}_2}(\mathsf{M}) - Q_{\mathsf{rev}}(\mathsf{M}) + 1$ is satisfied, A_i returns $(\hat{\mathbf{J}}_i, \hat{\boldsymbol{\chi}}_i)$. Otherwise, it returns $(0,0)$.

Analysis of Adversary A_i. Consider the variables $\hat{\mathbf{J}}_i, \hat{\boldsymbol{\chi}}_i, \hat{\mathbf{h}}_i$, and $\hat{\mathbf{s}}'_i$. Concretely, the variable $\hat{\boldsymbol{\chi}}_i := \hat{\mathbf{s}}'_i - \hat{\mathbf{h}}_i \cdot \mathsf{sk}$ results from the i-th valid blind signature, whereas the index $\hat{\mathbf{J}}_i$ indicates which random oracle query corresponds to this signature.

We will fix an execution of A_i via the tuples $I = (\mathsf{sk}, \mathsf{par}), \mathbf{h}$, and A_i's randomness ω. Define the set $\mathcal{W}$ of *successful inputs* of A_i as the set of all tuples $(I, \omega, \mathbf{h})$ which lead to a successful run of A_i, i.e.,[10]

$$\mathcal{W} := \{(I, \omega, \mathbf{h}) \mid \hat{\mathbf{J}}_i \neq 0; (\hat{\mathbf{J}}_i, \hat{\boldsymbol{\chi}}_i) \leftarrow \mathsf{A}_i(I, \mathbf{h}; \omega)\}$$

Note that $\mathcal{W}$ is independent of i and, by construction of A_i, i.e.,

$$\begin{aligned} \varepsilon &= \mathbf{Adv}^{\mathbf{OMUF}}_{\mathsf{BSRS[LHF,H,G]}}(\mathsf{M}) \\ &= \Pr_{\mathsf{par} \leftarrow_{\$} \mathsf{BSRS.PG}(1^\lambda), (\mathsf{sk}, \omega, \mathbf{h}) \leftarrow_{\$} (\mathcal{D}_{sk} \times \Omega \times C^{Q_\mathsf{H}})} [(\mathsf{sk}, \mathsf{par}, \omega, \mathbf{h}) \in \mathcal{W}] \end{aligned}$$

The following lemma is the first component required for our analysis:

Lemma 5.3.6. *Let $\mathcal{P}$ denote the set of parameters* par *such that:*

$$\Pr_{(\mathsf{sk}, \omega, \boldsymbol{h}) \leftarrow_{\$} (\mathcal{D}_{sk} \times \Omega \times C^{Q_\mathsf{H}})} [(\mathsf{sk}, \mathsf{par}, \omega, \boldsymbol{h}) \in \mathcal{W}] \geq \varepsilon/2.$$

[10] For simplicity, below we ignore the statistical additive errors incurred by the previous game hops.

Then, $\Pr_{\mathsf{par}\leftarrow_{\$}\mathsf{BSRS.PG}(1^\lambda)}[\mathsf{par}\in\mathcal{P}] \geq \varepsilon/2$.

Proof. We argue by contradiction. To this end, assume that $\Pr_{\mathsf{par}\leftarrow_{\$}\mathsf{BSRS.PG}(1^\lambda)}[\mathsf{par}\in\mathcal{P}] < \varepsilon/2$. But then we have:

$$
\begin{aligned}
\varepsilon &:= \Pr_{\mathsf{par}\leftarrow_{\$}\mathsf{BSRS.PG}(1^\lambda),(\mathsf{sk},\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{D}_{sk}\times\Omega\times C^{Q_\mathsf{H}})}[(\mathsf{sk},\mathsf{par},\omega,\mathbf{h})\in\mathcal{W}] \\
&= \sum_{\mathsf{par}} \Pr_{(\mathsf{sk},\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{D}_{sk}\times\Omega\times C^{Q_\mathsf{H}})}[(\mathsf{sk},\mathsf{par},\omega,\mathbf{h})\in\mathcal{W}] \cdot \Pr_{\mathsf{par}'\leftarrow_{\$}\mathsf{BSRS.PG}(1^\lambda)}[\mathsf{par}'=\mathsf{par}] \\
&= \sum_{\mathsf{par}\in\mathcal{P}} \Pr_{(\mathsf{sk},\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{D}_{sk}\times\Omega\times C^{Q_\mathsf{H}})}[(\mathsf{sk},\mathsf{par},\omega,\mathbf{h})\in\mathcal{W}] \cdot \Pr_{\mathsf{par}'\leftarrow_{\$}\mathsf{BSRS.PG}(1^\lambda)}[\mathsf{par}'=\mathsf{par}] \\
&+ \sum_{\mathsf{par}\notin\mathcal{P}} \Pr_{(\mathsf{sk},\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{D}_{sk}\times\Omega\times C^{Q_\mathsf{H}})}[(\mathsf{sk},\mathsf{par},\omega,\mathbf{h})\in\mathcal{W}] \cdot \Pr_{\mathsf{par}'\leftarrow_{\$}\mathsf{BSRS.PG}(1^\lambda)}[\mathsf{par}'=\mathsf{par}] \\
&< 1 \cdot \sum_{\mathsf{par}\in\mathcal{P}} \Pr_{\mathsf{par}\leftarrow_{\$}\mathsf{BSRS.PG}(1^\lambda)}[\mathsf{par}'=\mathsf{par}] + \varepsilon/2 \cdot \sum_{\mathsf{par}\notin\mathcal{P}} \Pr_{\mathsf{par}'\leftarrow_{\$}\mathsf{BSRS.PG}(1^\lambda)}[\mathsf{par}'=\mathsf{par}] \\
&= \Pr_{\mathsf{par}\leftarrow_{\$}\mathsf{BSRS.PG}(1^\lambda)}[\mathsf{par}\in\mathcal{P}] + \varepsilon/2 \cdot \Pr_{\mathsf{par}\leftarrow_{\$}\mathsf{BSRS.PG}(1^\lambda)}[\mathsf{par}\notin\mathcal{P}] \\
&< \varepsilon/2 + 1\cdot\varepsilon/2 = \varepsilon
\end{aligned}
$$

which is a contradiction. □

From this point on, we fix some arbitrary parameters $\mathsf{par}\in\mathcal{P}$ and make the convention that $I = (\mathsf{sk},\mathsf{par}) \leftarrow_{\$} \mathcal{I}$ samples sk uniformly from $\mathcal{D}_{sk}(\mathsf{par})$ (but keeps par fixed). Thus, we will assume, from here on out, that $\Pr_{(I,\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_\mathsf{H}})}[(I,\omega,\mathbf{h})\in\mathcal{W}] = \mathbf{Adv}^{\mathbf{OMUF}}_{\mathsf{BSRS[LHF,H,G]}}(\mathsf{M}) \geq \varepsilon/2$. Moreover, $\hat{\mathbf{J}}_i, \hat{\boldsymbol{\chi}}_i, \hat{\mathbf{h}}_i$, and $\hat{\mathbf{s}}'_i$ can be seen as random variables whose distribution is induced by the uniform distribution on $(\mathcal{I}\times\Omega\times C^{Q_\mathsf{H}})$, and whose outcome is uniquely determined given $(I,\omega,\mathbf{h})\in\mathcal{W}$. Thus, we will write

$$(\hat{\mathbf{J}}_i(I,\omega,\mathbf{h}), \hat{\boldsymbol{\chi}}_i(I,\omega,\mathbf{h})) \leftarrow \mathsf{A}_i(I,\mathbf{h};\omega)$$

In the following, when stating probability distributions over I,ω, and $\mathbf{h}$, unless specified differently, we will always refer to uniform distributions. That is, $(I,\omega,\mathbf{h}) \leftarrow_{\$}$

$(\mathcal{I} \times \Omega \times C^{Q_\mathsf{H}})$. We consider the following probability for fixed $(I, \omega, \mathbf{h})$, j, c, and i:

$$\Pr_{\mathbf{h}' \leftarrow_\$ C^{Q_\mathsf{H}} | \mathbf{h}_{[j-1]}} [\hat{\mathbf{J}}_i(I, \omega, \mathbf{h}') = j \wedge \hat{\boldsymbol{\chi}}_i(I, \omega, \mathbf{h}') = c] \tag{5.3}$$

We denote by $c_{i,j}(I, \omega, \mathbf{h})$ the lexicographically first value c s.t. probability (5.3) is maximized when $(I, \omega, \mathbf{h})$, j, and i are fixed. We then write $C_i(I, \omega, \mathbf{h}) = c_{i,\hat{\mathbf{J}}_i(I,\omega,\mathbf{h})}(I, \omega, \mathbf{h})$. Put differently, $C_i(I, \omega, \mathbf{h})$ represents the most likely value that random variable $\hat{\boldsymbol{\chi}}_i$ takes after *re-running* A_i on some instance I, randomness ω, and a conditionally sampled vector of RO responses $\mathbf{h}'$. For fixed i, j, we define $\mathcal{B}_{i,j} \subset \mathcal{W}$ as

$$\mathcal{B}_{i,j} := \{(I, \omega, \mathbf{h}) \in \mathcal{W} \mid \hat{\mathbf{J}}_i(I, \omega, \mathbf{h}) = j \wedge \hat{\boldsymbol{\chi}}_i(I, \omega, \mathbf{h}) \neq C_i(I, \omega, \mathbf{h})\}$$

and let

$$\beta_{i,j} := \Pr_{(I,\omega,\mathbf{h}) \leftarrow_\$ (\mathcal{I} \times \Omega \times C^{Q_\mathsf{H}})} [(I, \omega, \mathbf{h}) \in \mathcal{B}_{i,j}]$$

$$\delta_{i,j} := \Pr_{(I,\omega,\mathbf{h}) \leftarrow_\$ (\mathcal{I} \times \Omega \times C^{Q_\mathsf{H}}), \mathbf{h}' \leftarrow_\$ C^{Q_\mathsf{H}} | \mathbf{h}_{[j-1]}} \begin{bmatrix} \hat{\boldsymbol{\chi}}_i(I, \omega, \mathbf{h}) \neq \hat{\boldsymbol{\chi}}_i(I, \omega, \mathbf{h}') \\ \wedge\ \hat{\mathbf{J}}_i(I, \omega, \mathbf{h}) = \hat{\mathbf{J}}_i(I, \omega, \mathbf{h}') = j \end{bmatrix}$$

Notice that $\beta_{i,j}$ represents the probability of *not* getting the most likely value for $\hat{\boldsymbol{\chi}}_i$ on the first run of A_i (i.e., before conditionally re-sampling $\mathbf{h}'$). Finally, $\delta_{i,j}$ represents the probability of successfully extracting a collision by running adversary A_i twice.
The following inequality can be proven in our context in a way verbatim to [97].

Lemma 5.3.7. *For all i, j : $\delta_{i,j} \geq \beta_{i,j}(\frac{\beta_{i,j}}{8} - \frac{1}{2|C|})$.*

We now prove the following lemma, which is the core component for reducing collision resistance to one-more unforgeability for our construction.

Lemma 5.3.8. *There exist $i \in [Q_{\mathsf{S}_2} - Q_{\mathsf{rev}} + 1], j \in [Q_\mathsf{H}]$ s.t.:*

$$\beta_{i,j} > \frac{\frac{\varepsilon^2}{8} - \left(\frac{|S_c|}{|C|}\right)^{Q_{\mathsf{S}_2}} \cdot \frac{Q_\mathsf{H}^{Q_{\mathsf{S}_2}+1} \cdot \binom{Q_{\mathsf{S}_1}+Q_{\mathsf{S}_2}}{Q_{\mathsf{S}_1}}}{|C|}}{2Q_\mathsf{H}(Q_{\mathsf{S}_2} - Q_{\mathsf{rev}} + 1)}.$$

Proof. The proof of this lemma consists of three parts: (i) modelling the OMUF adver-

sary's confusion about the challenger's input leading to a particular transcript, and showing that under certain restrictions, for each such input, there exists exactly one other "mirror" input that could have led to the exact same transcript, (ii) bounding the event of getting the most likely candidate collision on the first run out of both a challenger using the actual input and a challenger using the "mirror" input, and (iii) showing that the set of inputs for which (i) holds is not too small.

Modelling M's Confusion about A_i's Input. Recall algorithm A_i and its internal variables. On input $(I = (\mathsf{sk}, \mathsf{par}), \omega = (\omega_{\mathsf{M}}{}^{11}, \mathbf{g}, \mathbf{r}), \mathbf{h})$, A_i runs M on $\mathsf{pk} = \mathsf{F}(\mathsf{sk})$ and randomness ω_{M} and answers its queries using the components of vectors $\mathbf{g}, \mathbf{r}$, and $\mathbf{h}$. We can thus fix an execution of M (within A_i) via a tuple of the form $(I, \omega, \mathbf{h}) = (I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h})$. In the following, $\mathbf{c}(I, \omega, \mathbf{h})$ denotes the vector of blinded challenges $\mathbf{c}_{sid}$ as defined each time oracle S_2 is invoked. Furthermore, we have assumed that $\mathsf{F} : \mathcal{D} \to \mathcal{R}$ has a torsion-free element $\mathbf{z}^* \in \mathcal{D} \setminus \{\mathbf{0}\}$ from the kernel. Thus, (i) $\mathsf{F}(\mathbf{z}^*) = \mathbf{0}$, and (ii) $\forall \mathbf{c}_1, \mathbf{c}_2 \in S_c$ s.t. $(\mathbf{c}_1 - \mathbf{c}_2) \cdot \mathbf{z}^* = \mathbf{0}$ we have $\mathbf{c}_1 = \mathbf{c}_2$.

We model adversary M's confusion about the challenger's input leading to a particular session transcript through the following mapping:

Lemma 5.3.9. *We define the mapping $\Phi : \mathcal{W} \to (\mathcal{I} \times \Omega \times C^{Q_{\mathsf{H}}})$:*

$$\Phi((\mathsf{sk}, \mathsf{par}), (\omega_{\mathsf{M}}, \boldsymbol{g}, \boldsymbol{r}), \boldsymbol{h}) := ((\mathsf{sk} + \boldsymbol{z}^*, \mathsf{par}), (\omega_{\mathsf{M}}, \boldsymbol{g}, \boldsymbol{r} - \boldsymbol{z}^* \cdot \boldsymbol{c}(I, (\omega_{\mathsf{M}}, \boldsymbol{g}, \boldsymbol{r}), \boldsymbol{h})), \boldsymbol{h}),$$

where we make the convention that for $v \in \mathcal{D} \cup \mathcal{C} \cup \mathcal{R}$, $v \cdot \perp := 0$. Let

$$\mathcal{W}_{inj} := \{(I, \omega, \boldsymbol{h}) \in \mathcal{W} : \Phi(I, \omega, \boldsymbol{h}) \in \Phi(\mathcal{W}) \cap \mathcal{W}\} \subset \mathcal{W}$$

Then the restriction $\Phi_{inj} : \mathcal{W}_{inj} \to \Phi(\mathcal{W}) \cap \mathcal{W}$ of Φ to $\mathcal{W}_{inj}$ is injective.

Proof. For the proof we require the following claim.

[11] In the analysis that follows, we will w.l.o.g. implicitly assume that κ and r are included as part of M's randomness ω_{M}.

Claim 6. *Let* $(I, \omega, \boldsymbol{h}) \in \mathcal{W}$. *If* $\Phi(I, \omega, \boldsymbol{h}) \in \mathcal{W}$, *then the tuples* $(I, \omega, \boldsymbol{h})$ *and* $\Phi(I, \omega, \boldsymbol{h})$ *fix the same execution of* M.

Proof. To this end we consider all values in the view of M.

- **Initial input to** M**.** Since Φ does not alter the values of ω_{M} and par, we only need to verify that M obtains the same public key in both executions. This is ensured via $\mathsf{F}(\mathsf{sk} + \mathbf{z}^*) = \mathsf{F}(\mathsf{sk}) + \mathsf{F}(\mathbf{z}^*) = \mathsf{F}(\mathsf{sk}) = \mathsf{pk}$.

- **Outputs of oracle** S_1**.** Oracle S_1 consecutively returns tuples of the form $(sid, \mathbf{R}_{sid})$, where $\mathsf{F}(\mathbf{r}_{sid}) = \mathbf{R}_{sid}$. These are the same in both executions since $\mathsf{F}(\mathbf{r}_{sid}) = \mathbf{R}_{sid} = \mathbf{R}_{sid} - 0 \cdot \mathbf{c}(I, \omega, \mathbf{h}) = \mathsf{F}(\mathbf{r}_{sid} - \mathbf{z}^* \cdot \mathbf{c}(I, \omega, \mathbf{h}))$.

- **Outputs of oracle** S_2**.** Oracle S_2 consecutively returns the values from $\mathbf{s}_{sid} := \mathbf{c} \cdot \mathsf{sk} + \mathbf{r}_{sid}$ (or $\perp$, in case $\mathbf{s}_{sid} \notin \mathcal{D}_s$). Note that the first value $\mathbf{c}_{sid,1}$ in both executions is the same (as it only depends on values that we have already argued to remain the same in both executions), i.e., $\mathbf{c}_{sid,1} = \mathbf{c}_{sid,1}(I, \omega, \mathbf{h}) = \mathbf{c}_{sid,1}(\Phi(I, \omega, \mathbf{h}))$. Thus,

$$\begin{aligned}
\mathbf{s}_{sid,1}(I, \omega, \mathbf{h}) &= \mathbf{r}_{sid,1} + \mathsf{sk} \cdot \mathbf{c}_{sid,1}(I, \omega, \mathbf{h}) \\
&= \mathbf{r}_{sid,1} - \mathbf{z}^* \cdot \mathbf{c}_{sid,1}(I, \omega, \mathbf{h}) + \mathbf{z}^* \cdot \mathbf{c}_{sid,1}(I, \omega, \mathbf{h}) + \mathsf{sk} \cdot \mathbf{c}_{sid,1}(I, \omega, \mathbf{h}) \\
&= (\mathbf{r}_{sid,1} - \mathbf{z}^* \cdot \mathbf{c}_{sid,1}(\Phi(I, \omega, \mathbf{h}))) + (\mathsf{sk} + \mathbf{z}^*) \cdot \mathbf{c}_{sid,1}(\Phi(I, \omega, \mathbf{h})) \\
&= \mathbf{s}_{sid,1}(\Phi(I, \omega, \mathbf{h}))
\end{aligned}$$

- **Outputs of oracle** Revoke**.** Oracle Revoke does not return any values.

- **Outputs of oracle** G**.** Oracle G consecutively returns the values from $\mathbf{g}$. These remain the same in both executions since they depend on $\mathbf{g}$, and the randomness ω_{M} of the adversary.

- **Outputs of oracle** H**.** Oracle H consecutively returns the values from $\mathbf{h}$. These remain the same in both executions since they depend on $\mathbf{R}, \mathbf{g}, \mathbf{h}$, and the randomness ω_{M}.

Thus, M sees identical values in both executions corresponding to $(I, \omega, \mathbf{h})$ and $\Phi(I, \omega, \mathbf{h})$. This implies that $(I, \omega, \mathbf{h})$ and $\Phi(I, \omega, \mathbf{h})$ fix the same execution of M, which proves the claim.

□

Towards a contradiction, suppose that Φ_{inj} is not injective. Thus, for distinct tuples $(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h}) \neq (I', (\omega'_{\mathsf{M}}, \mathbf{g}', \mathbf{r}'), \mathbf{h}')$ in $\mathcal{W}_{inj}$, we have $\Phi_{inj}(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h}) = \Phi_{inj}(I', (\omega'_{\mathsf{M}}, \mathbf{g}', \mathbf{r}'), \mathbf{h}')$. This implies $\omega_{\mathsf{M}} = \omega'_{\mathsf{M}}, \mathbf{g} = \mathbf{g}'$ and $\mathbf{h} = \mathbf{h}'$. Similarly, $\mathsf{sk} + \mathbf{z}^* = \mathsf{sk}' + \mathbf{z}^*$, which implies $\mathsf{sk} = \mathsf{sk}'$. Finally, $\mathbf{r} - \mathbf{z}^* \cdot \mathbf{c}(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h}) = \mathbf{r}' - \mathbf{z}^* \cdot \mathbf{c}(I', (\omega'_{\mathsf{M}}, \mathbf{g}', \mathbf{r}'), \mathbf{h}')$. Since $\Phi_{inj}(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h}) = \Phi_{inj}(I', (\omega'_{\mathsf{M}}, \mathbf{g}', \mathbf{r}'), \mathbf{h}')$, by Claim 6, $(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h})$ and $(I', (\omega'_{\mathsf{M}}, \mathbf{g}', \mathbf{r}'), \mathbf{h}')$ fix the same execution and therefore also $\mathbf{c}(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h}) = \mathbf{c}(I', (\omega'_{\mathsf{M}}, \mathbf{g}', \mathbf{r}'), \mathbf{h}')$. This implies $\mathbf{r} = \mathbf{r}'$, leading to the contradiction $(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h}) = (I', (\omega'_{\mathsf{M}}, \mathbf{g}', \mathbf{r}'), \mathbf{h}')$. Hence, Φ_{inj} is injective. □

Bounding the Probability of Obtaining the Most Likely Preimage. We now define the sets $\mathcal{B} := \bigcup_{i,j} \mathcal{B}_{i,j}$ and its complement $\mathcal{G} := \mathcal{W} \setminus \mathcal{B}$. That is, for all $(I, \omega, \mathbf{h}) \in \mathcal{G}$, we have $\hat{\boldsymbol{\chi}}_k(I, \omega, \mathbf{h}) = C_k(I, \omega, \mathbf{h}), \forall k \in [Q_{\mathsf{S}_2} + 1]$. The following lemma will help to upper bound the probability that $\hat{\boldsymbol{\chi}}$ takes different values (i.e., differs in at least one component) as a result of distinct instances $I = (\mathsf{sk}, \mathsf{par}), I' = (\mathsf{sk} + \mathbf{z}^*, \mathsf{par})$.

Lemma 5.3.10. *For any fixed* $(I, (\omega_{\mathsf{M}}, \boldsymbol{g}, \boldsymbol{r})) \in \mathcal{I} \times \Omega$,

$$\Pr_{\boldsymbol{h} \leftarrow_{\$} C^{Q_{\mathsf{H}}}}[(I, (\omega_{\mathsf{M}}, \boldsymbol{g}, \boldsymbol{r}), \boldsymbol{h}) \in \mathcal{G} \wedge \Phi(I, (\omega_{\mathsf{M}}, \boldsymbol{g}, \boldsymbol{r}), \boldsymbol{h}) \in \mathcal{G}] \leq \left(\frac{|S_c|}{|C|}\right)^{Q_{\mathsf{S}_2}} \cdot \frac{Q_{\mathsf{H}}^{Q_{\mathsf{S}_2}+1} \cdot \binom{Q_{\mathsf{S}_1}+Q_{\mathsf{S}_2}}{Q_{\mathsf{S}_1}}}{|C|}.$$

Proof. We argue by contradiction. Thus, assume that there exists some $(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r})) \in \mathcal{I} \times \Omega$:

$$\Pr_{\mathbf{h} \leftarrow_{\$} C^{Q_{\mathsf{H}}}}[(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h}) \in \mathcal{G} \wedge \Phi(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h}) \in \mathcal{G}] > \left(\frac{|S_c|}{|C|}\right)^{Q_{\mathsf{S}_2}} \cdot \frac{Q_{\mathsf{H}}^{Q_{\mathsf{S}_2}+1} \cdot \binom{Q_{\mathsf{S}_1}+Q_{\mathsf{S}_2}}{Q_{\mathsf{S}_1}}}{|C|}.$$

Notice that the internal variables $\hat{\mathbf{J}}_j(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h}), j = 1, \ldots, Q_{\mathsf{S}_2} + 1$ can be fixed to

pairwise-distinct values $\{u_1, \ldots, u_{Q_{\mathsf{S}_2}+1}\} \in [Q_{\mathsf{H}}]^{Q_{\mathsf{S}_2}+1}$ such that:

$$\Pr_{\mathbf{h} \leftarrow_{\$} C^{Q_{\mathsf{H}}}} \left[\begin{array}{c} ((I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h}) \in \mathcal{G}) \wedge (\Phi(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h}) \in \mathcal{G}) \\ \wedge \left(\forall j : \hat{\mathbf{J}}_j(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h}) = u_j\right) \end{array} \right] > \left(\frac{|S_c|}{|C|}\right)^{Q_{\mathsf{S}_2}} \cdot \frac{\binom{Q_{\mathsf{S}_1}+Q_{\mathsf{S}_2}}{Q_{\mathsf{S}_1}}}{|C|}.$$

Then, the vector of blinded challenges $\mathbf{c}(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h})$ can also be fixed to a vector $\mathbf{d} \in (S_c \cup \{\perp\})^{Q_{\mathsf{S}_1}+Q_{\mathsf{S}_2}}$ with exactly Q_{S_1} entries equal to $\perp$ which has the property:

$$\Pr_{\mathbf{h} \leftarrow_{\$} C^{Q_{\mathsf{H}}}} \left[\begin{array}{c} ((I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h}) \in \mathcal{G}) \wedge (\Phi(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h}) \in \mathcal{G}) \\ \wedge \left(\forall j : \hat{\mathbf{J}}_j(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h}) = u_j\right) \wedge (\mathbf{c}(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h}) = \mathbf{d}) \end{array} \right] > \frac{1}{|C|^{Q_{\mathsf{S}_2}} \cdot |C|}.$$

Finally, all entries of $\mathbf{h}$ with (distinct) indices $\{v_1, \ldots, v_{Q_{\mathsf{H}}-Q_{\mathsf{S}_2}-1}\}$ from $[Q_{\mathsf{H}}] \backslash \{u_1, \ldots, u_{Q_{\mathsf{S}_2}+1}\}$ can be fixed to those of some vector $\bar{\mathbf{h}} := (\bar{h}_1, \ldots, \bar{h}_{Q_{\mathsf{H}}-Q_{\mathsf{S}_2}-1}) \in C^{Q_{\mathsf{H}}-Q_{\mathsf{S}_2}-1}$:

$$\Pr_{\mathbf{h} \leftarrow_{\$} C^{Q_{\mathsf{H}}}} \left[\begin{array}{c} ((I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h}) \in \mathcal{G}) \wedge (\Phi(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h}) \in \mathcal{G}) \\ \wedge \left(\forall j : \hat{\mathbf{J}}_j(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h}) = u_j\right) \wedge (\mathbf{c}(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h}) = \mathbf{d}) \\ \wedge \left(\forall \iota : \mathbf{h}_{v_\iota} = \bar{\mathbf{h}}_\iota\right) \end{array} \right]$$
$$> \frac{1}{|C|^{Q_{\mathsf{S}_2}+1}|C|^{Q_{\mathsf{H}}-Q_{\mathsf{S}_2}-1}} = \frac{1}{|C|^{Q_{\mathsf{H}}}}.$$

Because random variable $\mathbf{h}$ takes a particular value $\mathbf{k} \in C^{Q_{\mathsf{H}}}$ with probability exactly $1/|C|^{Q_{\mathsf{H}}}$, the event in the above probability must contain at least two *distinct* elements $\mathbf{k}, \mathbf{k}' \in C^{Q_{\mathsf{H}}}$. Furthermore, the above probability statement guarantees that $\mathbf{k}$ and $\mathbf{k}'$ have identical entries, except for those with indices in $\{u_1, \ldots, u_{Q_{\mathsf{S}_2}+1}\}$. Thus w.l.o.g., let $i = u_t, t \in [Q_{\mathsf{S}_2} + 1]$ be the first index such that: $\mathbf{k}_i \neq \mathbf{k}'_i$ and $\mathbf{k}_m = \mathbf{k}'_m, \forall m < i$.

- **Case 1:** If sSess_t was completed (i.e., flag set to closed), then

$$\begin{aligned} C_t(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{k}) &= c_{t, \hat{\mathbf{J}}_t(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{k})}(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{k}) \\ &= c_{t,u_t}(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{k}) = c_{t,u_t}(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{k}') \\ &= c_{t, \hat{\mathbf{J}}_t(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{k}')}(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{k}') = C_t(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{k}') \end{aligned} \tag{5.4}$$

By Claim 6, $\hat{\mathbf{J}}_t(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{k}) = \hat{\mathbf{J}}_t(\Phi(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{k})) = u_t$ and $\hat{\mathbf{s}}'_t(I, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{k}) =$

$\hat{\mathbf{s}}'_t(\Phi(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k}))$. Furthermore, because $(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k}) \in \mathcal{G}$ and $\Phi(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k})) \in \mathcal{G}$, we know that $\hat{\boldsymbol{\chi}}_t(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k}) = C_t(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k}) = \hat{\mathbf{s}}'_t(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k}) - \mathsf{sk}\cdot\mathbf{k}_{u_t}$ and $\hat{\boldsymbol{\chi}}_t(\Phi(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k})) = C_t(\Phi(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k})) = \hat{\mathbf{s}}'_t(\Phi(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k}))-(\mathsf{sk}+\mathbf{z}^*)\cdot\mathbf{k}_{u_t}$. Putting this together, we obtain:

$$\begin{aligned}
C_t(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k}) &= \hat{\mathbf{s}}'_t(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k}) - \mathsf{sk}\cdot\mathbf{k}_{u_t} \\
&= \hat{\mathbf{s}}'_t(\Phi(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k})) - \mathsf{sk}\cdot\mathbf{k}_{u_t} \\
&= \hat{\mathbf{s}}'_t(\Phi(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k})) - \mathsf{sk}\cdot\mathbf{k}_{u_t} + \mathbf{z}^*\cdot\mathbf{k}_{u_t} - \mathbf{z}^*\cdot\mathbf{k}_{u_t} \\
&= \hat{\mathbf{s}}'_t(\Phi(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k})) - (\mathsf{sk}+\mathbf{z}^*)\cdot\mathbf{k}_{u_t} + \mathbf{z}^*\cdot\mathbf{k}_{u_t} \\
&= C_t(\Phi(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k})) + \mathbf{z}^*\cdot\mathbf{k}_{u_t} \qquad (5.5)
\end{aligned}$$

Similarly, we can show that:

$$C_t(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k}') = C_t(\Phi(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k}')) + \mathbf{z}^*\cdot\mathbf{k}'_{u_t}. \qquad (5.6)$$

Combining (in this order) equations (5.5), (5.4), and (5.6), we obtain:

$$\begin{aligned}
C_t(\Phi(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k})) + \mathbf{z}^*\cdot\mathbf{k}_{u_t} &= C_t(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k}) \\
&= C_t(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k}') \\
&= C_t(\Phi(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k}')) + \mathbf{z}^*\cdot\mathbf{k}'_{u_t} \qquad (5.7)
\end{aligned}$$

Because we have fixed $\mathbf{c}(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k}) = \mathbf{c}(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k}') = \mathbf{d}$, we have:

$$\begin{aligned}
&C_t(\Phi(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k})) \qquad (5.8) \\
&= C_t(I',(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}-\mathbf{z}^*\cdot\mathbf{c}(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k})),\mathbf{k}) \\
&= C_t(I',(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}-\mathbf{z}^*\cdot\mathbf{d}),\mathbf{k}) \\
&= C_t(I',(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}-\mathbf{z}^*\cdot\mathbf{d}),\mathbf{k}') \qquad (5.9) \\
&= C_t(I',(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}-\mathbf{z}^*\cdot\mathbf{c}(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k}')),\mathbf{k}') \\
&= C_t(\Phi(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k}')), \qquad (5.10)
\end{aligned}$$

where (5.9) follows again from the facts that $\forall j < u_t : \mathbf{k}_j = \mathbf{k}'_j$ and $\hat{\mathbf{J}}_t(\Phi(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k})) = \hat{\mathbf{J}}_t(\Phi(I,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{k}')) = u_t$. By combining (5.7) and (5.10), it now follows that $\mathbf{z}^* \cdot \mathbf{k}_{u_t} = \mathbf{z}^* \cdot \mathbf{k}'_{u_t}$. Thus, torsion-freeness of $\mathbf{z}^*$ implies that $\mathbf{k}_{u_t} = \mathbf{k}'_{u_t}$ which contradicts the assumption that $\mathbf{k}_{u_t} \neq \mathbf{k}'_{u_t}$.

- **Case 2:** If $\mathbf{sSess}_t$ was revoked, then let $\mathbf{a}_{u_t}, \mathbf{b}_{u_t}$ (respectively, $\mathbf{a}'_{u_t}, \mathbf{b}'_{u_t}$) be the blinding parameters used to obtain $\mathbf{k}_{u_t}$ (respectively, $\mathbf{k}'_{u_t}$). Notice that these are the same for both random oracle vectors since the query to oracle G is made before $\mathbf{k}_{u_t}$ and $\mathbf{k}'_{u_t}$ are returned through random oracle H. Furthermore, by the collision-freeness of G, the commitment binds to the same blinding parameters. Therefore, the blinded challenges are $\mathbf{k}_{u_t} + \mathbf{b}_{u_t} = \mathbf{d}_{u_t} = \mathbf{k}'_{u_t} + \mathbf{b}_{u_t}$ which implies that $\mathbf{k}_{u_t} = \mathbf{k}'_{u_t}$, which again contradicts the hypothesis that $\mathbf{k}_{u_t} \neq \mathbf{k}'_{u_t}$.

□

We can lift the restriction of fixed $(I,\omega) \in \mathcal{I} \times \Omega$, thus obtaining the following:

Corollary 5.3.2. $\Pr_{(I,\omega,\boldsymbol{h}) \leftarrow_{\$} (\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,(\omega_{\mathsf{M}},\boldsymbol{g},\boldsymbol{r}),\boldsymbol{h}) \in \mathcal{G} \wedge \Phi(I,(\omega_{\mathsf{M}},\boldsymbol{g},\boldsymbol{r}),\boldsymbol{h}) \in \mathcal{G}] \leq \left(\frac{|S_c|}{|C|}\right)^{Q_{\mathsf{S}_2}} \cdot \frac{Q_{\mathsf{H}}^{Q_{\mathsf{S}_2}+1} \cdot \binom{Q_{\mathsf{S}_1}+Q_{\mathsf{S}_2}}{Q_{\mathsf{S}_1}}}{|C|}$.

Showing that $\mathcal{W}_{inj}$ is not too small. Let $\mathcal{D}_{sk}$ and $\mathcal{D}_r$ denote the sets of secret keys and $\mathbf{r}$'s, respectively; then $\mathcal{E} := \mathcal{D}_{sk} \times \mathcal{D}_r^{Q_{\mathsf{S}_1}}$ and $\mathcal{E}_{inj} := \{(\mathsf{sk},\mathbf{r}) \in \mathcal{E} \mid (\mathsf{sk}+\mathbf{z}^*, \mathbf{r}-\mathbf{c}\cdot\mathbf{z}^*) \in \mathcal{E}, \forall \mathbf{c} \in S_c\} \subset \mathcal{E}$. Accordingly, $\forall (\mathsf{sk},\mathbf{r}) \in \mathcal{E} \setminus \mathcal{E}_{inj} : (\mathsf{sk}+\mathbf{z}^*, \mathbf{r}-\mathbf{c}\cdot\mathbf{z}^*) \notin \mathcal{E}$. Since Φ maps sk to $\mathsf{sk}+\mathbf{z}^*$ and $\mathbf{r}$ to $\mathbf{r}-\mathbf{c}(I,\omega,\mathbf{h})\cdot\mathbf{z}^*$, if $(\mathsf{sk},\mathbf{r}) \in \mathcal{E}_{inj}$ and $(\mathsf{sk},(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{h}) \in \mathcal{W}$ then $(\mathsf{sk},(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{h}) \in \mathcal{W}_{inj}$.

The following two lemmas can easily be proven:

Lemma 5.3.11. $\Pr_{(\mathsf{sk},(\omega_{\mathsf{M}},\boldsymbol{g},\boldsymbol{r}),\boldsymbol{h}) \leftarrow_{\$} (\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(\mathsf{sk},(\omega_{\mathsf{M}},\boldsymbol{g},\boldsymbol{r}),\boldsymbol{h}) \in \mathcal{W}_{inj}] \geq \varepsilon^2/8$.

Proof. In the Generalized Splitting Lemma, we set $\mathcal{X}_1 := \mathcal{D}_{sk}$ and $\mathcal{X}_3 := \mathcal{D}_r^{Q_{\mathsf{S}_1}}$ and accordingly, $S := \{1,3\}$. This implies the existence of a set $\mathcal{B}_{S,\varepsilon/2}$ such that:

$$\Pr_{(\mathsf{sk},(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{h}) \leftarrow_{\$} (\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(\mathsf{sk},(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{h}) \in \mathcal{B}_{S,\varepsilon/2}] \geq \varepsilon/2.$$

and such that for all $(\mathsf{sk}, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h}) \in \mathcal{B}_{S,\varepsilon/2}$,

$$\Pr_{(\mathsf{sk}',\mathbf{r}')\leftarrow_{\$}\mathcal{E}}[(\mathsf{sk}', (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}'), \mathbf{h}) \in \mathcal{W}] \geq \varepsilon/2.$$

The latter inequality can be rewritten as

$$\begin{aligned}
&\Pr_{(\mathsf{sk}',\mathbf{r}')\leftarrow_{\$}\mathcal{E}}[(\mathsf{sk}', (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}'), \mathbf{h}) \in \mathcal{W} \wedge (\mathsf{sk}', \mathbf{r}') \in \mathcal{E} \setminus \mathcal{E}_{inj}] \\
&+ \Pr_{(\mathsf{sk}',\mathbf{r}')\leftarrow_{\$}\mathcal{E}}[(\mathsf{sk}', (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}'), \mathbf{h}) \in \mathcal{W} \wedge (\mathsf{sk}', \mathbf{r}') \in \mathcal{E}_{inj}] \\
&= \Pr_{(\mathsf{sk}',\mathbf{r}')\leftarrow_{\$}\mathcal{E}}[(\mathsf{sk}', (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}'), \mathbf{h}) \in \mathcal{W}] \geq \varepsilon/2. \qquad (5.11)
\end{aligned}$$

By the $(\varepsilon, Q_{\mathsf{S}_1})$-regularity of H, at most an $\varepsilon/4$ fraction of $(\mathsf{sk}', \mathbf{r}') \in \mathcal{E}$ satisfy $(\mathsf{sk}', \mathbf{r}') \in \mathcal{E} \setminus \mathcal{E}_{inj}$. Thus, for all $(I, \omega, \mathbf{h}) = (\mathsf{sk}, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h}) \in \mathcal{B}_{S,\varepsilon/2}$,

$$\begin{aligned}
&\Pr_{(\mathsf{sk}',\mathbf{r}')\leftarrow_{\$}\mathcal{E}}[(\mathsf{sk}', (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}'), \mathbf{h}) \in \mathcal{W} \wedge (\mathsf{sk}', \mathbf{r}') \in \mathcal{E} \setminus \mathcal{E}_{inj}] \\
&\leq \Pr_{(\mathsf{sk}',\mathbf{r}')\leftarrow_{\$}\mathcal{E}}[(\mathsf{sk}', \mathbf{r}') \in \mathcal{E} \setminus \mathcal{E}_{inj}] \leq \varepsilon/4
\end{aligned}$$

By inequality (5.11), we obtain that for all $(I, \omega, \mathbf{h}) = (\mathsf{sk}, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h}) \in \mathcal{B}_{S,\varepsilon/2}$,

$$\begin{aligned}
\varepsilon/2 &\leq \Pr_{(\mathsf{sk}',\mathbf{r}')\leftarrow_{\$}\mathcal{E}}[(\mathsf{sk}', (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}'), \mathbf{h}) \in \mathcal{W}] \\
&= \Pr_{(\mathsf{sk}',\mathbf{r}')\leftarrow_{\$}\mathcal{E}}[(\mathsf{sk}', (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}'), \mathbf{h}) \in \mathcal{W} \wedge (\mathsf{sk}', \mathbf{r}') \in \mathcal{E}_{inj}] \\
&+ \Pr_{(\mathsf{sk}',\mathbf{r}')\leftarrow_{\$}\mathcal{E}}[(\mathsf{sk}', (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}'), \mathbf{h}) \in \mathcal{W} \wedge (\mathsf{sk}', \mathbf{r}') \in \mathcal{E} \setminus \mathcal{E}_{inj}] \\
&\leq \Pr_{(\mathsf{sk}',\mathbf{r}')\leftarrow_{\$}\mathcal{E}}[(\mathsf{sk}', (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}'), \mathbf{h}) \in \mathcal{W} \wedge (\mathsf{sk}', \mathbf{r}') \in \mathcal{E}_{inj}] + \varepsilon/4 \\
&\leq \Pr_{(\mathsf{sk}',\mathbf{r}')\leftarrow_{\$}\mathcal{E}}[(\mathsf{sk}', (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}'), \mathbf{h}) \in \mathcal{W}_{inj}] + \varepsilon/4
\end{aligned}$$

which implies that for all $(I, \omega, \mathbf{h}) = (\mathsf{sk}, (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}), \mathbf{h}) \in \mathcal{B}_{S,\varepsilon/2}$,

$$\Pr_{(\mathsf{sk}',\mathbf{r}')\leftarrow_{\$}\mathcal{E}}[(\mathsf{sk}', (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}'), \mathbf{h}) \in \mathcal{W}_{inj}] \geq \varepsilon/4.$$

Putting things together, we have:

$$
\begin{aligned}
&\Pr_{(sk,(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(\mathsf{sk},(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{h})\in\mathcal{W}_{inj}]\\
&=\Pr_{(\mathsf{sk},(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}}),(\mathsf{sk}',\mathbf{r}')\leftarrow_{\$}\mathcal{E}}[(\mathsf{sk}',(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}'),\mathbf{h})\in\mathcal{W}_{inj}]\\
&=\sum_{(\hat{I},\hat{\omega},\hat{\mathbf{h}})\in(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}\Pr[(\mathsf{sk}',(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}'),\mathbf{h})\in\mathcal{W}_{inj}\wedge(\mathsf{sk},(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{h})=(\hat{I},\hat{\omega},\hat{\mathbf{h}})]\\
&\geq\sum_{(\hat{I},\hat{\omega},\hat{\mathbf{h}})\in\mathcal{B}_{S,\varepsilon/2}}\Pr[(\mathsf{sk}',(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}'),\mathbf{h})\in\mathcal{W}_{inj}\wedge(\mathsf{sk},(\omega_{\mathsf{M}},\mathbf{g},\mathbf{r}),\mathbf{h})=(\hat{I},\hat{\omega},\hat{\mathbf{h}})]\\
&=\sum_{(\hat{I},\hat{\omega},\hat{\mathbf{h}})\in\mathcal{B}_{S,\varepsilon/2}}\Pr_{(\mathsf{sk}',\mathbf{r}')\leftarrow_{\$}\mathcal{E}}[(\mathsf{sk}',(\hat{\omega}_{\mathsf{M}},\hat{\mathbf{g}},\mathbf{r}'),\hat{\mathbf{h}})\in\mathcal{W}_{inj}]\cdot\Pr_{(\hat{I},\hat{\omega},\hat{\mathbf{h}})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\mathbf{h})=(\hat{I},\hat{\omega},\hat{\mathbf{h}})]\\
&\geq\frac{\varepsilon}{4}\cdot\sum_{(\hat{I},\hat{\omega},\hat{\mathbf{h}})\in\mathcal{B}_{S,\varepsilon/2}}\Pr_{(I,\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\mathbf{h})=(\hat{I},\hat{\omega},\hat{\mathbf{h}})]\\
&=\frac{\varepsilon}{4}\cdot\Pr_{(I,\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\mathbf{h})\in\mathcal{B}_{S,\varepsilon/2}]\geq\frac{\varepsilon}{4}\cdot\frac{\varepsilon}{2}=\frac{\varepsilon^2}{8}.
\end{aligned}
$$

□

The following lemma can be proven analogously to [98].

Lemma 5.3.12. $\Pr_{(I,\omega,\boldsymbol{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\boldsymbol{h})\in\mathcal{B}]\geq\frac{1}{2}\left(\frac{\varepsilon^2}{8}-\left(\frac{|S_c|}{|C|}\right)^{Q_{\mathsf{S}_2}}\cdot\frac{Q_{\mathsf{H}}^{Q_{\mathsf{S}_2}+1}\cdot\binom{Q_{\mathsf{S}_1}+Q_{\mathsf{S}_2}}{Q_{\mathsf{S}_1}}}{|C|}\right).$

Proof. In the following, let $\mathcal{G}_{inj}:=\mathcal{W}_{inj}\cap\mathcal{G}$ and $\mathcal{B}_{inj}:=\mathcal{W}_{inj}\cap\mathcal{B}$. Since for all $(I,\omega,\mathbf{h})\in\mathcal{W}_{inj}$, we have $\Phi(I,\omega,\mathbf{h})=\Phi_{inj}(I,\omega,\mathbf{h})\in\mathcal{W}=\mathcal{G}\cup\mathcal{B}$, we can partition $\mathcal{G}_{inj}$ into subsets $\mathcal{G}^g_{inj}$ and $\mathcal{G}^b_{inj}$, such that all elements in $\mathcal{G}^g_{inj}$ are mapped into $\mathcal{G}$ via Φ_{inj}, while elements in $\mathcal{G}^b_{inj}$ are mapped into $\mathcal{B}$ via Φ_{inj}. It follows that:

$$
\begin{aligned}
&\Pr_{(I,\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\mathbf{h})\in\mathcal{G}_{inj}]\\
&=\Pr_{(I,\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\mathbf{h})\in\mathcal{G}^g_{inj}]+\Pr_{(I,\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\boldsymbol{h})\in\mathcal{G}^b_{inj}]
\end{aligned}
\tag{5.12}
$$

By Corollary 5.3.2 we also know that:

$$\Pr_{(I,\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\mathbf{h})\in\mathcal{G}^{g}_{inj}]\leq\left(\frac{|S_c|}{|C|}\right)^{Q_{\mathsf{S}_2}}\cdot\frac{Q_{\mathsf{H}}^{Q_{\mathsf{S}_2}+1}\cdot\binom{Q_{\mathsf{S}_1}+Q_{\mathsf{S}_2}}{Q_{\mathsf{S}_1}}}{|C|} \tag{5.13}$$

Because Φ_{inj} is injective, we have:

$$\Pr_{(I,\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\mathbf{h})\in\mathcal{G}^{b}_{inj}]\leq\Pr_{(I,\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\mathbf{h})\in\mathcal{B}] \tag{5.14}$$

By combining (5.12), (5.13), and (5.14), we infer that:

$$\begin{aligned}&\Pr_{(I,\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\mathbf{h})\in\mathcal{G}_{inj}]\\&\leq\left(\frac{|S_c|}{|C|}\right)^{Q_{\mathsf{S}_2}}\cdot\frac{Q_{\mathsf{H}}^{Q_{\mathsf{S}_2}+1}\cdot\binom{Q_{\mathsf{S}_1}+Q_{\mathsf{S}_2}}{Q_{\mathsf{S}_1}}}{|C|}+\Pr_{(I,\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\mathbf{h})\in\mathcal{B}].\end{aligned}$$

From this, we can lower bound $\Pr_{(I,\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\mathbf{h})\in\mathcal{B}]$ as

$$\begin{aligned}&\Pr_{(I,\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\mathbf{h})\in\mathcal{B}]\geq\Pr_{(I,\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\mathbf{h})\in\mathcal{B}_{inj}]\\&=\Pr_{(I,\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\mathbf{h})\in\mathcal{W}_{inj}]-\Pr_{(I,\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\mathbf{h})\in\mathcal{G}_{inj}]\\&\geq\Pr_{(I,\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\mathbf{h})\in\mathcal{W}_{inj}]-\Pr_{(I,\omega,\boldsymbol{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\mathbf{h})\in\mathcal{B}]\\&-\left(\frac{|S_c|}{|C|}\right)^{Q_{\mathsf{S}_2}}\cdot\frac{Q_{\mathsf{H}}^{Q_{\mathsf{S}_2}+1}\cdot\binom{Q_{\mathsf{S}_1}+Q_{\mathsf{S}_2}}{Q_{\mathsf{S}_1}}}{|C|}.\end{aligned}$$

However, by Lemma 5.3.11 we know that $\Pr_{(I,\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\mathbf{h})\in\mathcal{W}_{inj}]=\varepsilon^2/8$, which in turn implies that:

$$\Pr_{(I,\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\mathbf{h})\in\mathcal{B}]\geq\frac{1}{2}\left(\frac{\varepsilon^2}{8}-\left(\frac{|S_c|}{|C|}\right)^{Q_{\mathsf{S}_2}}\cdot\frac{Q_{\mathsf{H}}^{Q_{\mathsf{S}_2}+1}\cdot\binom{Q_{\mathsf{S}_1}+Q_{\mathsf{S}_2}}{Q_{\mathsf{S}_1}}}{|C|}\right).$$

□

Completing the Proof of Lemma 5.3.8. We are now ready to prove Lemma 5.3.8, i.e., we show that there exist $i \in [Q_{\mathsf{S}_2} - Q_{\mathsf{rev}} + 1], j \in [Q_{\mathsf{H}}]$ such that $\beta_{i,j} > \frac{1}{2Q_{\mathsf{H}}(Q_{\mathsf{S}_2}-Q_{\mathsf{rev}}+1)} \cdot \left(\frac{\varepsilon^2}{8} - \left(\frac{|S_c|}{|C|}\right)^{Q_{\mathsf{S}_2}} \cdot \frac{Q_{\mathsf{H}}^{Q_{\mathsf{S}_2}+1} \cdot \binom{Q_{\mathsf{S}_1}+Q_{\mathsf{S}_2}}{Q_{\mathsf{S}_1}}}{|C|}\right)$. Toward a contradiction, suppose instead that for all $i \in [Q_{\mathsf{S}_2} - Q_{\mathsf{rev}} + 1], j \in [Q_{\mathsf{H}}]$, we have that:

$$\Pr_{(I,\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\mathbf{h}) \in \mathcal{B}_{i,j}] < \frac{1}{2Q_{\mathsf{H}}(Q_{\mathsf{S}_2} - Q_{\mathsf{rev}} + 1)} \cdot \left(\frac{\varepsilon^2}{8} - \left(\frac{|S_c|}{|C|}\right)^{Q_{\mathsf{S}_2}} \cdot \frac{Q_{\mathsf{H}}^{Q_{\mathsf{S}_2}+1} \cdot \binom{Q_{\mathsf{S}_1}+Q_{\mathsf{S}_2}}{Q_{\mathsf{S}_1}}}{|C|}\right).$$

By Lemma 5.3.12, we have:

$$\begin{aligned}
&\frac{1}{2}\left(\frac{\varepsilon^2}{8} - \left(\frac{|S_c|}{|C|}\right)^{Q_{\mathsf{S}_2}} \cdot \frac{Q_{\mathsf{H}}^{Q_{\mathsf{S}_2}+1} \cdot \binom{Q_{\mathsf{S}_1}+Q_{\mathsf{S}_2}}{Q_{\mathsf{S}_1}}}{|C|}\right) \leq \Pr_{(I,\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\mathbf{h}) \in \mathcal{B}] \\
&= \Pr_{(I,\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\mathbf{h}) \in \bigcup_{i,j} \mathcal{B}_{i,j}] \leq \sum_{i,j} \Pr_{(I,\omega,\mathbf{h})\leftarrow_{\$}(\mathcal{I}\times\Omega\times C^{Q_{\mathsf{H}}})}[(I,\omega,\mathbf{h}) \in \mathcal{B}_{i,j}] \\
&< \frac{1}{2}\left(\frac{\varepsilon^2}{8} - \left(\frac{|S_c|}{|C|}\right)^{Q_{\mathsf{S}_2}} \cdot \frac{Q_{\mathsf{H}}^{Q_{\mathsf{S}_2}+1} \cdot \binom{Q_{\mathsf{S}_1}+Q_{\mathsf{S}_2}}{Q_{\mathsf{S}_1}}}{|C|}\right)
\end{aligned}$$

which is a contradiction. □

Extracting Collisions for LHF from the Forgery. We now describe an adversary B that makes use of the previously defined wrapping adversaries A_i in a black-box manner, in order to win in game $\mathbf{CR}_{\mathsf{LHF}}$. Notice that because B guesses the number of closed and revoked sessions that the OMUF adversary M will have at the end of the simulation, the advantage of winning in $\mathbf{CR}_{\mathsf{LHF}}$ is multiplied by a factor of $\frac{1}{(Q_{\mathsf{S}_2}+1)(Q_{\mathsf{rev}}+1)}$.

- **Setup:** On input $\mathsf{par} \leftarrow_{\$} \mathsf{LHF.PGen}(1^\lambda)$, B guesses the number of closed sessions $\kappa^* \leftarrow_{\$} \{0\} \cup [Q_{\mathsf{S}_2}]$, the number of revoked sessions $r^* \leftarrow_{\$} \{0\} \cup [Q_{\mathsf{rev}}]$, and an index $i^* \leftarrow_{\$} [Q_{\mathsf{S}_2} - Q_{\mathsf{rev}} + 1]$, a secret key $\mathsf{sk} \leftarrow_{\$} \mathcal{D}_{sk}$, a vector of random oracle responses $\mathbf{h} \leftarrow_{\$} C^{Q_{\mathsf{H}}}$, randomness $\omega := (\omega_{\mathsf{M}}, \mathbf{g}, \mathbf{r}) \leftarrow_{\$} \Omega$, and it runs A_{i^*} on input $I = ((\mathsf{sk}, \mathsf{par}), \mathbf{h}; ((\omega_{\mathsf{M}}, \kappa^*, r^*), \mathbf{g}, \mathbf{r}))$.

- **Intermediate Phase:** When A_{i^*} returns a pair $(\hat{\mathbf{J}}_{i^*}, \hat{\boldsymbol{\chi}}_{i^*})$, and as long as $\hat{\mathbf{J}}_{i^*} \neq 0$ (if not, B aborts by returning $\perp$), B conditionally resamples a second random vector $\mathbf{h}' \leftarrow_{\$} C^{Q_{\mathsf{H}}}|\mathbf{h}_{[\hat{\mathbf{J}}_{i^*}-1]}$ and runs A_{i^*} a second time with the same randomness $((\omega_{\mathsf{M}}, \kappa^*, r^*), \mathbf{g}, \mathbf{r})$ and the same instance I but replacing $\mathbf{h}$ by $\mathbf{h}'$.

- **Output Determination:** When A_{i^*} returns a pair $(\hat{\mathbf{J}}'_{i^*}, \hat{\boldsymbol{\chi}}'_{i^*})$, B outputs $(\hat{\boldsymbol{\chi}}_{i^*}, \hat{\boldsymbol{\chi}}'_{i^*})$ iff $\hat{\mathbf{J}}_{i^*} \neq \hat{\mathbf{J}}'_{i^*}$ and $\hat{\boldsymbol{\chi}}_{i^*} \neq \hat{\boldsymbol{\chi}}'_{i^*}$. Otherwise, it outputs $\perp$.

Analysis of Adversary B. Notice that in the case that B does not abort, by definition of A_{i^*} we have,

$$\mathsf{F}(\hat{\boldsymbol{\chi}}_{i^*}) = \mathsf{F}(\hat{\mathbf{s}}'_{i^*} - \hat{\mathbf{h}}_{i^*} \cdot \mathsf{sk}) = \mathbf{S}'_{\hat{\mathbf{J}}_{i^*}} - \mathbf{h}_{\hat{\mathbf{J}}_{i^*}} \cdot \mathsf{pk} = \mathbf{R}'_{\hat{\mathbf{J}}_{i^*}}$$

Because A_{i^*} sees identical answers for the first $\hat{\mathbf{J}}_{i^*}-1$ queries to H, it behaves identically in both runs until it receives the answer to the $\hat{\mathbf{J}}_{i^*}$-th query to H. In particular, A_{i^*} poses the same $\hat{\mathbf{J}}_{i^*}$-th query to H which means that $\mathsf{F}(\hat{\boldsymbol{\chi}}_{i^*}) = \mathbf{R}'_{\hat{\mathbf{J}}_{i^*}}$ and therefore $\mathsf{F}(\hat{\boldsymbol{\chi}}_{i^*}) = \mathsf{F}(\hat{\boldsymbol{\chi}}'_{i^*})$.

For $\mathsf{par} \in \mathcal{P}$, we now consider:

$$\begin{aligned}
&\Pr_{(\hat{\boldsymbol{\chi}}_{i^*}, \hat{\boldsymbol{\chi}}'_{i^*}) \leftarrow_{\$} \mathsf{B}(\mathsf{par})}[\hat{\boldsymbol{\chi}}_{i^*} \neq \hat{\boldsymbol{\chi}}'_{i^*} \wedge \mathsf{F}(\hat{\boldsymbol{\chi}}_{i^*}) = \mathsf{F}(\hat{\boldsymbol{\chi}}'_{i^*})] \\
&= \sum_{j=1}^{Q_{\mathsf{H}}} \Pr[\hat{\boldsymbol{\chi}}_{i^*} \neq \hat{\boldsymbol{\chi}}'_{i^*} \wedge \mathsf{F}(\hat{\boldsymbol{\chi}}_{i^*}) = \mathsf{F}(\hat{\boldsymbol{\chi}}'_{i^*}) \wedge \hat{\mathbf{J}}_{i^*} = \hat{\mathbf{J}}'_{i^*} = j] \\
&= \sum_{j=1}^{Q_{\mathsf{H}}} \Pr[\hat{\boldsymbol{\chi}}_{i^*} \neq \hat{\boldsymbol{\chi}}'_{i^*} \wedge \hat{\mathbf{J}}_{i^*} = \hat{\mathbf{J}}'_{i^*} = j] = \sum_{j=1}^{Q_{\mathsf{H}}} \delta_{i^*,j} \geq \frac{1}{Q_{\mathsf{S}_2} - Q_{\mathsf{rev}} + 1} \cdot \max_{i \in [Q_{\mathsf{S}_2} - Q_{\mathsf{rev}} + 1]} \sum_{j=1}^{Q_{\mathsf{H}}} \delta_{i,j} \\
&\geq \max_{i,j} \frac{\beta_{i,j}}{2(Q_{\mathsf{S}_2} - Q_{\mathsf{rev}} + 1)} \left(\frac{\beta_{i,j}}{4} - \frac{1}{|C|} \right) \\
&> \frac{\frac{\varepsilon^2}{8} - \left(\frac{|S_c|}{|C|}\right)^{Q_{\mathsf{S}_2}} \cdot \frac{Q_{\mathsf{H}}^{Q_{\mathsf{S}_2}+1} \cdot \binom{Q_{\mathsf{S}_1}+Q_{\mathsf{S}_2}}{Q_{\mathsf{S}_1}}}{|C|}}{32 Q_{\mathsf{H}}^2 (Q_{\mathsf{S}_2} - Q_{\mathsf{rev}} + 1)^3} \cdot \left(\frac{\varepsilon^2}{8} - \left(\frac{|S_c|}{|C|}\right)^{Q_{\mathsf{S}_2}} \cdot \frac{Q_{\mathsf{H}}^{Q_{\mathsf{S}_2}+1} \cdot \binom{Q_{\mathsf{S}_1}+Q_{\mathsf{S}_2}}{Q_{\mathsf{S}_1}}}{|C|} - \frac{8 Q_{\mathsf{H}} (Q_{\mathsf{S}_2} - Q_{\mathsf{rev}} + 1)}{|C|} \right),
\end{aligned}$$

where for the first inequality we used that $\sum \delta_{i^*,j} = \max_i \sum \delta_{i,j}$ with probability at least $\frac{1}{Q_{\mathsf{S}_2} - Q_{\mathsf{rev}} + 1}$. Moreover, we have applied Lemmas 5.3.7 and 5.3.8 in the second to last and last inequality, respectively (relative to our choice of par). By reintroducing

randomness over the choice of parameters par, we finally obtain:

$$\varepsilon' = \mathbf{Adv}^{\mathbf{CR}}_{\mathsf{LHF}}(\mathsf{B}) = \Pr_{\mathsf{par}\leftarrow_{\$}\mathsf{LHF.PGen}(1^{\lambda}),(\hat{\mathbf{x}}_{i^*},\hat{\mathbf{x}}'_{i^*})\leftarrow_{\$}\mathsf{B}(\mathsf{par})}[\hat{\mathbf{x}}_{i^*} \neq \hat{\mathbf{x}}'_{i^*} \wedge \mathsf{F}(\hat{\mathbf{x}}_{i^*}) = \mathsf{F}(\hat{\mathbf{x}}'_{i^*})]$$

$$\geq \Pr_{\mathsf{par}\leftarrow_{\$}\mathsf{LHF.PGen}(1^{\lambda}),(\hat{\mathbf{x}}_{i^*},\hat{\mathbf{x}}'_{i^*})\leftarrow_{\$}\mathsf{B}(\mathsf{par})}[\hat{\mathbf{x}}_{i^*} \neq \hat{\mathbf{x}}'_{i^*} \wedge \mathsf{F}(\hat{\mathbf{x}}_{i^*}) = \mathsf{F}(\hat{\mathbf{x}}'_{i^*}) \mid \mathsf{par} \in \mathcal{P}] \cdot \Pr_{\mathsf{par}\leftarrow_{\$}\mathsf{LHF.PGen}(1^{\lambda})}[\mathsf{par} \in \mathcal{P}]$$

$$\geq \frac{\frac{\varepsilon^2}{8} - \left(\frac{|S_c|}{|C|}\right)^{Q_{\mathsf{S}_2}} \cdot \frac{Q_{\mathsf{H}}^{Q_{\mathsf{S}_2}+1} \cdot \binom{Q_{\mathsf{S}_1}+Q_{\mathsf{S}_2}}{Q_{\mathsf{S}_1}}}{|C|}}{32Q_{\mathsf{H}}^2(Q_{\mathsf{S}_2} - Q_{\mathsf{rev}} + 1)^3} \cdot \left(\frac{\varepsilon^2}{8} - \left(\frac{|S_c|}{|C|}\right)^{Q_{\mathsf{S}_2}} \cdot \frac{Q_{\mathsf{H}}^{Q_{\mathsf{S}_2}+1} \cdot \binom{Q_{\mathsf{S}_1}+Q_{\mathsf{S}_2}}{Q_{\mathsf{S}_1}}}{|C|} - \frac{8Q_{\mathsf{H}}(Q_{\mathsf{S}_2} - Q_{\mathsf{rev}} + 1)}{|C|}\right) \cdot \frac{\varepsilon}{2}$$

$$= O\left(\left(\varepsilon^2 - \left(\frac{|S_c|}{|C|}\right)^{Q_{\mathsf{S}_2}} \cdot \frac{Q_{\mathsf{H}}^{Q_{\mathsf{S}_2}+1} Q_{\mathsf{S}_1}^{Q_{\mathsf{S}_2}}}{|C|}\right)^2 \frac{1}{Q_{\mathsf{H}}^2 Q_{\mathsf{S}_2}^3}\right) \cdot \frac{\varepsilon}{2}$$

$$= O\left(\left(\varepsilon^2 - \left(\frac{|S_c|}{|C|}\right)^{Q_{\mathsf{S}_2}} \cdot \frac{Q_{\mathsf{H}}^{Q_{\mathsf{S}_2}+1} Q_{\mathsf{S}_1}^{Q_{\mathsf{S}_2}}}{|C|}\right)^3 \frac{1}{Q_{\mathsf{H}}^2 Q_{\mathsf{S}_2}^3}\right)$$

where the second-to-last equality holds for $Q_{\mathsf{S}_1} \geq Q_{\mathsf{S}_2} + Q_{\mathsf{rev}}$ and because $Q_{\mathsf{rev}} \leq Q_{\mathsf{S}_2}$. Finally, the last equality holds because $\varepsilon = O(\varepsilon^2)$.

□

5.4 A Concrete Instantiation Based on R – SIS

We now present a concrete instantiation of our generic construction from the R – SIS lattice assumption. The LHF we will rely on is the one from [122], which is also used in [98, 159].

Solving $\mathsf{R-SIS}_{q,n,m,d}$ is equivalent to finding short vectors in the related lattice $\Lambda_q^{\perp}(\hat{\mathbf{a}}) := \{\hat{\mathbf{z}} \in \mathcal{R}_q^m : \sum_{i=1}^m \mathbf{a}_i \cdot \mathbf{z}_i = 0\}$, where $\hat{\mathbf{a}} \leftarrow_{\$} \mathcal{R}_q^m$. [81] provides a classification of algorithms for finding short vectors in random lattices, in terms of the Hermite factor η. These algorithms compute a vector whose length is η^n times the length of the shortest lattice vector. To estimate the concrete hardness of $\mathsf{R-SIS}_{q,n,m,d}$ we use a Hermite factor of $\eta \approx 1.005$. To estimate the length of the shortest vector (in l_∞ norm), we use the following estimation from [122]: $\mathsf{sv}_\eta(n,q) := \min\{q, 2^{2\sqrt{n\log(q)\log(\eta)}} \cdot (n\log(q)/\log(\eta))^{-1/4}\}$.

Table 5.1: Parameter definitions for the lattice-based LHF.

Parameter	Definition	Instantiation
λ	main security parameter	128
n	integer power of 2	1024
m	dimension of a secret key vector $> \log(q)/\log(2d)$	179
q	prime $\geq 4dmn\sqrt{n}\log(n)$	2^{3574}
ι	# of irreducible factors of $X^n + 1$ modulo q	32
δ	l_∞ of a torsion-free element from the kernel $(2\delta+1)^{mn} > q^n, \delta < q/2$	2^{19}
d_{sk}	l_∞ of a secret key	2^{169}
$d_{c'}$	l_∞ of random oracle H's outputs	2^{85}
u	integer	4
v	integer	4
w	integer	4
d_β	$und_{c'}$	$\approx 2^{97}$
d_c	$d_\beta - d_{c'}, d_c < \frac{q^{1/\iota}}{2\sqrt{\iota}}$	$\approx 2^{97}$
d_r	$\geq vmn^2 d_{sk} d_c$	$\approx 2^{295}$
d_s	$d_r - nd_{sk}d_c$	$\approx 2^{295}$
d_α	$wmnd_s$	$\approx 2^{315}$
$d_{s'}$	$d_\alpha - d_s$	$\approx 2^{315}$
d	$d < \frac{1}{2}\mathsf{sv}_{1.005}(n,q)$	$\approx 2^{316}$
Signature size	$\approx n\log(2d_{c'}) + mn\log(2d_{s'}) + 2\lambda$	6.91 MB
Communication overhead (in case of a revocation)	$\approx mn\log(2d_r) + n\log(2d_c) + mn\log(2d_s)$ $+mn\log(2d_\alpha) + n\log(2d_\beta) + n\log(2d_{c'}) + 2\lambda$	12.97 MB +6.93 MB

For our scheme's unforgeability under revokes we will rely on the same conjecture as [98]:

Conjecture 2. *If $d < \mathsf{sv}_{1.005}(n,q)$ then no efficient algorithm can solve* $\mathsf{R-SIS}_{q,n,m,d}$.

The $\mathsf{R-SIS}$ **Linear Hash Function Family.** We set our parameters as described in column 2 of Table 5.1. In addition, we define the sets:

$$\mathcal{S} := \mathcal{R}_q, \quad \mathcal{D} := \mathcal{R}_q^m, \quad \text{and} \quad \mathcal{R} := \mathcal{R}_q,$$

As well as the family of sets:

$$\mathcal{B}(w) := \{\boldsymbol{f} \in \mathcal{R} : \|\boldsymbol{f}\|_\infty \leq w\}, w \in \mathbb{N}$$

To ensure that LHF is collision resistant, we need to set $d < \frac{1}{2}\mathsf{sv}_{1.005}(n,q)$ and select

$\mathcal{D}' \subseteq \mathcal{B}(d)$[12].

The filter sets are defined for xxx $\in \{\beta, c, c'\}$ and yyy $\in \{sk, r, s, s', \alpha\}$ as:

$$\mathcal{S}_{\text{xxx}} := \mathcal{B}(d_{\text{xxx}}) \subseteq \mathcal{S}, \quad \mathcal{D}_{\text{yyy}} := \mathcal{B}^m(d_{\text{yyy}}) \subseteq \mathcal{D}.$$

The following lemma from [159] is essential for estimating the membership of sums and products of elements from $\mathcal{D}$ to specific filter sets.

Lemma 5.4.1. *Let k, d_a, d_b and ϕ be positive integers, s.t. $d_b \geq \phi k n d_a$. Then, for all $\hat{\boldsymbol{a}} \in \mathcal{B}_q^k(d_a)$, we have*

$$\Pr_{\hat{\boldsymbol{b}} \leftarrow_{\$} \mathcal{B}_q^k(d_b)} [\|\hat{\boldsymbol{a}} + \hat{\boldsymbol{b}}\|_\infty \leq d_b - d_a] > e^{-1/\phi} - o(1).$$

Enclosedness Errors. First, we compute the enclosedness errors of LHF based on the parameters defined in Table 5.1. The following lemma is implied directly from the way the parameters are defined and by applying Lemma 5.4.1.

Lemma 5.4.2. *If $d_\beta, d_r, d_\alpha, d_c, d_s, d_{s'}$ are defined as in Table 5.1 and* LHF *is defined as above, then* LHF *has enclosedness errors equal to:*

$$(1 - e^{-1/u} + o(1), 1 - e^{-1/v} + o(1), 1 - e^{-1/w} + o(1)).$$

By Theorem 5.3.1, BSRS[LHF, H, G] has a correctness error of approximately 0.44.

Smoothness. It is easy to show that LHF is smooth. The proof is identical to the one for Lemma 4 in [98].

Lemma 5.4.3. *If d_s and d_c are defined as in Table 5.1, then* LHF *is smooth.*

Torsion-Free Elements from the Kernel. If d_c is set to be sufficiently small, then by selecting an appropriate prime q, we can apply the main result of [127] to ensure that there exists $\hat{\mathbf{z}}^* \in \mathcal{R}_q^m \setminus \{\mathbf{0}\}$ s.t. $\mathsf{F}(\hat{\mathbf{z}}^*) = \mathbf{0}$.

[12] The set $\mathcal{D}'$ is defined in equation (5.1).

Lemma 5.4.4. *Let $n \geq \iota > 1$ be powers of 2 and $q \equiv 2\iota + 1 \pmod{4\iota}$ be a prime. Then $X^n + 1$ factors into ι irreducible polynomials $X^{n/\iota} - r_j$ modulo q and any $\boldsymbol{y} \in \mathcal{R}_q \setminus \{\mathbf{0}\}$ satisfying $0 < \|\boldsymbol{y}\|_\infty < \frac{1}{\sqrt{\iota}} \cdot q^{1/\iota}$ is invertible in $\mathcal{R}_q$.*

Thus, we require $d_c < \frac{q^{1/\iota}}{2\sqrt{\iota}}$. Then for $\mathbf{c}_1, \mathbf{c}_2 \in S_c, (\mathbf{c}_1 - \mathbf{c}_2)\hat{\boldsymbol{z}}^* = \mathbf{0} \implies \mathbf{c}_1 = \mathbf{c}_2$ because otherwise $\mathbf{c}_1 - \mathbf{c}_2$ has an inverse and thus, $\hat{\boldsymbol{z}}^* = \mathbf{0}$. Thus, $\hat{\boldsymbol{z}}^*$ is a torsion-free element from the kernel.

We also need to ensure that there exists $\hat{\boldsymbol{z}}^* \in \mathcal{R}_q^m$ s.t. $\|\hat{\boldsymbol{z}}^*\|_\infty < q/2$. To this end, let $B_\delta := \{\hat{\mathbf{x}} \in \mathcal{R}_q^m : \|\hat{\mathbf{x}}\|_\infty \leq \delta\}$. If δ is set s.t. $|B_\delta| = (2\delta + 1)^{mn} > q^n = |\mathcal{R}|$, then by the pigeonhole principle, there exist distinct $\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2 \in B_\delta$ s.t. $\mathsf{F}(\hat{\mathbf{x}}_1) = \mathsf{F}(\hat{\mathbf{x}}_2)$. Then $\hat{\boldsymbol{z}}^* := \hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_2$ is a (short) torsion-free element from the kernel because $\|\hat{\boldsymbol{z}}^*\|_\infty = \|\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_2\|_\infty \leq \delta < q/2$.

Collision Resistance. To estimate the hardness of finding collisions in LHF we state the following simple lemma.

Lemma 5.4.5. *If $\mathsf{R-SIS}_{q,n,m,2d}$ is (ε, t)-hard, then LHF is (ε, t) collision-resistant.*

Proof. Let A be an adversary winning in the collision resistance game. A returns $(\hat{\boldsymbol{\chi}}_1, \hat{\boldsymbol{\chi}}_2)$ s.t. $\mathsf{F}(\hat{\boldsymbol{\chi}}_2 - \hat{\boldsymbol{\chi}}_1) = \mathbf{0}$. In particular, the values returned by our adversary against collision resistance from the proof of Theorem 5.3.5 are bounded by $\|\hat{\boldsymbol{\chi}}_i\|_\infty \leq d_{s'} + nd_{c'}d_{sk} < 2d_{s'}$. Thus, for the reduction we set $d = 2d_{s'}$. With the parameter set of Table 5.1, both d and $\frac{1}{2}\mathsf{sv}_\delta(n, q)$ are approximately equal to $\approx 2^{316}$. This implies that $(\hat{\boldsymbol{\chi}}_1, \hat{\boldsymbol{\chi}}_2)$ is a solution to $\mathsf{R-SIS}_{q,n,m,2d}$ because $\|\hat{\boldsymbol{\chi}}_2 - \hat{\boldsymbol{\chi}}_1\|_\infty < 2d < \mathsf{sv}_\delta(n, q)$. □

Unforgeability Under Revokes. To make the bound in Lemma 5.3.10 negligible, we need a sufficiently large challenge space $C = S_{c'}$. Concretely, we set $Q_{\mathsf{H}} = Q_{\mathsf{S}_1} = 2^{128}$ and $Q_{\mathsf{S}_2} = \mathsf{Q}_{\mathsf{rev}} = \log(128) = 7$ to safeguard against any attacks through a lattice variant of the Generalized ROS attack described in [98]. For our parameter choices, the bound in Lemma 5.3.10 is less than 2^{-128}.

Regularity. By selecting d_{sk} and d_r as in Table 5.1, our LHF is (ε, Q')-regular, where $\varepsilon = 2^{-128}$ and $Q' = Q_{\mathsf{S}_2} = 7$.

Lemma 5.4.6. *Let* $\varepsilon = 2^{-128}$ *and* $Q' = 7$. *Then for our selection of* d_{sk}, d_r, LHF *is* (ε, Q')*-regular, i.e.,*

$$\frac{|\mathcal{D}'_{sk}|}{|\mathcal{D}_{sk}|} \cdot \left(\frac{|\mathcal{D}'_r|}{|\mathcal{D}_r|}\right)^{Q'} \geq 1 - \varepsilon/4 = 1 - 2^{-130},$$

where

$$\mathcal{D}'_{sk} := \{\hat{\boldsymbol{sk}} \in \mathcal{D}_{sk} : \hat{\boldsymbol{sk}} + \hat{\boldsymbol{z}}^* \in \mathcal{D}_{sk}\}$$

and

$$\mathcal{D}'_r := \{\hat{\boldsymbol{r}} \in \mathcal{D}_r : \hat{\boldsymbol{r}} + \boldsymbol{c}\hat{\boldsymbol{z}}^* \in \mathcal{D}_r, \forall \boldsymbol{c} \in S_c\}.$$

Proof. By our choices of d_{sk} and d_r, we have:

$$\frac{|\mathcal{D}'_{sk}|}{|\mathcal{D}_{sk}|} \geq 1 - 2^{-130}, \left(\frac{|\mathcal{D}'_r|}{|\mathcal{D}_r|}\right)^{Q'} \geq 1 - 2^{-130}, \text{ and } d_r \geq vmn^2 d_{sk} d_c.$$

The result is then immediate by the Bernoulli inequality. □

Sizes. We select a prime $q \approx 2^{3574}$ so that $q \equiv 2\iota + 1 (\text{mod } 4\iota)$, where $\iota = 32$ and $X^n + 1$ factors into a product of ι irreducible polynomials modulo q and Lemma 5.4.4 can be applied. The signature consists of four parts: $\mathbf{c}', \hat{\mathbf{s}}', \gamma$, and ρ. The size for $\mathbf{c}'$ and $\hat{\mathbf{s}}'$ are respectively $n \log(2d'_c)$ and $mn \log(2d_{s'})$. Also, γ and ρ take up 128 bits each. Concretely, our instantiation outputs signatures of approximately 6.91 MB which is slightly smaller than the instantiation of [98] whose signatures are approximately 7.73 MB for the same security level (i.e., 128 bits). However, despite having an additional move, our scheme has a significantly smaller communication overhead of only 19.9 MB in the worst case (i.e., when a PoR is transmitted) compared to [98] which needs to transmit 440.83 MB per session (i.e., approximately 22.2 times larger compared to our instantiation). This is largely due to the fact that in the scheme of [98], the signer needs to transmit a large number of commitments during his first move. This results in the aforementioned bottleneck.

5.5 Conclusions, Open Problems and Future Work

In this chapter, we revisited the problem of rendering blind signature from linear hash function families with noticeable correctness error in the random oracle model (ROM) [98]. We proposed for the first time a new cryptographic primitive called *blind signatures with revocable sessions* (BSRS) to model four-move schemes in which the user is able to prove that he did not obtain a valid signature from a session and can ask for the session to be revoked [159]. We provided a general framework for constructing BSRS schemes from *any* linear hash function family with noticeable correctness error and proved its security in the ROM by expanding upon the techniques introduced in [98]. We instantiated our general framework from the standard $\mathsf{R}-\mathsf{SIS}$ assumption. Our lattice-based scheme relies on very simple cryptographic primitives and greatly outperforms the blind signature scheme of [98] in terms of communication overhead and achieves slightly more compact signature sizes. While our construction is still not practical enough to be used in practice, we believe that it may motivate future works in this area.

Chapter 6

Proxy Signatures from Ideal Lattices–Secure in All Rings

6.1 Introduction

A digital signature scheme allows a signer possessing a valid public/secret key pair to issue signatures on digital documents in a way that provides authentication of both the contents of the document, and the identity of its creator. However, in a real-world setting, the signer may be faced with overwhelming traffic (e.g.: having to accommodate thousands of users at any given time), or may need to become unavailable (e.g.: for maintenance) for a set amount of time, without denying users its services. Unfortunately, typical digital signatures do not cope with these problems very well.

Proxy signature schemes were introduced in [128] and aim to address precisely this problem. They typically allow an entity termed *the designator* to authorize another entity called a *proxy signer*, allowing the latter to issue signatures on its behalf. This provided flexibility is particularly useful when resources such as time and/or computational power are a concern. As such, proxy signature schemes have found numerous applications in distributed computing where delegation of rights is very commonplace. Some applications include e-cash systems [175], business-to-consumer mobile agents for e-commerce [56], grid computing [102], and global distribution networks.

6.1.1 Contributions and Related work.

Since their conception [128], numerous proxy signature schemes (and variants thereof) have appeared in the literature [112, 176, 175, 102]. However, the vast majority of them rely on classical number-theoretic hardness assumptions like the discrete logarithm problem or large integer factorization, which are known to be vulnerable against Shor's algorithm [169]. There have also been a few lattice-based proxy signature schemes [103, 183]. However, the construction of [103] was broken in [178], while the construction of [183] relies[1] on R – SIS with a specific choice for $\mathbf{f}$ (i.e., the quotient ring's underlying polynomial). While there are currently no algorithms exploiting the additional algebraic structure induced by choosing $\mathbf{f} = X^{2^k} + 1$, the recent works of [17, 61] show that the choice ring affects the hardness of either the worst-case or average-case problems. Following this observation, [124] proposed basing the security of lattice-based cryptographic schemes on the *simultaneous* average-case hardness of problems in *every* ring.

In this chapter, we present a provably-secure (in the ROM) proxy signature scheme with warrant. Our scheme is inspired by the digital signature scheme of [124], which has the benefit of basing its security on the *simultaneous* hardness of ideal lattice problems in *all* polynomial rings whose quotient polynomial's degree satisfies a certain relation. This very strong security guarantee gives us a lot of confidence on our construction because unlike the work of [183], our proposal could potentially remain unassailable even if weaknesses for a particular choice of quotient polynomial are found.

6.1.2 Organization

In Section 6.2 we recall the definitions of lattice problem variants that will serve as the foundation for our constructions, as well as the syntax and security model of proxy signature schemes. Section 6.3.1 provides a detailed description of our constructions, followed by formal proofs of security. In this chapter, we reprint the main construction in [141], of which the dissertation author was the main investigator and author.

[1] For efficiency reasons

6.2 Preliminaries

Throughout this chapter, we denote the main security parameter by n. We will denote the polynomial ring $\mathbb{Z}_q[X]$ by $\mathcal{R}_q$ and all operations will implicitly take place in it. Elements in $\mathcal{R}_q$ are of the form $\mathbf{a} = \sum_{i=0}^{k} a_i X^i$, where $a_i \in \{-\lfloor \frac{q}{2} \rfloor, \ldots, \lfloor \frac{q}{2} \rfloor\}$ and $k \in \mathbb{N}_0$. For polynomial $\mathbf{a} \in \mathcal{R}_q$ with degree $\deg(\mathbf{a})$, we will denote by $\|\mathbf{a}\|_\infty := \max_{i=0,\ldots,\deg(\mathbf{a})} |a_i|$ and $\|\mathbf{a}\|_1 := \sum_{i=0}^{\deg(\mathbf{a})} |a_i|$ the l_∞ and l_1 norms, respectively. We also define the sets $\mathcal{R}_q^{<n} := \{\mathbf{a} \in \mathcal{R}_q : \deg(\mathbf{a}) < n\}$ and $\mathcal{R}_{q,i}^{<n} := \{\mathbf{a} \in \mathcal{R}_q : \deg(\mathbf{a}) < n \wedge \|\mathbf{a}\|_\infty \leq i\}$. For a polynomial $\mathbf{a} \in \mathcal{R}_q$, and a monic polynomial $\mathbf{f}$ with $\deg(\mathbf{f}) = n$, the expression $\mathbf{a} \pmod{\mathbf{f}}$ denotes the unique polynomial $\mathbf{r} \in \mathcal{R}_q^{<n}$ for which there exists a polynomial $\mathbf{q} \in \mathcal{R}_q$ s.t.: $\mathbf{a} = \mathbf{qf} + \mathbf{r}$.

6.2.1 Lattice Problem Variants

The construction presented in this chapter will have its security based on the following variants of SVP and SIS originally defined in [124]. These variants of SVP and R – SIS are defined w.r.t. a specific irreducible polynomial $\mathbf{f}$ in the worst-case.

Definition 6.2.1. ($\mathbf{f}$–$\mathsf{SVP}_\gamma(\Lambda)$) We define the $\mathbf{f}$–$\mathsf{SVP}_\gamma(\Lambda)$ problem through the following game:

Game $\mathbf{f}$ – $\mathbf{SVP}_\gamma(\mathbf{B})$:

- **Setup.** On input a lattice $\Lambda = \Lambda(\mathbf{B})$ that corresponds to an ideal of the quotient ring $\mathbb{Z}[X]/\langle \mathbf{f} \rangle$ and an approximation factor $\gamma \in [1, \infty)$, the game invokes adversary A on input $\mathbf{B}$ and γ.
- **Output Determination.** When A outputs a vector $\hat{\mathbf{v}}$, the game returns 1 iff $\hat{\mathbf{v}} \in \Lambda$ and $\|\hat{\mathbf{v}}\|_\infty \leq \gamma \cdot \min_{\hat{\mathbf{w}} \in \Lambda - \{\mathbf{0}\}}(\|\hat{\mathbf{w}}\|_\infty)$. Otherwise, it returns 0.

Definition 6.2.2. ($\mathbf{f}$ – $\mathsf{SIS}_{k,q,\beta}$) We define the $\mathbf{f}$ – $\mathsf{SIS}_{k,q,\beta}$ problem through the following game:

Game $\mathbf{f}$ – $\mathbf{SIS}_{k,q,\beta}$:

- **Setup.** On input polynomials $\mathbf{a}_1, \ldots, \mathbf{a}_k \leftarrow_\$ \mathcal{R}_q/\langle \mathbf{f} \rangle$, the game invokes adversary A on input k, q, β, and $\mathbf{a}_1, \ldots, \mathbf{a}_k$.

- **Output Determination.** When A outputs polynomials $\mathbf{z}_1, \ldots, \mathbf{z}_k \in \mathbb{Z}[X]$, the game returns 1 iff $\|\mathbf{z}_i\|_\infty \leq \beta, \forall i \in [k]$ and $\sum_{i=1}^{k} \mathbf{a}_i \mathbf{z}_i = \mathbf{0} \pmod{\mathbf{f}}$. Otherwise, it returns 0.

The $\mathbf{f} - \mathsf{SIS}_{k,q,\beta}$ problem is on the average case, at least as hard as $\mathbf{f} - \mathsf{SVP}_\gamma$ on the worst case.

Theorem 6.2.1. *(Adapted from Theorem 5.1 in [125]) For any monic, irreducible (over the integers) polynomial $\boldsymbol{f}$ and $q > 2\theta_{\boldsymbol{f}}\beta k n^{1.5} \log n$, if there exists a PPT algorithm that solves the $\boldsymbol{f}-\mathsf{SIS}_{k,q,\beta}$ problem with noticeable probability, then there exists a PPT algorithm that solves $\boldsymbol{f} - \mathsf{SVP}_\gamma$ problem with $\gamma = \theta_{\boldsymbol{f}}\beta k n \log^2 n$ for any lattice Λ that corresponds to an ideal in $\mathbb{Z}[x]/\langle \boldsymbol{f} \rangle$.*

In [124], the following two average-case problems over the polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[x]$ are also defined. Notice that they are defined *independently* of any particular $\mathbf{f}$. As such, there is no reduction (mod $\mathbf{f}$) and we furthermore upper-bound the degrees of the solution polynomials.

Definition 6.2.3. (Homogeneous $\mathcal{R}_q^{<n} - \mathsf{SIS}_{k,d,\beta}$) We define the homogeneous $\mathbf{f} - \mathsf{SIS}_{k,q,\beta}$ problem through the following game:
Game $\mathcal{R}_q^{<n} - \mathbf{SIS}_{k,d,\beta}$:

- **Setup.** On input polynomials $\mathbf{a}_1, \ldots, \mathbf{a}_k \leftarrow_\$ \mathcal{R}_q^{<n}$, the game invokes adversary A on input k, d, β, and $\mathbf{a}_1, \ldots, \mathbf{a}_k$.

- **Output Determination.** When A outputs polynomials $\mathbf{z}_1, \ldots, \mathbf{z}_k \in \mathcal{R}_{q,\beta}^{<d}$, the game outputs 1 iff $\exists j \in [k]$ s.t. $\mathbf{z}_j \neq \mathbf{0}$ and $\sum_{i=1}^{k} \mathbf{a}_i \mathbf{z}_i = \mathbf{0}$. Otherwise, it outputs 0.

Definition 6.2.4. (Inhomogeneous $\mathcal{R}_q^{<n} - \mathsf{SIS}_{k,d_1,d_2,s,c,\beta}$) We define the inhomogeneous $\mathcal{R}_q^{<n} - \mathsf{SIS}_{k,d_1,d_2,s,c,\beta}$ problem through the following game:
Game $\mathcal{R}_q^{<n} - \mathbf{SIS}_{k,d_1,d_2,s,c,\beta}$:

- **Setup.** On input polynomials $\mathbf{a}_1, \ldots, \mathbf{a}_k \leftarrow_\$ \mathcal{R}_q^{<n}$ and $\mathbf{s}_1, \ldots, \mathbf{s}_k \leftarrow_\$ \mathcal{R}_{q,s}^{<d_1}$, the game sets $\mathbf{t} := \sum_{i=1}^{k} \mathbf{a}_i \mathbf{s}_i$ and invokes A on input $k, d_1, d_2, s, c, \beta, \mathbf{a}_1, \ldots, \mathbf{a}_k$, and $\mathbf{t}$.

- **Output Determination.** When A outputs polynomials $\mathbf{z}_1, \ldots, \mathbf{z}_k \in \mathcal{R}_{q,\beta}^{<d_2}$ and $\mathbf{c} \in \mathcal{R}^{<d_2-d_1+1}$, the game returns1 iff $0 < \|\mathbf{c}\|_1 \leq c$ and $\sum_{i=1}^{k} \mathbf{a}_i \mathbf{z}_i = \mathbf{tc}$. Otherwise, it outputs 0.

Under certain circumstances, the inhomogeneous $\mathcal{R}_q^{<n} - \mathsf{SIS}$ problem is at least as hard as the homogeneous $\mathcal{R}_q^{<n} - \mathsf{SIS}$ problem.

Lemma 6.2.2. *(Lemma 3.4 from [124]) Suppose that the following relationships are satisfied:*

1. $0 \leq d_1 < d_2 \leq n$
2. $s > 2^{\frac{\lambda}{kd_1}-1} \cdot q^{\frac{n+d_1}{kd_1}}$
3. $sc < q/4$

If there is a PPT algorithm that solves the inhomogeneous $\mathcal{R}_q^{<n} - \mathsf{SIS}_{k,d_1,d_2,s,c,\beta}$ problem in time t with probability θ, then there is s PPT algorithm that solves the homogeneous $\mathcal{R}_q^{<n} - \mathsf{SIS}_{k,d_2,\beta+sc}$ problem in time $poly(n) + t$ and with probability $\geq \frac{1}{2}(\theta - 2^{-\lambda})$.

Finally, the next lemma establishes a reduction from the $\mathbf{f} - \mathsf{SIS}_{k,q,\beta}$ problem to the homogeneous $\mathcal{R}_q^{<n} - \mathsf{SIS}_{k,d,\beta}$ problem.

Lemma 6.2.3. *(Lemma 3.2 in [124]) If there exists a PPT algorithm that solves the homogeneous $\mathcal{R}_q^{<n} - \mathsf{SIS}_{k,d,\beta}$ problem in time t and with probability θ, then there is a PPT algorithm that solves $\boldsymbol{f} - \mathsf{SIS}_{k,q,\beta}$ in time $poly(n) + t$ with probability θ as long as $d \leq \deg(\boldsymbol{f}) \leq n$.*

6.2.2 Syntax and Security Model

Proxy signature schemes were originally proposed in [128]. Their security model was later refined in [112] and formalized in [37]. According to [37], a proxy signature scheme is comprised by four algorithms (KeyGen, ProxyKeyGen, ProxySign, ProxyVerify). Their specification is the following:

Key Generation. Algorithm KeyGen inputs the security parameter n and produces the scheme's public parameters. Additionally, it outputs the public and secret keys $(\mathsf{pk}^{(i)}, \mathsf{sk}^{(i)})$ for $i \in \{\mathsf{D}, \mathsf{P}\}$ of the designator and proxy signer, respectively.

Proxy Key Generation. Algorithm ProxyKeyGen inputs the designator's keys $(\mathsf{pk}^{(\mathsf{D})}, \mathsf{sk}^{(\mathsf{D})})$ and a warrant[2] $W \in \{0,1\}^*$ and outputs a proxy key psk.

Proxy Sign. Algorithm ProxySign inputs the proxy signer's keys $(\mathsf{pk}^{(\mathsf{P})}, \mathsf{sk}^{(\mathsf{P})})$, the proxy key psk, and the message-to-be-signed $\mu \in \{0,1\}^*$ and outputs a valid proxy signature sig.

Proxy Verify. Algorithm ProxyVerify inputs a message μ, a purported signature sig, and the public keys of the designator and proxy signer. The algorithm outputs 1 if the signature is valid, and 0 otherwise.

A proxy signature is secure if it satisfies the following properties [37]:

Unforgeability: Valid proxy keys can only be created by the designator. Furthermore, valid proxy signatures can only be issued by an authorized proxy signer.

Verifiability: Given a proxy signature, it must be possible to verify that it truly originated from a proxy signer, authorized by the designator.

Strong Identifiability: Any issued proxy signature must uniquely identify the proxy signer who issued it.

Strong Undeniability: Once having issued a valid proxy signature, the proxy signer cannot deny its issuance.

Key Dependence: Proxy keys created by the designator must depend on his secret key.

6.3 A Proxy Signature from Ideal Lattices - Secure in All Rings

6.3.1 Our Construction

We go on to provide definitions for the algorithms (KeyGen, ProxyKeyGen, ProxySign, ProxyVerify) comprising our proxy signature scheme.

Key Generation. Algorithm KeyGen initializes the parameters summarized in Table 6.1, which we will consider publicly known by everyone, and which relate to the

[2]The warrant may contain legal information such as the authorization period, identifying information of the designator and proxy signer, etc.

Table 6.1: Scheme parameters for main security parameter n.

Parameter	Description	Bounds
n	main security parameter	-
q	coefficient modulus	prime
k	number of polynomials used in secret keys	positive integer
s	bound for the absolute value of secret-key polynomial coefficients	positive integer s.t.: $\|\mathbf{s}_i\|_\infty = s, \forall i$
d_1	maximum degree of secret-key polynomials + 1	$0 \leq d_1 < d_2 \leq n$
d_2	maximum degree of masking polynomials + 1	$0 \leq d_1 < d_2 \leq n$
c	Hamming weight of challenges	positive integer in $\{0, \ldots, d_2 - d_1\}$
σ	designator and proxy signer standard deviation	$11sc\sqrt{d_2 k}$
$\mathcal{R}_q := \mathbb{Z}_q[X]$	polynomial ring in which operations take place	-
$C := \{\mathbf{c} \in \mathcal{R}_{q,1}^{<d_2-d_1+1} : \|\mathbf{c}\|_1 \leq c\}$	Challenge space	-

parametrization of $\mathcal{R}_q^{<n} - \mathsf{SIS}$. We also make use of a cryptographic hash function $\mathsf{H} : \{0,1\}^* \rightarrow C$ modelled as a programmable RO.

The designator picks polynomials $\mathbf{s}_1^{(\mathsf{D})}, \ldots, \mathbf{s}_k^{(\mathsf{D})} \leftarrow_\$ \mathcal{R}_{q,s}^{<d_1}$ and $\mathbf{a}_1^{(\mathsf{D})}, \ldots, \mathbf{a}_k^{(\mathsf{D})} \leftarrow_\$ \mathcal{R}_q^{<n}$ and computes: $\mathbf{t}^{(\mathsf{D})} := \sum_{i=1}^k \mathbf{a}_i^{(D)} \mathbf{s}_i^{(\mathsf{D})}$. The designator's public key is comprised by polynomials $(\mathbf{s}_1^{(\mathsf{D})}, \ldots, \mathbf{s}_k^{(\mathsf{D})})$, whereas his public key is comprised by $(\mathbf{a}_1^{(\mathsf{D})}, \ldots, \mathbf{a}_k^{(\mathsf{D})}, \mathbf{t}^{(\mathsf{D})})$. Similarly, the proxy signer picks polynomials $\mathbf{s}_1^{(\mathsf{P})}, \ldots, \mathbf{s}_k^{(\mathsf{P})} \leftarrow_\$ \mathcal{R}_{q,s}^{<d_1}$ and $\mathbf{a}_1^{(\mathsf{P})}, \ldots, \mathbf{a}_k^{(\mathsf{P})} \leftarrow_\$ \mathcal{R}_q^{<n}$ and computes: $\mathbf{t}^{(\mathsf{P})} := \sum_{i=1}^k \mathbf{a}_i^{(\mathsf{P})} \mathbf{s}_i^{(\mathsf{P})}$. He keeps polynomials $(\mathbf{s}_1^{(\mathsf{P})}, \ldots, \mathbf{s}_k^{(\mathsf{P})})$ secret, and publishes $(\mathbf{a}_1^{(\mathsf{P})}, \ldots, \mathbf{a}_k^{(\mathsf{P})}, \mathbf{t}^{(\mathsf{P})})$.

Proxy Key Generation. To generate a proxy key for warrant $W \in \{0,1\}^*$, the designator picks masking polynomials $\mathbf{e}_1^{(\mathsf{D})}, \ldots, \mathbf{e}_k^{(\mathsf{D})} \in \mathcal{R}_q^{<d_2}$ s.t.: $\mathbf{e}_i^{(\mathsf{D})} \sim D_\sigma^{d_2}, \forall i$. He then computes a challenge $\mathbf{c}$ as a function of the "commitment" $\sum_{i=1}^k \mathbf{a}_i^{(\mathsf{D})} \mathbf{e}_i^{(\mathsf{D})}$ and the warrant W, as well as "responses" $\mathbf{z}_i^{(\mathsf{D})} := \mathbf{s}_i^{(\mathsf{D})} \mathbf{c}^{(\mathsf{D})} + \mathbf{e}_i^{(\mathsf{D})}, \forall i$. He also rejection-samples the response polynomials to make them statistically independent from his secret key. If rejection sampling succeeds and the response polynomials satisfy a certain bound, the designator sets $W_{\mathsf{D}\rightarrow\mathsf{P}} := (\mathbf{z}_1^{(\mathsf{D})}, \ldots, \mathbf{z}_k^{(\mathsf{D})}, \mathbf{c}^{(\mathsf{D})})$ as the proxy key and sends it, along with the warrant W to the intended proxy signer. The process is summarized in Algorithm 5.

Proxy Sign. The proxy signer receives $(W, W_{\mathsf{D}\rightarrow\mathsf{P}})$, parses $(\mathbf{z}_1^{(\mathsf{D})}, \ldots, \mathbf{z}_k^{(\mathsf{D})}, \mathbf{c}^{(\mathsf{D})}) := W_{\mathsf{D}\rightarrow\mathsf{P}}$ and verifies that it is a valid signature of the warrant W under the designator's public key. If this is the case, he picks masking polynomials $\mathbf{e}_1^{(\mathsf{P})}, \ldots, \mathbf{e}_k^{(\mathsf{P})} \sim D_\sigma^{d_2}, \forall i$ and computes his own challenge $\mathbf{c}^{(\mathsf{P})}$ as a function of "commitment" polynomials

Algorithm 5 ProxyKeyGen$((\mathbf{s}_1^{(\mathsf{D})}, \ldots, \mathbf{s}_k^{(\mathsf{D})}), (\mathbf{a}_1^{(\mathsf{D})}, \ldots, \mathbf{a}_k^{(\mathsf{D})}), W)$

$\mathbf{e}_1^{(\mathsf{D})}, \ldots, \mathbf{e}_k^{(\mathsf{D})} \in \mathcal{R}_q^{<d_2}$, s.t.: $\mathbf{e}_i^{(\mathsf{D})} \sim D_\sigma^{d_2}, \forall i$;
$\mathbf{c}^{(\mathsf{D})} := \mathsf{H}(\sum_{i=1}^k \mathbf{a}_i^{(\mathsf{D})} \mathbf{e}_i^{(\mathsf{D})}, W)$;
$\mathbf{z}_i^{(\mathsf{D})} := \mathbf{s}_i^{(\mathsf{D})} \mathbf{c}^{(\mathsf{D})} + \mathbf{e}_i^{(\mathsf{D})}, \forall i$;
$b :=$ Rejection_Sample$((\mathbf{z}_1^{(\mathsf{D})}, \ldots, \mathbf{z}_k^{(\mathsf{D})}), (\mathbf{s}_1^{(\mathsf{D})} \mathbf{c}^{(\mathsf{D})}, \ldots, \mathbf{s}_k^{(\mathsf{D})} \mathbf{c}^{(\mathsf{D})}), 11, sc\sqrt{d_2 k})$;
if $(b = 0$ or $\exists i \in [k] : \|\mathbf{z}_i^{(\mathsf{D})}\|_\infty > 5\sigma)$ **then**
    goto Step 1;
$W_{\mathsf{D}\to\mathsf{P}} = (\mathbf{z}_1^{(\mathsf{D})}, \ldots, \mathbf{z}_k^{(\mathsf{D})}, \mathbf{c}^{(\mathsf{D})})$;
return $(W, W_{\mathsf{D}\to\mathsf{P}})$;

$\sum_{i=1}^k \mathbf{a}_i^{(\mathsf{P})} \mathbf{e}_i^{(\mathsf{P})}$ and the message-to-be-signed μ. He then computes "response" polynomials $\mathbf{z}_i^{(\mathsf{P})} := \mathbf{s}_i^{(\mathsf{P})} \mathbf{c}^{(\mathsf{P})} + \mathbf{e}_i^{(\mathsf{P})}, \forall i \in [k]$ and rejection-samples them to ensure that his own secret key does not leak any information. If rejection sampling succeeds and the "response" polynomials satisfy a certain bound, the proxy signer outputs $(\mathbf{z}_1^{(\mathsf{D})}, \ldots, \mathbf{z}_k^{(\mathsf{D})}, \mathbf{z}_1^{(\mathsf{P})}, \ldots, \mathbf{z}_k^{(\mathsf{P})}, \mathbf{c}^{(\mathsf{D})}, \mathbf{c}^{(\mathsf{P})})$ as the proxy signature of message μ.

Algorithm 6 ProxySign$(\mu, W, W_{\mathsf{D}\to\mathsf{P}})$

Parse $(\mathbf{z}_1^{(\mathsf{D})}, \ldots, \mathbf{z}_k^{(\mathsf{D})}, \mathbf{c}^{(\mathsf{D})}) := W_{\mathsf{D}\to\mathsf{P}}$;
if $\left((\exists i \in [k] : \deg(\mathbf{z}_i^{(\mathsf{D})}) \geq d_2 \vee \|\mathbf{z}_i^{(\mathsf{D})}\|_\infty > 5\sigma) \vee (\mathbf{c}^{(\mathsf{D})} \neq \mathsf{H}(\sum_{i=1}^k \mathbf{a}_i^{(\mathsf{D})} \mathbf{z}_i^{(\mathsf{D})} - \mathbf{t}^{(\mathsf{D})} \mathbf{c}^{(\mathsf{D})}, W)\right)$ **then**
    Reject $(W, W_{\mathsf{D}\to\mathsf{P}})$;
$\mathbf{e}_1^{(\mathsf{P})}, \ldots, \mathbf{e}_k^{(\mathsf{P})} \in \mathcal{R}_q^{<d_2}$, s.t.: $\mathbf{e}_i^{(\mathsf{P})} \sim D_\sigma^{d_2}, \forall i$;
$\mathbf{c}^{(\mathsf{P})} := \mathsf{H}(\sum_{i=1}^k \mathbf{a}_i^{(\mathsf{P})} \mathbf{e}_i^{(\mathsf{P})}, \mu)$;
$\mathbf{z}_i^{(\mathsf{P})} := \mathbf{s}_i^{(\mathsf{P})} \mathbf{c}^{(\mathsf{P})} + \mathbf{e}_i^{(\mathsf{P})}, \forall i$;
$b :=$ Rejection_Sample$((\mathbf{z}_1^{(\mathsf{P})}, \ldots, \mathbf{z}_k^{(\mathsf{P})}), (\mathbf{s}_1^{(\mathsf{P})} \mathbf{c}^{(\mathsf{P})}, \ldots, \mathbf{s}_k^{(\mathsf{P})} \mathbf{c}^{(\mathsf{P})}), 11, sc\sqrt{d_2 k})$;
if $(b = 0$ or $\exists i \in [k] : \|\mathbf{z}_i^{(\mathsf{P})}\|_\infty > 5\sigma)$ **then**
    goto Step 4;
return $(\mathbf{z}_1^{(\mathsf{D})}, \ldots, \mathbf{z}_k^{(\mathsf{D})}, \mathbf{z}_1^{(\mathsf{P})}, \ldots, \mathbf{z}_k^{(\mathsf{P})}, \mathbf{c}^{(\mathsf{D})}, \mathbf{c}^{(\mathsf{P})})$;

The Rejection_Sample procedure used in Algorithms 5 and 6 is the same as the one described in Algorithm 1, with the only difference being that here we rely on the infinity norm, instead of the Euclidean norm.

Proxy Verify. A proxy signature $\left(\mathbf{z}_1^{(\mathsf{D})}, \ldots, \mathbf{z}_k^{(\mathsf{D})}, \mathbf{z}_1^{(\mathsf{P})}, \ldots, \mathbf{z}_k^{(\mathsf{P})}, \mathbf{c}^{(\mathsf{D})}, \mathbf{c}^{(\mathsf{P})}\right)$ is valid w.r.t. message μ, and warrant W, against designator public key $(\mathbf{a}_1^{(\mathsf{D})}, \ldots \mathbf{a}_k^{(\mathsf{D})}, \mathbf{t}^{(\mathsf{D})})$, and proxy signer public key $(\mathbf{a}_1^{(\mathsf{P})}, \ldots \mathbf{a}_k^{(\mathsf{P})}, \mathbf{t}^{(\mathsf{P})})$ iff all of the following conditions hold:

1. $\|\mathbf{z}_i^{(\mathsf{D})}\|_\infty \leq 5\sigma, \forall i = 1, \ldots, k$,

2. $\mathbf{c}^{(\mathsf{D})} = \mathsf{H}(\sum_{i=1}^{k} \mathbf{a}_i^{(\mathsf{D})}\mathbf{z}_i^{(\mathsf{D})} - \mathbf{t}^{(\mathsf{D})}\mathbf{c}^{(\mathsf{D})}, W)$,

3. $\|\mathbf{z}_i^{(\mathsf{P})}\|_\infty \leq 5\sigma, \forall i = 1, \ldots, k$,

4. $\mathbf{c}^{(\mathsf{P})} = \mathsf{H}(\sum_{i=1}^{k} \mathbf{a}_i^{(\mathsf{P})}\mathbf{z}_i^{(\mathsf{P})} - \mathbf{t}^{(\mathsf{P})}\mathbf{c}^{(\mathsf{P})}, \mu)$.

6.3.2 Analysis and Security

We now provide security proofs for our construction. We show that our scheme is: verifiable, strongly identifiable, strongly undeniable, unforgeable and that the proxy key depends on the designator's secret key.

Proposition 6.3.1. *(Verifiability) The proposed scheme is verifiable.*

Proof. Conditions 1 and 2 of Algorithm ProxyVerify certify that the proxy signer is warranted, by means of $(\mathbf{z}_1^{(\mathsf{D})}, \ldots, \mathbf{z}_k^{(\mathsf{D})}, \mathbf{c}^{(\mathsf{D})})$, to issue signatures on the designator's stead. □

Proposition 6.3.2. *(Strong Identifiability) The proposed scheme is strongly identifiable.*

Proof. Algorithm ProxyVerify receives the proxy signer's public key $(\mathbf{a}_1^{(\mathsf{P})}, \ldots, \mathbf{a}_k^{(\mathsf{P})}, \mathbf{t}^{(\mathsf{P})})$ as input and thus uniquely identifies a signature's issuer. □

Proposition 6.3.3. *(Strong Undeniability) The proposed scheme is strongly undeniable.*

Proof. Polynomials $(\mathbf{z}_1^{(\mathsf{P})}, \ldots, \mathbf{z}_k^{(\mathsf{P})}, \mathbf{c}^{(\mathsf{P})})$ were computed by the proxy signer during the ProxySign algorithm of Figure 6, using his secret key $(\mathbf{s}_1^{(\mathsf{P})}, \ldots, \mathbf{s}_k^{(\mathsf{P})})$. Any proxy signature produced by the proxy signer is linked to his secret key and thus, he cannot deny its issuance. □

Proposition 6.3.4. *(Key Dependence) The proxy key used by the proxy signer depends on the designator's secret key.*

Proof. The proxy key $W_{\mathsf{D}\to\mathsf{P}}$ was produced using the designator's secret key and thus depends on it. □

We establish unforgeability in two parts: unforgeability of the proxy key and unforgeability of the proxy signature. To this end, we consider the following "hybrid" signing algorithm HybridSign.

Algorithm 7 HybridSign$(\mu, (\mathbf{a}_1, \ldots, \mathbf{a}_k, \mathbf{t}))$

1: $\mathbf{c} \leftarrow_{\$} C$;
2: Select $\mathbf{z}_1, \ldots, \mathbf{z}_k \in \mathcal{R}_q^{<d_2}$ s.t. $\mathbf{z}_i \leftarrow_{\$} D_\sigma^{d_2}, \forall i$;
3: **if** $(\|\mathbf{z}_i\|_\infty > 5\sigma)$ **then**
4: Go to Step 1;
5: Program H s.t. $\mathbf{c} = \mathsf{H}(\sum_{i=1}^k \mathbf{a}_i\mathbf{z}_i - \mathbf{tc}, \mu)$;
6: **return** $(\mathbf{z}_1, \ldots, \mathbf{z}_k, \mathbf{c})$ with probability $1/M \approx 1/3$, otherwise go to Step 1;

Lemma 6.3.1. *(Lemma 4.1 in [124]) Suppose that the random oracle* H *is already programmed on* v *values. Then the statistical distance between the output of the signing procedure and the above hybrid signing algorithm, which does not take any secret keys* s_i *as inputs, is at most* $2^{-95} + v(\sqrt{2\pi}\sigma - 1)^{-d_2}$.

We now establish our scheme's unforgeability property.

Theorem 6.3.2. *(Unforgeability) Suppose that there exists a PPT adversary* A*, that successfully forges a proxy signature after a total of at most* t *queries to the* HybridSign *oracle and the random oracle* H*, with noticeable probability* θ*. Then there exists a PPT algorithm with the same complexity as* A *that solves* $\mathcal{R}_q^{<n} - \mathsf{SIS}_{k,d_1,d_2,s,2c,10\sigma}$ *or* $\mathcal{R}_q^{<n} - \mathsf{SIS}_{k,d_2,10\sigma}$ *with noticeable probability* $\geq \frac{1}{2}\left(\theta - \frac{1}{|C|}\right)\left(\frac{\theta - 1/|C|}{t} - \frac{1}{|C|}\right)$.

Proof. We first observe that any party other than the designator and a proxy signer possesses less information than the designator. As such, we can assume w.l.o.g. that the adversary attempting to forge is the designator himself. The adversary uses his secret key to compute a proxy key $W_{\mathsf{D}\to\mathsf{P}}$ for warrant W. Therefore, in order to completely forge a proxy signature $(\mathbf{z}_1^{(\mathsf{D})}, \ldots, \mathbf{z}_k^{(\mathsf{D})}, \mathbf{z}_1^{(\mathsf{P})}, \ldots, \mathbf{z}_k^{(\mathsf{P})}, \mathbf{c}^{(\mathsf{D})}, \mathbf{c}^{(\mathsf{P})})$, the adversary needs to forge the $(\mathbf{z}_1^{(\mathsf{P})}, \ldots, \mathbf{z}_k^{(\mathsf{P})}, \mathbf{c}^{(\mathsf{P})})$ part of the signature.

To prove the reduction, we construct a simulator S that uses the adversary as a black-box in order to solve the $\mathcal{R}_q^{<n} - \mathsf{SIS}_{k,d_1,d_2,s,2c,10\sigma}$ problem or the $\mathcal{R}_q^{<n} - \mathsf{SIS}_{k,d_2,10\sigma}$ problem. We observe that the input distributions of the two problems are identical. Indeed, let $(\mathbf{a}_1, \ldots, \mathbf{a}_k, \mathbf{t})$ be an instance of the $\mathcal{R}_q^{<n} - \mathsf{SIS}_{k,d_1,d_2,s,2c,10\sigma}$ problem and

$(\mathbf{a}'_1, \ldots, \mathbf{a}'_k)$ be an instance of the $\mathcal{R}_q^{<n} - \mathsf{SIS}_{k,d_2,10\sigma}$ problem. If S chooses $\mathbf{s}'_1, \ldots, \mathbf{s}'_k \leftarrow_{\$} \mathcal{R}_{q,s}^{<d_1}$ and computes $\mathbf{t}' = \sum_{i=1}^{k} \mathbf{a}'_i \mathbf{s}'_i$, then $\Delta\left((\mathbf{a}_1, \ldots, \mathbf{a}_k, \mathbf{t}); (\mathbf{a}'_1, \ldots, \mathbf{a}'_k, \mathbf{t}')\right) = 0$. Thus the two distributions are perfectly indistinguishable.

Setup. The simulator S receives the scheme parameters as input, prepares a random input instance for one of the two aforementioned problems and publishes it as its public key. We denote this public key as $(\mathbf{a}_1^{(\mathsf{S})}, \ldots, \mathbf{a}_k^{(\mathsf{S})}, \mathbf{t}^{(\mathsf{S})})$. We will show that if the adversary successfully forges on a message that was never seen by S, then the latter can solve the $\mathcal{R}_q^{<n} - \mathsf{SIS}_{k,d_1,d_2,s,2c,10\sigma}$ problem. On the contrary, if A forges on a message that was previously seen by S, then the latter can solve the $\mathcal{R}_q^{<n} - \mathsf{SIS}_{k,d_2,10\sigma}$ problem. The probability that we get a match is $1/2$. S also initializes two lists $L_{\mathsf{sig}} \leftarrow \emptyset$ and $L_{\mathsf{H}} \leftarrow \emptyset$ to be able to remain consistent in case the adversary repeats queries. We simulate the adversary's environment as follows:

Random Oracle queries. On input a tuple of the form $(\mathbf{u}, \mu)$, S checks if there exists an entry $(\mathbf{c}, \mathbf{u}, \mu) \in L_{\mathsf{H}}$ and returns $\mathbf{c}$ if that is the case. Otherwise, it picks $\mathbf{c} \leftarrow_{\$} C$, stores $(\mathbf{c}, \mathbf{u}, \mu)$ in L_{H} and returns $\mathbf{c}$ to the adversary.

Sign queries. On input message μ, S first checks if there already exists a tuple of the form $(\mu, \mathbf{z}_1^{(\mathsf{S})}, \ldots, \mathbf{z}_k^{(\mathsf{S})}, \mathbf{c}) \in L_{\mathsf{sig}}$ for the queried message. If that is the case, it returns $(\mathbf{z}_1^{(\mathsf{S})}, \ldots, \mathbf{z}_k^{(\mathsf{S})}, \mathbf{c})$ to the adversary. Otherwise, it runs $\mathsf{HybridSign}\left(\mu, (\mathbf{a}_1^{(\mathsf{S})}, \ldots, \mathbf{a}_k^{(\mathsf{S})}, \mathbf{t}^{(\mathsf{S})})\right)$ and stores the output tuple $(\mu, \mathbf{z}_1^{(\mathsf{S})}, \ldots, \mathbf{z}_k^{(\mathsf{S})}, \mathbf{c})$ in L_{sig}. It outputs $(\mathbf{z}_1^{(\mathsf{S})}, \ldots, \mathbf{z}_k^{(\mathsf{S})}, \mathbf{c})$ to the adversary. Notice that by Lemma 6.3.1, the adversary has a very small chance of distinguishing between a signer running $\mathsf{HybridSign}$ and $\mathsf{ProxySign}$. This property allows us to simulate the adversary's environment.

Forgery and Analysis. Eventually[3], A produces a forged, valid signature $(\mathbf{z}_1^{(\mathsf{S})}, \ldots, \mathbf{z}_k^{(\mathsf{S})}, \mathbf{c})$ for some message $\mu \in \{0,1\}^*$. The signature's validity implies that $\mathbf{c} = \mathsf{H}(\sum_{i=1}^{k} \mathbf{a}_i^{(\mathsf{S})} \mathbf{z}_i^{(\mathsf{S})} - \mathbf{t}^{(\mathsf{S})}\mathbf{c}, \mu)$. The probability that the adversary outputs a polynomial $\mathbf{c} \in C$ satisfying this relation *without* querying to the random oracle is only $1/|C|$. Therefore, with probability $1 - 1/|C|$, $\mathbf{c}$ was programmed into H. We further distinguish between the following cases:

[3]After polynomially upper-bounded time since A runs in PPT and all of its queries are handled efficiently.

- **Case 1:** If $\mathbf{c}$ was *directly* programmed into H by a random oracle query and A forged a signature $(\mathbf{z}_1, \ldots, \mathbf{z}_k, \mathbf{c})$ for a *new* message μ' that S never saw, then that signature satisfies the relations $\|\mathbf{z}_i\|_\infty \leq 5\sigma$ and $\mathbf{c} = \mathsf{H}(\sum_{i=1}^{k} \mathbf{a}_i^{(\mathsf{S})} \mathbf{z}_i - \mathbf{t}^{(\mathsf{S})}\mathbf{c}, \mu')$. Because A never queried H on input $(\sum_{i=1}^{k} \mathbf{a}_i^{(\mathsf{S})} \mathbf{z}_i - \mathbf{t}^{(\mathsf{S})}\mathbf{c}, \mu')$, the probability that a valid forged signature was successfully produced is $1/|C|$. We "rewind" A to the point where $\mathbf{c}$ was programmed into H, changing all responses from that point onward with new uniform responses from C. Suppose that $\mathbf{c}'$ is the *new* reply of H to the *same* query. According to the General Forking Lemma [32], the probability that A successfully forges again and that $\mathbf{c}' \neq \mathbf{c}$ is $\geq \left(\theta - \frac{1}{|C|}\right)\left(\frac{\theta - 1/|C|}{t} - \frac{1}{|C|}\right)$. Thus, S has now obtained a new relation $\mathbf{c}' = \mathsf{H}(\sum_{i=1}^{k} \mathbf{a}_i^{(\mathsf{S})} \mathbf{z}'_i - \mathbf{t}^{(\mathsf{S})}\mathbf{c}', \mu')$, where: $\sum_{i=1}^{k} \mathbf{a}_i^{(\mathsf{S})} \mathbf{z}_i - \mathbf{t}^{(\mathsf{S})}\mathbf{c} = \sum_{i=1}^{k} \mathbf{a}_i^{(\mathsf{S})} \mathbf{z}'_i - t^{(\mathsf{S})}\mathbf{c}'$. That last relation is equivalent to:

$$\sum_{i=1}^{k} \mathbf{a}_i^{(\mathsf{S})}(\mathbf{z}_i - \mathbf{z}'_i) = \mathbf{t}^{(\mathsf{S})}(\mathbf{c} - \mathbf{c}'),$$

where $0 < \|\mathbf{c} - \mathbf{c}'\|_1 \leq c + c = 2c$, $\deg(\mathbf{c} - \mathbf{c}') < d_2 - d_1 + 1$, $\|\mathbf{z}_i - \mathbf{z}'_i\|_\infty \leq 5\sigma + 5\sigma = 10\sigma$ and $\deg(\mathbf{z}_i - \mathbf{z}'_i) < d_2$. Thus, in this case, S outputs $(\mathbf{z}_1 - \mathbf{z}'_1, \ldots, \mathbf{z}_k - \mathbf{z}'_k, \mathbf{c} - \mathbf{c}')$ as a solution to the Inhomogeneous $\mathcal{R}_q^{<n} - \mathsf{SIS}_{k,d_1,d_2,s,2c,10\sigma}$ problem.

- **Case 2:** Let us now assume that $\mathbf{c}$ has been programmed into H *indirectly*, through a signing query, and thus the forged signature pertains to some message μ that S has seen. Further, let $(\mathbf{z}'_1, \ldots, \mathbf{z}'_k, \mathbf{c})$ be the signature that S outputs w.r.t. message μ. For $(\mathbf{z}'_1, \ldots, \mathbf{z}'_k, \mathbf{c})$ to be a valid signature, at least one of the $\mathbf{z}'_i$ polynomials needs to be different from the corresponding $\mathbf{z}_i$. Because $\mathbf{c}$ is the same for both signatures, we have: $\mathsf{H}(\sum_{i=1}^{k} \mathbf{a}_i^{(\mathsf{S})} \mathbf{z}_i - \mathbf{t}^{(\mathsf{S})}\mathbf{c}, \mu) = \mathsf{H}(\sum_{i=1}^{k} \mathbf{a}_i^{(\mathsf{S})} \mathbf{z}'_i - \mathbf{t}^{(\mathsf{S})}\mathbf{c}, \mu)$, which by the collision-resistance of H implies that:

$$\sum_{i=1}^{k} \mathbf{a}_i^{(\mathsf{S})} \mathbf{z}_i - \mathbf{t}^{(\mathsf{S})}\mathbf{c} = \sum_{i=1}^{k} \mathbf{a}_i^{(\mathsf{S})} \mathbf{z}'_i - \mathbf{t}^{(\mathsf{S})}\mathbf{c}$$

Therefore,

$$\sum_{i=1}^{k} \mathbf{a}_i^{(\mathsf{S})}(\mathbf{z}_i - \mathbf{z}'_i) = \mathbf{0},$$

where $\mathbf{z}_i \neq \mathbf{z}'_i$ for some index $i \in [k]$. Furthermore, $\|\mathbf{z}_i - \mathbf{z'}_i\|_\infty \leq 5\sigma + 5\sigma = 10\sigma$ and $\deg(\mathbf{z}_i - \mathbf{z}'_i) < d_2$. Thus, in this case, S outputs $(\mathbf{z}_1 - \mathbf{z}'_1, \ldots, \mathbf{z}_k - \mathbf{z}'_k)$ as a solution to the Homogeneous $\mathcal{R}_q^{<n}$ – $\mathsf{SIS}_{k,d_2,10\sigma}$ problem.

□

By combining Lemmas 6.2.2 and 6.2.3 with Theorems 6.2.1 and 6.3.2 as shown in Figure 6.1, we obtain the following corollary:

Corollary 6.3.1. *The only entity that can create proxy signatures in the proxy signature scheme of Section 6.3 is an authorized proxy signer. Any other entity that can forge proxy signatures can be twisted into an algorithm for solving the* $\boldsymbol{f}$– SVP *problem for all* $\boldsymbol{f}$ *with* $d_2 \leq \deg(\boldsymbol{f}) \leq n$.

Unforgeability of the proxy key is analogous and omitted for brevity.

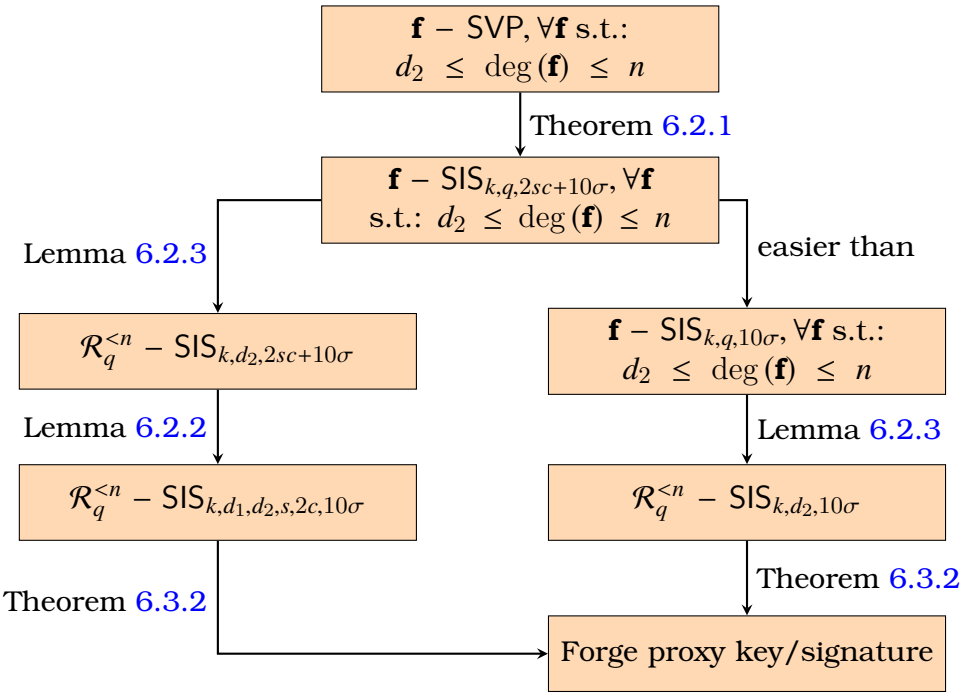

Figure 6.1: Schematic representation of the sequence of reductions culminating in our scheme's unforgeability.

6.4 Conclusions, Open Problems and Future Work

In this chapter we presented a new, provably-secure (in the ROM) proxy signature scheme from worst-case lattice assumptions. Like the proxy signature scheme of [183], breaking our scheme's unforgeability implies solving $\mathbf{f}-\mathsf{SVP}$. However, unlike [183], which requires carefully picking the quotient polynomial $\mathbf{f}$, our construction relies on the simultaneous hardness of $\mathbf{f}-\mathsf{SVP}$ on *exponentially-many* polynomials $\mathbf{f}$ with $d_2 \leq \deg(\mathbf{f}) \leq n$.

A potential future research direction would be the investigation of constructing proxy blind signature schemes [176] based on lattice assumptions, as all currently known constructions rely on classical number theoretic assumptions.

Bibliography

[1] Digital Single Market, howpublished = https://eur-lex.europa.eu/legal-content/fr/txt/html/?uri=celex:32006l0123, note = Accessed: 2021-12-02.

[2] Number of cryptocurrencies worldwide from 2013 to November 2021, howpublished = https://www.statista.com/statistics/863917/number-crypto-coins-tokens, note = Accessed: 2021-12-02.

[3] Post-Quantum Cryptography, howpublished = https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions, note = Accessed: 2021-12-02.

[4] Michel Abdalla, Chanathip Namprempre, and Gregory Neven. On the (im)possibility of blind message authentication codes. In David Pointcheval, editor, *Proc. of the 2006 The Cryptographers' Track at the RSA Conference on Topics in Cryptology*, pages 262–279, Berlin, Heidelberg, 2006. Springer-Verlag.

[5] Masayuki Abe. A secure three-move blind signature scheme for polynomially many signatures. In *Proc. of the Int. Conf. on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT '01, pages 136–151, London, UK, UK, 2001. Springer-Verlag.

[6] Masayuki Abe and Eiichiro Fujisaki. How to date blind signatures. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Proc. of the Int. Conf. on the Theory and Applications of Cryptology and Information Security: Advances in Cryptology - ASIACRYPT*, pages 244–251, Berlin, Heidelberg, 1996. Springer-Verlag.

[7] Masayuki Abe and Tatsuaki Okamoto. Provably secure partially blind signatures. In Mihir Bellare, editor, *Proc. of the 20th Annual Int. Cryptology Conference on Advances in Cryptology*, pages 271–286, London, UK, UK, 2000. Springer-Verlag.

[8] Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. Solving the shortest vector problem in 2^n time using discrete gaussian sampling: Extended abstract. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, page 733{742, New York, NY, USA, 2015. Association for Computing Machinery.

[9] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in p. *Ann. of Math*, 2:781–793, 2002.

[10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (h)ibe in the standard model. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010*, pages 553–572, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[11] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical ibe. In *Proceedings of the 30th Annual Conference on Advances in Cryptology*, CRYPTO'10, page 98{115, Berlin, Heidelberg, 2010. Springer-Verlag.

[12] Shweta Agrawal, Damien Stehle, and Anshu Yadav. Towards practical and round-optimal lattice-based threshold and blind signatures. Cryptology ePrint Archive, Report 2021/381, 2021. https://eprint.iacr.org/2021/381.

[13] Dorit Aharonov and Oded Regev. Lattice problems in np conp. *J. ACM*, 52(5):749{765, September 2005.

[14] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proc. of the 28th Annual ACM Symposium on Theory of Computing*, STOC '96, pages 99–108, New York, NY, USA, 1996. ACM.

[15] Miklós Ajtai. The shortest vector problem in l2 is np-hard for randomized reductions (extended abstract). In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, page 10{19, New York, NY, USA, 1998. Association for Computing Machinery.

[16] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, STOC '01, page 601{610, New York, NY, USA, 2001. Association for Computing Machinery.

[17] Martin Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched ntru assumptions. In *Proceedings, Part I, of the 36th Annual International Cryptology Conference on Advances in Cryptology — CRYPTO 2016 - Volume 9814*, page 153{178, Berlin, Heidelberg, 2016. Springer-Verlag.

[18] Nabil Alkeilani Alkadri, Rachid El Bansarkhani, and Johannes Buchmann. Blaze: Practical lattice-based blind signatures for privacy-preserving applications. Cryptology ePrint Archive, Report 2019/1167, 2019. https://ia.cr/2019/1167.

[19] Nabil Alkeilani Alkadri, Rachid El Bansarkhani, and Johannes Buchmann. On lattice-based interactive protocols: An approach with less or no aborts. Cryptology ePrint Archive, Report 2020/007, 2020. https://eprint.iacr.org/2020/007.

[20] Nabil Alkeilani Alkadri, Rachid El Bansarkhani, and Johannes Buchmann. On lattice-based interactive protocols with aborts. Cryptology ePrint Archive, Report 2020/007, 2020. https://eprint.iacr.org/2020/007.

[21] Nabil Alkeilani Alkadri, Rachid El Bansarkhani, and Johannes Buchmann. Blaze: Practical lattice-based blind signatures for privacy-preserving applications. In Joseph Bonneau and Nadia Heninger, editors, *FC*, pages 484–502, Cham, 2020. Springer International Publishing.

[22] Erdem Alkim, Nina Bindel, Johannes A. Buchmann, and Özgür Dagdelen. Tesla: Tightly-secure efficient signatures from standard lattices. Cryptology ePrint Archive, Report 2015/755, 2015. https://eprint.iacr.org/2015/755.

[23] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. *Theor. Comp. Sys.*, 48(3):535{553, apr 2011.

[24] Andreas Antonopoulos and Gavin Wood. *Mastering Ethereum: Building Smart Contracts and DApps*. O'Reilly Media, 1st edition, 2018.

[25] Yuriy Arbitman, Gil Dogon, Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. Swifftx: A proposal for the sha-3 standard. Available from: https://www.eecs.harvard.edu/~alon/PAPERS/lattices/swifftx.pdf, 11 2008.

[26] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer Communications Security*, CCS '13, page 1087{1098, New York, NY, USA, 2013. Association for Computing Machinery.

[27] Foteini Baldimtsi and Anna Lysyanskaya. On the security of one-witness blind signature schemes. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013*, pages 82–99, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[28] W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(4):625–636, 1993.

[29] Mihir Bellare and Shafi Goldwasser. The complexity of decision versus search. *SIAM Journal on Computing*, 23(1):97{119, February 1994.

[30] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *Proc. of the 13th ACM Conf. on Computer and Communications Security*, CCS '06, pages 390–399, New York, NY, USA, 2006. ACM.

[31] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. of the 1st ACM Conf. on Computer and Communications Security*, CCS '93, pages 62–73, New York, NY, USA, 1993. ACM.

[32] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, pages 409–426, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[33] O. Blazy, P. Gaborit, J. Schrek, and N. Sendrier. A code-based blind signature. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 2718–2722, June 2017.

[34] Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Short blind signatures. *Journal of Computer Security*, 21(5):627–661, September 2013.

[35] Olivier Blazy, David Pointcheval, and Damien Vergnaud. Compact round-optimal partially-blind signatures. In Ivan Visconti and Roberto De Prisco, editors, *Security and Cryptography for Networks*, pages 95–112, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[36] Johannes Blömer and Jean-Pierre Seifert. On the complexity of computing short linearly independent vectors and short bases in a lattice. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, STOC '99, page 711{720, New York, NY, USA, 1999. Association for Computing Machinery.

[37] Alexandra Boldyreva, Adriana Palacio, and Bogdan Warinschi. Secure proxy signature schemes for delegation of signing rights. *J. Cryptol.*, 25(1):57{115, January 2012.

[38] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In *Pro-*

ceedings of the 17th International Conference on The Theory and Application of Cryptology and Information Security, ASIACRYPT'11, page 41{69, Berlin, Heidelberg, 2011. Springer-Verlag.

[39] Dan Boneh and David Mandell Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In *Proc. of the 14th Int. Conf. on Practice and Theory in Public Key Cryptography Conf. on Public Key Cryptography*, PKC'11, pages 1–16, Berlin, Heidelberg, 2011. Springer-Verlag.

[40] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Lattice-based snargs and their application to more efficient obfuscation. Cryptology ePrint Archive, Report 2017/240, 2017. https://ia.cr/2017/240.

[41] R. B. Boppana, J. Hastad, and S. Zachos. Does co-np have short interactive proofs? *Inf. Process. Lett.*, 25(2):127{132, may 1987.

[42] Samuel Bouaziz-Ermann, Sébastien Canard, Gautier Eberhart, Guillaume Kaim, Adeline Roux-Langlois, and Jacques Traoré. Lattice-based (partially) blind signature without restart. Cryptology ePrint Archive, Report 2020/260, 2020. https://eprint.iacr.org/2020/260.

[43] Florian Bourse, Rafaël Del Pino, Michele Minelli, and Hoeteck Wee. Fhe circuit privacy almost for free. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO*, pages 62–89, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[44] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, page 575{584, New York, NY, USA, 2013. Association for Computing Machinery.

[45] Zvika Brakerski and Renen Perlman. Lattice-based fully dynamic multi-key fhe with short ciphertexts. In *Proceedings, Part I, of the 36th Annual International Cryptology Conference on Advances in Cryptology — CRYPTO 2016 - Volume 9814*, pages 190–213, Berlin, Heidelberg, 2016. Springer-Verlag.

[46] Murray R. Bremner. *Lattice Basis Reduction: An Introduction to the LLL Algorithm and Its Applications*. CRC Press, Inc., USA, 1st edition, 2011.

[47] Emmanuel Bresson, Jean Monnerat, and Damien Vergnaud. Separation results on the "one-more" computational problems. In Tal Malkin, editor, *Topics in Cryptology - CT-RSA 2008*, pages 71–87, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[48] Daniel R. L. Brown. Irreducibility to the one-more evaluation problems: More may be less. Cryptology ePrint Archive, Report 2007/435, 2007. https://eprint.iacr.org/2007/435.

[49] Jan Camenisch, Gregory Neven, and Abhi Shelat. Simulatable adaptive oblivious transfer. In Moni Naor, editor, *Proc. of the 26th Annual Int. Conf. on Advances in Cryptology*, pages 573–590, Berlin, Heidelberg, 2007. Springer-Verlag.

[50] Jan L. Camenisch, Jean-Marc Piveteau, and Markus A. Stadler. Blind signatures based on the discrete logarithm problem. In Alfredo De Santis, editor, *Advances in Cryptology — EUROCRYPT'94*, pages 428–432, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[51] Bo Carlsson. The digital economy: what is new and what is not? *Structural Change and Economic Dynamics*, 15(3):245–264, 2004. Contains the special issue New and Old Economy: The Role of ICT in Structural Change and Economic Dynamics.

[52] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010*, pages 523–552, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[53] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology*, pages 199–203, Boston, MA, 1983. Springer US.

[54] David Chaum. *Blind Signature System*, pages 153–153. Springer US, Boston, MA, 1984.

[55] David Chaum. Blinding for unanticipated signatures. In David Chaum and Wyn L. Price, editors, *Advances in Cryptology — EUROCRYPT' 87*, pages 227–233, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[56] H. Chen, P. P. Y. Lam, H. C. B. Chan, T. S. Dillon, J. Cao, and R. S. T. Lee. Business-to-consumer mobile agent-based internet commerce system (magics). *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(6):1174–1189, Nov 2007.

[57] Jung Hee Cheon, JinHyuck Jeong, and Ji Sun Shin. Cryptoanalysis on á round-optimal lattice-based blind signature scheme for cloud services. *Fut. Gener. Comp. Systems*, 95:100–103, 2019.

[58] Sherman S. M. Chow, Lucas C. K. Hui, S. M. Yiu, and K. P. Chow. Two improved partially blind signature schemes from bilinear pairings. In Colin Boyd and Juan Manuel González Nieto, editors, *Proc. of the 10th Australasian Conf. on Information Security and Privacy*, pages 316–328, Berlin, Heidelberg, 2005. Springer-Verlag.

[59] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[60] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo G. Desmedt, editor, *Proc. of the 14th Annual Int. Cryptology Conf. on Advances in Cryptology*, pages 174–187, Berlin, Heidelberg, 1994. Springer-Verlag.

[61] Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016*, pages 559–585, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[62] Ivan Damgård. Commitment schemes and zero-knowledge protocols. In Ivan Bjerre Damgård, editor, *Lectures on Data Security, Modern Cryptology in Theory and Practice, Summer School, Aarhus, Denmark, July 1998*, pages 63–86. Springer-Verlag, Berlin, Heidelberg, 1999.

[63] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theor.*, 22(6):644{654, sep 2006.

[64] Irit Dinur. Approximating svp to within almost-polynomial factors is np-hard. *Theoretical Computer Science*, 285(1):55 – 71, 2002. Algorithms and Complexity.

[65] Nico Döttling, Nils Fleischhacker, Johannes Krupp, and Dominique Schröder. Two-message, oblivious evaluation of cryptographic functionalities. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO*, pages 619–648, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[66] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013*, pages 40–56, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[67] Leo Ducas, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehle. Crystals – dilithium: Digital signatures from module lattices. Cryptology ePrint Archive, Report 2017/633, 2017. https://ia.cr/2017/633.

[68] Leo Ducas, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehle. Crystals – dilithium: Digital signatures from module lattices. Cryptology ePrint Archive, Report 2017/633, 2017. Available from: https://eprint.iacr.org/2017/633/20170627:201152.

[69] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR TCHES*, 2018(1):238–268, 2018.

[70] Nico Döttling, Nils Fleischhacker, Johannes Krupp, and Dominique Schröder. Two-message, oblivious evaluation of cryptographic functionalities. Cryptology ePrint Archive, Report 2017/958, 2017. https://ia.cr/2017/958.

[71] Edward Eaton and Fang Song. A note on the instantiability of the quantum random oracle. Cryptology ePrint Archive, Report 2019/1466, 2019. https://eprint.iacr.org/2019/1466.

[72] Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. Side-channel attacks on bliss lattice-based signatures: Exploiting branch tracing against strongswan and electromagnetic emanations in microcontrollers. In *Proc. of the 2017 ACM SIGSAC Conf. on Computer and Communications Security*, CCS '17, pages 1857–1874, New York, NY, USA, 2017. ACM.

[73] Chun-I Fan and Chin-Laung Lei. User efficient blind signatures. *Electronics Letters*, 34:544 - 546, 04 1998.

[74] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proc. on Advances in cryptology — CRYPTO '86*, pages 186–194, London, UK, UK, 1987. Springer-Verlag.

[75] Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In Cynthia Dwork, editor, *CRYPTO 2006*, pages 60–77, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[76] Marc Fischlin and Nils Fleischhacker. Limitations of the meta-reduction technique: The case of schnorr signatures. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013*, pages 444–460, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[77] Marc Fischlin and Dominique Schröder. Security of blind signatures under aborts. In Stanisław Jarecki and Gene Tsudik, editors, *Proc. of the 12th Int. Conf. on Practice and Theory in Public Key Cryptography: PKC '09*, pages 297–316, Berlin, Heidelberg, 2009. Springer-Verlag.

[78] Marc Fischlin and Dominique Schröder. On the impossibility of three-move blind signature schemes. In *Proc. of the 29th Annual Int. Conf. on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'10, pages 197–215, Berlin, Heidelberg, 2010. Springer-Verlag.

[79] John Fraleigh. *A First Course in Abstract Algebra*. Pearson, USA, seventh edition, 2002.

[80] Georg Fuchsbauer and Damien Vergnaud. Fair blind signatures without random oracles. In Daniel J. Bernstein and Tanja Lange, editors, *Proc. of the 3rd Int. Conf. on Cryptology in Africa*, pages 16–33, Berlin, Heidelberg, 2010. Springer-Verlag.

[81] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In *Eurocrypt*, EUROCRYPT'08, page 31{51, Berlin, Heidelberg, 2008. Springer-Verlag.

[82] Wen Gao, Yupu Hu, Baocang Wang, Jia Xie, and Momeng Liu. Identity-based blind signature from lattices. *Wuhan University Journal of Natural Sciences*, 22:355–360, 08 2017.

[83] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman Co., USA, 1990.

[84] Sanjam Garg, Vanishree Rao, Amit Sahai, Dominique Schröder, and Dominique Unruh. Round optimal blind signatures. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011*, pages 630–648, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[85] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. A simple bgn-type cryptosystem from lwe. In *Proceedings of the 29th Annual International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'10, page 506{522, Berlin, Heidelberg, 2010. Springer-Verlag.

[86] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proc. of the 40th Annual ACM*

Symposium on Theory of Computing, STOC '08, pages 197–206, New York, NY, USA, 2008. ACM.

[87] Essam Ghadafi. Efficient round-optimal blind signatures in the standard model. In Aggelos Kiayias, editor, *Financial Cryptography and Data Security*, pages 455–473, Cham, 2017. Springer International Publishing.

[88] John T. Gill. Computational complexity of probabilistic turing machines. In *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, STOC '74, page 91{95, New York, NY, USA, 1974. Association for Computing Machinery.

[89] Oded Goldreich. *Foundations of Cryptography: Volume 1*. Cambridge University Press, USA, 2006.

[90] Oded Goldreich. *P, NP, and NP-Completeness: The Basics of Computational Complexity*. Cambridge University Press, USA, 1st edition, 2010.

[91] Oded Goldreich and Shafi Goldwasser. On the limits of nonapproximability of lattice problems. *J. Comput. Syst. Sci.*, 60(3):540{563, June 2000.

[92] S Goldwasser and M Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, page 59{68, New York, NY, USA, 1986. Association for Computing Machinery.

[93] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281{308, April 1988.

[94] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from lwe. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015*, pages 503–523, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[95] Leon Groot Bruinderink, Andreas Hülsing, Tanja Lange, and Yuval Yarom. Flush, gauss, and reload – a cache attack on the bliss lattice-based signature scheme. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *IACR-CHES*, pages 323–345, Berlin, Heidelberg, 2016. Springer-Verlag.

[96] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In *Cryptographic Hardware and Embedded Systems - CHES 2012*, CHES'12, pages 530–547, Berlin, Heidelberg, 2012. Springer-Verlag.

[97] Eduard Hauck, Eike Kiltz, and Julian Loss. A modular treatment of blind signatures from identification schemes. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019*, pages 345–375, Cham, 2019. Springer International Publishing.

[98] Eduard Hauck, Eike Kiltz, Julian Loss, and Ngoc Khanh Nguyen. Lattice-based blind signatures, revisited. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO*, pages 500–529, Cham, 2020. Springer International Publishing.

[99] Ishay Haviv and Oded Regev. Tensor-based hardness of the shortest vector problem to within almost polynomial factors. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07, page 469{477, New York, NY, USA, 2007. Association for Computing Machinery.

[100] Ethan Heilman, Foteini Baldimtsi, and Sharon Goldberg. Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions. Cryptology ePrint Archive, Report 2016/056, 2016. https://ia.cr/2016/056.

[101] R. Impagliazzo. A personal view of average-case complexity. In *Proceedings of Structure in Complexity Theory. Tenth Annual IEEE Conference*, pages 134–147, 1995.

[102] M. A. Jabri and S. Matsuoka. Dealing with grid-computing authorization using identity-based certificateless proxy signature. In *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 544–553, May 2011.

[103] Y. Jiang, F. Kong, and X. Ju. Lattice-based proxy signature. In *2010 International Conference on Computational Intelligence and Security*, pages 382–385, Dec 2010.

[104] Ari Juels, Michael Luby, and Rafail Ostrovsky. Security of blind digital signatures. In Burton S. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, pages 150–164, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[105] David Kahn. *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet*. Scribner, USA, 1996.

[106] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. Chapman Hall/CRC, 3rd edition, 2020.

[107] Jonathan Katz, Julian Loss, and Michael Rosenberg. Boosting the security of blind signature schemes. Cryptology ePrint Archive, Report 2021/806, 2021. https://ia.cr/2021/806.

[108] Jonathan Katz, Dominique Schröder, and Arkady Yerukhimovich. Impossibility of blind signatures from one-way permutations. In Yuval Ishai, editor, *Theory of Cryptography*, pages 615–629, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[109] Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In Mitsuru Matsui, editor, *Advances in Cryptology {- ASIACRYPT 2009*, pages 703–720, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[110] Subhash Khot. Hardness of approximating the shortest vector problem in lattices. *J. ACM*, 52(5):789{808, sep 2005.

[111] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of fiat-shamir signatures in the quantum random-oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology {- EUROCRYPT 2018*, pages 552–586, Cham, 2018. Springer International Publishing.

[112] Seungjoo Kim, Sangjoon Park, and Dongho Won. Proxy signatures, revisited. In *Proceedings of the First International Conference on Information and Communication Security*, ICICS '97, pages 223–232, Berlin, Heidelberg, 1997. Springer-Verlag.

[113] Ilan Komargodski. Leakage resilient one-way functions: The auxiliary-input setting. In *Proc., Part I, of the 14th Int. Conf. on Theory of Cryptography - Volume 9985*, pages 139–158, New York, NY, USA, 2016. Springer-Verlag New York, Inc.

[114] Mahender Kumar, C. P. Katti, and P. C. Saxena. A secure anonymous e-voting system using identity-based blind signature scheme. In Rudrapatna K. Shyamasundar, Virendra Singh, and Jaideep Vaidya, editors, *Information Systems Security*, pages 29–49, Cham, 2017. Springer International Publishing.

[115] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptography*, 75(3):565–599, June 2015.

[116] Huy Quoc Le, Willy Susilo, Thanh Xuan Khuc, Minh Kim Bui, and Dung Hoang Duong. A blind signature from module latices. In *IEEE Conference on Dependable and Secure Computing (DSC)*, pages 1–8, 2019.

[117] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, Dec 1982.

[118] Fagen Li, Mingwu Zhang, and Tsuyoshi Takagi. Identity-based partially blind signature in the standard model for electronic cash. *Mathematical and Computer Modelling*, 58(1):196 – 203, 2013. Financial IT & Security and 2010 International Symposium on Computational Electronics.

[119] Chen Liang, Cui Yongquan, Tang Xueming, Hu Dongping, and Wan Xin. Hierarchical id-based blind signature from lattices. In *2011 Seventh International Conference on Computational Intelligence and Security*, pages 803–807, 2011.

[120] Yehuda Lindell. Bounded-concurrent secure two-party computation without setup assumptions. In *Proc. of the 35th Annual ACM Symposium on Theory of Computing*, STOC '03, pages 683–692, New York, NY, USA, 2003. ACM.

[121] Vadim Lyubashevsky. Lattice-based identification schemes secure under active attacks. In *Proc. of the Practice and Theory in Public Key Cryptography, 11th Int. Conf. on Public Key Cryptography*, PKC'08, pages 162–179, Berlin, Heidelberg, 2008. Springer-Verlag.

[122] Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *Proc. of the 15th Int. Conf. on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, pages 598–616, Berlin, Heidelberg, 2009. Springer-Verlag.

[123] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *Proc. of the 31st Annual Int. Conf. on Theory and Applications of Cryptographic Techniques*, pages 738–755, Berlin, Heidelberg, 2012. Springer-Verlag.

[124] Vadim Lyubashevsky. Digital signatures based on the hardness of ideal lattice problems in all rings. In *Proc., Part II, of the 22nd Int. Conf. on Advances in Cryptology — ASIACRYPT 2016 - Volume 10032*, pages 196–214, New York, NY, USA, 2016. Springer-Verlag New York, Inc.

[125] Vadim Lyubashevsky and Daniele Micciancio. Generalized compact knapsacks are collision resistant. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Proc. of the 33rd Int. Conf. on Automata, Languages and Programming - Volume Part II*, pages 144–155, Berlin, Heidelberg, 2006. Springer-Verlag.

[126] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6), nov 2013.

[127] Vadim Lyubashevsky and Gregor Seiler. Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT*, pages 204–224, Cham, 2018. Springer International Publishing.

[128] Masahiro Mambo, Keisuke Usuda, and Eiji Okamoto. Proxy signatures for delegating signing operation. In *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, CCS '96, pages 48–57, New York, NY, USA, 1996. Association for Computing Machinery.

[129] Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *Advances in Cryptology — CRYPTO '87*, pages 369–378, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[130] D. Micciancio and O. Regev. Worst-case to average-case reductions based on gaussian measures. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 372–381, 2004.

[131] Daniele Micciancio. The shortest vector in a lattice is hard to approximate to within some constant. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, FOCS '98, page 92, USA, 1998. IEEE Computer Society.

[132] Daniele Micciancio. Foundations of security analysis and design vi. chapter The Geometry of Lattice Cryptography, pages 185–210. Springer-Verlag, Berlin, Heidelberg, 2011.

[133] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012*, pages 700–718, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[134] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM Journal on Computing*, 37(1):267–302, April 2007.

[135] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum Cryptography*, pages 147–191. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[136] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, STOC '10, page 351{358, New York, NY, USA, 2010. Association for Computing Machinery.

[137] Kaisa Nyberg and Rainer A. Rueppel. A new signature scheme based on the dsa giving message recovery. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, page 58{61, New York, NY, USA, 1993. Association for Computing Machinery.

[138] Kazuo Ohta and Tatsuaki Okamoto. On concrete security treatment of signatures derived from identification. In Hugo Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, pages 354–369, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[139] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *CRYPTO*, pages 31–53, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[140] Tatsuaki Okamoto. Efficient blind and partially blind signatures without random oracles. In Shai Halevi and Tal Rabin, editors, *Proceedings of the 3rd Conference on Theory of Cryptography*, pages 80–99, Berlin, Heidelberg, 2006. Springer-Verlag.

[141] D. G. Papachristoudis, S. T. Halkidis, and G. Stephanides. An experimental comparison of some lll–type lattice basis reduction algorithms. *International Journal of Applied and Computational Mathematics*, 1:327–342, January 2015.

[142] Dimitrios Papachristoudis. A survey on lattice-based blind signatures and their feasibility. *Archives of Economic History*, XXXIII, No. 2, 2021.

[143] Dimitrios Papachristoudis, Dimitrios Hristu-Varsakelis, Foteini Baldimtsi, and George Stephanides. Leakage-resilient lattice-based partially blind signatures. *IET Information Security*, 13:670–684(14), Nov 2019.

[144] Dimitrios Papachristoudis, Julian Loss, Foteini Baldimtsi, and George Stephanides. A framework for blind signatures with revocable sessions, 2021. (to be submitted to CRYPTO 2022).

[145] Dimitrios Papachristoudis and George Stephanides. Proxy signatures from ideal lattices - secure in all rings. In *Proceedings of the 35th Panhellenic Conference on Mathematical Education "Mathematics: Research and Education in the 21st century"*, pages 853–868, Athens, Greece, 2018. Panhellenic Mathematical Society.

[146] Christian Paquin and Greg Zaverucha. U-prove cryptographic specification v1.1 (revision 3). Technical report, Microsoft Corporation, December 2013.

[147] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: Extended abstract. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, pages 333–342, New York, NY, USA, 2009. Association for Computing Machinery.

[148] Chris Peikert. A decade of lattice cryptography. *Found. Trends Theor. Comput. Sci.*, 10(4):283–424, March 2016.

[149] Peter Pessl, Leon Groot Bruinderink, and Yuval Yarom. To bliss-b or not to be: Attacking strongswan's implementation of post-quantum signatures. In *Proc. of*

the 2017 ACM SIGSAC Conf. on Computer and Communications Security, CCS '17, pages 1843–1855, New York, NY, USA, 2017. ACM.

[150] Valentin Petrov. *Sums of Independent Random Variables*. Springer-Verlag Berlin Heidelberg, 1st edition, 1975.

[151] Albrecht Petzoldt, Alan Szepieniec, and Mohamed Saied Emam Mohamed. A practical multivariate blind signature scheme. In Aggelos Kiayias, editor, *Financial Cryptography and Data Security*, pages 437–454, Cham, 2017. Springer International Publishing.

[152] David Pointcheval. Strengthened security for blind signatures. In Kaisa Nyberg, editor, *EUROCRYPT*, pages 391–405, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

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

[154] David Pointcheval and Jacques Stern. New blind signatures equivalent to factorization (extended abstract). In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, CCS '97, page 92{99, New York, NY, USA, 1997. Association for Computing Machinery.

[155] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, Jun 2000.

[156] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05, page 84{93, New York, NY, USA, 2005. Association for Computing Machinery.

[157] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120{126, feb 1978.

[158] F. Rodríguez-Henríquez, Daniel Ortiz-Arroyo, and Claudia García-Zamora. Yet another improvement over the mu{varadharajan e-voting protocol. *Computer Standards Interfaces*, 29:471–480, 05 2007.

[159] Markus Rückert. Lattice-based blind signatures. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, pages 413–430, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[160] Markus Rückert. *Lattice-based Signature Schemes with Additional Features*. PhD thesis, Technische Universität, Darmstadt, January 2011.

[161] Markus Rückert and Dominique Schröder. Fair partially blind signatures. In Daniel J. Bernstein and Tanja Lange, editors, *Proc. of the 3rd Int. Conf. on Cryptology in Africa*, pages 34–51, Berlin, Heidelberg, 2010. Springer-Verlag.

[162] Markus Rückert and Michael Schneider. Estimating the security of lattice-based cryptosystems. Cryptology ePrint Archive, Report 2010/137, 2010. eprint.iacr.org/2010/137.

[163] C. P. Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 239–252, New York, NY, 1990. Springer New York.

[164] Claus-Peter Schnorr. Security of blind discrete log signatures against interactive attacks. In *Proceedings of the Third International Conference on Information and Communications Security*, ICICS '01, pages 1–12, Berlin, Heidelberg, 2001. Springer-Verlag.

[165] C.P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53(2):201–224, jun 1987.

[166] Berry Schoenmakers. Lecture notes: Cryptographic protocols (version: 1.4). Available from: https://www.win.tue.nl/~berry/CryptographicProtocols/LectureNotes.pdf, 2 2019.

[167] Dominique Schröder and Dominique Unruh. Security of blind signatures revisited. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *Public Key Cryptography - PKC 2012*, pages 662–679, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[168] Dominique Schröder and Dominique Unruh. Round optimal blind signatures. Cryptology ePrint Archive, Report 2011/264, 2011. https://ia.cr/2011/264.

[169] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing (SICOMP)*, 26(5):1484–1509, October 1997.

[170] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. https://eprint.iacr.org/2004/332.

[171] Victor Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, USA, 2 edition, 2009.

[172] Jimmy Song. *Programming Bitcoin: Learn How to Program Bitcoin from Scratch*. O'Reilly Media, Inc., 1st edition, 2019.

[173] Markus Stadler, Jean-Marc Piveteau, and Jan Camenisch. Fair blind signatures. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology — EUROCRYPT '95*, pages 209–219, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[174] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009*, pages 617–635, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[175] Z. Tan and H. Xiao. Hierarchical proxy blind signature: A solution to e-cash in the real world. In *2008 The 9th International Conference for Young Computer Scientists*, pages 1476–1480, Nov 2008.

[176] Zuowen Tan, Zhuojun Liu, and Chunming Tang. Proxy blind signature scheme based on dlp. *Journal of Software*, 14, 11 2003.

[177] Haibo Tian, Fangguo Zhang, and Baodian Wei. A lattice-based partially blind signature. *Security and Communication Networks*, 9(12):1820–1828, August 2016.

[178] Miaomiao Tian and Liusheng Huang. Breaking A proxy signature scheme from lattices. *I. J. Network Security*, 14(6):320–323, 2012.

[179] Peter van Emde Boas. Another np-complete partition problem and the complexity of computing short vectors in lattices. 1980.

[180] J Von von Neumann, A. W. Taub, and A. H. Taub. Various techniques used in connection with random digits, notes by g e forsythe. *National Bureau of Standards Applied Math Series*, pages 36–38, 12 1951. Reprinted in von Neumann's Collected Works, 5 (1963), Pergamon Press, pp 768-770.

[181] Sebastiaan Von Solms and David Naccache. On blind signatures and perfect crimes. *Computer Security*, 11(6):581–583, October 1992.

[182] P. Weill, S.S. Management, and S.S.M.C. for. *Information Technology Infrastructure for E-Business*. Creative Media Partners, LLC, 2018.

[183] C. Yang, P. Qiu, S. Zheng, and L. Wang. An efficient lattice-based proxy signature scheme without trapdoor. In *2015 International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*, pages 189–194, Sep. 2015.

[184] Xun Yi and Kwok-Yan Lam. A new blind ecdsa scheme for bitcoin transaction anonymity. In *Asia-CCS*, Asia CCS '19, pages 613–620, New York, NY, USA, 2019. Association for Computing Machinery.

[185] Lili Zhang and Yanqin Ma. A lattice-based identity-based proxy blind signature scheme in the standard model. *Mathematical Problems in Engineering*, 2014, 09 2014.

[186] R. Zhang, Y. Zhang, and K. Ren. Distributed privacy-preserving access control in sensor networks. *IEEE Trans. on Par. and Distr. Systems*, 23(8):1427–1438, Aug 2012.

[187] Yanhua Zhang. Forward-secure identity-based shorter blind signature from lattices. *American Journal of Networks and Communications*, 5:17, 01 2016.

[188] Hongfei Zhu, Yu-an Tan, Xiaosong Zhang, Liehuang Zhu, Changyou Zhang, and Jun Zheng. A round-optimal lattice-based blind signature scheme for cloud services. *Future Generation Computer Systems*, 73(C):106–114, August 2017.