COLLEGE MATHS, TELECOMS
DOCTORAL INFORMATIQUE, SIGNAL
BRETAGNE SYSTEMES, ELECTRONIQUE

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES

ÉCOLE DOCTORALE N° 601
Mathématiques, Télécommunications, Informatique, Signal, Systèmes, Électronique
Spécialité : *Informatique*

Par
Guilhem NIOT

Achieving Practical Post-Quantum Threshold Signatures from Lattices

Constructions pratiques de signature à seuil post-quantique à partir de réseaux euclidiens

Thèse présentée et soutenue à Paris, le 19 Mai 2026
Unité de recherche : IRISA

Rapporteur·trice·s avant soutenance :

Damien VERGNAUD	Professeur, Sorbonne Université
Vadim LYUBASHEVSKY	Researcher, IBM Research Zurich

Composition du Jury :

Examinateur·trice·s :	Dennis HOFHEINZ	Professor, ETH Zurich
	Phong Q. NGUYEN	Directeur de recherche, ENS PSL
	Katharina BOUDGOUST	Chargée de recherche, CNRS Montpellier
	Russell W. F. LAI	Assistant Professor, Aalto University
Dir. de thèse :	Pierre-Alain FOUQUE	Professeur, Université de Rennes
Co-dir. de thèse :	Thomas PREST	Chercheur, PQShield

Université de Rennes

PQ SHIELD

ACKNOWLEDGEMENTS

As this thesis comes to an end, I would like to thank everyone who has supported me along the way.

First and foremost, I would like to thank my PQShield advisor, Thomas Prest. You have been an incredible mentor, always available for a chat or a deep discussion, and your knowledge and insights have been invaluable. I especially appreciated the times you came to me with new ideas or challenges, pushing me to think critically and creatively, which greatly enriched my research experience. Your guidance has been central to my growth as a researcher. I am also grateful for the research environment you created at PQShield, which enabled many fruitful collaborations.

I would also like to thank my academic advisor, Pierre-Alain Fouque, for his guidance and support throughout this work. Although we did not have the opportunity to meet very frequently due to the industrial setting of this thesis, I greatly appreciated the discussions we had during our meetings.

I would like to express my gratitude to the members of my jury for agreeing to evaluate this work. In particular, I thank Damien Vergnaud and Vadim Lyubashevsky for agreeing to review the manuscript, and Dennis Hofheinz, Phong Q. Nguyen, Katharina Boudgoust, and Russell W. F. Lai for agreeing to serve as examiners.

Next, I would like to extend my thanks to my colleagues at PQShield. Working alongside such talented and passionate individuals has been a privilege. I am particularly grateful to my colleagues from the research team, including Thomas Espitau, Rafael del Pino, Pierre-Yves Strub, Guilhem Mizrahi and Bachir Lachgel in Paris, as well as colleagues around the world: Shuichi Katsumata, Kaoru Takemure, Thom Wiggers, Ida Tucker, Alexandre Wallet, Gustavo Delerue, Benedikt Auerbach, and Wessel van Woerden. I feel fortunate to have had the chance to collaborate with you all, and I learned a lot from our interactions. We shared many great moments, both in and out of work, and I hope to have many more in the future. I am especially thankful to Thomas Espitau and Shuichi Katsumata, who went above and beyond to support me during these years. Thomas, you were the first to mentor me at PQShield, and you introduced me to lattice-based cryptography. Beyond research, you were always happy to answer my random texts about life or career advice, for which I am very grateful. Shuichi, it is a pleasure to work with you. You often took the time to explain complex topics to me and introduce me to new research areas. I have always admired your capacity to juggle so many different projects while maintaining such a high level of quality in your work, and I hope to one day reach your level! Thank you both also for the opportunity of visiting Japan several times, which was an amazing experience.

Also, I would like to thank all the people I had the pleasure to collaborate with during this thesis who I did not mention yet. This includes Michael Reichle, Sofía Celi, Giacomo Borin, Sebastian Faller, Keitaro Hashimoto, Daniel Collins, Loïs Huguenin-Dumittan, Kévin Duverger, Charlie Jacomme, Cristina Onete, Daniel Escudero, Muhammed F. Esgin, Amin Sakzad, Ron Steinfeld, Mehdi Tibouchi. In particular, we started long and fruitful collaborations with Michael Reichle, Sofía Celi and Giacomo Borin, which I hope will continue.

Further, I would like to thank Damien Stehlé for introducing me to cryptography during my studies at ENS Lyon, and for your advice throughout these years. You inspired me to pursue research in this field, and introduced me to the wonderful people at PQShield and at NTT who made this thesis possible. I would also like to thank Serge Vaudenay for your insightful courses on cryptography at EPFL, which deepened my understanding of the field.

I am also grateful to the colleagues and institutions who invited me to present my work during these years. This includes JP Morgan, Télécom Paris, Berkeley, SNU and LIP6 Almasty. It was a

pleasure to meet you all, and those presentations were very helpful for disseminating my research and for getting valuable feedback. I am especially thankful to Yongsoo Song for making my visit to Korea possible, and for the great hospitality I received during my stay. I also wish to deeply thank Tancrède Lepoint for welcoming me at AWS over an entire summer, and for introducing me to the world of industry research.

Finally, I would like to thank my family and friends for their constant support and encouragement throughout these years. All the times we spent together were the necessary balance to my research life, and I am grateful for your presence in my life.

CONTENTS

NOTATIONS AND ABBREVIATIONS

The following tables collect the main notations and abbreviations used throughout this thesis. Notations restricted to a single chapter are indicated explicitly.

GENERAL MATHEMATICAL NOTATIONS

$a := b$	a is defined to be equal to b.
$\mathbb{R}$, $\mathbb{Q}$, $\mathbb{Z}$, $\mathbb{N}$	Sets of real numbers, rational numbers, integers, and non-negative integers.
$\mathbb{Z}_q$	Ring of integers modulo q.
$y \leftarrow f(x)$	y is the output of the (possibly probabilistic) function f on input x.
$y \leftarrow \mathcal{D}$	y is a sample drawn from distribution $\mathcal{D}$.
$x \xleftarrow{\$} S$	x is sampled uniformly at random from the set S.
$\mathcal{U}(S)$	Uniform distribution over the set S.
$[N]$	The set $\{1, 2, \ldots, N\}$.
$\Pr[\cdot]$	Probability.
$\mathbb{E}[\cdot]$	Expectation.
$\lfloor x \rceil$	Rounding of x to the nearest integer (half-up convention).
$\lfloor x \rceil_\nu$	Shift-rounding: $\lfloor x/2^\nu \rceil$, dropping the ν least significant bits of x.
$\mathsf{lift}(x)$	Canonical representative of $x \in \mathbb{Z}_q$ in the interval $(-q/2, q/2]$.
$\perp$	Error symbol / null value.

VECTORS, MATRICES AND NORMS

$\mathbf{x}$, $\mathbf{y}$, ...	Vectors in bold lowercase (e.g. $\mathbf{x} \in \mathbb{R}^k$ or $\mathbf{x} \in R^k$).
$\mathbf{A}$, $\mathbf{B}$, ...	Matrices in bold uppercase (e.g. $\mathbf{A} \in \mathbb{R}^{k \times \ell}$ or $\mathbf{A} \in R^{k \times \ell}$).
$\mathbf{A}^\top$, $\mathbf{x}^\top$	Transpose of a matrix or vector.
$\lVert \mathbf{x} \rVert_2$	Euclidean (ℓ_2) norm of vector $\mathbf{x}$. Apply lift to each coefficient if $\mathbf{x}$ is over $\mathbb{Z}_q$ or R_q.
$\lVert \mathbf{x} \rVert_\infty$	Infinity (ℓ_∞) norm of vector $\mathbf{x}$.
$\lVert \mathbf{A} \rVert_2$	Spectral norm of matrix $\mathbf{A}$ (largest singular value).
$\langle \mathbf{x}, \mathbf{y} \rangle$	Inner product of $\mathbf{x}$ and $\mathbf{y}$.
$\mathbf{I}_k$	Identity matrix of size k (written $\mathbf{I}$ when size is clear from context).

RINGS AND MODULE LATTICES

R	Cyclotomic ring $\mathbb{Z}[X]/(X^n+1)$, where n is a power of two.
R_q	Quotient ring R/qR.
$\eta'_\varepsilon(\Lambda)$	Scaled smoothing parameter $\frac{1}{\sqrt{2\pi}}\eta_\varepsilon(\Lambda)$ (factor-$\frac{1}{2}$ convention).
$D_{\mathbb{Z}^n,\sigma,\mathbf{c}}$	Discrete Gaussian over $\mathbb{Z}^n$ with parameter σ and center $\mathbf{c}$.

CRYPTOGRAPHIC SECURITY

λ	Security parameter.
$\mathsf{negl}(\lambda)$	Negligible function in λ.
$\mathsf{Time}(\mathcal{A})$	Running time of algorithm $\mathcal{A}$.
$\mathsf{Adv}^{X}_{\mathcal{A}}(1^\lambda)$	Advantage of adversary $\mathcal{A}$ against security notion X.
$\mathcal{A}^{\mathcal{O}}$	Adversary $\mathcal{A}$ with oracle access to $\mathcal{O}$.
$\mathsf{Game}^{X}_{\mathcal{A}}$	Security game for notion X played against adversary $\mathcal{A}$.

HARDNESS ASSUMPTIONS

MSIS	Module Short Integer Solution problem.
SelfTargetMSIS	Variant of MSIS used in Fiat-Shamir proofs.
MLWE	Module Learning With Errors problem.
Hint-MLWE	MLWE with additional linear hints.

THRESHOLD SIGNATURES

(N, T, t)	N total parties, signing threshold T, corruption threshold $t < T$.
sk, vk	Secret (signing) key and verification key.
sk_i	Private key share of party i in a threshold scheme.
CS, HS	Sets of corrupted and honest parties; $\mathsf{HS} = [N]\backslash\mathsf{CS}$, $\lvert\mathsf{CS}\rvert \leqslant t$.
act	Active signing set, $\mathsf{act} \subseteq [N]$, $\lvert\mathsf{act}\rvert = T$.
msg	Message (in $\{0,1\}^*$).
sig	Signature.
pm^i_r	Message broadcast by party i in round r ($\perp$ if aborted).
st_i	Internal state of party i.
sid	Session identifier (used in the synchronous model).

aux	Auxiliary public information produced during key generation.
SYNC-TS-UF	Unforgeability in the static, synchronous model.
ASYNC-TS-UF	Unforgeability in the static, asynchronous model.
TS-CORR	Threshold Signature Correctness game.

NAMED SCHEMES AND PROTOCOLS

ECDSA	Elliptic Curve Digital Signature Algorithm.
FROST	Flexible Round-Optimized Schnorr Threshold signatures.
Raccoon	Fiat-Shamir masking-friendly lattice signature scheme.
TRaccoon	Threshold variant of Raccoon.
RB-Raccoon	Robust threshold variant (Ch. 4).
IA-Raccoon	Threshold variant with identifiable aborts (Ch. 5).
ML-DSA	NIST post-quantum digital signature standard.
Threshold ML-DSA	Threshold ML-DSA (Ch. 6).
Plover	Hash-and-sign masking-friendly lattice signature.

CHAPTER 4 (ROBUST THRESHOLD SIGNATURES) — SPECIFIC NOTATIONS

$L_{\mathsf{act},i}$	Lagrange reconstruction coefficient of party i for set act.
$[\![\mathbf{x}]\!]$	Masked or secretly-shared value $\mathbf{x}$.
TS-RB	Robustness security notion for threshold signatures.
SSS.Share, SSS.Decode	Shamir sharing algorithms: Share and Decode.
SSS.Decode, SSS.ErrorCorrect	Shamir sharing algorithms: Decode and ErrorCorrect (for robust reconstruction).
V3S	The specific VSS scheme used in Ch. 4 for short secrets.

CHAPTER 5 (IDENTIFIABLE ABORTS) — SPECIFIC NOTATIONS

TS-ID	Threshold signature with identifiable abort.
IdentifyAbort	Algorithm for identifying the party that caused a signing abort.

CHAPTER 6 (THRESHOLD ML-DSA) — SPECIFIC NOTATIONS

HighBits, LowBits	ML-DSA decomposition functions for a ring element.
MakeHint, UseHint	Hint vector computation and application functions.
Power2Round	Power2Round function (bit-dropping rounding in ML-DSA).
TS-CORR-ABORT	Correctness-with-abort security notion for threshold signatures.
acc, rej	Accept / reject outcomes in the rejection sampling.

GENERAL ABBREVIATIONS

BKZ	Block Korkine-Zolotarev: a lattice basis reduction algorithm.
DKG	Distributed Key Generation.
DL	Discrete Logarithm.
HE	Homomorphic Encryption.
IA	Identifiable Abort.
KEM	Key Encapsulation Mechanism.
LSSS	Linear Secret Sharing Scheme (general class of secret sharing schemes).
MPC	Multi-Party Computation.
NIST	National Institute of Standards and Technology.
NIZK	Non-Interactive Zero-Knowledge proof.
NTT	Number Theoretic Transform.
PKE	Public Key Encryption.
PPT	Probabilistic Polynomial-Time algorithm.
PRF	Pseudorandom Function.
RO / ROM	Random Oracle / Random Oracle Model.
SKE	Symmetric Key Encryption.
SSS	Shamir's Secret Sharing.
TSS	Threshold Signature Scheme.
UC	Universal Composability framework.
VSS	Verifiable Secret Sharing.

1 INTRODUCTION

1.1 CRYPTOLOGY

Cryptology is the science dedicated to securing communication in the presence of adversarial behavior. It is composed of two interconnected branches: cryptography, which focuses on hiding information and protecting communication, and cryptanalysis, which aims to break or compromise the protection granted by cryptography. While the need for secure communication is as old as communication itself, the formal study of cryptology emerged in the 20th century, building on centuries of earlier cipher design and accelerated by the advent of mechanical and electronic communication technologies.

Although early mechanical ciphers such as Enigma already embodied the essence of modern cryptosystems through algorithmic encryption and key-based security, the modern era of cryptology truly began with Claude Shannon's information-theoretic framework for secrecy [Sha49] and the subsequent development of computational security models in the 1970s. The introduction of public-key cryptography by Diffie and Hellman [DH76] fundamentally reshaped the field, allowing secure communication over open networks without pre-shared secrets.

Until the work of Diffie and Hellman, cryptography was primarily concerned with symmetric-key schemes, where both the sender and receiver share a common secret key. This paradigm, however, is inherently limited in scalability: in a large network, every pair of participants must securely exchange and manage distinct shared keys. Historically, such exchanges required physical meetings or trusted couriers, making cryptography impractical for widespread use. Public-key cryptography introduced asymmetric schemes, where users possess a public key (shared openly) and a private key (kept secret). This breakthrough enabled secure communication with a large number of participants as one can simply communicate securely using the recipient's public key – typically distributed via digital infrastructures – without prior *physical* key exchange. Two main primitives were introduced: public-key encryption, allowing anyone to send encrypted messages to a recipient using their public key, and digital signatures, enabling users to sign messages in a way that anyone can verify the signature using the signer's public key, but only the signer can produce valid signatures.

The next year, Rivest, Shamir, and Adleman [RSA78] introduced the RSA algorithm, which became the first widely adopted public-key cryptosystem, enabling one to both encrypt messages and create digital signatures.

Since then, cryptology has evolved alongside technology, addressing emerging challenges such as distributed systems, privacy-preserving computation, and, more recently, the threat posed by quantum computing.

1.2 QUANTUM COMPUTING AND POST-QUANTUM CRYPTOGRAPHY

Quantum computing emerged as a new computational paradigm in the 1980s to leverage principles of quantum mechanics, such as superposition and entanglement. It promises to perform computations for certain problems exponentially faster than classical computers. Notably, Shor's algorithm [Sho94] can efficiently factor large integers and compute discrete logarithms, which are the foundational hard problems underlying widely used public-key cryptosystems like RSA and ECC (Elliptic Curve Cryptography).

Although large-scale quantum computers capable of running Shor's algorithm do not yet exist, their potential development poses a significant threat to current cryptographic systems. To prepare, the cryptographic community has developed the field of post-quantum cryptography, which designs cryptographic schemes whose security reduces to problems believed to resist quantum attacks.

Several problems have been proposed as the basis for post-quantum cryptography, including lattice problems [Ajt96; Reg05], code-based problems [McE78], multivariate polynomial problems [MI88], isogeny problems [RS06; JD11], and hash-based problems [Mer90]. Among these, lattice-based cryptography has emerged as a leading candidate. Lattice problems have been subject to cryptanalysis in depth, and benefit from strong worst-case to average-case hardness connections, building trust in their security. Additionally, they offer efficient implementations and competitive sizes.

In 2017, the National Institute of Standards and Technology (NIST) initiated a process to evaluate and standardize post-quantum cryptographic algorithms. After several rounds of selection, NIST announced in 2022 the standardization of several algorithms, including three lattice-based schemes out of four: Kyber [SABD+22] (ML-KEM) for public-key encryption and key establishment, Dilithium [LDKL+22] (ML-DSA) and Falcon [PFHK+22] (FN-DSA) for digital signatures. They are now being actively integrated into protocols and software, with several notable cases such as the support of ML-KEM in TLS by Cloudflare [WR22].

While this is a significant step, many advanced cryptographic primitives remain underexplored in the post-quantum setting.

1.3 THE NEED FOR THRESHOLD SIGNATURES

Digital signatures guarantee that only the holder of the secret key can produce valid signatures, while anyone can verify signatures using the public key. They are widely used with crucial applications such as authenticating software updates, secure TLS handshakes for internet communications, authorizing transactions, and enabling consensus in distributed systems. A signature scheme typically consists of three algorithms: key generation, signing, and verification.

In the case of sensitive applications, such as cryptocurrency wallets or certificate authorities, relying on a single party to hold the signing key creates significant risks. If that party experiences a failure, the entire system may cease to function, with potentially severe financial losses (e.g., in a cryptocurrency wallet). In parallel, if that party is compromised, an attacker may gain the ability to produce valid signatures, leading to unauthorized use of the system.

Threshold signature schemes (TSS) were introduced by Desmedt and Frankel in 1991 [DF92] to solve this challenge by distributing trust across multiple participants, based on the earlier concept of distributed computation [Yao86; Des88]. In a (T, N)-threshold scheme, any set of at least T out of N parties can jointly produce a signature that verifies under a single public key (this is known as *correctness*), while any subset of fewer than T parties is unable to sign on its own (this is *unforgeability*). They ensure resistance against both failures (up to $N - T$ parties can be offline or unresponsive) and compromises (up to $T - 1$ parties can be corrupted by an adversary without endangering the signing key).

This initial work has been refined and extended over the years, leading to a rich literature on the security models [GJKR96; CGJK+99; Can01; BCKM+22], properties of threshold signature schemes [Ped91; GJKR96; AL07; KG20], and many constructions [GJKR96; GRJK00; GGN16; KG20; Lin24; DKMM+24] for various underlying signature algorithms. In particular, several threshold signature schemes have reached a high level of maturity and are actively being deployed in practice.

Recognizing the growing importance of threshold cryptography, NIST launched the *Multi-Party Threshold Cryptography* (MPTC) project to drive the development of threshold schemes. Sev-

eral workshops have been held since 2019 to discuss evaluation criteria, and these efforts culminated in a formal call for proposals in 2026 [NIST-IR8214C]. This initiative is further igniting interest in the design and optimization of practical threshold signature schemes.

For now, let us put aside the formal security models, and focus on the high-level goals and properties of threshold signatures. In addition to correctness and unforgeability, threshold signature schemes often aim to achieve the following properties:

- **Distributed Key Generation (DKG).** A protocol that allows parties to jointly generate a public/private key pair without any single party knowing the entire private key. This eliminates the need for a trusted dealer and enhances security by distributing trust among the participants even during key generation.
- **Robustness.** The ability to produce a valid signature even when some parties behave maliciously or send incorrect information during the signing process. Robust schemes include mechanisms to ensure that honest parties can always complete the signing operation.
- **Identifiable Aborts.** Achieving robustness comes at a significant cost, and in some scenarios, it is acceptable for the signing process to fail as long as misbehaving parties are identified. Identifiable aborts ensure that when an abort occurs, honest parties can identify which participants caused the abort. This property allows to take corrective actions, such as banning misbehaving parties.
- **Standard Compatibility.** The ability to produce signatures that are compatible with existing signature standards and verification algorithms. This ensures that threshold signatures can be seamlessly integrated into existing systems without requiring modifications to verification procedures.

In the classical setting, threshold signature schemes have reached a high level of maturity, with efficient protocols satisfying all the above desirable properties for widely used signature algorithms such as RSA [Sho00], EdDSA [KG20; RRJS+22], and ECDSA [CGGM+20]. These schemes are actively being deployed.

However, despite the standardization of post-quantum signature schemes, practical threshold signatures in the post-quantum setting remained an open question until very recently. Initial works on post-quantum threshold signatures [BKP13; BGGJ+18; CS19] laid important theoretical foundations and showed that it is feasible to design post-quantum constructions, but their protocols are far from practical. More recently, [ASY22; GKS24; BTT22] focused on lattice-based constructions, and significantly improved efficiency and practicality. Still, they remained out of reach for real-world deployments, with signature sizes in the order of 100kB and high signing latencies expected to be in the order of seconds or minutes. A breakthrough came with the work of [DKMM+24], which introduced the first efficient post-quantum threshold signature scheme, based on the lattice-based signature scheme Raccoon [PKPR24], achieving signature sizes of about 10kB – an order of magnitude smaller than prior works, and high efficiency, with signing latencies in the order of milliseconds. It was followed-up with other threshold variants of Raccoon: [EKT24; BKLM+25] proposed two-round signing protocols at the cost of a higher communication overhead, and [KRT24] proposed a 5-round protocol that achieves security in a stronger model where adversaries can adaptively corrupt parties during the protocol execution. However, these Raccoon-based schemes do not distribute their key generation nor provide robustness or identifiable aborts, and they are not compatible with standardized signature schemes, leaving many questions open beyond efficiency.

Due to the promising results on lattice-based threshold cryptography, we decided to focus this thesis on designing practical post-quantum threshold signatures from lattices, addressing the remaining challenges toward deployable schemes.

1.4 CHALLENGES IN LATTICE-BASED THRESHOLD SIGNATURES

Before detailing our contributions, we would like to provide some context to help readers understand the challenges specific to designing threshold signatures in the lattice setting.

The first step is to understand the way lattice-based signatures work. There are two main families of lattice-based signatures: hash-and-sign schemes [GPV08; MP12; PFHK+22; JRS24] and Fiat-Shamir schemes [Lyu12; PKPR24; CCDG+24].

HASH-AND-SIGN. Hash-and-sign schemes build on the concept of one-way functions, i.e. a function that is easy to compute on every input, but hard to invert given a random image. The idea is to introduce a secret trapdoor that allows the signer to efficiently compute pre-images of the one-way function. Concretely, given a one-way function f and a message msg, the signer computes a hash of the message to obtain a target value $t = H(\mathsf{msg})$, and then uses its secret trapdoor on f to sample a pre-image z such that $t = f(z)$.

In the lattice setting, the trapdoor one-way function is typically based on hard lattice problems such as the Short Integer Solution (SIS) and Learning With Errors (LWE) problems, and constrains the pre-image z to be a short vector in a lattice. However, when signing, it is crucial that the signature z does not leak information about the trapdoor, as this would compromise the security of the scheme. To achieve this, lattice-based hash-and-sign schemes employ techniques such as rejection sampling [LW15; PP21; JRS24] and Gaussian sampling [GPV08; PFHK+22] to ensure that the distribution of signatures is close to a target distribution that is independent of the trapdoor.

FIAT-SHAMIR. Fiat-Shamir schemes are based on the foundational Fiat-Shamir transformation [FS87], which converts *interactive* identification protocols – proving knowledge of a secret – into *non-interactive* signature schemes using hash functions. In a typical Fiat-Shamir signature scheme, the signing process involves three steps:

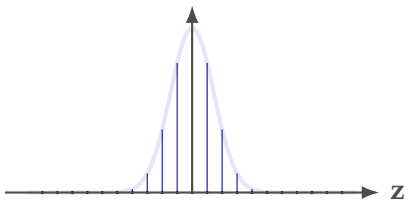
1. The signer generates a random commitment value based on its secret key and some randomness.
2. The signer computes a challenge by hashing the commitment and the message to be signed.
3. The signer computes a response using its secret key, the commitment, and the challenge.

In the lattice setting, Fiat-Shamir schemes typically prove knowledge of a short vector solution to a lattice problem, such as SIS or LWE, without revealing the vector itself. To ensure soundness, i.e. that only someone knowing the secret key can produce valid signatures, these schemes typically require producing short responses. As for hash-and-sign schemes, this is often achieved using techniques ensuring that the response distribution is independent of the secret key, so as to not leak it, including rejection sampling [Lyu09; DDLL13; LDKL+22] and Gaussian sampling [DPS23].

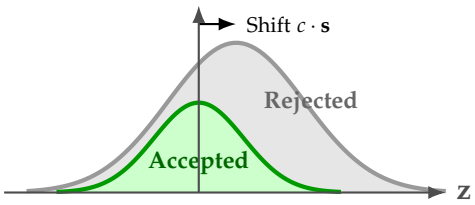
SAMPLING SIGNATURES. Both of the above paradigms require sampling signatures from a distribution that is independent of the secret key, while ensuring shortness. There are several techniques to do so: Gaussian sampling, rejection sampling, and the more recent noise flooding technique. These techniques feature different trade-offs in terms of efficiency and implementation complexity, which in turn appear as a major factor impacting the complexity of designing threshold signatures.

We provide a high-level overview of these techniques below, and an intuitive one-dimensional illustration of the resulting signature distributions in Fig. 1.1; the actual schemes operate over high-dimensional lattices.

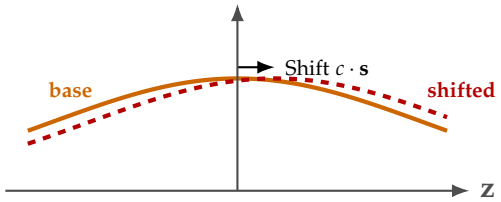
- **Gaussian sampling.** This technique samples signatures from a Gaussian distribution conditioned on the target value. It provides the most compact preimages, and in turn the smallest signatures. However, it is also the most complex to implement, typically requiring high-precision arithmetic [DN12; MW17] and careful numerical analysis [Pre17] to ensure security and correctness.
- **Rejection sampling.** This technique samples candidate signatures from a broader distribution and then accepts or rejects them based on a probability that ensures that the final distribution is independent of the secret key. Rejection sampling is generally simpler to implement than Gaussian sampling, but it leads to larger signatures and potentially increased signing times due to the potential need for multiple sampling attempts.
- **Noise flooding.** This more recent technique adds a noise to the signature to mask the secret key. Given a sufficiently large noise, the distribution of signatures remains sufficiently close to the original noise distribution, thus not revealing the secret key. This technique is typically the simplest to implement. However, it often results in larger signatures.



(a) Gaussian Sampling. Signatures are sampled from a Discrete Gaussian in the lattice coset defined by the target (i.e., $\mathbf{z}$ is sampled around 0 such that $f(\mathbf{z}) =$ target). This yields the most compact signatures but is the most complex to implement.



(b) Rejection Sampling. Rejects signature candidates so that the output follows the "Accepted" distribution – independent of $c \cdot \mathbf{s}$. Simpler, but wider distribution.



(c) Noise Flooding. A large base noise distribution is used to drown out the secret-dependent shift $c \cdot \mathbf{s}$. While conceptually the simplest, it produces wider signature distributions to maintain security.

Figure 1.1: Visual comparison of signature generation techniques.

Interestingly, the above hierarchy of complexity in implementing these sampling techniques is mirrored in the complexity of designing threshold signatures using them. The most efficient lattice-based threshold signature scheme to date uses noise flooding [DKMM+24], while prior works using rejection sampling [BTT22] were significantly more complex and less efficient, and it remains an open question to practically thresholdize a Gaussian-sampling based signature. We illustrate this design space in Fig. 1.2.

The typical process to design a lattice-based threshold signature scheme involves two main steps (i) choosing the paradigm (hash-and-sign or Fiat-Shamir) and the sampling technique (Gaussian, rejection, or noise flooding) underlying the base signature scheme, and then (ii) distributing the signing algorithm among multiple parties, while preserving correctness and unforgeability, and ideally achieving the additional desirable properties we mentioned previously.

The usual approach for (ii) is to consider each step of the signing algorithm, and design a subprotocol to distribute it among the parties. We distinguish two main directions to do so, which we describe below.

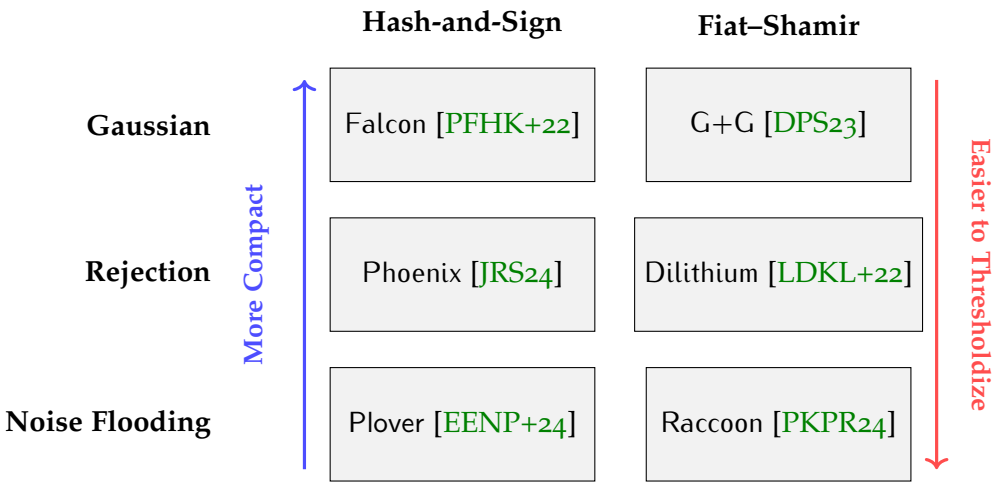


Figure 1.2: Design space of lattice-based signature schemes. We include representative signature schemes for each paradigm and sampling technique.

GENERIC MPC WITH UC SECURITY. The first idea that comes to mind is to use generic techniques from secure multiparty computation (MPC), and prove the security of the protocol in the Universal Composability (UC) framework [Can01]. This framework provides strong security guarantees, by proving a form of indistinguishability between the real protocol execution and an ideal execution with a trusted party. For signatures, works typically either model a generic signature functionality [Can03; BH04; ADN06] or directly the target signature signing [DKLs18; CCLS+19; CEN25]. We can theoretically design a protocol that allows multiple parties to jointly compute any function over their inputs without leaking information on the secret or intermediate steps, and ensuring that the operation is correctly performed. However, it is typically too inefficient for the dimensions and operations involved in lattice-based signatures, including the sampling from Gaussian or short distributions, polynomial arithmetic over rings, and comparisons.

Remark 1.4.1. Other composability frameworks for secure multi-party computation have been proposed [Mau10; HS15; KTR20] over the years. However, the main challenge of efficiency remains.

CUSTOM PROTOCOLS WITH GAME-BASED SECURITY. Another direction is to define tailored game-based security notions for the desirable properties of threshold signatures, and then design custom protocols for each step of the signing algorithm. This approach allows to significantly improve efficiency by leveraging lattice-specific properties to accept some controlled leakage or imperfect operations that do not compromise security. This is notably the approach taken in [DKMM+24], which distributes the noise-flooding-based signing process by having each party locally sample noise, compute a noisy partial signature, and then the scheme aggregates these partial signatures to form a valid signature. By carefully analyzing what information can safely be revealed during the protocol execution, and adapting the scheme parameters to support larger aggregated signatures, this approach achieves significantly better efficiency than generic MPC.

This customization crucially relies on the possibility to decompose the signing algorithm into simple operations that can be distributed securely. So far, the most amenable signing algorithms to such decomposition are those using noise flooding, which mainly depend on short vector sampling. Several works have applied these ideas to rejection-sampling-based signatures [BTT22], by performing both short vector sampling and rejection sampling locally within each signer, but with significantly less efficiency.

We provide an extended overview of the techniques used in prior works to design lattice-based threshold signatures in Section 3.5.

As previously mentioned, beyond efficiency, several challenges remain toward practical lattice-based threshold signatures, including achieving robustness against malicious parties, supporting identifiable aborts to improve efficiency in many scenarios, and designing standard-compliant schemes compatible with existing verification algorithms and key formats.

1.5 CONTRIBUTIONS OF THIS THESIS

This thesis solves several open questions in the design of practical lattice-based threshold signatures. We introduce novel techniques to augment noise-flooding-based signatures with the robustness and identifiable aborts properties they lacked, and we overcome the challenges that prevented efficiently distributing signature algorithms based on rejection-sampling, culminating with the first threshold signature scheme compatible with the ML-DSA standard. Throughout the thesis, we instantiate our techniques on two representative schemes: Raccoon [PKPR24], a noise-flooding-based Fiat-Shamir signature shown to be threshold-friendly in [DKMM+24], and Dilithium (ML-DSA) [LDKL+22], the NIST-standardized signature based on rejection sampling. While our techniques could be applied to the two-round threshold variant of Raccoon [EKT24] to achieve robustness and identifiable aborts[1], we focus on the three-round protocol of [DKMM+24] for clarity of the exposition, as two-round schemes imply a more involved signing which is orthogonal to the main ideas we introduce in this thesis. More precisely, our main contributions are as follows:

1. **Robust DKG and signing for lattices.** We introduce the first robust distributed key generation and threshold signing protocols based on noise flooding. These protocols guarantee the production of a valid signature even in the presence of malicious parties. We rely on a new lightweight verifiable secret-sharing mechanism that avoids costly zero-knowledge proofs by exploiting simple noisy linear relations. We prove its security under the Hint-MLWE assumption and integrate these checks into signing to ensure correctness. This demonstrates, for the first time, that full robustness can be achieved for lattice-based threshold signatures using only basic building blocks. This is detailed in Chapter 4.

2. **Efficient protocols with identifiable aborts.** While full robustness requires an honest majority during signing and can remain costly in practice, in many real-world scenarios, it suffices that parties can detect misbehavior and identify the culprits when an abort occurs, allowing the system to recover quickly without the overhead of full robustness.

   We develop new secret sharing techniques ensuring the shortness of user shares, and small reconstruction coefficients. This enables the unique realization of the *"threshold designer's dream"*: we design a noise-flooding based threshold scheme where signature shares double as valid signatures under the corresponding secret key shares and enable the detection of misbehavior simply by verifying signature shares. This leads to highly efficient protocols for distributed key generation and signing with identifiable aborts. This is detailed in Chapter 5.

3. **Standard-compliant threshold Dilithium (ML-DSA).** Finally, we resolve the main obstacle that had prevented the design of competitive threshold schemes relying on rejection-sampling-based signatures. We develop new simulation techniques to show that *rejected transcripts in Fiat-Shamir-with-aborts signatures can be safely revealed for practical parameters*. Combined with our short-secret sharing techniques and targeted optimizations, this leads to the *first practical threshold signature scheme for Dilithium (*ML-DSA*)*, fully compatible with

1 Although we choose to not describe it in this thesis, the NIST submission Hermine [BCPE+26a] consists in applying our identifiable aborts techniques to the two-round setting, extending the protocol [EKT24].

the standardized verification algorithm and key formats, and that works for a small number of parties ($N \leqslant 6$ for a communication of at most a few MB). This is detailed in Chapter 6.

Taken together, these results address several open challenges on the path to practical post-quantum threshold signatures. It strengthens the position of lattice-based TSS based on noise flooding as a versatile and practical solution, even for a large number of parties, and supporting advanced properties such as robustness and identifiable aborts. It also demonstrates that rejection-sampling-based signatures can also be efficiently thresholdized for the first time, achieving higher compactness, standard compatibility and competitive efficiency for a small number of parties, albeit advanced properties such as robustness remain open in that setting.

As a demonstration of the relevance of our contributions, our techniques underlie several of the schemes proposed in response to NIST's MPTC call for proposals [NIST-IR8214C]:

- **Amber [BPLP26]:** Amber is an efficient lattice-based IND-CCA Threshold KEM proposal that explicitly incorporates our techniques for identifying aborts presented in Chapter 5.
- **Hermine [BCPE+26a]:** Hermine is a partially non-interactive two-round threshold signature scheme based on Raccoon that applies our abort identification techniques from Chapter 5 to the two-round setting, and further adds the support of key refreshing.
- **Mithril [CDPE+26]:** Mithril is a threshold signature scheme for ML-DSA that builds on our techniques from Chapter 6 to achieve standard compatibility for a small number of parties.

1.6 SUPPORTING PUBLICATIONS

We have contributed several of the results presented in this thesis as part of papers under review or that have been published in peer-reviewed conferences. Below is a list of these publications, along with their corresponding chapters in this thesis:

- **Flood and Submerse: Distributed Key Generation and Robust Threshold Signature from Lattices [ENP24].** This paper studies the design of robust distributed key generation and signing protocols for lattice-based threshold signatures using noise flooding. It introduces a new verifiable secret-sharing mechanism and integrates it into a threshold signing protocol for the signature scheme Plover [EENP+24] to ensure correctness even in the presence of misbehaving parties. This work is presented in Chapter 4, albeit adapted to the Raccoon signature scheme [PKPR24] for consistency with the rest of the thesis.

  It has been published in the proceedings of CRYPTO 2024.

- **Simple and Efficient Lattice Threshold Signatures with Identifiable Aborts [dENP25].** This paper introduces new secret sharing techniques that sample short secret shares and enable the reconstruction of partial secrets with small norms. Leveraging these techniques, it designs efficient distributed key generation and signing protocols with *identifiable aborts* for noise-flooding-based signatures. The formalism introduced in this work, as well as its *replicated secret sharing* with short shares is presented in Chapter 5. We further extend this chapter with the *Vandermonde secret sharing* scheme that scales to a larger number of parties. This contribution is part of a recent rework of the original preprint [BCPE+26b], which builds on the same core techniques to introduce Hermine, a two-round threshold signature scheme based on Raccoon that supports key refreshing and identifiable aborts. We focus on the original three-round scheme in this thesis for clarity of the exposition.

  This work is available on ePrint.

- **Finally! A Compact Lattice-Based Threshold Signature [PN25].** This paper resolves the main obstacle that had prevented the design of practical threshold signatures based on rejection-sampling-based signatures. It introduces new simulation techniques to safely reveal rejected transcripts in Fiat-Shamir-with-aborts signatures, and combines them with short-secret sharing techniques to design the most compact threshold signature to date, scaling up to 8 parties. Its simulation results make up the theoretical core of Chapter 6.

  This work has been published in the proceedings of PKC 2025.

- **Efficient Threshold ML-DSA [CPEN+26].** This paper presents the first practical threshold signature scheme for Dilithium (ML-DSA), fully compatible with the standardized verification algorithm and key formats. It builds on the simulation techniques introduced in [PN25] and combines them with targeted optimizations to achieve competitive efficiency for a small number of parties. This work is presented in Chapter 6. Note that this work additionally contains a distributed key generation, and a posteriori sharing of existing ML-DSA keys that we chose not to present in this thesis for conciseness.

  This work will be published in the proceedings of USENIX Security 2026.

1.7 ADDITIONAL PUBLICATIONS

In addition to the works related to this thesis, we have contributed to several other publications in the field of cryptography:

- **Plover: Masking-Friendly Hash-and-Sign Lattice Signatures [EENP+24].** This paper introduces Plover, the first lattice-based hash-and-sign signature scheme designed to be masking-friendly. Building on concepts from Raccoon [PKPR24] to counter side-channel attacks, the paper provides a systematic framework for analyzing their security and demonstrates their applicability to hash-and-sign signatures. By leveraging noise flooding, it achieves quasi-linear complexity in the number of masking shares. Plover can be considered the hash-and-sign counterpart of Raccoon, and we demonstrate in [ENP24] how the threshold techniques developed in Chapter 4 for noise-flooding signatures apply naturally to it to achieve robust signing.

  This work has been published in the proceedings of EUROCRYPT 2024.

- **Unmasking TRaccoon: A Lattice-Based Threshold Signature with An Efficient Identifiable Abort Protocol [PKNR+25].** Akin to our contributions in Chapter 5, this work focuses on enhancing the TRaccoon threshold signature scheme with an identifiable abort mechanism. We take a different approach by preserving the original design of TRaccoon and introduce a lightweight add-on that runs zero-knowledge proofs to identify malicious signers only in the event of a failure. This incurs an added communication cost of $60 + 6.4|T|$ KB when the protocol fails. Additionally, we provide the first formal security analysis of a zero-knowledge variant of LaBRADOR and introduce a new game-based definition for interactive identifiable abort protocols, extending standard unforgeability definitions.

  This work has been published in the proceedings of CRYPTO 2025.

- **Share the MAYO: thresholdizing MAYO [CEN25].** In this paper, we study the practical thresholdization of OV-based signature schemes, specifically focusing on MAYO and UOV, which are candidates in the NIST process for standardization of additional digital signature schemes. Our approach begins by addressing the challenges associated with thresholdizing algorithms that sample solutions to linear equation systems of the form $\mathbf{A}x = y$, which are fundamental to OV-based signature schemes. Previous attempts have introduced levels of

leakage that we deem insecure. We propose a new solution, discuss its security, and assess its practicality. We then explore thresholdizing the entire signing functionality of these signature schemes.

This work has been published in the proceedings of PQCrypto 2025.

- **Comprehensive Deniability Analysis of Signal Handshake Protocols: X3DH, PQXDH to Fully Post-Quantum with Deniable Ring Signatures [KNTW25].** The Signal protocol relies on a handshake (formerly X3DH, now PQXDH) to set up secure conversations, valuing *deniability* so users can deny participation. Prior analyses use varying, ad-hoc models that obscure guarantees and prevent comparison.

  Building on the abstraction by Hashimoto et al. [HKW25], we present a unified framework for analyzing Signal handshake deniability. We examine X3DH and PQXDH, clarifying PQXDH's deniability against *harvest-now–judge-later* quantum adversaries. We also analyze post-quantum alternatives such as RingXKEM that use ring signatures. By introducing a deniability metric inspired by differential privacy, we offer relaxed, pragmatic guarantees. This metric further allows us to define *deniable ring signatures* (a relaxation of the traditional anonymity notion), enabling efficient constructions from the NIST standard Falcon and the candidate for standardization MAYO, which are deniable despite not being fully anonymous.

  This work has been published in the proceedings of USENIX Security 2025.

- **Practical Deniable Post-Quantum X3DH: A Lightweight Split-KEM for K-Waay [Nio25].** The Signal Protocol faces the challenge of migrating to a post-quantum world while preserving critical properties such as asynchrony and deniability. While PQXDH grants post-quantum confidentiality, full migration of the X3DH handshake remains elusive. As studied in [KNTW25], the main approach to migrate X3DH involves using ring signatures to achieve deniability. K-Waay [CHNR+24] proposed to explore an alternative path via split KEMs but suffers from size limitations compared to ring signature-based approaches.

  This work introduces Sparrow-KEM and Sym-Sparrow-KEM, novel asymmetric and symmetric split KEMs designed to optimize K-Waay. Leveraging the MLWE assumption, we reduce communication by 5.1× and improve speed by 40× over prior split KEMs. Sym-Sparrow-KEM is the first symmetric split-KEM to offer deniability along with strong implicit authentication properties (IND-1KCA, IND-1BatchCCA). Our results demonstrate the feasibility of a compact, deniable post-quantum X3DH based on split KEMs.

  This work has been published in the proceedings of ASIACCS 2025.

- **Subversion-resilient Key-exchange in the Post-quantum World [DFJN+25].** Subversion-resilient Authenticated Key-Exchange (AKE) ensures security even when parts of the protocol implementation are tampered with. One way to achieve AKE is by using Reverse Firewalls (RFs) to restore security.

  In this work, we extend RF-based subversion resilience in security definitions, constructions, and formal verification. First, we introduce a useful relaxation of the notion of security in subversion-resilient AKE with RFs: the goal is no longer to prevent all exfiltration, but rather to restore to the AKE protocol a property lost upon subversion. We focus specifically on authenticating and (key-)securing RFs, and consider a spectrum of compromises, designing a framework in which adversaries can tamper with some components of the implementation but perhaps not others. Aiming for post-quantum security, we define 're-randomizable Key Encapsulation Mechanisms', providing instantiations based on classical Diffie-Hellman and Kyber. Finally, we establish foundations for the formal verification of RF-based protocols, proving our construction secure using the CryptoVerif prover, in addition to computational-security proofs in usual Bellare-Rogaway methodology.

This work has been published in the proceedings of CCS 2025.

- **Adaptively-Secure Three-Round Threshold Schnorr from DL [NRT26].** This work presents the first threshold Schnorr signature scheme secure against adaptive corruptions and requiring only three rounds (two online, one offline) under the Discrete Logarithm (DL) assumption in the random oracle model. Previous adaptively-secure threshold Schnorr protocols either relied on stronger assumptions or required five rounds. The scheme preserves the standard Schnorr interface while minimizing latency. The construction introduces novel techniques, including an equivocal commitment with simulation-extractable NIZK and a masking-based aggregated opening strategy for homomorphic commitments, and provides a formalization of a strong adaptive security notion.

  This work will be published in the proceedings of EUROCRYPT 2026.

- **Revisiting PQ WireGuard: A Comprehensive Security Analysis With a New Design Using Reinforced KEMs [HKNW25].** WireGuard is a VPN providing high performance based on the Noise protocol. A recent post-quantum (PQ) variant was proposed by Hülsing et al. [HNSW+21], however since Wireguard requires the handshake message to fit in one UDP packet of size roughly 1200 B, they rely on Classic McEliece, whose large public keys significantly increase server memory requirements and complicates kernel-level deployment.

  In this work, we revisit PQ WireGuard to improve its design, security, and efficiency. We address binding issues in PQ KEMs and prove security in a new computational model. We introduce 'reinforced KEM' (RKEM) and a construction named 'Rebar' to compress ML-KEM-like ciphertexts. This enables a PQ WireGuard protocol where the server avoids storing large keys, reducing public key memory usage by 190 to 390×.

  This work will be published in the proceedings of S&P 2026.

2 INTRODUCTION (EN FRANÇAIS)

2.1 CRYPTOLOGIE

La cryptologie est la science qui étudie la sécurisation des communications en la présence d'adversaires. Elle se compose de deux branches étroitement liées : la cryptographie, qui vise à protéger les communications et à dissimuler l'information, et la cryptanalyse, qui cherche au contraire à affaiblir ou à contourner ces protections. Si le besoin de communiquer de manière sûre est aussi ancien que la communication elle-même, l'étude formelle de la cryptologie s'est structurée au XX^{e} siècle, s'appuyant sur des siècles de conception de chiffrements et accélérée par l'essor des technologies de communication mécaniques puis électroniques.

Bien que des chiffrements mécaniques tels qu'Enigma incarnent déjà l'essence des cryptosystèmes modernes (chiffrement algorithmique et sécurité fondée sur une clé), l'ère moderne de la cryptologie débute véritablement avec le cadre informationnel de Claude Shannon pour la confidentialité [Sha49], puis avec le développement des modèles de sécurité computationnelle dans les années 1970. L'introduction de la cryptographie à clé publique par Diffie et Hellman [DH76] a ensuite profondément transformé le domaine, en rendant possible la communication sûre sur des réseaux ouverts sans secret pré-partagé.

Jusqu'aux travaux de Diffie et Hellman, la cryptographie concernait principalement des schémas à clé symétrique, dans lesquels l'émetteur et le récepteur partagent une même clé secrète. Ce paradigme est cependant limité en termes de passage à l'échelle : dans un grand réseau, chaque paire de participants doit échanger et gérer une clé distincte. Historiquement, ces échanges exigeaient des rencontres physiques ou des courriers de confiance, rendant la cryptographie peu pratique à grande échelle. La cryptographie à clé publique introduit des schémas asymétriques, où chaque utilisateur possède une clé publique (partagée librement) et une clé secrète (conservée confidentielle). Cette avancée permet de communiquer de manière sûre avec un grand nombre de participants : il suffit d'utiliser la clé publique du destinataire – typiquement distribuée via des infrastructures numériques – sans échange de clé *physique* préalable. Deux primitives majeures émergent : le chiffrement à clé publique, qui permet à quiconque d'envoyer un message chiffré à un destinataire à l'aide de sa clé publique, et la signature numérique, qui permet de signer un message de sorte que tout le monde puisse vérifier la signature via la clé publique du signataire, tandis que seul le signataire peut produire des signatures valides.

L'année suivante, Rivest, Shamir et Adleman [RSA78] introduisent l'algorithme RSA, qui deviendra le premier cryptosystème à clé publique massivement déployé, permettant à la fois de chiffrer des messages et de produire des signatures.

Depuis, la cryptologie a évolué au rythme des technologies, répondant à des défis nouveaux tels que les systèmes distribués, le calcul respectueux de la vie privée et, plus récemment, la menace que fait peser l'informatique quantique.

2.2 INFORMATIQUE QUANTIQUE ET CRYPTOGRAPHIE POST-QUANTIQUE

L'informatique quantique est apparue dans les années 1980 comme un nouveau paradigme de calcul, exploitant des principes de mécanique quantique tels que la superposition et l'intrication. Elle promet, pour certains problèmes, des gains exponentiels par rapport aux ordinateurs classiques. En particulier, l'algorithme de Shor [Sho94] permet de factoriser efficacement de grands

entiers et de calculer des logarithmes discrets, qui sont les problèmes difficiles à la base de cryptosystèmes à clé publique largement utilisés, tels que RSA et l'ECC (cryptographie sur courbes elliptiques).

Même si des ordinateurs quantiques capables d'exécuter l'algorithme de Shor n'existent pas encore, leur développement potentiel constitue une menace importante pour les systèmes cryptographiques actuels. Pour s'y préparer, la communauté cryptographique a développé la cryptographie post-quantique, qui conçoit des schémas dont la sécurité se réduit à des problèmes supposés résister aux attaques quantiques.

Plusieurs familles de problèmes ont été proposées comme base pour la cryptographie post-quantique, notamment les problèmes sur réseaux euclidiens [Ajt96; Reg05], les problèmes de codes [McE78], les systèmes polynomiaux multivariés [MI88], les isogénies [RS06; JD11] et les constructions fondées sur les fonctions de hachage [Mer90]. Parmi elles, la cryptographie sur réseaux s'est imposée comme un candidat majeur. Les problèmes sur réseaux ont fait l'objet d'analyses cryptanalytiques approfondies et bénéficient de liens solides entre difficulté pire cas et cas moyen, renforçant la confiance dans leur sécurité. Ils offrent en outre des implémentations efficaces et des tailles compétitives.

En 2017, le National Institute of Standards and Technology (NIST) a lancé un processus d'évaluation et de standardisation d'algorithmes post-quantiques. Après plusieurs années de sélection, le NIST a annoncé en 2022 la standardisation de plusieurs algorithmes, dont trois schémas sur quatre sont basés sur les réseaux : Kyber [SABD+22] (ML-KEM) pour le chiffrement à clé publique et l'établissement de clés, ainsi que de Dilithium [LDKL+22] (ML-DSA) et Falcon [PFHK+22] (FN-DSA) pour les signatures numériques. Ils sont désormais activement intégrés dans des protocoles et des logiciels, avec notamment la prise en charge de ML-KEM dans TLS par Cloudflare [WR22].

Il s'agit d'une étape importante, mais de nombreuses primitives cryptographiques avancées restent encore peu explorées dans le cadre post-quantique.

2.3 LE BESOIN DE SIGNATURES À SEUIL

Les signatures numériques garantissent que seul le détenteur de la clé secrète peut produire des signatures valides, tandis que toute personne peut vérifier une signature à l'aide de la clé publique. Elles sont omniprésentes dans des applications critiques : authentification des mises à jour logicielles, échanges TLS pour les communications sur Internet, autorisation de transactions, ou encore consensus dans les systèmes distribués. Un schéma de signature se compose typiquement de trois algorithmes : génération de clés, signature et vérification.

Dans des applications sensibles, telles que les portefeuilles de cryptomonnaies ou les autorités de certification, confier la clé de signature à une seule entité crée des risques importants. En cas de panne de cette entité, le système peut cesser de fonctionner, avec des pertes potentiellement sévères (par exemple pour un portefeuille de cryptomonnaies). Par ailleurs, si cette entité est compromise, un attaquant peut produire des signatures valides, conduisant à un usage non autorisé du système.

Les schémas de signature à seuil (*threshold signature schemes*, TSS) ont été introduits par Desmedt et Frankel en 1991 [DF92] afin de répondre à ce défi en répartissant la confiance entre plusieurs participants, s'appuyant sur le concept antérieur de calcul distribué [Yao86; Des88]. Dans un schéma à seuil (T, N), tout ensemble d'au moins T parties parmi N peut produire conjointement une signature vérifiable sous une seule clé publique (propriété dite de *correction*), tandis que tout sous-ensemble de taille strictement inférieure à T est incapable de signer seul (*inforgeabilité*). De tels schémas offrent une résistance à la fois aux pannes (jusqu'à $N - T$ parties peuvent être hors ligne ou non réactives) et aux compromissions (jusqu'à $T - 1$ parties peuvent être corrompues sans mettre en péril la clé de signature).

Ces premiers travaux ont été affinés et étendus au fil des années, donnant naissance à une vaste littérature sur les modèles de sécurité [GJKR96; CGJK+99; Can01; BCKM+22], les propriétés des schémas à seuil [Ped91; GJKR96; AL07; KG20], ainsi que de nombreuses constructions à seuil [GJKR96; GRJK00; GGN16; KG20; Lin24; DKMM+24] pour différents algorithmes de signature sous-jacents. En particulier, plusieurs schémas à seuil ont atteint un haut niveau de maturité et sont effectivement déployés en pratique.

Reconnaissant l'importance croissante de la cryptographie à seuil, le NIST a lancé le projet *Multi-Party Threshold Cryptography* (MPTC) afin d'explorer une potentielle standardisation de schémas à seuil. Plusieurs événements ont été organisés depuis 2019 pour discuter des critères d'évaluation, et ces efforts ont abouti à un appel formel à propositions en 2026 [NIST-IR8214C]. Cette initiative renforce encore l'intérêt pour la conception et l'optimisation de schémas de signature à seuil pratiques.

Pour l'instant, mettons de côté les modèles de sécurité formels et concentrons-nous sur les objectifs et propriétés de haut niveau des signatures à seuil. Au-delà de la correction et de l'inforgeabilité, les schémas à seuil visent souvent les propriétés suivantes :

- **Génération distribuée de clés (DKG).** Un protocole permettant aux parties de générer conjointement une paire de clés publique/privée sans qu'aucune partie ne connaisse la clé privée entière. Cela élimine le besoin d'un tiers de confiance et renforce la sécurité en répartissant la confiance dès la génération des clés.

- **Robustesse.** La capacité à produire une signature valide même si certaines parties se comportent de manière malveillante ou envoient des informations incorrectes pendant le processus de signature. Les schémas robustes incluent des mécanismes garantissant que le protocole termine toujours.

- **Abandons identifiables.** Obtenir une robustesse complète peut être coûteux et, dans certains scénarios, il est acceptable que la signature échoue dès lors que les parties fautives sont identifiées. Les abandons identifiables garantissent qu'en cas d'abandon, il est possible d'identifier quels participants en sont la cause, ce qui permet de prendre des mesures correctives (par exemple exclure des parties malveillantes).

- **Compatibilité avec les standards.** La capacité à produire des signatures compatibles avec des standards existants et avec les algorithmes de vérification standard. Cela permet d'intégrer des signatures à seuil dans des systèmes existants sans modifier les procédures de vérification.

Dans le cadre d'adversaire uniquement classique, les schémas de signature à seuil ont atteint un haut niveau de maturité, avec des protocoles efficaces satisfaisant l'ensemble des propriétés souhaitables ci-dessus pour des algorithmes largement utilisés tels que RSA [Sho00], EdDSA [KG20; RRJS+22] et ECDSA [CGGM+20]. Ces schémas sont déployés en pratique.

Cependant, malgré la standardisation de schémas de signature post-quantiques, la conception de signatures à seuil *pratiques* dans le cadre post-quantique est restée une question ouverte jusqu'à très récemment. Les premiers travaux sur les signatures à seuil post-quantiques [BKP13; BGGJ+18; CS19] ont posé des fondations théoriques importantes et montré la faisabilité de constructions post-quantiques, mais leurs protocoles sont loin d'être pratiques. Plus récemment, [ASY22; GKS24; BTT22] se sont concentrés sur des constructions sur réseaux et ont nettement amélioré l'efficacité et le caractère pratique. Malgré cela, ces schémas restaient hors de portée d'un déploiement réel, avec des signatures de l'ordre de 100 kB et des latences attendues de l'ordre de la seconde ou de la minute. Une avancée importante a été réalisée avec [DKMM+24], qui introduit le premier schéma de signature à seuil post-quantique pratique, fondé sur le schéma de signature sur réseaux Raccoon [PKPR24], obtenant des signatures d'environ 10 kB – un ordre de grandeur plus compact que les travaux antérieurs – et une grande efficacité, avec des latences

de l'ordre de la milliseconde. Ce travail a été suivi par d'autres variantes à seuil de Raccoon: [EKT24; BKLM+25] proposent des protocoles de signature en deux tours au prix d'un surcoût de communication, et [KRT24] propose un protocole en cinq tours assurant la sécurité dans un modèle plus fort où l'adversaire peut corrompre adaptativement des parties pendant l'exécution du protocole. Néanmoins, ces schémas à l'état de l'art basés sur Raccoon ne distribuent ni la génération de clés, ni n'offrent robustesse ou des abandons identifiables, et ils ne sont pas compatibles avec les schémas de signature standardisés, laissant de nombreuses questions ouvertes au-delà de l'efficacité.

Au vu des résultats prometteurs en cryptographie à seuil sur réseaux, nous avons choisi de consacrer cette thèse à la conception de signatures à seuil post-quantiques pratiques fondées sur les réseaux, en s'attaquant aux défis restants vers des schémas complètement déployables.

2.4 DÉFIS DES SIGNATURES À SEUIL SUR RÉSEAUX

Avant de détailler nos contributions, nous souhaitons donner quelques éléments de contexte afin d'aider le lecteur à comprendre les difficultés propres à la conception de signatures à seuil dans le cadre des réseaux euclidiens.

La première étape consiste à comprendre le fonctionnement des signatures sur réseaux. On distingue deux principales familles : les schémas selon le paradigme *hash-and-sign* [GPV08; MP12; PFHK+22; JRS24] et les schémas de type *Fiat–Shamir* [Lyu12; PKPR24; CCDG+24].

HASH-AND-SIGN. Les schémas *hash-and-sign* reposent sur la notion de fonction à sens unique : une fonction facile à calculer sur toute entrée, mais difficile à inverser à partir d'une image aléatoire. L'idée est d'introduire une trappe secrète permettant au signataire de calculer efficacement des préimages de la fonction. Concrètement, étant donnée une fonction à sens unique f et un message msg, le signataire hache le message pour obtenir une cible $t = H(\mathsf{msg})$, puis utilise la trappe secrète associée à f pour échantillonner une préimage z telle que $t = f(z)$.

Dans le cadre des réseaux, la fonction à sens unique avec trappe est typiquement fondée sur des problèmes difficiles tels que SIS (*Short Integer Solution*) et LWE (*Learning With Errors*), et contraint la préimage z à être un vecteur court d'un réseau. Lors de la signature, il est cependant crucial que la signature z ne révèle pas d'information sur la trappe : sinon la sécurité du schéma serait compromise. Pour l'éviter, les schémas *hash-and-sign* sur réseaux utilisent des techniques telles que l'échantillonnage par rejet [LW15; PP21; JRS24] et l'échantillonnage gaussien [GPV08; PFHK+22] afin de garantir que la distribution des signatures est proche d'une distribution cible indépendante de la trappe.

FIAT-SHAMIR. Les schémas Fiat–Shamir reposent sur la transformation fondatrice de Fiat et Shamir [FS87], qui convertit des protocoles d'identification *interactifs* – prouvant la connaissance d'un secret – en schémas de signature *non interactifs* via des fonctions de hachage. Dans un schéma de signature Fiat–Shamir typique, la signature se déroule en trois étapes :

1. Le signataire génère un engagement aléatoire à partir de sa clé publique et d'une source d'aléa.
2. Le signataire calcule un défi en hachant l'engagement et le message à signer.
3. Le signataire calcule une réponse à partir de sa clé secrète, de l'engagement et du défi.

Dans le cadre des réseaux, les schémas Fiat–Shamir prouvent typiquement la connaissance d'un vecteur court solution d'un problème sur réseaux, tel que SIS ou LWE, sans révéler ce vecteur. Pour garantir l'unforgeabilité, ces schémas exigent généralement des réponses courtes. Comme pour les schémas *hash-and-sign*, on utilise souvent des techniques garantissant que la

distribution des réponses est indépendante de la clé secrète, afin d'éviter toute fuite, notamment l'échantillonnage par rejet [Lyu09; DDLL13; LDKL+22] et l'échantillonnage gaussien [DPS23].

ÉCHANTILLONNAGE DES SIGNATURES. Les deux paradigmes ci-dessus requièrent une technique d'échantillonnage des signatures selon une distribution indépendante de la clé secrète, tout en garantissant qu'elles restent courtes. Il existe plusieurs techniques pour cela : l'échantillonnage gaussien, l'échantillonnage par rejet et la technique plus récente dite d'*inondation de bruit* (*noise flooding*). Ces techniques présentent des compromis différents en termes d'efficacité et de complexité d'implémentation, ce qui impacte directement la difficulté de conception de signatures à seuil.

Nous donnons ci-dessous une vue d'ensemble de ces techniques, et illustrons les distributions de signatures qu'elles produisent dans la Figure 2.1.

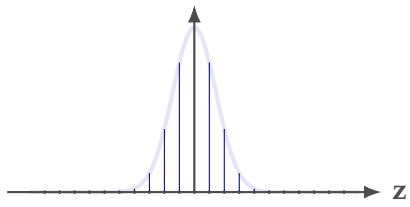
- **Échantillonnage gaussien.** Cette technique échantillonne des signatures selon une distribution gaussienne conditionnée par la valeur cible. Elle fournit les préimages les plus compactes, et donc les signatures les plus petites. En contrepartie, c'est la plus complexe à implémenter, nécessitant souvent de l'arithmétique à haute précision [DN12; MW17] et une analyse poussée [Pre17] pour garantir la sécurité.
- **Échantillonnage par rejet.** Cette technique échantillonne des signatures candidates selon une distribution plus large, puis les accepte ou les rejette selon une probabilité choisie de façon à rendre la distribution finale indépendante de la clé secrète. Elle est plus simple à implémenter que l'échantillonnage gaussien, mais peut conduire à des signatures plus grandes et à des temps de signature plus longs en raison du nombre potentiellement important de tentatives.
- **Inondation de bruit.** Cette technique plus récente ajoute du bruit à la signature afin de masquer la clé secrète. Pour un bruit suffisamment grand, la distribution des signatures devient suffisamment proche d'une distribution cible indépendante de la clé secrète. Elle est typiquement la plus simple à implémenter, au prix de signatures souvent plus grandes.

Cette hiérarchie de complexité d'implémentation se reflète dans la complexité de conception de signatures à seuil fondées sur celles-ci. Le schéma de signature à seuil sur réseaux le plus efficace actuellement utilise l'inondation de bruit [DKMM+24], tandis que les travaux antérieurs s'appuyant sur l'échantillonnage par rejet [BTT22] étaient nettement plus complexes et moins efficaces. Le design à seuil pratique d'un schéma fondé sur l'échantillonnage gaussien reste une question ouverte. Nous illustrons cet espace de conception dans la figure ci-dessous.

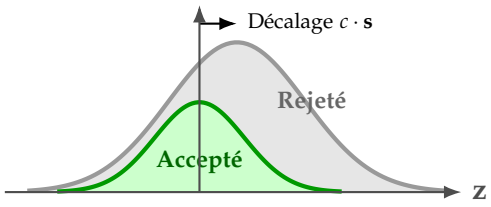
Le processus typique de conception d'un schéma de signature à seuil sur réseaux comporte deux étapes principales : (i) choisir le paradigme (*hash-and-sign* ou *Fiat–Shamir*) et la technique d'échantillonnage (gaussien, rejet ou inondation de bruit) du schéma de signature de base, puis (ii) distribuer l'algorithme de signature entre plusieurs parties tout en préservant correction et infalsifiabilité, et idéalement en atteignant les propriétés supplémentaires mentionnées plus haut.

L'approche habituelle pour (ii) consiste à considérer chaque étape de l'algorithme de signature et à concevoir un sous-protocole permettant de la répartir entre les parties. Nous distinguons deux grandes directions pour procéder, décrites ci-dessous.

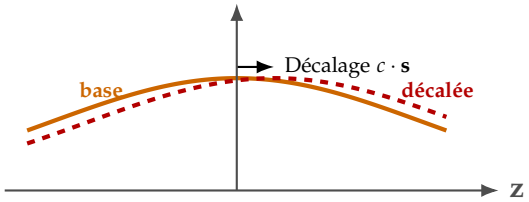
MPC GÉNÉRIQUE ET SÉCURITÉ UC. Une première idée est d'utiliser des techniques génériques de calcul multipartite sécurisé (MPC) et de prouver la sécurité du protocole dans le cadre du framework de composabilité universelle (UC) [Can01]. Ce cadre fournit de fortes garanties en établissant une forme d'indistinguabilité entre l'exécution réelle du protocole et une exécution idéale avec un tiers de confiance. Pour les signatures, on trouve dans la litérature à la fois des travaux modélisant de façon générique la fonctionnalité de signature [Can03; BH04; ADN06], ou bien directement les algorithmes d'une signature cible [DKLs18; CCLS+19; CEN25]. En



(a) Échantillonnage gaussien. Les signatures sont échantillonnées selon une gaussienne discrète dans le coset du réseau défini par la cible (**z** est échantillonné autour de 0 tel que $f(\mathbf{z}) = \text{cible}$). Cette méthode produit les signatures les plus compactes, mais est aussi la plus complexe à implémenter.



(b) Échantillonnage par rejet. Les signatures candidates sont rejetées de manière probabiliste afin que la distribution finale soit "Acceptée" — indépendante du décalage $c \cdot \mathbf{s}$. Plus simple, mais la distribution est plus large.



(c) Inondation de bruit. Une distribution de bruit de base massive est utilisée pour masquer statistiquement le décalage $c \cdot \mathbf{s}$ lié à la clé secrète. Bien que conceptuellement plus simple, cette technique produit des distributions plus larges pour garantir la sécurité.

Figure 2.1: Comparaison visuelle des différentes techniques de génération de signatures.

théorie, on peut ainsi concevoir un protocole permettant à plusieurs parties de calculer conjointement n'importe quelle fonction sur leurs entrées sans révéler d'information sur les secrets ou les valeurs intermédiaires, tout en garantissant la correction du calcul. En pratique, cette approche est généralement trop inefficace pour les dimensions et opérations des signatures sur réseaux, notamment l'échantillonnage gaussien, pour échantillonner des distributions courtes, ou encore les comparaisons.

Remark 2.4.1. D'autres outils de preuves composables pour le calcul multi-partite sécurisé ont été proposés au fil des années [Mau10; HS15; KTR20]. Toutefois, le défi principal de l'efficacité demeure.

PROTOCOLES SUR MESURE ET SÉCURITÉ PAR JEUX. Une autre direction consiste à définir des notions de sécurité *sur mesure* (formalisées par des jeux de sécurité) correspondant aux propriétés

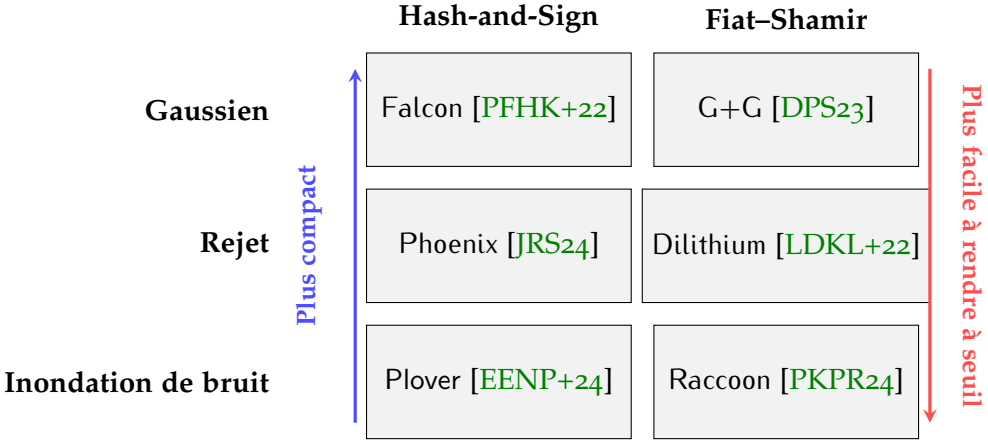


Figure 2.2: Espace de conception des schémas de signature sur réseaux. Nous incluons des schémas représentatifs pour chaque paradigme et technique d'échantillonnage.

souhaitées des signatures à seuil, puis à concevoir des protocoles spécifiques pour chaque étape de la signature. Cette approche permet d'améliorer fortement l'efficacité en tirant parti de propriétés propres aux réseaux et en acceptant certaines fuites contrôlées ou opérations imparfaites qui ne compromettent pas la sécurité. C'est notamment l'approche de [DKMM+24], qui distribue un processus de signature fondé sur l'inondation de bruit : chaque partie échantillonne localement du bruit, calcule une signature partielle bruitée, puis ces signatures partielles sont agrégées pour former une signature valide. En analysant soigneusement quelles informations peuvent être révélées sans danger durant l'exécution du protocole, et en adaptant les paramètres afin de supporter des signatures agrégées plus grandes, cette approche obtient une efficacité nettement supérieure à celle du MPC générique.

Cette approche repose crucialement sur la possibilité de décomposer l'algorithme de signature en opérations simples qui se distribuent de façon sûre. Actuellement, les algorithmes les plus adaptés à une telle décomposition sont ceux reposant sur l'inondation de bruit, qui dépendent principalement d'échantillonnage de vecteurs courts. Plusieurs travaux ont appliqué ces idées à des signatures fondées sur l'échantillonnage par rejet [BTT22], en effectuant localement (chez chaque signataire) l'échantillonnage de vecteurs courts et le rejet, mais avec une efficacité sensiblement moindre. La conception à seuil pratique d'un schéma fondé sur l'échantillonnage gaussien reste une question ouverte.

Au-delà de l'efficacité, plusieurs défis subsistent pour obtenir des signatures à seuil sur réseaux réellement pratiques : obtenir la robustesse face à des parties malveillantes, supporter des abandons identifiables afin d'améliorer l'efficacité dans de nombreux scénarios, et concevoir des schémas conformes aux standards, compatibles avec les algorithmes de vérification et formats de clés existants.

2.5 CONTRIBUTIONS DE CETTE THÈSE

Cette thèse résout plusieurs questions ouvertes dans la conception de signatures à seuil sur réseaux pratiques. Nous introduisons de nouvelles techniques permettant d'enrichir des signatures basées sur l'inondation de bruit avec des propriétés de robustesse et d'abandon identifiable dont elles étaient dépourvues, et nous surmontons les obstacles qui empêchaient de distribuer efficacement des algorithmes de signature basés sur l'échantillonnage par rejet, aboutissant au premier schéma de signature à seuil compatible avec le standard ML-DSA. Tout au long de la thèse, nous instancions nos techniques sur deux schémas représentatifs : Raccoon [PKPR24], une signature Fiat–Shamir basée sur l'inondation de bruit et identifiée comme particulièrement adaptée aux constructions à seuil dans [DKMM+24], et Dilithium (ML-DSA) [LDKL+22], la signature standardisée par le NIST reposant sur la technique d'échantillonnage par rejet. Bien que nos techniques puissent être appliquées à la variante à seuil en deux tours de Raccoon [EKT24] afin d'obtenir la robustesse et des abandons identifiables[1], nous nous concentrons sur le protocole en trois tours de [DKMM+24] afin de clarifier l'exposé, les schémas en deux tours impliquant une phase de signature plus complexe, orthogonale aux idées principales introduites dans cette thèse. Plus précisément, nos contributions principales sont les suivantes :

1. **Génération distribuée de clés et signatures robustes sur réseaux euclidiens.** Nous introduisons les premiers protocoles robustes de génération distribuée de clés et de signature à seuil basés sur l'inondation de bruit. Ces protocoles garantissent la production d'une signature valide même en présence de parties malveillantes. Nous nous appuyons sur un nouveau mécanisme léger de partage de secret vérifiable qui évite des preuves à divulgation nulle coûteuses en exploitant de simples relations linéaires bruitées. Nous prouvons

1 Bien que nous ayons choisi de ne pas le décrire dans cette thèse, la soumission au NIST Hermine [BCPE+26a] consiste à appliquer nos techniques d'abandons identifiables au cadre en deux tours basé sur [EKT24].

sa sécurité sous l'hypothèse Hint-MLWE et intégrons ces vérifications au protocole de signature afin d'assurer la correction. Cela montre, pour la première fois, qu'une robustesse complète peut être atteinte pour des signatures à seuil sur réseaux en n'utilisant que des briques de base. Ceci est détaillé dans le chapitre 4.

2. **Protocoles efficaces avec abandons identifiables.** Alors que la robustesse complète nécessite une majorité honnête pendant la signature et peut être coûteuse en pratique, des garanties plus faibles suffisent dans de nombreux scénarios réalistes. Dans ce cas, les parties peuvent détecter un comportement malveillant et identifier les responsables lors d'un abandon, permettant au système de se rétablir rapidement sans le surcoût d'une robustesse totale.

   Nous développons de nouvelles techniques de partage de secret garantissant la petitesse des parts et des coefficients de reconstruction, ainsi que la reconstruction de secrets partiels de faible norme. Cela permet une conception particulièrement simple : nous concevons un schéma à seuil basé sur l'inondation de bruit dans lequel les parts de signature sont elles-mêmes des signatures valides sous les clés secrètes partielles correspondantes, et permettent de détecter les comportements malveillants simplement en vérifiant ces parts. Cela conduit à des protocoles très efficaces de génération distribuée de clés et de signature avec abandons identifiables. Ceci est détaillé dans le chapitre 5.

3. **Signatures à seuil pratiques pour Dilithium (ML-DSA).** Enfin, nous levons l'obstacle principal qui empêchait la conception de schémas à seuil compétitifs s'appuyant sur des signatures basées sur l'échantillonnage par rejet. Nous développons de nouvelles techniques de simulation montrant que *les signatures rejetées dans des signatures Fiat–Shamir-with-aborts peuvent être révélées sans risque pour des paramètres pratiques*. Combinées à nos techniques de partage de secrets courts et à des optimisations ciblées, elles mènent au *premier schéma de signature à seuil pratique pour Dilithium (*ML-DSA*)*, entièrement compatible avec l'algorithme de vérification standardisé et les formats de clés, et fonctionnant pour un petit nombre de parties ($N \leqslant 6$ pour une communication d'au plus quelques Mo). Ceci est détaillé dans le chapitre 6.

Pris ensemble, ces résultats comblent plusieurs lacunes importantes sur la voie de signatures à seuil post-quantiques pratiques. Ils renforcent la position des TSS sur réseaux basés sur l'inondation de bruit comme solution polyvalente et efficace, y compris pour un grand nombre de parties, et supportant des propriétés avancées telles que la robustesse et les abandons identifiables. Ils montrent aussi, pour la première fois, que des signatures basées sur l'échantillonnage par rejet peuvent être rendues à seuil efficacement, atteignant une plus grande compacité, la compatibilité avec les standards et une efficacité compétitive pour un petit nombre de parties, même si des propriétés avancées comme la robustesse restent ouvertes pour cette approche.

Nous soulignons que nos techniques sous-tendent plusieurs schémas proposés en réponse à l'appel à propositions du NIST (MPTC) [NIST-IR8214C], ce qui illustre la pertinence de nos contributions :

- **Amber [BPLP26] :** Amber est une proposition efficace de KEM IND-CCA à seuil sur réseaux euclidiens, qui intègre nos techniques d'identification d'abandons présentées dans le chapitre 5.

- **Hermine [BCPE+26a] :** Hermine est un schéma de signature à seuil partiellement non interactif en deux tours basé sur Raccoon, qui applique nos techniques d'identification d'abandons du chapitre 5 à un protocole à deux tours et ajoute un support du rafraîchissement de clés.

- **Mithril [CDPE+26] :** Mithril est un schéma de signature à seuil pour ML-DSA qui s'appuie sur nos techniques du chapitre 6 afin d'obtenir la compatibilité au standard ML-DSA pour un petit nombre de parties.

2.6 PUBLICATIONS ASSOCIÉES

Plusieurs résultats présentés dans cette thèse ont été développés dans le cadre d'articles en cours d'évaluation ou publiés dans des conférences avec comité de lecture. Ci-dessous figure la liste de ces publications, ainsi que les chapitres correspondants dans la thèse :

- **Flood and Submerse: Distributed Key Generation and Robust Threshold Signature from Lattices [ENP24].** Cet article étudie la conception de protocoles robustes de génération distribuée de clés et de signature à seuil sur réseaux basés sur l'inondation de bruit. Il introduit un nouveau mécanisme de partage de secret vérifiable et l'intègre à un protocole de signature à seuil pour le schéma de signature Plover [EENP+24] afin d'assurer la correction même en présence de parties déviantes. Ce travail est présenté dans le chapitre 4, avec une adaptation au schéma Raccoon [PKPR24] pour la cohérence avec le reste de la thèse.

  Cet article a été publié dans les actes de CRYPTO 2024.

- **Simple and Efficient Lattice Threshold Signatures with Identifiable Aborts [dENP25].** Cet article introduit de nouvelles techniques de partage de secret qui échantillonnent des parts courtes et permettent la reconstruction de secrets partiels de faible norme. Il conçoit des protocoles efficaces de génération distribuée de clés et de signature avec *abandons identifiables* pour des signatures basées sur l'inondation de bruit en s'appuyant sur ces techniques. Le formalisme introduit dans ce travail, ainsi que le *partage de secret répliqué* à parts courtes, sont présentés dans le chapitre 5. Nous étendons également ce chapitre avec un schéma de *partage de secret de Vandermonde* qui passe à l'échelle pour un plus grand nombre de parties. Cette contribution provient d'une réécriture récente du préprint original [BCPE+26b], qui s'appuie sur les mêmes idées de base pour introduire Hermine, un schéma de signature à seuil en deux tours basé sur Raccoon qui supporte le rafraîchissement de clés et les abandons identifiables. Nous nous concentrons sur le schéma en trois tours original dans cette thèse pour la clarté de l'exposition.

  Ce travail est disponible sur ePrint.

- **Finally! A Compact Lattice-Based Threshold Signature [PN25].** Cet article résout l'obstacle principal qui empêchait la conception de signatures à seuil pratiques basées sur la technique d'échantillonnage par rejet. Il introduit de nouvelles techniques de simulation permettant de révéler sans risque des signatures rejetées dans des schéma de signatures Fiat–Shamir-with-aborts, et les combine à des techniques de partage de secrets courts pour concevoir la signature à seuil la plus compacte à ce jour, passant à l'échelle jusqu'à 8 parties. Ses résultats de simulation constituent le noyau théorique du chapitre 6.

  Cet article a été publié dans les actes de PKC 2025.

- **Efficient Threshold ML-DSA [CPEN+26].** Cet article présente le premier schéma de signature à seuil pratique pour Dilithium (ML-DSA), entièrement compatible avec l'algorithme de vérification standardisé et ses formats de clés. Il s'appuie sur les techniques de simulation introduites dans [PN25] et les combine à des optimisations ciblées afin d'obtenir une efficacité compétitive pour un petit nombre de parties. Ce travail est présenté dans le chapitre 6. Notons que cet article contient également une génération distribuée de clés, ainsi qu'un partage *a posteriori* de clés ML-DSA existantes, que nous avons choisi de ne pas présenter dans cette thèse par souci de concision.

Ce travail sera publié dans les actes de USENIX Security 2026.

2.7 PUBLICATIONS SUPPLÉMENTAIRES

En plus des travaux directement liés à cette thèse, nous avons contribué à plusieurs autres publications en cryptographie :

- **Plover: Masking-Friendly Hash-and-Sign Lattice Signatures [EENP+24].** Cet article-ci présente Plover, le premier schéma de signature hash-and-sign basé sur les réseaux de type lattice et conçu pour être compatible avec le masquage. En s'appuyant sur les idées de Raccoon [PKPR24] pour résister aux attaques par canaux auxiliaires, il propose un cadre systématique pour analyser leur sécurité et montre leur applicabilité aux signatures hash-and-sign. En utilisant la technique d'inondation de bruit, Plover atteint une complexité quasi-linéaire en fonction du nombre de parts de masquage. Plover peut être vu comme l'équivalent hash-and-sign de Raccoon, et nous montrons dans [ENP24] comment les techniques de seuil développées dans Chapter 4 pour les signatures avec inondation de bruit s'appliquent naturellement pour obtenir une signature robuste.

  Ce travail a été publié dans les actes de EUROCRYPT 2024.

- **Unmasking TRaccoon: A Lattice-Based Threshold Signature with An Efficient Identifiable Abort Protocol [PKNR+25].** Dans l'esprit de nos contributions du chapitre 5, ce travail vise à enrichir le schéma de signature à seuil TRaccoon avec un mécanisme d'abandon identifiable. Nous adoptons une approche différente en préservant la conception originale de TRaccoon et en introduisant un module léger qui exécute des preuves à divulgation nulle afin d'identifier les signataires malveillants uniquement en cas d'échec. Cela induit un surcoût de communication de $60 + 6.4|T|$ KB lorsque le protocole échoue. Nous fournissons également la première analyse formelle de sécurité d'une variante à divulgation nulle de LaBRADOR et introduisons de nouveaux jeux de sécurité pour formaliser des protocoles interactifs d'abandon identifiable, étendant les définitions standards d'inforgeabilité.

  Ce travail a été publié dans les actes de CRYPTO 2025.

- **Share the MAYO: thresholdizing MAYO [CEN25].** Dans cet article, nous étudions la conception à seuil de signatures pratiques basées sur le principe *Oil-and-Vinegar* (OV), en nous concentrant en particulier sur MAYO et UOV, candidats dans le processus du NIST pour la standardisation de schémas de signature supplémentaires. Notre approche commence par traiter les difficultés liées à rendre à seuil l'algorithme qui échantillonne des solutions de systèmes linéaires de la forme $\mathbf{A}x = y$, fondamentaux pour les signatures OV. Les tentatives précédentes introduisaient des niveaux de fuite que nous jugeons non sûrs. Nous proposons une nouvelle solution, discutons sa sécurité, et évaluons sa praticabilité. Enfin, nous explorons la conception à seuil de la totalité de la fonctionnalité de signature de ces schémas.

  Ce travail a été publié dans les actes de PQCrypto 2025.

- **Comprehensive Deniability Analysis of Signal Handshake Protocols: X3DH, PQXDH to Fully Post-Quantum with Deniable Ring Signatures [KNTW25].** Le protocole Signal s'appuie sur une phase d'"handshake" (anciennement X3DH, aujourd'hui PQXDH) pour établir des conversations sûres, et valorise la *déniabilité* afin que les utilisateurs puissent nier leur participation. Les analyses antérieures reposent sur des modèles ad hoc variés, qui brouillent les garanties et rendent les comparaisons difficiles.

  En nous appuyant sur l'abstraction de Hashimoto et al. [HKW25], nous présentons un cadre unifié pour analyser la déniabilité du handshake de Signal. Nous étudions X3DH

et PQXDH, en clarifiant la déniabilité de PQXDH face à des adversaires quantiques de type *harvest-now–judge-later*. Nous analysons aussi des alternatives post-quantiques telles que RingXKEM, basées sur des signatures de cercle. En introduisant une métrique de déniabilité inspirée de la confidentialité différentielle, nous obtenons des garanties relâchées mais pragmatiques. Cette métrique nous permet en outre de définir des *signatures de cercle déniables* (un affaiblissement de la notion traditionnelle d'anonymat), permettant des constructions efficaces à partir du standard Falcon du NIST et du candidat à la standardisation MAYO, qui sont déniables bien que pas pleinement anonymes.

Ce travail a été publié dans les actes de USENIX Security 2025.

- **Practical Deniable Post-Quantum X3DH: A Lightweight Split-KEM for K-Waay [Ni025].** Le protocole Signal fait face au défi de migrer vers un monde post-quantique tout en préservant ses propriétés critiques. Si PQXDH offre une confidentialité post-quantique, la migration complète du handshake X3DH reste difficile. Comme étudié dans l'article [KNTW25], l'approche principale pour migrer X3DH consiste à utiliser des signatures de cercle pour obtenir la déniabilité. K-Waay [CHNR+24] explore une voie alternative via des *split KEMs*, mais souffre de limitations de taille par rapport aux approches basées sur les signatures de cercle.

  Ce travail introduit Sparrow-KEM et Sym-Sparrow-KEM, deux nouveaux *split KEMs*, respectivement asymétrique et symétrique, conçus pour optimiser K-Waay. En exploitant l'hypothèse MLWE, nous réduisons la communication d'un facteur 5.1× et améliorons la vitesse d'un facteur 40× par rapport à des split KEMs antérieurs. Sym-Sparrow-KEM est le premier split-KEM symétrique à offrir la déniabilité tout en garantissant de fortes propriétés d'authentification implicite (IND-1KCA, IND-1BatchCCA). Nos résultats montrent la faisabilité d'un X3DH post-quantique compact et déniable basé sur des split KEMs.

  Ce travail a été publié dans les actes de ASIACCS 2025.

- **Subversion-resilient Key-exchange in the Post-quantum World [DFJN+25].** Les *Authenticated Key-Exchange* (AKE) résistants à la subversion garantissent la sécurité même lorsque certaines parties de l'implémentation du protocole sont altérées. Une façon d'obtenir de tels AKE consiste à utiliser des *Reverse Firewalls* (RF) pour restaurer la sécurité.

  Dans ce travail, nous étendons la résistance à la subversion basée sur les RF au niveau des définitions de sécurité, des constructions et de la vérification formelle. Tout d'abord, nous introduisons un relâchement utile de la notion de sécurité pour les AKE résistants à la subversion avec RF : l'objectif n'est plus d'empêcher toute exfiltration, mais plutôt de restaurer au protocole AKE une propriété perdue lors de la subversion. Nous nous concentrons en particulier sur l'authentification et la (sécurisation de) clés des RF, et considérons un spectre de compromissions, en conçevant un cadre où l'adversaire peut altérer certains composants de l'implémentation mais pas nécessairement les autres. Dans une optique post-quantique, nous définissons des *Key Encapsulation Mechanisms re-randomizables* et proposons des instanciations basées sur Diffie–Hellman classique et Kyber. Enfin, nous posons les bases de la vérification formelle de protocoles basés sur les RF, en prouvant la sécurité de notre construction à l'aide du prouveur CryptoVerif, en plus de preuves de sécurité computationnelle selon la méthodologie classique de Bellare–Rogaway.

  Ce travail a été publié dans les actes de CCS 2025.

- **Adaptively-Secure Three-Round Threshold Schnorr from DL [NRT26].** Ce travail présente le premier schéma de signature Schnorr à seuil sécurisé contre les corruptions adaptatives et nécessitant seulement trois tours (deux en ligne, un hors ligne) sous l'hypothèse du logarithme discret (DL) dans le modèle de l'oracle aléatoire. Les protocoles Schnorr à seuil adaptativement sécurisés précédents reposaient soit sur des hypothèses plus fortes, soit

nécessitaient cinq tours. Ce schéma préserve l'interface standard des signatures Schnorr tout en minimisant la latence. Ce travail introduit de nouvelles techniques, notamment un engagement équivoque avec NIZK extractible par simulation et une stratégie d'ouverture pour des engagements homomorphes agrégés basée sur une technique de masquage, ainsi que fournit une formalisation d'une notion forte de sécurité adaptative.

Ce travail sera publié dans les actes de EUROCRYPT 2026.

- **Revisiting PQ WireGuard: A Comprehensive Security Analysis With a New Design Using Reinforced KEMs [HKNW25].** WireGuard est un VPN très performant basé sur le protocole Noise. Une variante post-quantique (PQ) a été proposée par Hülsing et al. [HNSW+21] ; cependant, comme WireGuard impose que le message de handshake tienne dans un seul paquet UDP d'environ 1200 B, ils s'appuient sur Classic McEliece, dont les grandes clés publiques augmentent fortement la mémoire côté serveur et compliquent un déploiement au niveau noyau.

  Dans ce travail, nous réexaminons PQ WireGuard afin d'en améliorer la conception, la sécurité et l'efficacité. Nous traitons des problèmes de *binding* dans des KEM post-quantiques et prouvons la sécurité dans un nouveau modèle computationnel. Nous introduisons les *reinforced KEM* (RKEM) ainsi qu'une construction nommée *Rebar* pour compresser des chiffrés de type ML-KEM. Cela permet un protocole WireGuard post-quantique où le serveur évite de stocker de grandes clés, réduisant l'empreinte mémoire liée aux clés publiques d'un facteur 190 à 390×.

  Ce travail sera publié dans les actes de S&P 2026.

3 BACKGROUND AND PRELIMINARIES

3.1 NOTATIONS

GENERAL NOTATIONS. The notation $a := b$ means that a is defined to be equal to b. We denote by $\mathbb{R}$ the set of real numbers, $\mathbb{Q}$ the set of rational numbers, $\mathbb{Z}$ the set of integers, and $\mathbb{N}$ the set of non-negative integers. For a probabilistic function f (resp. distribution $\mathcal{D}$), we write $y \leftarrow f(x_1, \ldots, x_k)$ (resp. $y \leftarrow \mathcal{D}$) to denote that y is the output of f on inputs $x_1, \ldots, x_k$ (resp. y is a sample from $\mathcal{D}$). For a set S, we denote by $\mathcal{U}(S)$ the uniform distribution over S, and write $x \xleftarrow{\$} S$ as shorthand for $x \leftarrow \mathcal{U}(S)$. For $N \in \mathbb{N}$, we denote $[N] := \{1, 2, \ldots, N\}$.

PROBABILITY AND ALGORITHMS. Let $\lambda \in \mathbb{N}$ be the security parameter. A function $f(\lambda)$ is negligible (or $f(\lambda) = \mathsf{negl}(\lambda)$) in λ if it is $\mathcal{O}(\lambda^{-c})$ for every constant $c > 0$. We denote by $\mathsf{Time}(\mathcal{A})$ the running time of an algorithm $\mathcal{A}$. A probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$ is an algorithm whose running time $\mathsf{Time}(\mathcal{A})$ is upper bounded by a polynomial in the security parameter λ. For some algorithm $\mathcal{A}$, we write $\mathcal{A}^{\mathcal{O}}$ to denote that $\mathcal{A}$ has oracle access to the oracle $\mathcal{O}$.

ALGORITHMIC NOTATIONS. We use **req** and **assert** in algorithms as a shortcut to check if a proposition x is false and abort the execution if so. More precisely, **req** stops the execution of the local algorithm and returns $\perp$ to the caller, while **assert** stops the execution up to the top-level caller and returns 0 as the output of the entire algorithm. **assert** is in particular useful in security games to denote conditions that must be satisfied by the adversary to avoid losing the game.

ALGEBRA AND REPRESENTATION OF $\mathbb{Z}_q$ ELEMENTS. Let $q \in \mathbb{N}$. We denote by $\mathbb{Z}_q$ the ring of integers modulo q, and by $\mathbb{Z}_q^n$ the set of length-n vectors over $\mathbb{Z}_q$. For an element $x \in \mathbb{Z}_q$, we write $\mathsf{lift}(x)$ for its canonical representative in the interval $(-\frac{q}{2}, \frac{q}{2}]$. All additions, subtractions, and scalar multiplications over $\mathbb{Z}_q$ are taken modulo q unless stated otherwise.

VECTORS AND MATRICES. Vectors are written in bold lowercase (e.g., $\mathbf{x} \in \mathbb{Z}_q^n$) and matrices in bold uppercase (e.g., $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$). We extend the lift notation to vectors and matrices by applying it component-wise. For a vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{Z}^n$, we denote its Euclidean norm by $\|\mathbf{x}\|_2$ and its infinity norm by $\|\mathbf{x}\|_\infty$. This extends to $\mathbb{Z}_q$ vectors by considering their lifted representatives. For a matrix $\mathbf{A}$ (resp. a vector $\mathbf{x}$), we denote by $\mathbf{A}^\top$ (resp. $\mathbf{x}^\top$) its transpose. The product $\mathbf{Ax}$ denotes standard matrix–vector multiplication over $\mathbb{Z}_q$ (or over $\mathbb{Z}$, when using lifted representatives). The scalar product of two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^n$ is denoted by $\langle \mathbf{x}, \mathbf{y} \rangle$. The spectral norm of a matrix $\mathbf{A}$ is denoted by $\|\mathbf{A}\|_2$ and corresponds to its largest singular value. The identity matrix of size k is denoted by $\mathbf{I}_k$ (or simply $\mathbf{I}$ when the size is clear from the context).

DISCRETE GAUSSIAN DISTRIBUTIONS. We will make use of discrete Gaussian distributions throughout this thesis. Given a positive definite matrix $\mathbf{\Sigma} \in \mathbb{R}^{n \times n}$, we define the Gaussian function $\rho_{\sqrt{\mathbf{\Sigma}}} : \mathbb{R}^n \to \mathbb{R}$ as

$$\rho_{\sqrt{\mathbf{\Sigma}}}(\mathbf{x}) := \exp\left(-\frac{\mathbf{x}^\top \mathbf{\Sigma}^{-1} \mathbf{x}}{2}\right).$$

Remark 3.1.1. Note that two different conventions are commonly used in the literature for defining Gaussian functions and distributions, either with a factor of $1/2$ [DDLL13] or with a factor

of π [MP12; APS15] in the exponent. We choose the convention with a factor of $1/2$ in the exponent, as it leads to a more direct relationship between the parameter $\mathbf{\Sigma}$ and the covariance of the distribution. Specifically, under this convention, the covariance matrix of the corresponding continuous Gaussian distribution is exactly $\mathbf{\Sigma}$, and this relationship approximately holds for the discrete Gaussian distribution as well.

We denote by $D_{\mathbb{Z}^n,\sqrt{\mathbf{\Sigma}},\mathbf{c}}$ the discrete Gaussian distribution over $\mathbb{Z}^n$ with parameter $\sqrt{\mathbf{\Sigma}}$ and center $\mathbf{c} \in \mathbb{R}^n$, i.e., the distribution that outputs each element $\mathbf{x} \in \mathbb{Z}^n$ with probability proportional to $\rho_{\sqrt{\mathbf{\Sigma}}}(\mathbf{x} - \mathbf{c})$. When $\mathbf{\Sigma} = \sigma^2 \mathbf{I}_n$ for some standard deviation $\sigma > 0$, we simply write $D_{\mathbb{Z}^n,\sigma}$ instead of $D_{\mathbb{Z}^n,\sqrt{\mathbf{\Sigma}}}$. We may even write D_σ instead of $D_{\mathbb{Z}^n,\sigma}$ when it is clear from context.

We may write $\rho_\sigma(S) := \sum_{\mathbf{x}\in S} \rho_{\sigma\mathbf{I}_n}(\mathbf{x})$ for any countable set $S \subseteq \mathbb{Z}^n$ such that this sum is finite.

3.2 LATTICES

This section recalls basic definitions and properties related to lattices. A reader familiar with lattices may skip to the problems we will rely on in this thesis, Section 3.2.3.

A *lattice* Λ is a discrete additive subgroup of $\mathbb{R}^m$. Any lattice can be represented as the set of all integer linear combinations of the columns of a full-rank matrix $\mathbf{B} \in \mathbb{R}^{m\times n}$, with $n \leqslant m$, called a *basis* of the lattice:

$$\Lambda = \mathcal{L}(\mathbf{B}) := \{\mathbf{Bx} : \mathbf{x} \in \mathbb{Z}^n\}.$$

Remark 3.2.1. n is called the *dimension* of the lattice. A lattice can have multiple bases, and the choice of basis can affect the efficiency of algorithms that operate on the lattice.

We provide a visual example of a two-dimensional lattice in Fig. 3.1.

Several problems over lattices are believed to be hard to solve, even for quantum computers, and form the basis of lattice-based cryptography.

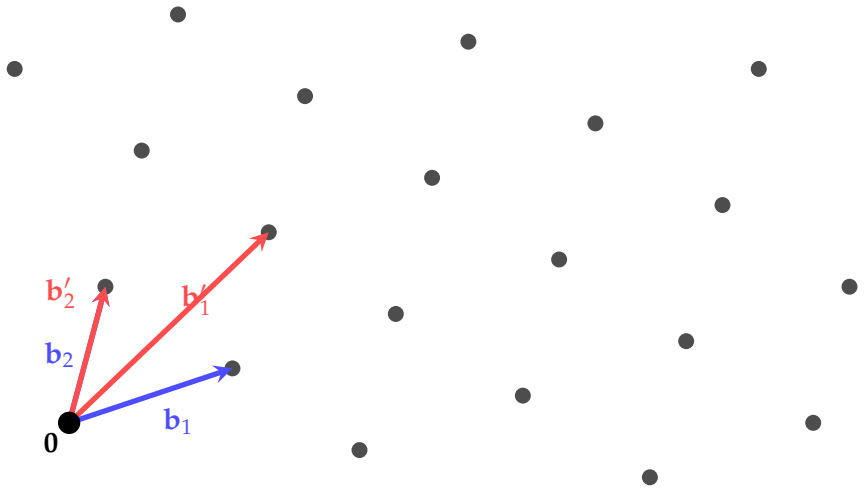


Figure 3.1: A two-dimensional lattice Λ. Two different bases $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2]$ (blue) and $\mathbf{B}' = [\mathbf{b}'_1, \mathbf{b}'_2]$ (red) generate the same lattice.

We recall a few useful notions related to lattices. The *dual lattice* of Λ is defined as

$$\Lambda^* := \{\mathbf{y} \in \mathbb{R}^n : \forall \mathbf{x} \in \Lambda, \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}\}.$$

The *smoothing parameter* $\eta_\varepsilon(\Lambda)$ of a lattice Λ for some $\varepsilon > 0$ is defined as the smallest real $s > 0$ such that $\rho_{1/(\sqrt{2\pi}s)\mathbf{I}_n}(\Lambda^* \backslash \{0\}) \leqslant \varepsilon$. We also define a scaled variant $\eta'_\varepsilon(\Lambda) := \frac{1}{\sqrt{2\pi}}\eta_\varepsilon(\Lambda)$, corresponding to Gaussians defined with a factor $1/2$ in the exponent – as used in this thesis – instead of π.

3.2.1 Plain Lattice Problems

The breakthrough work of Ajtai [Ajt96] established the first worst-case to average-case reduction for lattices. His reduction shows that solving certain average-case problems on finding short vectors in random lattices is at least as hard as approximating the Shortest Vector Problem (SVP) – recalled below – in the worst case. This work introduced the core idea behind lattice-based cryptography: provable security via reductions from worst-case lattice problems.

Definition 3.2.2 (Shortest Vector Problem (SVP)). *Given a lattice Λ, the goal of the Shortest Vector Problem is to find a non-zero vector $\mathbf{v} \in \Lambda$ such that $\|\mathbf{v}\|_2$ is minimized, i.e., find*

$$\mathbf{v} \in \Lambda \backslash \{0\} \text{ such that } \|\mathbf{v}\|_2 = \lambda_1(\Lambda) := \min_{\mathbf{x} \in \Lambda \backslash \{0\}} \|\mathbf{x}\|_2.$$

Building on Ajtai's ideas, Micciancio and Regev later introduced the modular Short Integer Solution (SIS) problem [MR04], which has since become a central hardness assumption in lattice-based cryptography and is amenable to constructing various cryptographic primitives. In contrast to Ajtai's original average-case formulation–where the norm bound depends on the geometry of the underlying lattice (e.g., proportional to $\lambda_1(\Lambda)$)–the SIS problem is defined over a random matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and asks for a nonzero integer vector $\mathbf{x} \in \mathbb{Z}_q^n$ of fixed bounded norm satisfying the modular equation $\mathbf{A}\mathbf{x} = 0$. The set of solutions to $\mathbf{A}\mathbf{x} = 0$ forms a lattice, and the SIS problem asks for a short non-zero vector in this lattice. Micciancio and Regev showed that SIS also enjoys worst-case to average-case reductions.

Definition 3.2.3 (Short Integer Solution (SIS) Problem). *Given parameters $n, m, q \in \mathbb{N}$ and a bound $\beta \in \mathbb{R}$, the Short Integer Solution (SIS) problem is defined as follows: Given a uniformly random matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$, find a non-zero vector $\mathbf{x} \in \mathbb{Z}^n$ such that*

- $\mathbf{A}\mathbf{x} = 0$,
- $\|\mathbf{x}\|_2 \leqslant \beta$.

A year later, Regev introduced the Learning With Errors (LWE) problem [Reg05], which has become another foundational assumption in lattice-based cryptography. The LWE problem asks one to distinguish noisy linear samples from uniformly random ones and enjoys worst-case to average-case reductions from "natural" lattice problems.

Definition 3.2.4 (Learning With Errors (LWE) Problem). *Given parameters $n, m, q \in \mathbb{N}$ and a secret (resp. error) distribution $\chi_{\mathbf{s}}$ (resp χ_e) over $\mathbb{Z}_q$, the Learning With Errors (LWE) problem is defined as follows: One is given m samples of the form $(\mathbf{a}_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, and must distinguish between the two cases:*

- ***LWE distribution:*** *$\mathbf{a}_i \xleftarrow{\$} \mathbb{Z}_q^n$ uniformly and $b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \bmod q$ for a secret $\mathbf{s} \xleftarrow{\$} \chi_{\mathbf{s}}^n$ and error $e_i \xleftarrow{\$} \chi_e$.*
- ***Uniform distribution:*** *$(\mathbf{a}_i, b_i)$ is sampled uniformly from $\mathbb{Z}_q^n \times \mathbb{Z}_q$.*

3.2.2 Module Lattices and Problems

Module lattices [LS15] are special types of lattices formed by considering modules over polynomial rings. They have become widely used in lattice-based cryptography due to their efficiency and structured properties, and underlie the recent NIST standards ML-KEM, ML-DSA and FN-DSA.

CYCLOTOMIC RINGS. Let ϕ be a cyclotomic polynomial, i.e. the minimal monic polynomial with integer coefficients whose roots are the primitive m-th roots of unity for some $m \in \mathbb{N}$. The ring $R = \mathbb{Z}[X]/(\phi(X))$ is called a cyclotomic ring, and is the ring of integers of the cyclotomic number field $K = \mathbb{Q}[X]/(\phi(X))$. The degree of ϕ (and thus of K) is denoted by n.

MODULE LATTICES. A subset $M \subseteq K^k$ is called an R-module if it is closed under addition and multiplication by elements of R. M is said to be discrete and have *rank* k if it is generated by k elements as an R-module, i.e., there exist $m_1, \ldots, m_k \in M$ linearly independent over R such that every element of M can be expressed as an R-linear combination of these generators.

An *R-module lattice* Λ (or simply a *module lattice*) is the image of a discrete R-module $M \subseteq R^k$ under an embedding $\sigma : K^k \to \mathbb{C}^{k \cdot n}$:

$$\Lambda = \sigma(M) \subset \mathbb{R}^{k \cdot n}$$

Remark 3.2.5. Note that if M is of rank k, then Λ is a lattice of dimension $n \cdot k$.

Two common embeddings are used in the literature, and all previous notations are extended to module elements using it, including norms and discrete Gaussians, by applying the embedding to the module elements. These embeddings are:

- The *canonical embedding*, which maps an element $a(X) \in K$ to the vector obtained by evaluating $a(X)$ at the n roots of $\phi(X)$. We extend σ component-wise to K^k, mapping each module element $(a_1, \ldots, a_k) \in K^k$ to $(\sigma(a_1), \ldots, \sigma(a_k)) \in \mathbb{R}^{k \cdot n}$. It is appreciated for its algebraic properties and is commonly used in theoretical analyses.
- The *coefficient embedding*, which maps an element $a(X) = a_0 + a_1 X + \ldots + a_{n-1} X^{n-1} \in K$ to the vector of its coefficients $(a_0, a_1, \ldots, a_{n-1})$. This embedding is often used in practice for its simplicity and efficient implementation. Note that the norm defined using the coefficient embedding is equivalent to the one defined using the canonical embedding up to a scaling factor bounded by $\sqrt{n}$ in the cyclotomic ring $R = \mathbb{Z}[X]/(X^n + 1)$ with n a power of two. Since the distortion between the two norms is polynomial in n, worst-case to average-case reductions proven in the canonical embedding remain valid when using the coefficient embedding.

For the story, people have considered using *ideal lattices* for cryptography, which are a special case of module lattices of rank one. However, efficient attacks against them have been proposed for certain approximation factors [CGS14; DPW19], raising concerns about their security. As a result, the cryptographic community has shifted toward using module lattices of higher rank, which are believed to be resistant to these attacks.

In this thesis, and as common in lattice-based cryptography, we focus on the cyclotomic rings $R = \mathbb{Z}[X]/(X^n + 1)$, where n is a power of two, equipped with the *coefficient embedding*. Further, we focus on module lattices over the quotient ring $R_q = R/qR$ for some integer $q \in \mathbb{N}$. This choice is in particular motivated by the fact that $X^n + 1$ is a cyclotomic polynomial whose roots are the primitive $2n$-th roots of unity, and it allows for efficient polynomial arithmetic using the Number Theoretic Transform (NTT) when working modulo q such that q is chosen appropriately (e.g., prime and satisfying $q \equiv 1 \mod 2n$).

MODULE LATTICE PROBLEMS. The module variants of the SIS and LWE problems, denoted MLWE and MSIS respectively, are defined similarly to their plain lattice counterparts but operate over module lattices. For ease of use, we consider them with regards to the advantage of a PPT adversary, and we include the errors in the secret by prepending the public matrix with the identity.

Definition 3.2.6 (Module Short Integer Solution (MSIS) Problem). *Let k, ℓ be integers and $\beta > 0$ be a real number. The advantage $\mathsf{Adv}^{\mathsf{MSIS}}_{\mathcal{A}}(1^\lambda)$ of an adversary $\mathcal{A}$ against the Module Short Integer Solutions problem $\mathsf{MSIS}_{R_q,k,\ell,\beta}$ is defined as:*

$$\mathsf{Adv}^{\mathsf{MSIS}}_{\mathcal{A}}(1^\lambda) = \Pr\left[\mathbf{A} \xleftarrow{\$} R_q^{k\times\ell},\ \mathbf{z} \leftarrow \mathcal{A}(\mathbf{A}) \ :\ 0 < \|\mathbf{z}\| \leqslant \beta \ \wedge \ \begin{bmatrix}\mathbf{A} & \mathbf{I}_k\end{bmatrix}\mathbf{z} = \mathbf{0} \bmod q\right].$$

The $\mathsf{MSIS}_{R_q,k,\ell,\beta}$ assumption states that any PPT adversary $\mathcal{A}$ has a negligible advantage $\mathsf{Adv}^{\mathsf{MSIS}}_{\mathcal{A}}(1^\lambda)$.

Definition 3.2.7 (Module Learning With Errors (MLWE) Problem). *Let k, ℓ be integers, and $\chi_{\mathbf{s}}$ be a distribution over $R_q^{\ell+k}$. The advantage $\mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{A}}(1^\lambda)$ of an adversary $\mathcal{A}$ against the Module Learning With Errors problem $\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{s}}}$ is defined as:*

$$\mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{A}}(1^\lambda) = \left|\Pr\left[\mathcal{A}\left(\mathbf{A}, \begin{bmatrix}\mathbf{A} & \mathbf{I}_k\end{bmatrix}\cdot\mathbf{s}\right) = 1\right] - \Pr\left[\mathcal{A}\left(\mathbf{A}, \mathbf{u}\right) = 1\right] - \frac{1}{2}\right|,$$

where $\mathbf{A} \xleftarrow{\$} R_q^{k\times\ell}$ uniformly, $\mathbf{s} \xleftarrow{\$} \chi_{\mathbf{s}}$, and $\mathbf{u} \xleftarrow{\$} R_q^k$ uniformly. The $\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{s}}}$ assumption states that any PPT adversary $\mathcal{A}$ has a negligible advantage $\mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{A}}(1^\lambda)$.

Analogously to their plain lattice counterparts, both MSIS and MLWE enjoy worst-case to average-case reductions [LS15].

In addition to the above, the research community has introduced several variants of the SIS (resp. MSIS) and LWE (resp. MLWE) problems to better suit specific cryptographic applications.

3.2.3 The Self-Target MSIS problem

The first problem we recall and will rely on in this thesis is the Self-Target MSIS (SelfTargetMSIS) problem [DKLL+18; KLS18], which is a variant of MSIS where the target vector $\mathbf{z}$ must satisfy a certain relation with respect to a hash function. This problem is particularly useful for proving the security of Fiat-Shamir signatures, as it captures the idea that an adversary must find a short vector that hashes to a specific value determined by the adversary's own query.

Definition 3.2.8 (SelfTargetMSIS). *Let k, ℓ be integers and $\beta_{\mathsf{stmsis}} > 0$ be a real number. Let $\mathcal{C}$ be a subset of R_q, and let $G : R_q^k \times \{0,1\}^{2\lambda} \mapsto \mathcal{C}$ be a cryptographic hash function modeled as a random oracle. The advantage $\mathsf{Adv}^{\mathsf{SelfTargetMSIS}}_{\mathcal{A}}(1^\lambda)$ of an adversary $\mathcal{A}$ against $\mathsf{SelfTargetMSIS}_{R_q,k,\ell,\beta_{\mathsf{stmsis}}}$ is defined as:*

$$\mathsf{Adv}^{\mathsf{SelfTargetMSIS}}_{\mathcal{A}}(1^\lambda) = \Pr\Bigg[\mathbf{A} \xleftarrow{\$} R_q^{k\times\ell}, (\mathsf{msg}, \mathbf{z}) \leftarrow \mathcal{A}^G(\mathbf{A}), (\mathsf{msg}, \mathbf{z}) \in \{0,1\}^{2\lambda} \times R_q^{\ell+k} : \left(\mathbf{z} = \begin{bmatrix}c\\ \mathbf{z}'\end{bmatrix}\right) \wedge \left(\|\mathbf{z}\|_2 \leqslant \beta_{\mathsf{stmsis}}\right) \wedge G(\begin{bmatrix}\mathbf{A} & \mathbf{I}_k\end{bmatrix}\cdot\mathbf{z}, \mathsf{msg}) = c\Bigg].$$

The $\mathsf{SelfTargetMSIS}_{R_q,k,\ell,\beta_{\mathsf{stmsis}}}$ assumption states that any PPT adversary $\mathcal{A}$ has a negligible advantage $\mathsf{Adv}^{\mathsf{SelfTargetMSIS}}_{\mathcal{A}}(1^\lambda)$.

SelfTargetMSIS is shown to be as hard as MSIS in the random oracle model when the range of the hash function is exponentially large with a standard proof using the forking lemma [FS87; BN06]. There is also a proof in the quantum random oracle model [JMW24].

More precisely, we have the following reduction:

Lemma 3.2.9 (Hardness of SelfTargetMSIS, c.f. [DKMM+24, Lemma B.1]). *For any adversary $\mathcal{A}$ against the $\mathsf{SelfTargetMSIS}_{R_q,k,\ell,\beta_{\mathsf{stmsis}}}$ problem making at most Q_G random oracle queries to the hash function G, there exists an adversary $\mathcal{B}$ against the $\mathsf{MSIS}_{R_q,k,\ell,\beta}$ problem with $\beta = 2\beta_{\mathsf{stmsis}}$ such that*

$$\mathsf{Adv}^{\mathsf{SelfTargetMSIS}}_{\mathcal{A}}(1^\lambda) \leqslant \sqrt{Q_G \cdot \mathsf{Adv}^{\mathsf{MSIS}}_{\mathcal{B}}(1^\lambda)} + \frac{Q_G}{|\mathcal{C}|},$$

where $\mathsf{Time}(\mathcal{B}) \approx 2 \cdot \mathsf{Time}(\mathcal{A})$.

In this thesis, we will directly work with SelfTargetMSIS rather than with MSIS, and will estimate its concrete security based on the best known attacks.

Specifically, we follow the footsteps of the analyses of Dilithium [LDKL+22, Section C.3] and Raccoon [dEKM+23, Section 4.3.5] and evaluate the best known attacks against SelfTargetMSIS as described in [KLS18, Section 4.2], which are either to (i) break the second preimage resistance of the underlying hash function, or (ii) to generate $\mathbf{w}$ at random, compute $c = \mathsf{H}_c(\mathbf{w}, \mathsf{msg})$ and finally to solve the following problem for $\mathbf{z}$:

$$\left(\begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{z} = \mathbf{w} - c \cdot \mathbf{b}\right) \quad \text{and} \quad \left(\|\mathbf{z}\| \leqslant \beta_{\mathsf{stmsis}}\right). \tag{1}$$

Solving (i) is hard as long as $2^{\omega}\binom{n}{\omega} \geqslant 2^{\lambda}$. On the other hand, solving (ii) requires solving Eq. (1), which is an instance of the (inhomogeneous) MSIS problem.

We will then evaluate the best known attacks against MSIS using the open source Lattice Estimator [APS15]. This tool estimates the cost of attacks based on lattice reduction algorithms such as BKZ against lattice problems, including MSIS and MLWE, relying on standard heuristics [SE94; GN08; GNR10; CN11; GJS15; ADPS16; BDGL16], notably on the lattice geometry (e.g., the geometric series assumption [Sch03]). Our concrete security estimates are obtained using the estimator's "rough" cost model, which follows the Core-SVP methodology, standard in the literature: the attack cost is expressed in terms of the block size β required for BKZ and the corresponding cost of solving a single SVP instance in dimension β. In particular, this omits certain polynomial factors such as the number of calls to the SVP oracle, or the cost of the BKZ reduction itself.

3.2.4 The Hint MLWE problem

Next, we recall the Hint MLWE problem [KLSS23]. This variant of MLWE captures the hardness of MLWE when the adversary is given some additional information (the "hint") about the secret and error vector of the form $\mathbf{M} \cdot \mathbf{s} + \mathbf{y}$, with some matrix $\mathbf{M}$ of small spectral norm and noise $\mathbf{y}$. These hints correspond to the noise flooding technique commonly used in lattice-based signatures, where we sample $\mathbf{y}$ from a sufficiently large distribution to mask the actual secret and error vectors. The definition with a matrix $\mathbf{M}$ was introduced in [ENP24], whereas [KLSS23] originally defined Hint-MLWE with hints of the form $c \cdot \mathbf{s} + \mathbf{y}$.

Several other similar assumptions have been introduced in the literature to capture security in the presence of a linear leakage, e.g. *Extended MLWE* [LNS21] which does not add noise to the hints, Algebraic One-More MLWE [EKT25] and Algebraic One-More MSIS [ZT25] which allows to adaptively query hints and to use a combination of reusable noises to mask the secret, and *Leaky LWE* [LSW25] which unifies some of the earlier approaches. While they all provide reductions from standard MLWE, Hint-MLWE appears to provide the tightest reduction, and is perfectly suited to our needs in this thesis.

Definition 3.2.10 (Hint-MLWE). *Let k, ℓ, Q be integers, $\chi_{\mathbf{s}}$, $\left(\chi_{\mathbf{y}}^{(i)}\right)_{i\in[Q]}$ be probability distributions over $R^{\ell+k}$, and $\mathcal{M}$ be a distribution over $\left(R^{(\ell+k)\times(\ell+k)}\right)^{Q}$. The advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hint\text{-}MLWE}}(1^{\lambda})$ of an adversary $\mathcal{A}$ against the Hint Module Learning with Errors problem $\mathsf{Hint\text{-}MLWE}_{R_q,k,\ell,Q,\chi_{\mathbf{s}},(\chi_{\mathbf{y}}^{(i)})_{i\in[Q]},\mathcal{M}}$ is defined as:*

$$\left|\Pr\left[1 \leftarrow \mathcal{A}\left(\mathbf{A}, \begin{bmatrix}\mathbf{A} & \mathbf{I}_k\end{bmatrix} \cdot \mathbf{s}, (\mathbf{M}_i, \mathbf{z}_i)_{i\in[Q]}\right)\right] - \Pr\left[1 \leftarrow \mathcal{A}\left(\mathbf{A}, \mathbf{u}, (\mathbf{M}_i, \mathbf{z}_i)_{i\in[Q]}\right)\right]\right|,$$

where $(\mathbf{A}, \mathbf{u}) \xleftarrow{\$} R_q^{k\times\ell} \times R_q^k$, $\mathbf{s} \leftarrow \chi_{\mathbf{s}}$, $(\mathbf{M}_i)_{i\in[Q]} \leftarrow \mathcal{M}$ and for $i \in [Q]$: $\mathbf{y}_i \leftarrow \chi_{\mathbf{y}}^{(i)}$, and $\mathbf{z}_i = \mathbf{M}_i \cdot \mathbf{s} + \mathbf{y}_i$. The $\mathsf{Hint\text{-}MLWE}_{R_q,k,\ell,Q,\chi_{\mathbf{s}},(\chi_{\mathbf{y}}^{(i)})_i,\mathcal{M}}$ assumption states that any efficient adversary $\mathcal{A}$ has a negligible advantage.

Remark 3.2.11. We may simply write $\mathsf{Hint\text{-}MLWE}_{R_q,k,\ell,Q,\sigma_{\mathbf{s}},(\sigma_{\mathbf{y}}^{(i)})_{i\in[Q]},\mathcal{M}}$ when using discrete Gaussians, i.e. $\chi_{\mathbf{s}} = D_{R^{\ell+k},\sigma_{\mathbf{s}}}, \chi_{\mathbf{y}}^{(i)} = D_{R^{\ell+k},\sigma_{\mathbf{y}}^{(i)}}$.

Interestingly, Hint-MLWE is as hard as standard MLWE when using Gaussian noise and under certain parameter regimes depending on the spectral norm of the matrices sampled from $\mathcal{M}$ and the size of the noise distributions $\chi_{\mathbf{y}}^{(i)}$ [KLSS23; ENP24].

Theorem 3.2.12 (Hardness of Hint-MLWE). *Let k, ℓ, Q, be positive integers, and $\mathcal{M}$ be a distribution over $\left(R^{n(\ell+k)\times n(\ell+k)}\right)^Q$. For $\sigma_{\mathbf{s}}, (\sigma_y^{(i)})_{i\in[Q]} > 0$, let $\sigma > 0$ be a real number defined as $\frac{1}{\sigma^2} = 2\left(\frac{1}{\sigma_{\mathbf{s}}^2} + \sum_{i\in[Q]} \frac{\|\mathbf{M}_i\|_2^2}{(\sigma_y^{(i)})^2}\right)$.*

If $\sigma \geqslant \sqrt{2}\cdot\eta'_\varepsilon(\mathbb{Z}^{n(\ell+k)})$ for some $\varepsilon \in (0, 1/2]$, then there exists an efficient reduction from $\mathsf{MLWE}_{R_q,k,\ell,\sigma,\sigma}$ to $\mathsf{Hint\text{-}MLWE}_{R_q,k,\ell,Q,\sigma_{\mathbf{s}},(\sigma_{\mathbf{y}}^{(i)})_{i\in[Q]},\mathcal{M}}$ that reduces the advantage by at most 2ε.

We will also rely on the Lattice Estimator [APS15] to evaluate the best known attacks against MLWE and Hint-MLWE, leveraging the reduction in Theorem 3.2.12 to translate the parameters of Hint-MLWE to those of MLWE.

3.3 SIGNATURE SCHEMES

Signature schemes are the main cryptographic primitive studied in this thesis, albeit distributed in a threshold manner. A digital signature scheme allows a *single* user to sign messages such that anyone can verify the authenticity of the signature using the corresponding verification key. Crucially, only the holder of the private key can produce valid signatures.

We recall below their syntax and security definitions, which will serve as a basis to define threshold signature schemes in Section 3.4.

A signature scheme is defined by three PPT algorithms:

$\mathsf{Keygen}(1^\lambda) \to (\mathsf{vk}, \mathsf{sk})$. A probabilistic algorithm that takes as input the security parameter λ and outputs a key pair $(\mathsf{vk}, \mathsf{sk})$, where vk is the verification key and sk is the private key;

$\mathsf{Sign}(\mathsf{sk}, \mathsf{msg}) \to \mathsf{sig}$. A (possibly probabilistic) algorithm that takes as input a private key sk and a message $\mathsf{msg} \in \{0,1\}^*$, and outputs a signature sig;

$\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}) \to \{0,1\}$. A deterministic algorithm that takes as input a verification key vk, a message $\mathsf{msg} \in \{0,1\}^*$, and a signature sig, and outputs a bit $b \in \{0,1\}$ indicating whether the signature is valid ($b = 1$) or invalid ($b = 0$).

It must satisfy two main properties of interest: *correctness* and *unforgeability*. *Correctness* guarantees that signatures generated using the signing algorithm will always be accepted by the verification algorithm. *Unforgeability* ensures that an adversary cannot produce new valid signatures without access to the private key.

Definition 3.3.1 (Correctness). *A signature scheme is* correct *if for any valid message $\mathsf{msg} \in \{0,1\}^*$, we have*

$$\Pr\left[(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Keygen}(1^\lambda) : \mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{Sign}(\mathsf{sk}, \mathsf{msg})) = 1\right] \geqslant 1 - \mathsf{negl}(\lambda).$$

Unforgeability can be defined in two main flavors, depending on which signatures are accepted as forgeries: *existential unforgeability* (EUF-CMA) and *strong unforgeability* (SUF-CMA). Both notions consider an adversary $\mathcal{A}$ that has access to a signing oracle $\mathsf{Sign}(\mathsf{sk}, \cdot)$, which allows it to request signatures on messages of its choice. The adversary's goal is to produce a valid signature, either on a new message (for EUF-CMA) or on any message, including those queried to the signing oracle as long as the signature is different (for SUF-CMA).

Definition 3.3.2 (Existential Unforgeability under Chosen Message Attacks (EUF-CMA)). *A signature scheme is* existentially unforgeable under chosen message attacks *if for any PPT adversary $\mathcal{A}$, its advantage*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A}}(1^\lambda) = \Pr\left[\begin{array}{l}(\mathsf{vk},\mathsf{sk}) \leftarrow \mathsf{Keygen}(1^\lambda),\ (\mathsf{msg}^*,\mathsf{sig}^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{vk}) : \\ \mathsf{Verify}(\mathsf{vk},\mathsf{msg}^*,\mathsf{sig}^*) = 1 \wedge \mathsf{msg}^* \notin L\end{array}\right]$$

is negligible in λ, where L is the set of messages queried by $\mathcal{A}$ to its signing oracle $\mathsf{Sign}(\mathsf{sk},\cdot)$.

Definition 3.3.3 (Strong Existential Unforgeability under Chosen Message Attacks (SUF-CMA)). *A signature scheme is* strongly unforgeable under chosen message attacks *if for any PPT adversary $\mathcal{A}$, its advantage*

$$\mathsf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathcal{A}}(1^\lambda) = \Pr\left[\begin{array}{l}(\mathsf{vk},\mathsf{sk}) \leftarrow \mathsf{Keygen}(1^\lambda),\ (\mathsf{msg}^*,\mathsf{sig}^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{vk}) : \\ \mathsf{Verify}(\mathsf{vk},\mathsf{msg}^*,\mathsf{sig}^*) = 1 \wedge (\mathsf{msg}^*,\mathsf{sig}^*) \notin L\end{array}\right]$$

is negligible in λ, where L is the set of message-signature pairs queried by $\mathcal{A}$ to its signing oracle $\mathsf{Sign}(\mathsf{sk},\cdot)$.

3.4 THRESHOLD SIGNATURES

Threshold signatures schemes extend digital signature schemes to a distributed setting, allowing the sharing of the signing capability among multiple parties. As we have discussed in Chapter 1, threshold signatures are particularly useful in scenarios where enhanced security and fault tolerance are required.

At a high level, in a (N, T, t)-threshold signature scheme, N parties jointly hold a shared private key, and any subset of T parties can cooperate to execute a distributed protocol and generate a valid signature on behalf of the group. The parameter $t < T$ denotes the maximum number of corrupted parties that the scheme can tolerate while maintaining a notion of unforgeability. Ideally, we want $t = T - 1$ to minimize the number of parties required to sign while maximizing the corruption threshold.

Throughout this thesis, we assume that parties are indexed by $[N] = \{1, 2, \ldots, N\}$, and we will denote CS as the set of corrupted parties and $\mathsf{HS} = [N] \backslash \mathsf{CS}$ as the set of honest parties, with the convention that $|\mathsf{CS}| \leqslant t$.

Remark 3.4.1. We may write simply (N, T)-threshold or T-out-of-N threshold signature scheme when $t = T - 1$.

Analogously to standard signature schemes, a threshold signature scheme is composed of key generation, signing, and verification procedures. The primary difference is that the signing procedure is distributed among T out of N parties, and the key generation may also be distributed. This requires special care to translate the correctness and unforgeability properties to the distributed setting.

There exist two main approaches to formalize threshold signatures: the game-based approach and the simulation-based approach. The former defines a concrete protocol flow and individual security properties via tailored security games played between an adversary and a challenger, and the latter uses the Universal Composability (UC) framework [Can01] to capture all desired security properties in a single definition, and allowing for composability with other protocols. While UC offers strong composability guarantees, it introduces significant complexity in the constructions and proofs, and can obscure the concrete round complexity and efficiency of the resulting schemes. Hence, in this thesis, we focus on the game-based approach to define threshold signatures.

3.4.1 Syntax

A (N, T, t)-threshold signature scheme is composed of the same three procedures $(\mathsf{Keygen}, \mathsf{Sign}, \mathsf{Verify})$ as standard signature schemes, but allowing Keygen and Sign to be distributed protocols executed among the N parties.

In the game-based approach, we formalize distributed protocols as stateful algorithms executed by interacting parties. To simplify the syntax, we define protocols in a round-based structure. In each round, a party processes the messages received from the network for the previous round, updates its internal state, and outputs new messages to be sent to other parties.

While the syntax defines the *logic* of message generation, the actual delivery of these messages – including timing, ordering, and potential dropping – is determined by the underlying communication model (e.g., synchronous or asynchronous; malicious/semi-trusted). We abstract this transport layer in the syntax and delegate the specifics of message delivery to the security definitions, discussed in Section 3.4.2.

KEY GENERATION. The key generation can be either centralized or distributed. When it is centralized, a trusted dealer executes the Keygen algorithm to generate a verification key vk, auxiliary information aux that may be leveraged in specific settings (we will only make use of it for identifiable aborts, c.f. Chapter 5), and then shares sk_i to each party $i \in [N]$. We denote this by $\mathsf{Keygen}(1^\lambda, N, T, t) \to (\mathsf{vk}, \mathsf{aux}, (\mathsf{sk}_i)_{i\in[N]})$, where each party i obtains a private key share sk_i, and vk and aux are made public.

When the key generation is distributed in R_{KG} rounds, we define it as a tuple of algorithms:

$\mathsf{Setup}(1^\lambda, N, T, t) \to (\mathsf{st}_i)_{i\in[N]}$. A probabilistic algorithm that takes as input the security parameter λ and outputs a state st_i for each party $i \in [N]$. This models any initial setup required before executing the key generation protocol, such as establishing secure channels.

$\mathsf{ShareKeygen}_r(i, \mathsf{st}_i, \mathsf{pm}_{r-1}) \to (\mathsf{pm}_r^i, \mathsf{st}_i')$ **FOR** $r \in [R_{\mathsf{KG}}]$. An algorithm that is executed by each party $i \in [N]$ in round r of the key generation protocol. It takes as input the party's current state st_i and the messages $\mathsf{pm}_{r-1} = (\mathsf{pm}_{r-1}^j)_{j\in[N]}$ received from all parties $j \in [N]$ in the previous round (with the convention $\mathsf{pm}_0 = \bot$). It outputs a message pm_r^i to be sent to all other parties and an updated state st_i'. $\mathsf{pm}_r^i = \bot$ if the party aborts in round r.

$\mathsf{CombineKey}((\mathsf{pm}_r)_{r\in[R_{\mathsf{KG}}]}) \to (\mathsf{vk}, \mathsf{aux})|\bot$. A deterministic algorithm that is *publicly* executed after the last round of the key generation protocol. It takes as input all messages pm_r^i sent by all parties $i \in [N]$ in all rounds $r \in [R_{\mathsf{KG}}]$, and outputs the verification key vk and auxiliary information aux, or $\bot$ if the protocol aborts.

$\mathsf{PartialSecret}(\mathsf{st}_i) \to \mathsf{sk}_i$. A deterministic algorithm executed by each party $i \in [N]$ when the key generation protocol succeeds. It takes as input the party's final state st_i and outputs the private key share sk_i.

SIGNING. The signing procedure is always distributed. The signing protocol allows a subset $\mathsf{act} \subseteq [N]$ of T parties to jointly sign a message msg using their private key shares $(\mathsf{sk}_i)_{i\in\mathsf{act}}$ and the auxiliary information aux generated during key generation. We define it as a tuple of algorithms modeling an execution in R_{sig} rounds.

$\mathsf{ShareSign}_r(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, i, \mathsf{sk}_i, \mathsf{st}_i, \mathsf{pm}_{r-1}) \to (\mathsf{pm}_r^i, \mathsf{st}_i')$ **FOR** $r \in [R_{\mathsf{sig}}]$. This is a probabilistic algorithm executed by each party $i \in [N]$ in round r of the signing protocol. It takes as input the signing set $\mathsf{act} \subseteq [N]$ (i.e., the set of parties participating in the signing), the message msg to be signed, the party's private key share sk_i, its current state st_i, and the messages $\mathsf{pm}_{r-1} = (\mathsf{pm}_{r-1}^j)_{j\in[N]}$ received from all parties $j \in [N]$ in the previous round

(with the convention $\mathsf{pm}_0 = \bot$). It may also take as input a session identifier sid to uniquely identify the protocol execution. It outputs a message pm_r^i to be sent to all other parties and an updated state st_i'.

In the literature, some protocols allow act and msg to be provided later than round 1, e.g., to model offline preprocessing. However, for simplicity, we require them to be provided from round 1 in this thesis.

$\mathsf{Combine}(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, (\mathsf{pm}_r)_{r\in[R_{\mathsf{sig}}]}) \to \mathsf{sig}|\bot$. A deterministic algorithm that is executed after the last round of the signing protocol. It takes as input all messages pm_r^i sent by all parties $i \in [N]$ in all rounds $r \in [R_{\mathsf{sig}}]$, and the message msg to be signed, and outputs the signature sig or $\bot$ if the protocol aborts.

$\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}) \to \{0, 1\}$. The same verification algorithm as standard signature schemes.

Further, we may define additional algorithms to model the abort identification property, which allows honest parties to identify misbehaving parties that caused the protocol to abort. We leave the formalization of this property to Chapter 5, dedicated to threshold signatures with abort identification.

3.4.2 Communication Models

As with all distributed protocols, threshold signatures require a communication layer to enable message exchange between parties. As is common in the literature, we abstract away the mechanics of the transport layer. We do not specify how parties receive the inputs act, msg, pm_{r-1} at the syntax level, and we capture the behavior of the communication layer in the security definitions.

In the game-based approach, we model different communication models by providing some control over the protocol execution and message delivery to the adversary, allowing it to manipulate the communication between parties in certain ways.

We consider two primary communication models:

- **Asynchronous Model.** We assume a fully untrusted network controlled by the adversary. The adversary can arbitrarily drop, delay, reorder, or modify messages. For simplicity, our syntax assumes that each party executes the next round only after receiving messages from all other parties. This can practically be achieved by having parties store received messages in a buffer and wait for the remaining ones, or by relying on an untrusted *coordinator* that collects and forwards messages between parties. Note that in this model, the protocol execution may never terminate.

  In the security games, this is formalized by allowing the adversary to trigger the key generation and signing algorithms of *each party* individually, controlling the messages they receive and send.

- **Synchronous Model with Reliable Broadcast.** This model assumes the existence of a broadcast channel that guarantees delivery and consistency, i.e. all honest parties receive the same messages in the same order. This can be achieved in practice either by relying on a trusted *coordinator* that faithfully forwards messages between parties, or by implementing a reliable broadcast protocol, although this often incurs significant overhead. This model is particularly relevant for robust threshold signatures (Chapter 4), as it is trivial to break robustness in an asynchronous setting by dropping messages or forwarding inconsistent messages to parties.

  This is formalized in the security games by providing the adversary with oracles that execute entire rounds of the protocol, ensuring that all honest parties receive the same messages in the same order. We additionally consider the use of unique session identifiers sid in this model.

3.4.3 Safety Properties: Unforgeability and Correctness

Threshold signatures schemes are required to satisfy two main properties, referred to as safety properties (e.g. "nothing bad happens"), that directly correspond to their centralized counterparts: *correctness* and *unforgeability*.

While in the honest majority setting (e.g., $t < N/2$) key generation and signing can be considered separately allowing for modular definitions [KGS23], in the dishonest majority setting ($t \geqslant N/2$) game-based definitions typically consider both jointly (see e.g., [BCKM+22]). We adopt this approach in this work. The main reason is that in the dishonest majority setting, we cannot reconstruct a valid private key from honest parties only, preventing security definitions enabling drop-in replacement of a centralized key generation in the security games. A solution could be to leverage techniques to extract private keys from corrupted parties (e.g., via zero-knowledge proofs), however at a significant cost in terms of efficiency and complexity.

CORRECTNESS. A threshold signature scheme is correct if when both the key generation and signing protocols are executed honestly by all parties (and the network is reliable), any signature generated by the signing protocol is accepted by the verification algorithm. This is formalized in the below definition.

Definition 3.4.2 (Correctness). *We define the correctness game* $\mathsf{Game}^{\mathsf{TS\text{-}CORR}}$ *in Fig. 3.2. A* (N, T, t)*-threshold signature scheme is* correct *if for any corruption threshold* t*, signing threshold* T*, number of parties* N*, signing set* $\mathsf{act} \subseteq [N]$ *such that* $|\mathsf{act}| = T$*, and message* $\mathsf{msg} \in \{0,1\}^*$*, we have:*

$$\Pr\left[\mathsf{Game}^{\mathsf{TS\text{-}CORR}}(1^\lambda, N, T, t, \mathsf{act}, \mathsf{msg}) = 1\right] \geqslant 1 - \mathsf{negl}(\lambda).$$

UNFORGEABILITY. Akin to standard signature schemes, unforgeability ensures that an adversary cannot produce valid signatures, provided they do not have access to the private key shares of at least $t+1$ parties. The adversary is further allowed to interact with a signing oracle that enables it to request signatures on messages of its choice. While simple in concept, many variants of unforgeability exist in the threshold setting, depending on the corruption model (static vs adaptive), the communication model (synchronous vs asynchronous), and the type of forgeries considered.

We illustrate the space of unforgeability notions for threshold signatures in Fig. 3.3, and detail the different dimensions below.

There are two main corruption models:

- **Static Corruption.** The adversary must choose which parties to corrupt before the execution of the key generation and signing protocols. This models scenarios where the adversary has limited capabilities or where parties are compromised in advance.
- **Adaptive Corruption.** The adversary can choose which parties to corrupt dynamically during the execution of the protocols, based on the information it gathers. This model is more realistic, however it is notoriously harder to prove security in this setting.

  While one can use *complexity leveraging* to achieve adaptive security from static security (e.g. by guessing the set of corrupted parties in advance), this results in an exponential loss in the security reduction which becomes unacceptable as the number of parties grows.

There are two main communication models, as discussed in Section 3.4.2:

- **Synchronous Model with Reliable Broadcast.** The adversary is restricted to delivering messages consistently to all honest parties, ensuring they receive the same messages in the same order.

$\mathsf{Game}^{\mathsf{TS\text{-}CORR}}(1^\lambda, N, T, t, \mathsf{act}, \mathsf{msg})$

1 : $\boxed{(\mathsf{vk}, \mathsf{aux}, (\mathsf{sk}_i)_{i\in[N]}) \leftarrow \mathsf{Keygen}(1^\lambda, N, T, t)}$

2 : $\boxed{(\mathsf{st}_i)_{i\in[N]} \leftarrow \mathsf{Setup}(1^\lambda, N, T, t)}$

3 : $\boxed{\textbf{for } r \in [R_{\mathsf{KG}}] \textbf{ do}}$

4 : $\quad\boxed{\textbf{for } i \in [N] \textbf{ do}}$

5 : $\quad\quad\boxed{\mathsf{pm}_r^i, \mathsf{st}_i \leftarrow \mathsf{ShareKeygen}(i, \mathsf{st}_i, \mathsf{pm}_{r-1})}$

6 : $\quad\quad\boxed{\textbf{if } \mathsf{pm}_r^i = \perp \textbf{ then return } 0}$

7 : $\boxed{(\mathsf{vk}, \mathsf{aux}) := \mathsf{CombineKey}((\mathsf{pm}_r)_{r\in[R_{\mathsf{KG}}]})}$

8 : $\boxed{\textbf{if } (\mathsf{vk}, \mathsf{aux}) = \perp \textbf{ then return } 0}$

9 : $\boxed{\textbf{for } i \in [N] \textbf{ do } \mathsf{sk}_i := \mathsf{PartialSecret}(\mathsf{st}_i)}$

10 : $\textbf{for } r \in [R_{\mathsf{sig}}] \textbf{ do}$

11 : $\quad\mathsf{pm}_r := \varnothing$

12 : $\quad\textbf{for } i \in \mathsf{act} \textbf{ do}$

13 : $\quad\quad\mathsf{pm}_r^i, \mathsf{st}_i \leftarrow \mathsf{ShareSign}_r(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, i, \mathsf{sk}_i, \mathsf{st}_i, \mathsf{pm}_{r-1})$

14 : $\mathsf{sig} := \mathsf{Combine}(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, (\mathsf{pm}_r)_{r\in[R_{\mathsf{sig}}]})$

15 : $\textbf{if } \mathsf{sig} = \perp \textbf{ then return } \perp$

16 : $\textbf{return } \mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$

Figure 3.2: Correctness game for a threshold signature. Dashed boxes correspond to the centralized key generation setting, and solid boxes to the distributed key generation. By convention, $\mathsf{pm}_0 = \perp$. The threshold signature is correct if the above game returns 1 with overwhelming probability.

- **Asynchronous Model.** The adversary has full control over the network, and can arbitrarily drop, delay, reorder, or modify messages.

Finally, there are two main types of forgeries:

- **Existential Unforgeability.** The adversary's goal is to produce a valid signature on a new message that it has not queried to the signing oracle.
- **Strong Unforgeability.** The adversary's goal is to produce more valid signatures than those it has obtained by querying the signing oracle.

Several levels of security can be defined [BCKM+22] for each type of forgery, depending on which signing query is considered as providing a valid signature, thus not counting toward the forgery. Roughly, the levels are defined as follows:

- **Level 0.** A forgery is trivial if the signing oracle for at least one honest party was called for that message.
- **Level 1.** A forgery is trivial if the signing oracle was called for at least $T - c$ honest parties for that message.
- **Level 2.** A forgery is trivial if the signing oracle was called *in a signing session for that message* for at least $T - c$ honest parties, where c is the number of corrupted parties.
- **Level 3.** A forgery is trivial if *in a signing session for that message*, all the honest parties in the signing set were queried for that message, or their message was modified by the adversary.
- **Level 4.** A forgery is trivial if *in a signing session for that message*, all the honest parties in the signing set were queried for that message.

Remark 3.4.3. As an interesting fact, in the classical setting, there exists a generic transformation in the standard model from existential to strong unforgeability for threshold signatures [ABHR25], however it is not known if it is also possible in the post-quantum setting.

In this thesis, we focus on:

- **Static Corruption.** We consider only static adversaries that choose which parties to corrupt before the execution of the protocols. This is common in the literature on lattice-based threshold signatures, as obtaining provable security against adaptive adversaries leads to significant efficiency overheads (see e.g. [KRT24]). Note that this is typically a limitation of the proof techniques rather than of the schemes themselves, which often remain secure against adaptive adversaries in practice.
- **Existential Unforgeability, Level 0.** We focus on existential unforgeability, where the adversary's goal is to produce a valid signature on a new message that it has not queried to the signing oracle. Further, we consider Level 0 security, where a forgery is trivial if the signing oracle for at least one honest party was called for that message. This is the most widely studied notion of unforgeability for threshold signatures. We note that restricting to level 0 is not an intrinsic limitation of our techniques, and stronger notions could be achieved with additional effort, but Level 0 often suffices in practice and allows for simpler proofs and explanations, keeping the focus on the main technical contributions of this thesis.

For the communication model, we consider the synchronous model with reliable broadcast during key generation, and for signing, both the synchronous model with reliable broadcast (for robust threshold signatures, Chapter 4), and the asynchronous model (for our other constructions).

We formalize these notions of unforgeability in Figs. 3.4 and 3.5, respectively for signing in the synchronous and asynchronous models.

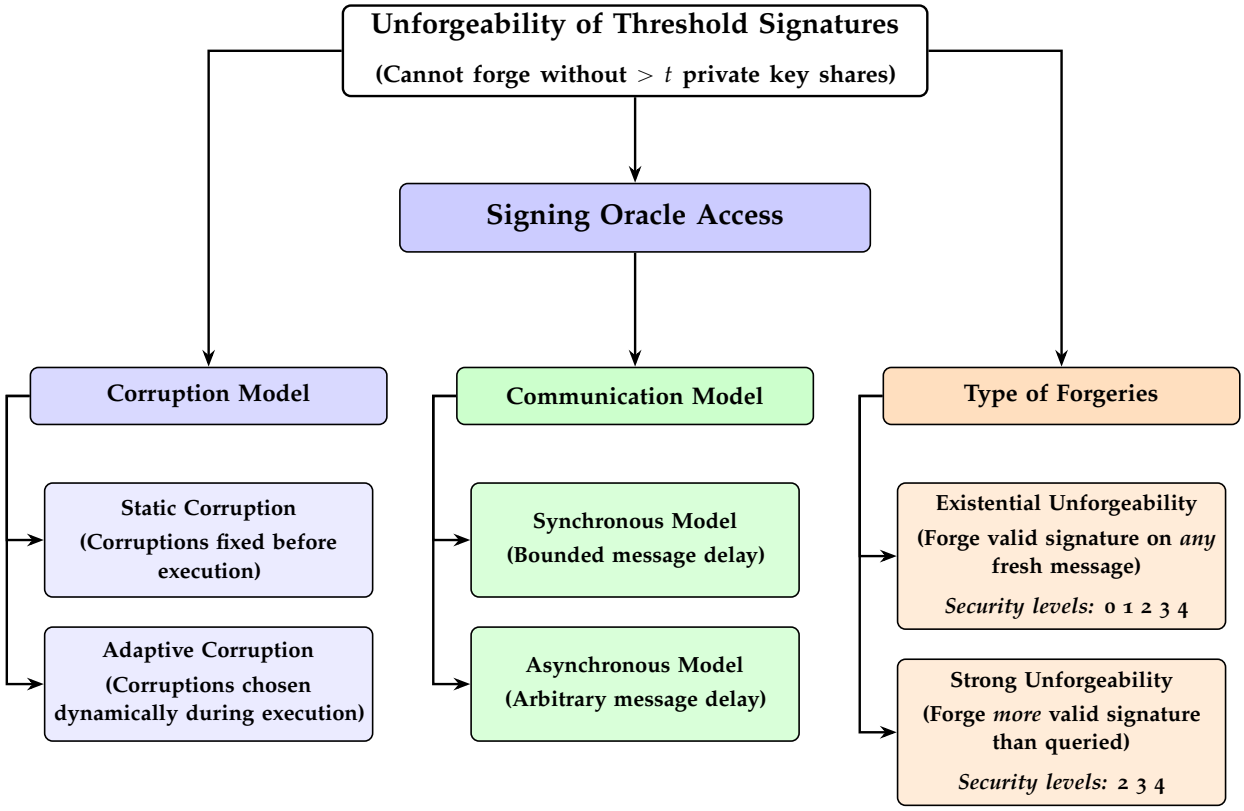


Figure 3.3: Taxonomy of unforgeability notions for threshold signatures. The adversary is given access to a signing oracle, and its goal is to produce valid signatures without access to more than t private key shares. The unforgeability notions vary based on the corruption model (static vs adaptive), communication model (synchronous vs asynchronous), and type of forgeries considered (existential vs strong). The security levels correspond to different winning conditions that modify which signing oracle queries count toward the forgery.

Definition 3.4.4 (Static, Synchronous Unforgeability). *A (N, T, t)-threshold signature scheme is* unforgeable in the static, synchronous model *if for any PPT adversary $\mathcal{A}$, its advantage*

$$\mathsf{Adv}_{\mathcal{A}}^{\text{SYNC-TS-UF}}(1^\lambda, N, T, t) = \Pr\left[\mathsf{Game}_{\mathcal{A}}^{\text{SYNC-TS-UF}}(1^\lambda, N, T, t) = 1\right]$$

is negligible in λ.

Definition 3.4.5 (Static, Asynchronous Unforgeability). *A (N, T, t)-threshold signature scheme is* unforgeable in the static, asynchronous model *if for any PPT adversary $\mathcal{A}$, its advantage*

$$\mathsf{Adv}_{\mathcal{A}}^{\text{ASYNC-TS-UF}}(1^\lambda, N, T, t) = \Pr\left[\mathsf{Game}_{\mathcal{A}}^{\text{ASYNC-TS-UF}}(1^\lambda, N, T, t) = 1\right]$$

is negligible in λ.

3.4.4 Liveness Properties: Robustness and Identifiable Aborts

Beyond safety properties, threshold signature schemes may also be required to satisfy certain liveness properties (e.g., "something good eventually happens"). In the context of threshold signatures, we wish to eventually obtain valid signatures, even in the presence of misbehaving parties. Two main liveness properties are commonly considered in the literature: *robustness* and *identifiable aborts*. They apply to both the key generation and signing protocols.

Robustness ensures that as long as a sufficient number of honest parties participate in the scheme execution, a valid signature is always produced, regardless of the behavior of the corrupted parties. However, this typically requires $T \geqslant 2t + 1$, or even $T \geqslant 3t + 1$, which can be a significant limitation in practice.

Identifiable aborts is an alternative property that ensures that if the protocol aborts due to misbehavior (e.g., the key generation or signing protocol does not complete successfully), the honest parties can identify at least one corrupted party responsible for the abort. Intuitively, this property allows honest parties to take corrective actions, such as excluding misbehaving parties from future protocol executions, and will eventually lead to successful signature generation.

We defer the formalization of these properties to Chapter 4 and Chapter 5, dedicated to robust threshold signatures and threshold signatures with identifiable aborts, respectively.

3.5 THRESHOLD SIGNATURE SCHEMES: CLASSICAL, POST-QUANTUM, AND PROPERTIES

Having established the security notions of interests for this thesis—unforgeability, robustness, identifiable aborts—we now survey the landscape of threshold signatures. Our goal is to understand how these properties have been achieved in the literature, tracing the evolution from classical schemes based on discrete logarithms and RSA to modern post-quantum constructions based on lattices.

We also need to consider the trade-offs between different security guarantees and the underlying assumptions, as well as the efficiency of the resulting schemes. Recall from the previous section and Fig. 3.3 that there are multiple dimensions to consider when defining security for threshold signatures (e.g., static vs adaptive corruption, synchronous vs asynchronous communication, existential vs strong unforgeability, and the hierarchy from Bellare et al. [BCKM+22]).

In the following, we examine how classical and post-quantum schemes navigate these trade-offs.

$\mathsf{Game}^{\mathsf{SYNC\text{-}TS\text{-}UF}}_{\mathcal{A}}(1^\lambda, N, T, t)$

$L_{\mathsf{sid}}, L_{\mathsf{sig}} := \varnothing$
$(\mathsf{CS}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathsf{H}}(1^\lambda, N, T, t)$ // $\mathcal{A}$ chooses corrupted parties
assert $\mathsf{CS} \subseteq [N] \wedge |\mathsf{CS}| \leqslant t$
$\mathsf{HS} := [N] \backslash \mathsf{CS}$
$(\mathsf{vk}, \mathsf{aux}, (\mathsf{sk}_i)_{i\in[N]}) \leftarrow \mathsf{Keygen}(1^\lambda, N, T, t)$
$(\mathsf{st}_i)_{i\in[N]} \leftarrow \mathsf{Setup}(1^\lambda, N, T, t)$
$\mathsf{st}_{\mathcal{A}} \leftarrow \mathcal{A}((\mathsf{st}_i)_{i\in\mathsf{CS}}, \mathsf{st}_{\mathcal{A}})$
for $r \in [R_{\mathsf{KG}}]$ **do**
  for $i \in \mathsf{HS}$ **do**
    $\mathsf{pm}^i_r, \mathsf{st}_i \leftarrow \mathsf{ShareKeygen}(i, \mathsf{st}_i, \mathsf{pm}_{r-1})$
    if $\mathsf{pm}^i_r = \perp$ **then return** 0
  $((\mathsf{pm}^i_r)_{i\in\mathsf{CS}}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathsf{H}}((\mathsf{pm}^i_r)_{i\in\mathsf{HS}}, \mathsf{st}_{\mathcal{A}})$
$(\mathsf{vk}, \mathsf{aux}) := \mathsf{CombineKey}((\mathsf{pm}_r)_{r\in[R_{\mathsf{KG}}]})$
if $(\mathsf{vk}, \mathsf{aux}) = \perp$ **then return** 0
for $i \in \mathsf{HS}$ **do** $\mathsf{sk}_i := \mathsf{PartialSecret}(\mathsf{st}_i)$
$(\mathsf{msg}, \mathsf{sig}) \leftarrow \mathcal{A}^{\mathsf{H}, (\mathsf{OPerformRound}_r)_{r\in[R_{\mathsf{sig}}]}}(\mathsf{vk}, \mathsf{aux}, (\mathsf{sk}_i)_{i\in\mathsf{CS}}, \mathsf{st}_{\mathcal{A}})$
if $(\mathsf{msg} \in L_{\mathsf{sig}})$ or $\neg\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$ **then**
  return 0 // Returned pair is not a valid forgery
return 1

$\mathsf{OPerformRound}_1(\mathsf{sid}, \mathsf{act}, \mathsf{msg})$

require $\mathsf{sid} \notin L_{\mathsf{sid}} \wedge |\mathsf{act}| = T$
for $i \in \mathsf{HS} \cap \mathsf{act}$ **do**
  $(\mathsf{pm}^i_1, \mathsf{st}_i) := \mathsf{ShareSign}_1(\mathsf{vk}, \mathsf{sid}, \mathsf{act}, \mathsf{msg}, i, \mathsf{sk}_i, \mathsf{st}_i, \mathsf{msg}, \perp)$
$L_{\mathsf{sig}} := L_{\mathsf{sig}} \cup \{\mathsf{msg}\}$
$L_{\mathsf{sid}}[\mathsf{sid}] = \{1, \mathsf{act}, \mathsf{msg}, (\mathsf{pm}^i_1)_{i\in\mathsf{HS}\cap\mathsf{act}}\}$
return $(\mathsf{pm}^i_1)_{i\in\mathsf{HS}\cap\mathsf{act}}$

$\mathsf{OPerformRound}_r(\mathsf{sid}, (\mathsf{pm}^i_{r-1})_{i\in\mathsf{CS}\cap\mathsf{act}})$, for $r = 2, ..., R_{\mathsf{sig}}$

$\{r', \mathsf{act}, \mathsf{msg}, (\mathsf{pm}^i_{r-1})_{i\in\mathsf{HS}\cap\mathsf{act}}\}$ **from** $L_{\mathsf{sid}}[\mathsf{sid}]$
require $r' = r - 1$
$\mathsf{pm}_{r-1} := (\mathsf{pm}^i_{r-1})_{i\in\mathsf{act}}$
for $i \in \mathsf{HS} \cap \mathsf{act}$ **do**
  $(\mathsf{pm}^i_r, \mathsf{st}_i) := \mathsf{ShareSign}_r(\mathsf{vk}, \mathsf{sid}, \mathsf{act}, \mathsf{msg}, i, \mathsf{sk}_i, \mathsf{st}_i, \mathsf{msg}, \mathsf{pm}_{r-1})$
$L_{\mathsf{sid}}[\mathsf{sid}] = \{r, \mathsf{act}, \mathsf{msg}, (\mathsf{pm}^i_r)_{i\in\mathsf{HS}\cap\mathsf{act}} \cup (\mathsf{pm}^i_r)_{i\in\mathsf{HS}\cap\mathsf{act}}\}$
return $(\mathsf{pm}^i_r)_{i\in\mathsf{HS}\cap\mathsf{act}}$

Figure 3.4: Unforgeability game for a threshold signature with key generation and signing in the *synchronous model with reliable broadcast*. We assume access to a hash oracle H modeled as a random oracle. Dashed boxes correspond to the centralized key generation setting, and solid boxes to the distributed key generation. By convention, $\mathsf{pm}_0 = \perp$. The adversary $\mathcal{A}$ wins if the game SYNC-TS-UF returns 1, i.e. if it forged a new signature.

$\mathsf{Game}_{\mathcal{A}}^{\mathsf{ASYNC\text{-}TS\text{-}UF}}(1^\lambda, N, T, t)$

$L_{\mathsf{sig}} \coloneqq \varnothing$
$(\mathsf{CS}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathsf{H}}(1^\lambda, N, T, t)$ // $\mathcal{A}$ chooses corrupted parties
assert $\mathsf{CS} \subseteq [N] \wedge |\mathsf{CS}| \leqslant t$
$\mathsf{HS} \coloneqq [N] \backslash \mathsf{CS}$
$(\mathsf{vk}, \mathsf{aux}, (\mathsf{sk}_i)_{i\in[N]}) \leftarrow \mathsf{Keygen}(1^\lambda, N, T, t)$
$(\mathsf{st}_i)_{i\in[N]} \leftarrow \mathsf{Setup}(1^\lambda, N, T, t)$
$\mathsf{st}_{\mathcal{A}} \leftarrow \mathcal{A}((\mathsf{st}_i)_{i\in\mathsf{CS}}, \mathsf{st}_{\mathcal{A}})$
for $r \in [R_{\mathsf{KG}}]$ **do**
  for $i \in \mathsf{HS}$ **do**
    $\mathsf{st}_i, \mathsf{pm}_r^i \leftarrow \mathsf{ShareKeygen}(i, \mathsf{st}_i, \mathsf{pm}_{r-1})$
    if $\mathsf{pm}_r^i = \bot$ **then return** 0
  $((\mathsf{pm}_r^i)_{i\in\mathsf{CS}}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathsf{H}}((\mathsf{pm}_r^i)_{i\in\mathsf{HS}}, \mathsf{st}_{\mathcal{A}})$
$(\mathsf{vk}, \mathsf{aux}) \coloneqq \mathsf{CombineKey}((\mathsf{pm}_r)_{r\in[R_{\mathsf{KG}}]})$
if $(\mathsf{vk}, \mathsf{aux}) = \bot$ **then return** 0
for $i \in \mathsf{HS}$ **do** $\mathsf{sk}_i \coloneqq \mathsf{PartialSecret}(\mathsf{st}_i)$
$(\mathsf{msg}, \mathsf{sig}) \leftarrow \mathcal{A}^{\mathsf{H}, (\mathcal{O}_{\mathsf{ShareSign}_r})_{r\in[R_{\mathsf{sig}}]}}(\mathsf{vk}, \mathsf{aux}, (\mathsf{sk}_i)_{i\in\mathsf{CS}}, \mathsf{st}_{\mathcal{A}})$
if $(\mathsf{msg} \in L_{\mathsf{sig}})$ or $\neg\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$ **then**
  return 0 // Returned pair is not a valid forgery
return 1

$\mathcal{O}_{\mathsf{ShareSign}_r}(\mathsf{act}, \mathsf{msg}, i, \mathsf{pm}_{r-1})$ for $r = 2, ..., R_{\mathsf{sig}}$

require $|\mathsf{act}| = T \wedge i \in \mathsf{act} \cap \mathsf{HS}$
$L_{\mathsf{sig}} \coloneqq L_{\mathsf{sig}} \cup \{\mathsf{msg}\}$
$(\mathsf{pm}_r^i, \mathsf{st}_i) \coloneqq \mathsf{ShareSign}_r(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, i, \mathsf{sk}_i, \mathsf{st}_i, \mathsf{msg}, \mathsf{pm}_{r-1})$
return pm_r^i

Figure 3.5: Unforgeability game for a threshold signature with key generation in the synchronous model with reliable broadcast, and signing in the *asynchronous model*. We assume access to a hash oracle H modeled as a random oracle. Dashed boxes correspond to the centralized key generation setting, and solid boxes to the distributed key generation. By convention, $\mathsf{pm}_0 = \bot$. The adversary $\mathcal{A}$ wins if the game ASYNC-TS-UF returns 1, i.e. if it forged a new signature.

3.5.1 The Golden Age: Classical Threshold Signatures

In the pre-quantum world, threshold signatures benefited from the algebraic structures of RSA and discrete logarithm groups, which enabled the best of both worlds: strong security guarantees (e.g., adaptive security, robustness, identifiable aborts) and efficient constructions.

RSA. A threshold RSA construction was proposed as early as 1998 [Rab98]. It has later been improved to be robust [Sho00; GRJK00], include a distributed key generation [FS01], to support dynamic updates of the set of parties [GHKR08] and adaptive security [ADN06]. However, interest in threshold RSA waned over time, as discrete-log schemes like Schnorr and DSA proved more efficient and easier to threshold, and as the community shifted toward elliptic curves.

OVERVIEW OF DISCRETE-LOG SCHEMES. The discrete logarithm setting has been the most fertile ground for threshold signature schemes, with a wide variety of constructions and security properties achieved over the years. The algebraic structure of groups where the discrete logarithm problem is hard allows for elegant protocols with strong security guarantees without sacrificing efficiency.

- **Schnorr:** Due to their linear structure, Schnorr signatures received significant attention in the literature [GJKR99; SS01; GRSS+21; CKM21; CKM23; Lin24; KG25]. While early works focused on feasibility, the FROST scheme [KG20] later set a benchmark for efficiency by introducing a pre-processing phase that enables a non-interactive signing protocol, with identifiable aborts. More recently, the ROAST framework [RRJS+22] showed that such partially non-interactive schemes could be made robust generically.

  Following this already strong baseline, recent works aimed to push the boundaries in terms of security guarantees, notably achieving full adaptive security [BDLR25b; BDLR25a; CKKT+25; Che25; NRT26]—sometimes combined with robustness [BLSW24], or stronger notions of unforgeability [BCKM+22].

- **DSA/ECDSA:** Contrary to Schnorr, DSA and ECDSA signatures do not have a linear structure, which makes them more challenging to threshold. Still, their widespread use in practice motivated significant research efforts [DKLs18; GG18; GGN16; Lin17; LN18; DKLs24]. [GJKR96] proposed the first threshold DSA scheme. Later works focused on improving its efficiency, and achieving properties such as identifiable aborts [CGGM+20; CCLS+23], and robustness [TX25].

- **BLS:** Threshold signatures based on pairings, notably BLS signatures, allow for *non-interactive* signature generation. [Bol03] provided the first such scheme. Later, [BL22] achieved the first proof of adaptive security for threshold BLS, and [GJMS+24] provided a proactive version of threshold BLS.

3.5.2 The Post-Quantum Shift: Challenges with Lattices

While lattice-based cryptography has become the leading candidate for post-quantum secure primitives, designing threshold signatures in this setting has proven to be significantly more challenging than in the classical setting. The core issue is that lattice signatures involve non-linear operations—including *rejection sampling* or *short vector sampling*. These techniques do not distribute naturally.

We start by examining lattice-based signatures following the Fiat-Shamir with Aborts (FSwA) [Lyu09] paradigm which underly the NIST standard ML-DSA, as they offer a direct parallel to Schnorr signatures and highlight important challenges. We will later discuss the noise flooding-

based approach, and how solutions to distribute lattice-based Fiat-Shamir signatures appear to naturally extend to hash-and-sign schemes.

Schnorr.Keygen()

1: $\mathsf{sk} \xleftarrow{\$} \mathbb{Z}_q$
2: $\mathsf{vk} := g^{\mathsf{sk}}$
3: **return** $(\mathsf{sk}, \mathsf{vk})$

Schnorr.Sign(sk, msg)

1: // Commit
2: $r \xleftarrow{\$} \mathbb{Z}_q$
3: $w := g^r$
4: // Challenge
5: $c := H(w, \mathsf{msg})$
6: // Respond
7: $z := c \cdot \mathsf{sk} + r$
8: **return** (c, z)

Schnorr.Verify(vk, msg, sig)

1: $w' := g^z \cdot \mathsf{vk}^{-c}$
2: **require** $H(w', \mathsf{msg}) = c$

(a) Schnorr Signatures (discrete-log setting)

FSwA.Keygen()

1: $\mathbf{A} \xleftarrow{\$} R_q^{k \times \ell}$; $\mathbf{s} \leftarrow \chi_{\mathbf{s}}$ // $\|\mathbf{s}\|$ small
2: $\mathbf{b} := [\mathbf{A} \mid \mathbf{I}] \cdot \mathbf{s}$
3: **return** $(\mathsf{sk} = \mathbf{s},\ \mathsf{vk} = (\mathbf{A}, \mathbf{b}))$

FSwA.Sign(sk, msg)

1: // Commit
2: $\mathbf{r} \leftarrow \chi_{\mathbf{r}}$ // $\|\mathbf{r}\|$ small
3: $\mathbf{w} := [\mathbf{A} \mid \mathbf{I}] \cdot \mathbf{r}$
4: // Challenge
5: $c := H(\mathbf{w}, \mathsf{msg})$
6: // Respond
7: $\mathbf{z} := c \cdot \mathsf{sk} + \mathbf{r}$
8: **if** $\|\mathbf{z}\| \geqslant \gamma$ **then restart**
9: **return** $(c, \mathbf{z})$

FSwA.Verify(vk, msg, sig)

1: **require** $\|\mathbf{z}\| < \gamma$
2: $\mathbf{w}' := [\mathbf{A} \mid \mathbf{I}] \cdot \mathbf{z} - c \cdot \mathbf{b}$
3: **require** $H(\mathbf{w}', \mathsf{msg}) = c$

(b) Fiat-Shamir with Aborts (FSwA) (lattice setting, e.g. ML-DSA)

Figure 3.6: Comparison of Schnorr and FSwA signatures. Both follow the same Fiat-Shamir blueprint (Commit → Challenge → Respond), but the lattice setting introduces *shortness constraints* on the secret key and nonce, and a *rejection sampling* loop to hide the secret key from the response. Differences in the lattice setting are highlighted. This figure presents a simplified, unoptimized view of ML-DSA (e.g. without HighBits/MakeHint).

We illustrate in Fig. 3.6 the structural similarities—and differences—between classical discrete logarithm signatures and modern lattice-based schemes. Broadly speaking, both Schnorr and ML-DSA follow the same Fiat-Shamir "sigma protocol" blueprint:

1. **Commitment:** The signer samples a random nonce (r or $\mathbf{r}$) and commits to it ($w = g^r$ or $\mathbf{w} = [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}$).
2. **Challenge:** A challenge c is derived by hashing the message and commitment.
3. **Response:** The signer computes a response $z = c \cdot \mathsf{sk} + \mathbf{r}$ that proves the knowledge of the secret key sk.

Despite this shared blueprint, transforming these schemes into threshold protocols presents quite different challenges. The left side of Fig. 3.6 (Schnorr) is entirely linear over $\mathbb{Z}_q$. To sign distributedly, intuitively, parties can simply sample additive shares of the nonce locally ($r = \sum r_i \pmod q$) and compute additive shares of the response. Since the sum of uniform elements is uniform, the resulting signature is indistinguishable from one produced by a single signer.

The right side (FSwA signatures), however, introduces complex operations:

- **Shortness Constraints:** Lattice-based schemes rely on the "shortness" of vectors (e.g., small coefficients). While sums of uniforms in $\mathbb{Z}_q$ remain uniform, the sum of short vectors is

strictly larger. One also cannot simply remove bias from a maliciously sampled short nonce by adding a honest nonce to it.

- **Rejection Sampling:** To prevent the signature $\mathbf{z} = c \cdot \mathbf{s} + \mathbf{r}$ from leaking the secret $\mathbf{s}$, FSwA signatures force the signer to *reject* and restart if $\mathbf{z}$ reveals statistical bias (for instance, if it falls outside a specific hypercube in ML-DSA). In the threshold setting, this creates a key challenge: parties must collectively decide whether to reject $\mathbf{z}$ *without* reconstructing it, as revealing a rejected signature would compromise the secret key.

Several works suggested to use general MPC protocols [BKP13; CS19; TPCZ24; DKLS25; BCEP+25] to compute these non-linear operations. While this approach provided feasibility results, this leads to either impractical schemes, or significant compromises in the security guarantees to achieve a reasonable practicality (see Table 6.1 for the case of ML-DSA). The pre-computation phase in this approach is also often left out in the concrete numbers as it remains an open problem to efficiently distribute the sampling of a large number of correlated randomness as required for lattice-based schemes. We thus focus on schemes that do not rely on general MPC protocols in this survey.

ACCEPTING IMPERFECT OPERATIONS. Designing efficient lattice-based threshold signatures requires a shift in perspective. Centralized schemes like ML-DSA are optimized for compactness, sampling secrets and nonces from distributions much narrower than the modulus q used in the following operations. However, sampling a short vector $\mathbf{r}$ in a distributed manner is hard: if parties sample small local shares $\mathbf{r}_i \bmod q$, their sum $\mathbf{r} = \sum \mathbf{r}_i \bmod q$ follows a convolution distribution (approaching a Gaussian), which is wider and the final value is partly leaked to corrupted parties. To avoid heavy MPC machinery, research has largely pivoted to accepting "imperfect" distributed operations—in particular, simply summing local short vectors—and dealing with the potential bias and parameter increase that come with it.

Strategy I: Distributing Rejection Sampling

The first threshold lattice signature schemes reaching practicality followed the approach of distributing the rejection sampling procedure itself [DOTT22; BTT22; Che23; ADP24]. Attempts in this direction focused on the N-out-of-N setting where the secret key is additively shared ($\mathbf{s} = \sum \mathbf{s}_i$) such that each share $\mathbf{s}_i$ is itself small. In this paradigm, parties perform N independent, *local* rejection samplings. Essentially, each party samples a local nonce share $\mathbf{r}_i$, computes a partial response $\mathbf{z}_i = c \cdot \mathbf{s}_i + \mathbf{r}_i$, and checks if $\mathbf{z}_i$ satisfies the rejection bound. If all parties succeed locally, they aggregate their shares to form the final signature.

However, even in this constrained setting, this line of research has faced significant hurdles in providing compact signatures as shown in Table 3.1. The core difficulty lies in proving security: there is no known way to prove that it is safe to reveal the partial commitments $\mathbf{w}_i$ *before* the rejection condition is fully satisfied. To mitigate this, these schemes employed expensive mechanisms—such as trapdoor commitments or heavy randomization techniques—to keep partial values hidden until rejection sampling is successful.

We break this efficiency barrier in Chapter 6 by proposing a threshold variant of the NIST standard ML-DSA, furthermore supporting any $T \leqslant N \leqslant 6$ and outcompeting all previous schemes in terms of signature and public key sizes.

Still, this research direction has inherent limitations that we leave open for future work. It appears to become increasingly difficult to achieve compact signatures as the number of parties increases, and identifiable aborts and robustness seem in opposition with rejection sampling as we implement the latter by explicitly allowing parties to abort the protocol. Should the scalability of this approach be improved, it would also be interesting to obtain tight proof of the adaptive security of these schemes.

Scheme	[DOTT22]*	[BTT22]	[ADP24]	[Che23]	Ours (Chapter 6)
|sig| (kB)	2538	98	12	20	2.4
|vk| (kB)	21	26	7	7	1.3
Type	M	M	(T,N) / M	M	(T,N)
N	7	32	7	32	6

Table 3.1: Comparison of lattice-based threshold signatures schemes following the Fiat-Shamir with Aborts paradigm. We include works on multi-signatures as they can easily be leveraged to build a N-out-of-N threshold signature. * Sizes for [DOTT22] were proposed in [ADP24].

Strategy II: The Noise Flooding Alternative

To avoid the inefficiencies of distributed rejection sampling, an alternative paradigm emerged for building efficient lattice threshold signatures: *noise flooding*. This paradigm removes the rejection sampling step in the signing algorithm and rather simply adds a large noise term to the signature to hide the secret key contribution. This is exemplified by the Raccoon signature scheme [PKPR24], illustrated in Fig. 3.7. The modern argument for the security of this approach typically relies on the Hint-MLWE assumption (c.f. Section 3.2.4), which is as secure as MLWE when the norm of the noise is proportional to $\sqrt{Q_s}$, where Q_s is the number of signatures supported. Interestingly, this approach makes the signing operation essentially linear, and thus easily distributable, at the cost of larger signatures (about 10 kB). In the threshold setting, it suffices for security that one honest party adds enough noise to the final signature to ensure security, without concerns about the exact distribution of the nonce or signature.

Raccoon.Sign(sk, msg)

1 : // Commit
2 : $\mathbf{r} \leftarrow \chi_{\mathbf{r}}$ // $\|\mathbf{r}\|$ *large* (flooding noise)
3 : $\mathbf{w} := [\mathbf{A} \mid \mathbf{I}] \cdot \mathbf{r}$
4 : // Challenge
5 : $c \leftarrow H(\mathbf{w}, \mathsf{msg})$
6 : // Respond
7 : $\mathbf{z} := c \cdot \mathsf{sk} + \mathbf{r}$
8 : (no rejection step)
9 : **return** $(c, \mathbf{z})$

Figure 3.7: Simplified pseudo-code of Raccoon [PKPR24] (noise flooding-based lattice signature). The nonce $\mathbf{r}$ is sampled from a large flooding distribution $\chi_{\mathbf{r}}$. (Simplified: rounding and hint vector omitted.)

One key advantage of this approach is that it no longer relies on an additive sharing of $\mathbf{s}$ with small shares, and it can leverage other secret-sharing schemes. Notably, assuming that parties possess shares $[\![\mathbf{s}]\!]_i$ of the secret key $\mathbf{s}$, such that for any subset act of T parties, $\mathbf{s} = \sum_{i \in \mathsf{act}} L_{\mathsf{act},i} \cdot [\![\mathbf{s}]\!]_i$ – where $L_{\mathsf{act},i}$ are linear reconstruction coefficients –, then parties can simply compute partial signatures $\mathbf{z}_i = c \cdot L_{\mathsf{act},i} \cdot [\![\mathbf{s}]\!]_i + \mathbf{r}_i$ and aggregate them as $\mathbf{z} = \sum_{i \in \mathsf{act}} \mathbf{z}_i$. This intuition almost directly translates into a threshold scheme, although one needs to be careful to not leak the secret shares $[\![\mathbf{s}]\!]_i$ through the partial signatures $\mathbf{z}_i$, since they are typically much larger than the noise $\mathbf{r}_i$. In the literature, several approaches have been proposed, leveraging different secret sharing schemes for the secret key:

- **Linear Secret Sharing Schemes with Small Reconstruction Coefficients.** In [CTZ24], Chairattana-Apirom, Tessaro and Zhu showed that one could securely use a linear secret

sharing scheme with *small reconstruction coefficients* to share the secret key, and directly use the shares in the partial signatures. However, this approach is limited to a small number of parties (e.g. $N = 32$) and requires increasing the noise to hide the shares, and thus leads to large signatures (about 140 kB [ZT25]).

- **Shamir's Secret Sharing.** It is also possible to use Shamir's secret sharing, which allows to support large thresholds (e.g., $T \leqslant N \leqslant 1024$ in the scheme TRaccoon [DKMM+24]). In this case, the partial signatures need to be hidden. Two approaches have been proposed in the literature to achieve this:
  - **Homomorphic Encryption.** Early works [ASY22; GKS24; GHKS+25] proposed to compute the signature shares under fully or linearly homomorphic encryption, and only decrypt the final signature. This approach is secure and allows to support large thresholds, but it is also rather inefficient, as it requires to perform expensive homomorphic operations on large ciphertexts.
  - **Masking with Zero-Shares.** To avoid such heavy machinery, the TRaccoon scheme from [DKMM+24] demonstrated that one could simply mask the partial signatures using "shares of zero". These shares sum to zero and thus vanish in the aggregation, but they effectively blind the individual partial signatures, protecting the secret shares. They are computed non-interactively using pairwise keys and a pseudorandom function (PRF).

TRaccoon can be seen as a lattice-based analogue of the classical Sparkle [CKM23] signature scheme, which builds a 3-round signature scheme with a commit and reveal approach to compute the commitment $\mathbf{w}$ in a distributed manner, tackling the presence of rushing adversaries trying to bias the challenge c by waiting to see the honest parties' commitments before sending their own. We present the signing algorithms of both schemes side by side in Fig. 3.8 to highlight their similarities and differences (we consider the leaner version of TRaccoon introduced in [KRT24]).

Sparkle.Sign$(\mathsf{act}, \mathsf{msg}, i, \mathsf{sk}_i)$

1 : // Round 1: Commit
2 : $r_i \xleftarrow{\$} \mathbb{Z}_q$; $w_i := g^{r_i}$
3 : Broadcast $h_i := H(w_i)$
4 : // Round 2: Reveal
5 : Broadcast w_i
6 : // Round 3: Respond
7 : **if** $\exists j, H(w_j) \neq h_j$ **then** abort
8 : $w := \prod w_j$; $c := H(w, \mathsf{msg})$
9 : Broadcast $z_i := c \cdot L_{\mathsf{act},i} \cdot \mathsf{sk}_i + r_i$
10 : // Reconstruct
11 : $z := \sum z_j$; **return** (c, z)

(a) Sparkle (Schnorr-based).

TRaccoon.Sign$(\mathsf{act}, \mathsf{msg}, i, \mathsf{sk}_i)$

1 : // Round 1: Commit
2 : $\mathbf{r}_i \leftarrow \chi_{\mathbf{r}}$; $\mathbf{w}_i := [\mathbf{A} \mid \mathbf{I}]\mathbf{r}_i$
3 : Broadcast $h_i := H(\mathbf{w}_i)$
4 : // Round 2: Reveal
5 : Broadcast $\mathbf{w}_i$
6 : // Round 3: Respond
7 : **if** $\exists j, H(\mathbf{w}_j) \neq h_j$ **then** abort
8 : $\mathbf{w} := \sum \mathbf{w}_j$; $c := H(\mathbf{w}, \mathsf{msg})$
9 : $\Delta_i \leftarrow \mathsf{ZeroShare}(i, \mathsf{transcript})$ // $\sum \Delta_j = 0$
10 : Broadcast $\mathbf{z}_i := c \cdot L_{\mathsf{act},i} \cdot \mathsf{sk}_i + \mathbf{r}_i + \Delta_i$
11 : // Reconstruct
12 : $\mathbf{z} := \sum \mathbf{z}_j$; **return** $(c, \mathbf{z})$

(b) TRaccoon (Lattice-based).

Figure 3.8: Comparison of distributed signing in the threshold schemes Sparkle [CKM23] and TRaccoon [DKMM+24]. While both use a similar 3-round structure (Commit-Reveal-Respond), TRaccoon introduces *zero-shares* to mask the partial responses.

As the most efficient lattice threshold signature scheme to date, TRaccoon has inspired several follow-up works: [KRT24] introduced a 5-round adaptively secure variant; [EKT24; BKLM+25; GTJL+26] introduced partially non-interactive variants using techniques inspired by the classical

threshold signature FROST [KG20] for the two first. In FROST-like schemes, parties sample multiple commitments $\mathbf{w}_{i,b}$ in a preprocessing round, and randomly combine them during signing to prevent rushing adversaries. We illustrate the signing algorithm of [EKT24] side by side with FROST in Fig. 3.9.

FROST.Sign(act, msg, i, sk_i)

1 : // Round 1: Preprocessing
2 : $(r_i, s_i) \xleftarrow{\$} \mathbb{Z}_q^2$
3 : $(R_i, S_i) := (g^{r_i}, g^{s_i})$
4 : // Round 2: Sign
5 : **for** $j \in \mathsf{act}$ **do**
6 : $\quad \rho_j := H(j, \mathsf{msg}, (u, R_u, S_u)_u)$
7 : $w := \prod_{j \in \mathsf{act}} R_j \cdot S_j^{\rho_j}$
8 : $c := H(w, \mathsf{msg})$
9 : $z_i := c \cdot \mathsf{sk}_i \cdot L_{\mathsf{act},i} + r_i + \rho_i \cdot s_i$
10 : // Combine
11 : $z := \sum_{j \in \mathsf{act}} z_j$
12 : **return** (c, z)

(a) FROST (Schnorr-based).

[EKT24].Sign(act, msg, i, sk_i)

1 : // Round 1: Preprocessing
2 : $\mathbf{r}_{i,1}, \ldots, \mathbf{r}_{i,\mathsf{rep}} \leftarrow \chi_{\mathbf{r}}$
3 : **for** $b \in [\mathsf{rep}]$ **do** $\mathbf{w}_{i,b} := [\mathbf{A} \mid \mathbf{I}] \cdot \mathbf{r}_{i,b}$
4 : // Round 2: Sign
5 : $(\beta_2 \| \ldots \| \beta_{\mathsf{rep}}) := H(\mathsf{msg}, \mathsf{act}, (\mathbf{w}_{j,b})_{j,b})$
6 : **for** $j \in \mathsf{act}$ **do** $\mathbf{w}_j := \mathbf{w}_{j,1} + \sum_{b>1} \beta_b \cdot \mathbf{w}_{j,b}$
7 : $\mathbf{w} := \sum_{j \in \mathsf{act}} \mathbf{w}_j$; $c := H(\mathbf{w}, \mathsf{msg})$
8 : $\Delta_i \leftarrow \mathsf{ZeroShare}(i, \mathsf{transcript})$ // $\sum \Delta_j = 0$
9 : $\mathbf{z}_i := c \cdot L_{\mathsf{act},i} \cdot \mathsf{sk}_i + \mathbf{r}_{i,1} + \sum_{b>1} \beta_b \cdot \mathbf{r}_{i,b} + \Delta_i$
10 : // Combine
11 : $\mathbf{z} := \sum_{j \in \mathsf{act}} \mathbf{z}_j$
12 : **return** $(c, \mathbf{z})$

(b) [EKT24] (Lattice-based). The β_b are sampled from the set of signed monomials $\mathbb{T} = \{\pm X^u \mid u \in [n]\} \subset R$.

Figure 3.9: Comparison of 2-round signing in FROST [KG20] and [EKT24]. Both prevent rushing attacks by forcing parties to commit to multiple nonces, and later randomly combine them.

ATTACKS AGAINST THE CORRECTNESS. While noise flooding solves the problem of distributing the signature generation for arbitrary thresholds $T \leqslant N$, even when N is large, it remains an open problem to efficiently deal with parties attempting to abort the protocol by sending malformed messages. While in the classical equivalents Sparkle and FROST, the linearity of the operations allows to easily detect and identify malicious parties, in the lattice setting, the masking techniques used to protect the secret shares also hide malicious behavior. Even the less efficient schemes using homomorphic encryption or linear secret sharing with small reconstruction coefficients are challenging to secure against such attacks, as the signing shares are large and thus one cannot easily verify that parties sampled short nonces. The only known solution to this problem is to use heavy Non-Interactive Zero-Knowledge (NIZK) proofs to verify the correctness of the partial signatures [ASY22; GKS24; GHKS+25; PKNR+25].

In this thesis, we introduce efficient solutions to this problem, aiming for robustness in Chapter 4 and identifiable aborts in Chapter 5. Note that our technique for identifiable aborts also applies to the FROST-like schemes, but we focus on the simpler TRaccoon for clarity, as the techniques we introduce are orthogonal to the specific signing algorithm used.

DISTRIBUTED KEY GENERATION. Most of the lattice-based threshold signature schemes discussed above assume a trusted setup where a dealer generates the secret key and distributes shares to the parties. While acceptable for some applications, a fully decentralized trust model requires a Distributed Key Generation (DKG) protocol where parties collaboratively generate the key pair. For lattice-based schemes, DKG is particularly challenging because the final secret key sk must be short while being shared among parties. Prior works have either relied on generic MPC protocols, which are inefficient, or on HE-based techniques, which also introduce signifi-

Scheme	Rounds	Communication	|vk|	|sig|	Robustness	IA	NIZK/HE	DKG
[GKS24]	2	> 1 MB	13.6 kB	46.6 kB	✗	✓	✓	✓
[GHKS+25]	3	563 kB	4.4 kB	11.3 kB	✗	✓	✓	✓
TRaccoon [DKMM+24]	3	41 kB	3.9 kB	12.7 kB	✗	✗	✗	✗
[EKT24]	2	276 kB	5.5 kB	10.8 kB	✗	✗	✗	✗
[BKLM+25]	2	612 kB	4.6 kB	13.7 kB	✗	✗	✗	✗
[CTZ24; ZT25]	2	1.2 MB	42 kB	≈ 140 kB	✗	✗	✗	✗
[GTJL+26]	2	27 kB	2.0 kB	17.7 kB	✗	✗	✗	✗
[PKNR+25]*	3	$88.2 + 6.4T$ kB	3.9 kB	12.7 kB	✗	✓	✓	✗
RB-Raccoon **(Chapter 4)**	4	$28 + 34T$ kB	4.6 kB	15.1 kB	✓	✗	✗	✓
IA-Raccoon **(Chapter 5)**	3	≈ 28 kB	4.3 kB	12.3 kB	✗	✓	✗	✓

Table 3.2: Comparison of noise flooding based threshold lattice signature schemes. We report parameters for NIST Level I security. Communication costs are given per party. * [PKNR+25] provides an *interactive* 3-round abort identification mechanism, deferring the heavy NIZK computations to after an execution of the signing protocol failed.

cant overhead. We will see in Chapter 4 that we can introduce efficient Verifiable Secret Sharing (VSS) techniques in the honest majority setting to *robustly* generate Shamir shares of a short secret key. For the malicious majority setting, we introduce in Chapter 5 a DKG protocol that generates shares for specialized *everywhere short secret sharing* schemes, particularly useful for noise flooding-based signatures, and that can be combined with our identifiable abort mechanism.

We summarize in Table 3.2 the properties of the noise flooding-based threshold signature schemes discussed above, including our own contributions in Chapter 4 and Chapter 5.

Hash-and-Sign Schemes.

While the above discussion focused on threshold schemes following the Fiat-Shamir paradigm, it is also possible to build efficient lattice-based threshold signatures following the Hash-and-Sign paradigm. This was demonstrated in our work [ENP24], introducing a threshold variant of the masking-friendly lattice signature scheme Plover [EENP+24] – which uses noise-flooding techniques akin to Raccoon. Additionally, Phoenix [JRS24] is an equivalent of Plover following the FSwA paradigm, and we believe it could be turned into a threshold scheme using the techniques of Chapter 6.

It is worth noting that the Hash-and-Sign paradigm does not seem to be as compact as the Fiat-Shamir paradigm for threshold signatures, notably due to more constrained parameterization (Plover works over R a power-of-two cyclotomic directly, and does not leverage higher module ranks). Still, it is an interesting direction for future work to explore the design space of Hash-and-Sign threshold signatures.

3.6 BASE SIGNATURE SCHEMES

In this section, we provide a more precise description of the two base signature schemes we will use in this thesis: ML-DSA [LDKL+22] and Raccoon [PKPR24], reintroducing in particular their rounding and hint mechanisms.

Detailed descriptions and notations for ML-DSA are deferred to Chapter 6, while we provide several shared notations and definitions for Raccoon here to facilitate the understanding of the subsequent chapters.

3.6.1 Modulus Rounding

Both signatures schemes rely on modulus rounding, a technique commonly used in lattice-based cryptography to reduce the size of elements.

We recall here the notation and definition of the modulus rounding of Raccoon. With an abuse of notation, we will use the same notation for the high-level description of ML-DSA. We refer to Chapter 6 for a description of the precise rounding mechanism used in ML-DSA.

Let $\nu \in \mathbb{N}^*$. Every $x \in \mathbb{Z}$ admits a unique decomposition $x = 2^\nu x_\top + x_\perp$, with $x_\perp \in [-2^{\nu-1}, 2^{\nu-1} - 1]$, and we define $\lfloor \cdot \rceil_\nu : \mathbb{Z} \to \mathbb{Z}$ by

$$\lfloor x \rceil_\nu = x_\top = \lfloor x/2^\nu \rceil, \qquad \lfloor t \rceil = \left\lfloor t + \tfrac{1}{2} \right\rfloor \text{ (half-up).}$$

If $q > 2^\nu$, we extend $\lfloor \cdot \rceil_\nu$ to $\mathbb{Z}_q$ by lifting $x \mapsto \bar{x} \in [0, q-1]$ and setting $\lfloor x \rceil_\nu = \lfloor \bar{x}/2^\nu \rceil \bmod q_\nu = (\bar{x})_\top \bmod q_\nu \in [0, \lfloor (q-1)/2^\nu \rfloor]$ where $q_\nu = \lfloor q/2^\nu \rceil$, and apply it coefficient-wise to vectors.

Remark 3.6.1. Unlike [dEKM+23; DKMM+24], we select $q_\nu = \lfloor q/2^\nu \rceil$ instead of $q_\nu = \lfloor q/2^\nu \rfloor$, which can allow to obtain slightly better parameters for Raccoon and TRaccoon.

3.6.2 ML-DSA Signatures

ML-DSA [LDKL+22] is one of the standard lattice-based signature schemes selected by NIST for post-quantum cryptography. It instantiates the Fiat-Shamir with Aborts paradigm over module lattices, relying on the hardness of the MLWE and MSIS problems.

At a high level, ML-DSA signatures work as follows.

KEY GENERATION. The public key consists of a (rounded) MLWE instance, i.e., a matrix $\mathbf{A} \in R_q^{k \times \ell}$, and a vector $\mathbf{b} = \left\lfloor \begin{bmatrix} \mathbf{A} & \mathbf{I} \end{bmatrix} \cdot \mathbf{s} \right\rceil_{\nu_\mathbf{b}}$, where $\mathbf{s} \in R_q^{\ell+k}$ is a secret vector uniformly sampled from a small hypercube.

SIGNING. To sign a message msg, the signer samples a random vector $\mathbf{y} \in R_q^\ell$ uniformly from a larger hypercube, computes the commitment $\mathbf{w} = \lfloor \mathbf{A}\mathbf{y} \rceil_{\nu_\mathbf{w}} \in R_q^k$, and derives a challenge c by hashing msg and $\mathbf{w}$. The signer then computes the response $\mathbf{z}^{(1)} = c \cdot \mathbf{s}^{(1)} + \mathbf{y}$, where $c = \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$. To ensure that signatures do not leak information about the secret key $\mathbf{s}$, the signer performs rejection sampling: if $\mathbf{z}$ is not within a target space, the signer discards the signature and restarts the signing process. The final signature is $(c, \mathbf{z}^{(1)}, \mathbf{h})$, where $\mathbf{h}$ is a hint vector that helps the verifier recompute $\mathbf{w}$ from c and $\mathbf{z}^{(1)}$.

VERIFICATION. To verify a signature $(c, \mathbf{z}^{(1)}, \mathbf{h})$ on a message msg, the verifier first checks that $\mathbf{z}^{(1)}$ and $\mathbf{h}$ are sufficiently small. Then, it uses $\mathbf{h}$ to recompute $\mathbf{w}$ from c and $\mathbf{z}^{(1)}$, and finally checks that hashing msg and $\mathbf{w}$ yields c.

We refer the reader to Chapter 6 for a more detailed description of the mechanisms of ML-DSA, notations, and for our distributed key generation and signing protocols for ML-DSA.

3.6.3 Raccoon Signatures

Raccoon [PKPR24] is a *masking-friendly* lattice-based signature scheme built with the Fiat-Shamir paradigm. It aims at protecting against side-channel attacks with minimal performance overheads compared to standard lattice-based signatures.

Its core observation is that the rejection sampling step – notably used in the ML-DSA standard [LDKL+22] – to ensure statistical independence between the secret key and the signature is the main obstacle to efficient masking. Raccoon circumvents this by employing *noise flooding* in

place of rejection sampling. Noise flooding consists in adding a large amount of noise to the signature, ensuring that the signature distribution is sufficiently close to a distribution independent of the secret key to ensure security.

In their work, they rely on sums of uniform distributions for their flooding noise, and analyze the signature distribution with Rényi divergence techniques to prove security in the random oracle model. They show that Raccoon signatures can be masked efficiently while ensuring security in the t-probing model [ISW03]. Roughly, in addition to standard masking techniques, this is because at most t of the uniforms used in the flooding noise can be known to the adversary, and the remaining uniforms ensure sufficient entropy in the flooding noise to protect the secret key.

In broad terms, Raccoon signatures work as follows.

KEY GENERATION. The public key consists of a (rounded) MLWE instance, i.e., a matrix $\mathbf{A} \in R_q^{k \times \ell}$, and a vector $\mathbf{b}_\top = \left\lfloor \begin{bmatrix} \mathbf{A} & \mathbf{I} \end{bmatrix} \cdot \mathbf{s} \right\rceil_{\nu_\mathbf{b}}$, where $\mathbf{s} \in R_q^{\ell+k}$ is a secret vector sampled as the sum of small uniforms.

SIGNING. To sign a message msg, the signer samples a random vector $\mathbf{r} \in R_q^{\ell+k}$ as the sum of small uniforms (though significantly larger than for $\mathbf{s}$), computes the commitment $\mathbf{w} = \left\lfloor \begin{bmatrix} \mathbf{A} & \mathbf{I} \end{bmatrix} \cdot \mathbf{r} \right\rceil_{\nu_\mathbf{w}} \in R_q^k$, and derives a challenge c by hashing msg and $\mathbf{w}$. The signer then computes the response $\mathbf{z} = c \cdot \mathbf{s} + \mathbf{r}$, where $c = \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$. The final signature is $(c, \mathbf{z}^{(1)}, \mathbf{h})$, where $\mathbf{h} = \mathbf{w} - \left\lfloor \mathbf{A} \cdot \mathbf{z}^{(1)} - 2^{\nu_\mathbf{b}} \cdot \mathbf{b}_\top \right\rceil_{\nu_\mathbf{w}}$ is a hint vector that helps the verifier recompute $\mathbf{w}$ from c and $\mathbf{z}^{(1)}$.

VERIFICATION. To verify a signature $(c, \mathbf{z}^{(1)}, \mathbf{h})$ on a message msg, the verifier first checks that $(\mathbf{z}^{(1)}, 2^{\nu_\mathbf{w}} \cdot \mathbf{h})$ is sufficiently small. Then, it uses $\mathbf{h}$ to recompute $\mathbf{w}$ from c and $\mathbf{z}^{(1)}$, and finally checks that hashing msg and $\mathbf{w}$ yields c.

Raccoon differs from ML-DSA in several ways:

- The core difference is the use of noise flooding instead of rejection sampling, which allows for efficient masking.
- The secret key and randomness are sampled as sums of small uniforms, rather than uniformly from hypercubes.
- The rounding and hint mechanisms are chosen differently in Raccoon, to simplify the algorithms while preserving similar compactness.

Interestingly, when focusing on the signature scheme without masking, a straightforward variant of Raccoon signatures can rely on discrete Gaussians for the secret key and signing noise, rather than sums of uniforms. In this case, the resulting signature scheme can be proven secure under the Hint-MLWE assumption, with a reduction from MLWE. This provides tighter parameters than Rényi divergence techniques, and we use this variant throughout this thesis.

THRESHOLD-FRIENDLINESS OF Raccoon. del Pino et al. [DKMM+24] showed that Raccoon is particularly well-suited for threshold signatures. Indeed, the absence of rejection sampling allows to design simple distributed signing protocols, where each party contributes to the randomness $\mathbf{r}$, and the parties collaborate to compute the commitment $\mathbf{w}$, and the response $\mathbf{z}$ such that $\mathbf{z}$ always includes some honest randomness guaranteeing security.

A high-level blueprint of their seminal work – coined TRaccoon – is as follows when $T = N$ (i.e., all parties participate in signing):

TRUSTED KEY GENERATION. The secret key $\mathbf{s}$ is shared among the N parties using Shamir's secret sharing. Each party obtains a share $[\![\mathbf{s}]\!]_i$. The public key is computed as in the standard Raccoon scheme.

DISTRIBUTED SIGNING. To sign a message msg, the N parties proceed as follows:

- **Round 1:** Each party i samples a random vector $\mathbf{r}_i$ from a discrete Gaussian, and computes its commitment share $\mathbf{w}_i = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{r}_i$. The parties then broadcast a hash of their commitment share to each other. This prevents malicious parties from adapting their commitment shares after seeing those of honest parties.
- **Round 2:** After receiving the hashes from all parties, each party broadcasts its commitment share $\mathbf{w}_i$.
- **Round 3:** Finally, after receiving all commitment shares and verifying their hashes, the parties obtain an overall commitment by computing $\mathbf{w} = \left\lfloor \sum_{i=1}^{N} \mathbf{w}_i \right\rceil_{\nu_\mathbf{w}}$, derive the challenge c by hashing msg and $\mathbf{w}$, and compute their response share $\mathbf{z}_i = c \cdot L_i \cdot [\![\mathbf{s}]\!]_i + \mathbf{r}_i$, where L_i is the Lagrange coefficient of party i. The parties then broadcast their response shares $\mathbf{z}_i$.
- **Signature reconstruction:** The final signature $(c, \mathbf{z}^{(1)}, \mathbf{h})$ can be derived by combining the response shares: $\mathbf{z} = \sum_{i=1}^{N} \mathbf{z}_i$ and computing the hint vector $\mathbf{h}$.

As discussed in Section 3.5, [DKMM+24; KRT24] further showed that their scheme can be extended to the $T < N$ setting by randomizing the partial responses $\mathbf{z}_i$ using shares of zero, ensuring that only the sum of the partial responses is revealed to the adversary, and with signatures to authentify messages from the second round. They introduced a deterministic algorithm based on PRF to derive these shares of zero from a common seed based on the transcript.

TRaccoon is proven secure against static adversaries in the asynchronous communication model. It can support even large numbers of parties $N = 1024$ with practical performance, and signatures of size around 12.7 KB with NIST Level 1 security.

However, as discussed in the introduction, there remains several limitations of TRaccoon and its variants, which we aim to address in this thesis.

3.7 NOTATIONS AND USEFUL LEMMAS FOR Raccoon

Since Raccoon is a core building block of several constructions in this thesis, we introduce here some notations and useful lemmas that will be used throughout the following chapters.

3.7.1 Bounds on Modulus Rounding

We bound the induced error of our rounding mechanism (Lemma 3.7.1) and its behavior under addition (Lemma 3.7.2).

Lemma 3.7.1. *Let $q, \nu \in \mathbb{N}^*$, with $q > 2^\nu$ and $2^\nu \nmid q$. For $x \in \mathbb{Z}_q$,*

$$\left| x - 2^\nu \cdot \overline{\lfloor x \rceil_\nu} \right| \leqslant \begin{cases} 2^\nu - 1 & \text{if } q_\nu = \lfloor q/2^\nu \rceil \\ 2^{\nu-1} & \text{if } q_\nu = \lceil q/2^\nu \rceil \end{cases}$$

Recall that here $\overline{\lfloor x \rceil_\nu}$ is the lifting of $\lfloor \bar{x} \rceil_\nu \bmod q_\nu$ to $[0, q_\nu - 1]$.

Proof. Let $x \in \mathbb{Z}_q$. We uniquely decompose $\bar{x} = 2^\nu \cdot \bar{x}_\top + \bar{x}_\bot$, with $\bar{x}_\top \in [0, q_\nu]$ and $\bar{x}_\bot \in [-2^{\nu-1}, 2^{\nu-1} - 1]$.

Then,

$$\overline{\lfloor x \rceil_\nu} = \lfloor \bar{x} \rceil_\nu \bmod q_\nu = \begin{cases} 0 & \text{if } x_\top = q_\nu \\ \bar{x}_\top & \text{otherwise} \end{cases}$$

In the first case, we have:

$$\begin{aligned}\left|x - 2^\nu \cdot \overline{\lfloor x \rceil_\nu}\right| &= |x| &(2)\\ &= |x_\perp + 2^\nu \cdot q_\nu| &(3)\\ &= |x_\perp - q_\perp| &(4)\\ &\leqslant 2^\nu - 1 &(5)\end{aligned}$$

where we used the fact that $q = 2^\nu \cdot q_\nu + q_\perp$ with $q_\perp \in [-2^{\nu-1}, 2^{\nu-1} - 1]$.

Interestingly, this can even be improved to $2^{\nu-1}$ when $q_\nu = \lceil q/2^\nu \rfloor$, as in this case, we have $x_\perp < q_\perp \in [-2^{\nu-1}, -1]$ since $x_\top = q_\nu$ but $x = 2^\nu x_\top + x_\perp < q = 2^\nu x_\top + q_\perp$, and thus $|x_\perp - q_\perp| \leqslant 2^{\nu-1}$.

In the second case, we simply have $x - 2^\nu \cdot \overline{\lfloor x \rceil_\nu} \bmod q = x_\perp \in [-2^{\nu-1}, 2^{\nu-1} - 1]$, which concludes the proof. □

Lemma 3.7.2. *Let $q, \nu \in \mathbb{N}^*$ such that $q > 2^\nu$ and $2^\nu \nmid q$. For any $x, \delta \in \mathbb{Z}_q$, we have*

$$\left|2^\nu \cdot \left(\overline{\lfloor x + \delta \rceil_\nu - \lfloor x \rceil_\nu}\right) - \delta\right| \leqslant \begin{cases} 3 \cdot 2^{\nu-1} \cdot \sqrt{n} & \text{if } q_\nu = \lceil q/2^\nu \rfloor \\ (3 \cdot 2^{\nu-1} + q_\perp) \cdot \sqrt{n} & \text{if } q_\nu = \lfloor q/2^\nu \rfloor \end{cases}$$

Proof. The proof proceeds similarly to the second part of [DKMM+24, Lemma 3.2]. We will note $\overline{x}$ the lift of x to $\mathbb{Z}^n$ with coefficients in $[0, q-1]$.

We start by uniquely writing $\overline{x} = 2^\nu \cdot x_\top + x_\perp \bmod q$, where $(x_\top, x_\perp) \in [0, \lfloor q/2^\nu \rfloor] \times [-2^{\nu-1}, 2^{\nu-1} - 1]$. Similarly, we define $\delta_\top$ and $\delta_\perp$.

Now, remark that $\overline{x} + \overline{\delta}$ is bounded by $2(q-1)$. Hence, we can write,

$$\begin{aligned}\lfloor x + \delta \rceil_\nu &= \lfloor 2^\nu \cdot (x_\top + \delta_\top) + x_\perp + \delta_\perp - \alpha \cdot q \rceil_\nu \\ &= \lfloor 2^\nu \cdot (x_\top + \delta_\top - \alpha \cdot q_\nu) + x_\perp + \delta_\perp - \alpha \cdot q_\perp \rceil_\nu\end{aligned}$$

where $\alpha \in \{0, 1\}$ such that $\alpha = 1$ if $x + \delta > q - 1$, and $\alpha = 0$ otherwise. Put differently, we reduce $x + \delta$ modulo q. This reduced value is the one we round as per the definition of the rounding procedure over $\mathbb{Z}_q$.

We can then explicit the different values of $2^\nu \cdot \overline{\lfloor x + \delta \rceil_\nu - \lfloor x \rceil_\nu} - \delta$ by case analysis on the values of α and $\delta_\top$. Let us start by distinguishing the values taken by $\lfloor x + \delta \rceil_\nu$.

We observe that $x_\perp + \delta_\perp - \alpha \cdot q_\perp$ has coordinates bounded by $3 \cdot 2^{\nu-1} - 1$ in absolute value. This is in particular due to our choice $q_\nu = \lfloor q/2^\nu \rceil$ which ensures that $|q_\perp| < 2^{\nu-1}$. Hence, it will change the rounded value of $x_\top + \delta_\top - \alpha \cdot q_\nu$ by at most 1. Concretely, we can write:

$$\lfloor x + \delta \rceil_\nu = x_\top + \delta_\top - \alpha \cdot q_\nu + \gamma \bmod q_\nu$$

where $\gamma \in \{-1, 0, 1\}$ is a vector that depends on the value of $x_\perp + \delta_\perp - \alpha \cdot q_\perp$ as follows:

$$\gamma_i = \begin{cases} -1 & \text{if } x_\perp + \delta_\perp - \alpha \cdot q_\perp \leqslant -2^{\nu-1} - 1 \\ 0 & \text{if } x_\perp + \delta_\perp - \alpha \cdot q_\perp \in [-2^{\nu-1}, 2^{\nu-1} - 1] \\ 1 & \text{if } x_\perp + \delta_\perp - \alpha \cdot q_\perp \geqslant 2^{\nu-1} \end{cases}$$

We can then perform the reduction modulo q_ν over $\overline{\lfloor x + \delta \rceil_\nu - \lfloor x \rceil_\nu}$:

$$\begin{aligned}2^\nu \cdot \overline{\lfloor x + \delta \rceil_\nu - \lfloor x \rceil_\nu} - \delta &= 2^\nu \cdot \overline{(x_\top + \delta_\top - \alpha \cdot q_\nu + \gamma) - x_\top} - \delta \\ &= 2^\nu \cdot \overline{\delta_\top + \gamma} - \delta \\ &= 2^\nu \cdot ((\delta_\top + \gamma) - \varepsilon \cdot q_\nu) - \delta \\ &= 2^\nu \cdot \gamma + \varepsilon \cdot q_\perp - \delta_\perp\end{aligned}$$

where the second equality follows from $\alpha \cdot q_\nu = 0 \bmod q_\nu$ and the last equality follows from $q = 2^\nu \cdot q_\nu + q_\perp$ and $\delta = 2^\nu \cdot \delta_\top + \delta_\perp$. We define $\varepsilon \in \{0, 1\}$ depending on whether $q_\nu = \lfloor q/2^\nu \rfloor$ or $q_\nu = \lceil q/2^\nu \rfloor$.

CASE $q_\nu = \lfloor q/2^\nu \rceil$. In this case, we define ε as in [DKMM+24]:

$$\varepsilon = \begin{cases} 0 & \text{if } \delta_\top + \gamma \in \{0, ..., q_\nu - 1\} \\ 1 & \text{if } (\delta_\top, \gamma) \in \{(q_\nu - 1, 1), (q_\nu, 0), (q_\nu, 1)\} \end{cases}$$

Note that it is not possible that $(\delta_\top, \gamma) = (0, -1)$, i.e. $\varepsilon = -1$ since in that case we would have $\delta_\perp \geqslant 0$ as we use the canonical non-negative representatives for the liftings. From the definition of γ, we would also have $\alpha = 1, q_\perp \geqslant 0$ and $x_\perp + \delta_\perp < 0$ in this case for γ to be -1, and thus $x + \delta = 2^\nu \cdot x_\top + (x_\perp + \delta_\perp) < 2^\nu \cdot x_\top < 2^\nu \lfloor q/2^\nu \rceil$, which contradicts the fact that $x + \delta > q - 1$ when $\alpha = 1$.

CASE $q_\nu = \lceil q/2^\nu \rceil$. In this case, the only possible wrap-arounds are when $(\delta_\top, \gamma) = (0, -1)$ or $(\delta_\top, \gamma) = (q_\nu - 1, 1)$, and we define ε as follows:

$$\varepsilon = \begin{cases} -1 & \text{if } (\delta_\top, \gamma) = (0, -1) \\ 0 & \text{if } \delta_\top + \gamma \in \{0, ..., q_\nu - 1\} \\ 1 & \text{if } (\delta_\top, \gamma) = (q_\nu - 1, 1) \end{cases}$$

Now we bound the norm. We have

$$\begin{aligned} \|2^\nu \cdot \overline{\lfloor x + \delta \rceil_\nu - \lfloor x \rceil_\nu} - \delta \bmod q\| &= \|2^\nu \cdot \gamma + \varepsilon \cdot q_\perp - \delta_\perp \bmod q\| \\ &\leqslant \|2^\nu \cdot \gamma + \varepsilon \cdot q_\perp - \delta_\perp \bmod q\|_\infty \cdot \sqrt{n}. \end{aligned}$$

Thus it is sufficient to bound the infinity norm of $2^\nu \cdot \gamma + \varepsilon \cdot q_\perp - \delta_\perp$.

When $q_\nu = \lfloor q/2^\nu \rceil$, we can naively bound it by $3 \cdot 2^{\nu-1} + q_\perp$.

When $q_\nu = \lceil q/2^\nu \rceil$, we can obtain a slightly better bound by performing a more careful case analysis. For the cases of $\gamma = 0$ or $\varepsilon = 0$, we have

$$\begin{aligned} \|2^\nu \cdot \gamma + \varepsilon \cdot q_\perp - \delta_\perp \bmod q\|_\infty &\leqslant \|2^\nu \cdot \gamma \bmod q\|_\infty + \|\varepsilon \cdot q_\perp - \delta_\perp \bmod q\|_\infty \\ &\leqslant 2^{\nu-1} + 2^\nu \\ &= 3 \cdot 2^{\nu-1}. \end{aligned}$$

From the definitions of (γ, ε), it remains the cases when both γ and ε are 1 or both are -1. We write

$$2^\nu \cdot \gamma + \varepsilon \cdot q_\perp - \delta_\perp = \pm(2^\nu + q_\perp) - \delta_\perp$$

which is smaller than $3 \cdot 2^{\nu-1}$ in absolute value since $|\delta_\perp| \leqslant 2^{\nu-1}$ and $q_\perp \leqslant 0$ as $q_\nu = \lceil q/2^\nu \rceil$.

Thus, for any case, we have $\|2^\nu \cdot \gamma + \varepsilon \cdot q_\perp - \delta_\perp \bmod q\|_\infty \leqslant 3 \cdot 2^{\nu-1}$.

Combining all the arguments, we deduce

$$\|2^\nu \cdot \overline{\lfloor x + \delta \rceil_\nu - \lfloor x \rceil_\nu} - \delta \bmod q\| \leqslant \begin{cases} 3 \cdot 2^{\nu-1} \cdot \sqrt{n} & \text{if } q_\nu = \lceil q/2^\nu \rceil \\ (3 \cdot 2^{\nu-1} + q_\perp) \cdot \sqrt{n} & \text{if } q_\nu = \lfloor q/2^\nu \rceil \end{cases}$$

This concludes this proof. □

Lemma 3.7.3. *Let $q, \nu \in \mathbb{N}^*$ such that $q > 2^\nu$ and $2^\nu \nmid q$. For any $x, \delta \in \mathbb{Z}_q$, we have*

$$|\lfloor x + \delta \rceil_\nu - \lfloor x \rceil_\nu \bmod q_\nu| \leqslant |\delta|/2^\nu + 3/2.$$

Proof. Recall from the proof of Lemma 3.7.2 that we can write $\lfloor x + \delta \rceil_\nu - \lfloor x \rceil_\nu = \delta_\top + \gamma \bmod q_\nu$ where $\gamma \in \{-1, 0, 1\}$.

We can rewrite $\delta_\top$ as $\delta_\top = \delta/2^\nu - \delta_\perp/2^\nu$. Hence, we have

$$\begin{aligned} |\lfloor x + \delta \rceil_\nu - \lfloor x \rceil_\nu \bmod q_\nu| &= |\delta_\top + \gamma \bmod q_\nu| \\ &\leqslant |\delta/2^\nu| + |\delta_\perp/2^\nu + \gamma| \\ &\leqslant |\delta|/2^\nu + 3/2 \end{aligned}$$

□

3.7.2 Correctness Bounds for Raccoon

We provide below a useful lemma to analyze the correctness of Raccoon-based threshold signatures. Specifically, we are interested in bounding the error introduced by the roundings of the public key and commitments, and the subsequent use of hint vectors.

Lemma 3.7.4. *Let $q, \nu_{\mathbf{b}}, \nu_{\mathbf{w}}$ be positive integers such that $q > \max(2^{\nu_{\mathbf{b}}}, 2^{\nu_{\mathbf{w}}})$. Additionally assume $\lfloor q/2^{\nu_{\mathbf{b}}} \rceil = \lceil q/2^{\nu_{\mathbf{b}}} \rceil$ and $\lfloor q/2^{\nu_{\mathbf{w}}} \rceil = \lceil q/2^{\nu_{\mathbf{w}}} \rceil$.*

Let $\mathbf{s}, \mathbf{r} \in R_q^{\ell+k}$ be arbitrary vectors. We define $\mathbf{b} = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{s}$, $\mathbf{b}_\top = \lfloor \mathbf{b} \rceil_{\nu_{\mathbf{b}}}$, $\mathbf{w} = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{r}$. Let $\mathbf{z} \in R_q^{\ell+k}$ verifying $\begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{z} = c \cdot \mathbf{b} + \mathbf{w}$ for c a challenge vector.

Then, for $\mathbf{h} = \lfloor \mathbf{w} \rceil_{\nu_{\mathbf{w}}} - \left\lfloor \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{z}^{(1)} - 2^{\nu_{\mathbf{b}}} \cdot c \cdot \mathbf{b}_\top \right\rceil_{\nu_{\mathbf{w}}}$, we have

$$\left\| 2^{\nu_{\mathbf{w}}} \cdot \mathbf{h} - \mathbf{z}^{(2)} \right\| \leqslant \sqrt{nk}(3 \cdot 2^{\nu_{\mathbf{w}}-1} + \omega \cdot 2^{\nu_{\mathbf{b}}-1}).$$

Proof. We have:

$$\begin{aligned}
\|2^{\nu_{\mathbf{w}}} \cdot \mathbf{h} - \mathbf{z}^{(2)}\| &= \left\| 2^{\nu_{\mathbf{w}}} \cdot \left(\lfloor \mathbf{w} \rceil_{\nu_{\mathbf{w}}} - \left\lfloor \mathbf{A}\mathbf{z}^{(1)} - 2^{\nu_{\mathbf{b}}} \cdot c \cdot \mathbf{b}_\top \right\rceil_{\nu_{\mathbf{w}}} \right) - \mathbf{z}^{(2)} \right\| \\
&= \left\| 2^{\nu_{\mathbf{w}}} \cdot \left(\lfloor \mathbf{w} \rceil_{\nu_{\mathbf{w}}} - \left\lfloor \left(c \cdot \mathbf{b} + \mathbf{w} - \mathbf{z}^{(2)} \right) - 2^{\nu_{\mathbf{b}}} \cdot c \cdot \mathbf{b}_\top \right\rceil_{\nu_{\mathbf{w}}} \right) - \mathbf{z}^{(2)} \right\| \\
&= \left\| 2^{\nu_{\mathbf{w}}} \cdot \left(\lfloor \mathbf{w} \rceil_{\nu_{\mathbf{w}}} - \left\lfloor \mathbf{w} + c\left(\mathbf{b} - 2^{\nu_{\mathbf{b}}} \cdot \mathbf{b}_\top\right) - \mathbf{z}^{(2)} \right\rceil_{\nu_{\mathbf{w}}} \right) - \mathbf{z}^{(2)} \right\| \\
&\leqslant 3 \cdot 2^{\nu_{\mathbf{w}}-1}\sqrt{nk} + \|c \cdot (\mathbf{b} - 2^{\nu_{\mathbf{b}}} \cdot \mathbf{b}_\top)\| \quad \text{(applying Lemma 3.7.2)} \\
&\leqslant 3 \cdot 2^{\nu_{\mathbf{w}}-1}\sqrt{nk} + \omega \|\mathbf{b} - 2^{\nu_{\mathbf{b}}} \cdot \mathbf{b}_\top\| \\
&\leqslant \sqrt{nk}(3 \cdot 2^{\nu_{\mathbf{w}}-1} + \omega \cdot 2^{\nu_{\mathbf{b}}-1}) \quad \text{(applying Lemma 3.7.1)}
\end{aligned}$$

where the second equality comes from the fact that $\mathbf{A} \cdot \mathbf{z}^{(1)} + \mathbf{z}^{(2)} = c \cdot \mathbf{b} + \mathbf{w}$. □

4 LIGHTWEIGHT VERIFIABLE SECRET SHARING AND ROBUSTNESS

In this chapter, we focus on the problem of achieving robustness in lattice-based threshold signature schemes (TSS) and distributing the key generation (DKG). We propose a new framework based on *random submersions*—that is a projection onto a random subspace blinded by a small Gaussian noise—for constructing verifiable short secret sharing and leverage it to construct efficient robust threshold lattice-based signatures, when based on *noise flooding*. This leads to the introduction of the first *robust* threshold variant of Raccoon [PKPR24], together with a DKG, and the most efficient robust lattice-based TSS to date.

4.1 INTRODUCTION

Although significant progress has been made in the design of efficient lattice-based threshold signatures, notably with the introduction of TRaccoon [DKMM+24], several challenges remain before such schemes can be deployed broadly in practice.

In particular, the security requirements of these protocols are nuanced, and require mechanisms to address potential misbehavior. While some applications are satisfied as long as dishonest parties cannot produce signatures on their own (i.e., unforgeability), it is often important to prevent attacks against correctness, such as ensuring that the protocol outputs a valid signature at the end of its execution. This property is referred to as *robustness*. It has been extensively studied in the classical context – as detailed in Section 3.5 (e.g., robust threshold RSA [GRJK00], DSS [GJKR96] and robust FROST [RRJS+22]) – however, the exploration of post-quantum solutions for robustness is relatively recent. The only[1] known construction [BKP13] for robust lattice threshold signatures relies on generic MPC techniques, and is not practical. In contrast, the efficient schemes based on noise flooding [DKMM+24] do not achieve robustness.

In the classical setting, a natural way to implement a robust DKG and signing is to use a *Verifiable Secret Sharing* (VSS) scheme [CGMA85], which allows a dealer to share a secret key share while parties can verify that the shares they receive are consistent with a valid secret. This concept was later extended to *publicly verifiable secret sharing* (pVSS) by Stadler [Sta96], allowing any party to verify the validity of shares distributed by a dealer. One can simply have each party act as a dealer, sharing a freshly generated secret key share using VSS, and then aggregate the shares from the intersection of trusted parties. We can expect that a similar approach could be used in the lattice setting, although, in that setting it is crucial to ensure that the shares are not only consistent, but also that the underlying secret is *short*, as required for the correctness of the signature scheme.

Several VSS have been proposed for the classical setting [BGW88; Ped92; ABCP23; KGS23] (see [ABCP23; KGS23] for comparative introductions). For the lattice setting, Gentry, Halevi and Lyubashevsky [GHL22] proposed a lattice-based pVSS that includes zero-knowledge proofs for the shortness of the secret and consistency of the shares. However, their construction still relies on discrete-log assumptions to compress the proof size, making it unsound in the presence of quantum adversaries. They left as an open question the design of a practical fully post-quantum pVSS with shortness proof.

1 Ji et al. [JTZ23] also proposed a robust threshold signature scheme based on Fiat-Shamir with Aborts. However, its security proofs appear incomplete or flawed, as they implicitly assume the success probability to remain $1/M$ even conditioned on the value p' – normally internal – that they reveal. Note also that it has large signatures and requires the participation of all the non-corrupted users.

In short, it seems that we can get *efficient* threshold from lattices at the cost of losing robustness, or we can get *robust* threshold at the cost of relying on highly inefficient primitives. This leaves the following question:

Can we reconcile efficiency and robustness in the post-quantum era, using lattices?

4.1.1 Our Contributions

We study and propose a solution for robust (short) secret sharing, turn it into a robust distributed key generation (DKG), and as a byproduct get the first practical lattice-based robust threshold signature using noise flooding. Before further presentation, we emphasize that all our primitives rely only on *standard lattice assumptions* and do not require advanced additional primitives such as (threshold) FHE or NIZK.

LATTICE-BASED VERIFIABLE (SHORT) SECRET SHARING: We propose a technique called "random submersion" for secret sharing, enabling the secure generation and distribution of Learning With Errors (LWE) samples. This approach provides a form of proof to affirm the shortness of the underlying shared secrets in a distributed environment. These proofs are not exactly zero-knowledge, but we ensure the secret preserves sufficient uncertainty for security with noise flooding techniques. Our proposal is inspired by [ABCP23] for its syntax and challenge sampling, and [GHL22] for its compactness proof, employing Johnson-Lindenstrauss-type projections. We adopt a pragmatic approach for our VSS and prove approximate shortness only, with a small loss factor in the norm proven of about 10. This small loss allows us to improve over the practicality of [GHL22] by removing complex zero-knowledge proofs – which are computationally intensive – and the use of classical hardness assumptions in their instantiation.

ROBUST DISTRIBUTED KEY GENERATION: Building upon our V3S, we build a *robust* DKG. The protocol is *three-round*, *synchronous*, and uses a general complaint round to reach a consensus on the trustable parties.

More precisely, we propose a broadcast protocol where each party performs its own verifiable secret sharing via the V3S protocol highlighted above. Next, each party broadcasts its list of trustees, and the (public) intersection of these lists is computed, effectively forming a public clique of mutually trusted parties. Finally, each party aggregates the shares they received from trusted parties, which yields a secret sharing of a jointly generated short secret key, thanks to the linearity of Shamir's secret sharing.

While this solution has a nonnegligible communication cost (as the proofs are sent to every party by every party), it ensures maximal robustness for a low number of rounds.

ROBUST THRESHOLD SIGNATURES: We can adapt this methodology, almost *verbatim*, to construct threshold signatures. This follows from the fact that in the underlying signature scheme[2], the signing algorithm is essentially the key generation procedure with extra steps and slightly different parameters. As a consequence, the robustness of such signatures follows from similar arguments to the ones used for the DKG protocol. Our approach relies on the *noise-flooding* technique to create signatures; here the *flooding* involves linearly concealing secret values with sufficiently large noise and ensuring security by quantifying the residual statistical leakage.

We instantiate our framework in RB-Raccoon, a robust threshold variant of the Raccoon signature scheme featuring a 3-round distributed key generation and a 4-round signing protocol.

2 In our case this scheme is the Fiat-Shamir scheme Raccoon, but the same comment would apply to the Hash-and-Sign signature Plover.

While RB-Raccoon builds upon the digital signature scheme Raccoon, we emphasize that our robust threshold signature and distributed key generation constructions are generic and can be applied to other noise-flooding based schemes, such as the Hash-and-Sign scheme Plover [EENP+24]. This was achieved in the published version of this work [ENP24].

Before turning to the details and security proofs of all these protocols, we propose a high-level overview of the main technicalities, caveats, and ideas used further.

4.2 TECHNICAL OVERVIEW

We now turn to a high-level introduction of our techniques and protocols. Our first contribution is a proposal for a lattice-based verifiable short secret sharing scheme, for which we can control its leakage very precisely, but which is not technically zero-knowledge. Building on this framework, we show how to extend it to devise a protocol for robust distributed key generation and robust threshold signatures. All of these are secure under *standard* lattice assumptions and *do not* require more advanced primitives such as FHE or NIZK (Non-Interactive Zero-Knowledge proofs).

4.2.1 A Lattice Verifiable Short Secret Sharing Proposal

Many lattice-based cryptographic schemes hinge on variants of the Learning With Errors (LWE) assumption, which is crucially based on the shortness of some secret elements. In particular, in the context of devising a robust threshold scheme, the verification of this shortness is critical. Addressing this challenge, we introduce a new technique coined *random submersion*. This method enables the secure generation and distribution of an LWE sample, concurrently providing a proof to confirm the shortness of the sample. It ensures both the integrity and the confidentiality of the secret, and we will leverage the technique to ensure the robustness of our threshold scheme in Section 4.5.2. Our technique aims for practicality, and only provides approximate shortness bounds in exchange for simple and mostly linear algorithms, and lighter computation cost than prior work [GHL22]. *Random submersion* relies on a Johnson-Lindenstrauss type lemma, blinded by a Gaussian noise for confidentiality. This type of lemma was already successfully applied for verifiable secret sharing with approximate shortness[3] – relying on rejection sampling – in [GHL22, Section 3.4], but differs in the distribution of its blindings, which removes the need for rejection sampling and tightens their approximate shortness proofs.

We present a tweakable framework for securely distributing a small secret vector $\mathbf{x}$ among N parties, with the provision that a maximum of $t < N/2$ among these parties may be untrustworthy. Our approach revolves on the sharing of secrets that are derived from a Gaussian distribution and also accommodates secret vectors chosen by adversaries, provided these vectors have a norm capped at B. We name our secret sharing technique V3S (Verifiable Short Secret Sharing).

Methodology.

The cornerstone of our method is to construct a scheme *as linear as feasible*, to ensure compatibility with lattice-based operations. Our initial step involves defining the most fundamental requirements for our scheme, starting with the secret itself.

1. The secret $\mathbf{x}$ is represented as a vector of small norm, which is then distributed linearly into a series of Shamir shares.

3 [GHL22] includes a mechanism to upgrade their approximate shortness proofs to exact shortness proofs by additionally proving non-linear relations. We avoid doing this here to preserve efficiency and avoid complex zero-knowledge proofs.

2. The verification of the secret's smallness is achieved through a random linear transformation applied to itself.

3. To prevent excessive leakage of the secret by the output of this transformation, we blind it by incorporating an additive mask within its image space.

4. The choice of the randomness used to construct the transformation shall be verifiable by the parties, and be independent of the previous steps.

Writing what precedes as an algorithm would yield a blueprint of the following shape, further using a Merkle tree to verifiably derive the randomness after the previous steps are completed – similarly to the classical VSS from [ABCP23]:

1. The dealer samples an ephemeral Gaussian blinding value $\mathbf{y}$, that will be used within their individual proof to prove the shortness of $\mathbf{x}$ without leaking its value.

2. Then they generate a random Shamir secret sharing for $\mathbf{x}$ and $\mathbf{y}$ of order T, denoted $[\![\mathbf{x}]\!]$, and $[\![\mathbf{y}]\!]$.

3. To generate verifiable randomness, the dealer hashes the shares $[\![\mathbf{x}]\!]_{i\in[N]}$ and $[\![\mathbf{y}]\!]_{i\in[N]}$ into a Merkle tree, with a hash h as root. It also produces individual proofs prf_i allowing each party to verify that $[\![\mathbf{x}]\!]_i, [\![\mathbf{y}]\!]_i$ belongs to the tree.

4. It derives a challenge matrix $\mathbf{R} = \mathsf{H}_{\mathbf{R}}(h)$ from a suitable distribution, and computes the value $[\![\mathbf{v}]\!] = \mathbf{R} \cdot [\![\mathbf{x}]\!] + [\![\mathbf{y}]\!]$.

5. The proof $\pi = (h, [\![\mathbf{v}]\!])$ is broadcasted to all parties. Parties are additionally provided with designated individual proofs $\pi_i = ([\![\mathbf{y}]\!]_i, \mathsf{prf}_i)$, exchanged over secure pairwise channels.

Any party i, given its share $[\![\mathbf{x}]\!]_i$ and proofs $\pi = (h, [\![\mathbf{v}]\!])$ and $\pi_i = ([\![\mathbf{y}]\!]_i, \mathsf{prf}_i)$ performs the following verifications:

1. prf_i correctly proves that $[\![\mathbf{x}]\!]_i$ and $[\![\mathbf{y}]\!]_i$ are included in Merkle tree h.

2. Derive $\mathbf{R} = \mathsf{H}_{\mathbf{R}}(h)$ and verify that $[\![\mathbf{v}]\!]_i = \mathbf{R} \cdot [\![\mathbf{x}]\!]_i + [\![\mathbf{y}]\!]_i$.

3. Verify that $\|\mathbf{v}\|$ is smaller than some fixed bound B'.

Remark that in the context of sampling an MLWE sample, it will also be of interest to derive robustly the value $\mathbf{b} = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{x}$. Seeing Shamir's secret sharing as a Reed-Solomon code, each party i can broadcast their share $[\![\mathbf{b}]\!]_i = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot [\![\mathbf{x}]\!]_i$. Then, by gathering all shares $[\![\mathbf{b}]\!]_i$ from all parties, it is possible to reconstruct $\mathbf{b}$ by Reed-Solomon decoding when sufficiently many honest parties participated.

Guaranteeing soundness and correctness.

While the verification of the randomness is classical thanks to the Merkle tree, the crux of the shortness proof will lie in the following requirements:

SMALL SECRET IMPLIES SMALL PROOF | The proof $\mathbf{R} \cdot \mathbf{x} + \mathbf{y}$ must be small—with overwhelming probability—if both $\mathbf{x}$ and $\mathbf{y}$ are small, for instance when Gaussian is drawn.

BIG SECRET IMPLIES BIG PROOF | It must become large if $\|\mathbf{x}\|$ is compromised, that is when it is larger than the desired acceptance bound B.

FEW COLLISIONS | Obviously, for correctness, we can not tolerate too many collisions in the values, so we should not have too many pairs $(\mathbf{x}, \mathbf{y})$ being sent to the same value $\mathbf{R}\mathbf{x} + \mathbf{y} \bmod q$.

This means that from a geometric perspective, we are asking the matrix $\mathbf{R}$ to act as a random *pseudo-isometry* in the mod$q$ hypercube. This is very reminiscent of the Johnson-Lindenstrauss type lemma [JLS86], which has been successfully applied to the modular setting in cryptography in [GHL22]. In the following, we say that such distributions satisfy property G (see Definition 4.4.5 for a formal definition).

Adding zero-knowledge.

Revealing the pair $(\mathbf{R}, \mathbf{R} \cdot \mathbf{x} + \mathbf{y})$ *leaks* some statistical information on $\mathbf{x}$. However, when the dealer is honest, $\mathbf{y}$ is sampled from a Gaussian distribution of sufficiently large variance with regards to $\|\mathbf{R} \cdot \mathbf{x}\|$, so that this information leakage is very mitigated. More precisely, we can view $\mathbf{R} \cdot \mathbf{x} + \mathbf{y}$ as a hint on $\mathbf{x}$ and show that the distribution of $\mathbf{x}$ conditioned on this hint is essentially a Gaussian distribution with a slightly smaller variance (c.f. Section 4.3.4). In particular, this conditioned distribution can still be leveraged to prove the pseudorandomness of an MLWE sample derived from $\mathbf{x}$. This is the same intuition behind the reduction from MLWE to the Hint-MLWE problem that we recalled in Section 3.2.4.

4.2.2 A Proposal for Robust Secret Sharing and Robust DKG

From* V3S *to Distributed Secret Sharing.

Consider a scenario where N users aim to collectively generate a secret $\mathbf{x}$, and each obtain a share of the resulting secret. Each user $i \in \{1, \ldots, N\}$ will possess $[\![\mathbf{x}]\!]_i = \boldsymbol{P}(i)$, where $\boldsymbol{P}$ is an interpolation polynomial such that $\boldsymbol{P}(0) = \mathbf{x}$. To achieve this, we require each participant to generate their own share of the global secret and transmit it to all other participants. However, for reasons of robustness and security, direct broadcast of their share is inadvisable, as it would allow any eavesdropper to learn all the shares and, by extension, the secret. Instead, each participant divides their share into smaller shares (termed local shares) and distributes these local shares to others. To ensure each local share between parties i and j remains confidential to them, we assume the existence of symmetric encryption between each pair of parties.

This method alone does not guarantee robustness, nor does it prevent potential dishonesty in the generation of local shares. To address this, our approach involves using a V3S scheme to allow parties to verify the local shares they receive. If a proof of a local share fails – say, if party i detects an incorrect proof from party j – then i broadcasts a complaint against j along with a proof of the error, enabling *all parties* to verify the incorrectness of j's proof to i and exclude j from their list of trusted parties. After addressing these complaints, every honest party knows the other parties that shared a secret honestly, and the correct local shares, allowing them to aggregate the local shares they have received to form their share of the secret. The protocol proceeds as follows:

(V1) **Round 1 (Secret-Share Individual Secret).** Each party i:

a. Generates a short vector $\mathbf{x}_i$.

b. Utilizes the V3S *as dealer* to secret-share $\mathbf{x}_i$ and generate a distributed proof of shortness, denoting $[\![\mathbf{x}_i]\!]$ as the secret-sharing and $\pi_{i \to j}$ as the partial proof for j.

c. For each j, encrypts $[\![\mathbf{x}_i]\!]_j, \pi_{i \to j}$ under a symmetric key known only to i and j.

d. Broadcasts the encrypted proofs and shares.

(V2) **Round 2 (Verify and Complain).** Each party i:

a. Decrypts all secret shares $[\![(\mathbf{s}_j, \mathbf{e}_j)]\!]_i$ and partial proofs $\pi_{j \to i}$ sent by others.

b. Only keep secret shares with valid partial V3S proofs. We note valid_i the set of participants j with valid proofs, and $\mathsf{complaints}_i$ as those without.

c. Broadcasts $\mathsf{complaints}_i$ along with proof of incorrectness for the partial V3S proofs, e.g. provide material allowing other parties to decrypt and verify the values that j sent to i.

(V3) **Round 3 (Review Complaints).** Each party i:

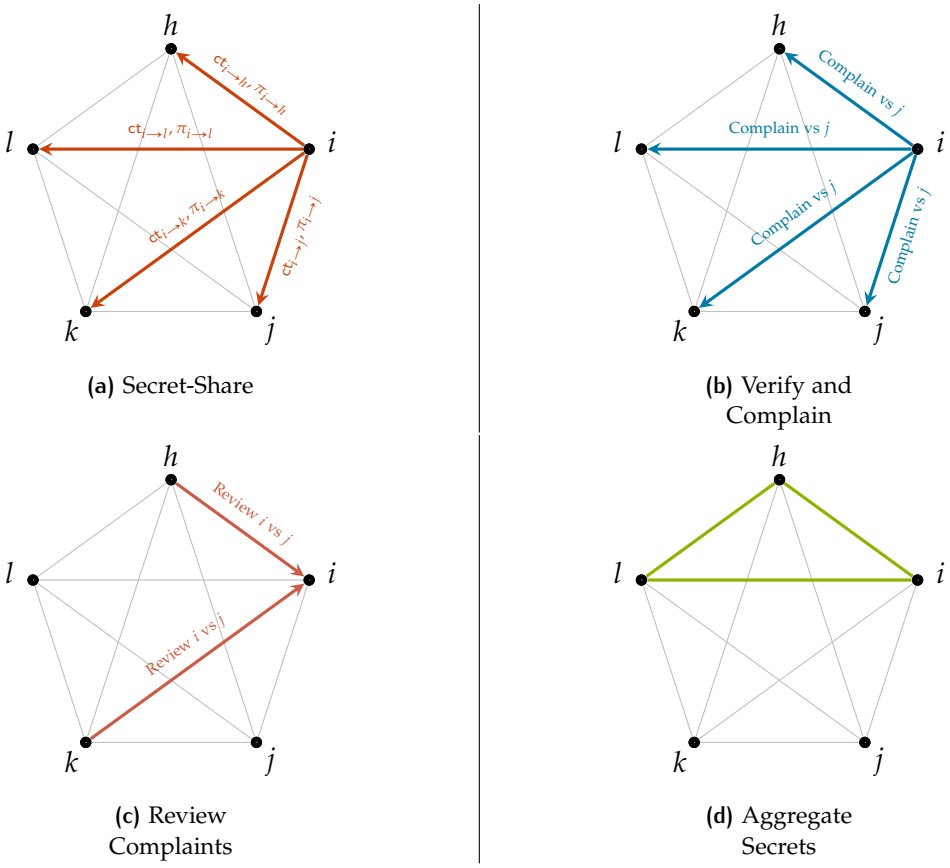
a. Receives $\mathsf{complaints}_j$ from user j, including the proof of incorrectness.

b. Removes any user from valid_i if proven invalid by j's complaint.

(V4) **Round 4 (Aggregate Valid Secrets).** Each party i, now with a confirmed list valid_i of valid users:

a. Aggregates the valid secret shares received: $[\![\mathbf{x}]\!]_i = \sum_{j\in\mathsf{valid}_i} [\![\mathbf{x}_j]\!]_i$, thus forming a secret sharing of $\mathbf{x} = \sum_{j\in\mathsf{valid}_i} \mathbf{x}_j$.

When the protocol ends, when a sufficient number of honest users participate, each honest user i holds a share $[\![\mathbf{x}]\!]_i$ of the aggregated secret $\mathbf{x} = \sum_{j\in\mathsf{valid}_i} \mathbf{x}_j$.



(a) Secret-Share

(b) Verify and Complain

(c) Review Complaints

(d) Aggregate Secrets

Figure 4.1: Our blueprint for secret-sharing a jointly generated short secret $\mathbf{x} = \sum_{i\in\mathsf{HS}} \mathbf{x}_i$. This structured underlies the distributed signing (Figs. 4.13 and 4.14) and key generation (Fig. 4.11) protocols of RB-Raccoon.

Turning a V3S *into a Robust DKG.*

Leveraging our robust secret sharing protocol, we can readily derive a robust distributed key generation (DKG) protocol. After all, a typical lattice key generation is essentially comprised of the generation of a short MLWE secret $\mathbf{x} = \mathbf{s}$ and a bit of salt to generate a random matrix $\mathbf{A}$. Our protocol is robust and secure as long as there are T users with $2t + 1 \leqslant T$ for the final reconstruction.

(K1) **Round 1.** Each participant i undertakes the following:

a. Generate, secret-share, prove, and encrypt the shares and partial proofs of a short secret vector $\mathbf{x}_i = \mathbf{s}_i$, following the procedure outlined in (V1).

b. Concurrently, generate an individual salt salt_i and broadcast it.

(K2) **Round 2.** Participant i continues by:

a. Decrypting and verifying the shares+proofs received, and broadcasting complaints against users who submitted invalid shares/proofs, mirroring the steps in (V2).

(K3) **Round 3.** Participant i proceeds to:

a. Review the complaints issued by all users, update valid_i accordingly, and compute an aggregate secret share $[\![\mathbf{s}]\!]_i = \sum_{j \in \mathsf{valid}_i} [\![\mathbf{s}_j]\!]_i$, akin to the method described in (V3).

b. Produce a public salt salt by hashing the salts of all users in valid_i.

c. Utilize salt to generate a public matrix $\mathbf{A}$, which is identical for all honest users. Calculate a partial public key $[\![\mathbf{b}]\!]_i = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot [\![\mathbf{s}]\!]_i$.

d. Broadcast the salt salt and the partial public key $[\![\mathbf{b}]\!]_i$.

(K4) **Round 4.** Participant i:

a. Retrieves the salt from the previous round, and in case of conflicting contributions, employ majority voting.

b. Reconstructs the secrets using the V3S for a robust reconstruction.

Upon completion, provided a sufficient number of honest users participate, the public key $\mathbf{A}, \mathbf{b} = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{s}$ can be recovered using the error-correcting features of Reed-Solomon codes/Shamir's secret sharing. Each honest participant i retains a share $[\![\mathbf{s}]\!]_i$ of $\mathbf{s}$. Our DKG is formally described and proven in Section 4.5.1.

4.2.3 Robust Threshold Lattice-Based Signature

Another application of our robust distributed secret-sharing protocol is threshold signing, wherein the secrets being shared and combined are the shares of the signature itself. Utilizing our technique, we can robustly adapt both signatures in the Hash-and-Sign paradigm and the Fiat-Shamir paradigm. For example, we can develop robust variants of both recent proposals Raccoon and Plover. In this section, we focus on presenting a *robust* threshold Fiat-Shamir protocol based on Raccoon, with its noise flooded signing drafted in Fig. 3.7. Specifically, we want to robustly sample a *short* nonce $\mathbf{r}$, compute the commitment $\mathbf{w} = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{r}$, and derive the response $\mathbf{z} = c \cdot \mathbf{s} + \mathbf{r}$ in a distributed manner.

Toward RB-Raccoon*: A Robust Threshold Fiat-Shamir signature.*

Now presenting our third application of the technique, we opt for a more condensed exposition. The principal idea remains unchanged from the distributed key generation: signers are tasked with generating and locally sharing their portion of what will become the nonce vector $\mathbf{r}$ – corresponding to the nonce from Raccoon, employing the V3S. Upon reaching a consensus on the nonce share (via a four-round protocol that identifies dishonest participants through a complaint round), each portion of the final signature is derived from the nonce share, given the linear nature of the Raccoon signing operations. We propose the following protocol:

(S1) **Round 1.** Each user i performs the following:

a. Generate, secret-share, prove, and encrypt the shares and partial proofs of a short nonce vector $\mathbf{r}_i$, akin to (V1).

(S2) **Round 2.** Each user i performs the following:

a. Decrypt and verify the shares and partial proofs received, broadcasting complaints against users who sent invalid shares/proofs, as in (V2). Complaints are also raised if user i fails to receive another user's share.

(S3) **Round 3.** Each user i performs the following:

a. Review complaints from all users, adjust valid_i accordingly, and compute an aggregate secret share $[\![\mathbf{r}]\!]_i = \sum_{j \in \mathsf{valid}_i} [\![\mathbf{r}_j]\!]_i$, as in (V3).

b. Compute a partial commitment $[\![\mathbf{w}]\!]_i = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot [\![\mathbf{r}]\!]_i$.

c. Broadcast valid_i and $[\![\mathbf{w}]\!]_i$.

(S4) **Round 4.** Each user i reconstructs $\mathbf{w} = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{r}$ from the shares $[\![\mathbf{w}]\!]_i$, derives a challenge c from $\mathbf{w}$, and concludes the signature as in the non-threshold scenario. The sole distinction lies in the necessity to calculate shares of $\mathbf{z}$ using the shares $[\![\mathbf{s}]\!]_i$ and $[\![\mathbf{r}]\!]_i$.

At the protocol's conclusion, if a sufficient number of honest users are present, a valid vector $\mathbf{z} = c \cdot \mathbf{s} + \mathbf{r}$ can be assembled using the error-correcting properties of Reed-Solomon codes. Consequently, $\mathsf{sig} = (c, \mathbf{z})$ constitutes a valid signature for the message msg under a Raccoon public key.

4.2.4 Some Open Problems and Directions

The protocol we constructed is synchronous as we must wait for each round to be completed, in particular for the complaint round. It would be in particular quite easy to perform a forking-like attack when we do not have all complaints in an asynchronous setting.

4.3 PRELIMINARIES

We reuse the general preliminaries from Chapter 3. We will in particular rely on computations mod$q$. Recall that we define the norm over $\mathbb{Z}_q^k$ and R_q^k as the norm of the lifted vectors in $\mathbb{Z}^{nk} \cap (-q/2, q/2]^{nk}$ (c.f. Chapter 3).

We introduce here additional notation and definitions specific to this chapter.

4.3.1 Merkle Trees

A *Merkle tree* is a binary tree where each leaf node contains the hash of a data block, and each non-leaf node contains the hash of the concatenation of its two child nodes. The root of the tree, known as the *Merkle root*, serves as a compact commitment on the entire dataset.

To prove that a specific data block is part of the dataset represented by the Merkle root, one can provide a *Merkle proof*. This proof consists of the hashes of the sibling nodes along the path from the leaf node containing the data block to the Merkle root. By iteratively hashing the data block with its sibling hashes, one can reconstruct the Merkle root and verify its authenticity.

In this work, we assume the access to a random oracle H_{MT} that models the hash function used in the Merkle tree construction.

We are further interested in a version of Merkle trees that support indexed leaves, ensuring that each leaf corresponds to a specific index in the dataset. This is easily achieved when the set

of indices is known in advance, by ordering the leaves according to their indices. This allows to uniquely identify each leaf by its index, and to provide Merkle proofs for specific indices.

We will rely on the following algorithms for indexed Merkle trees:

$\mathsf{MerkleTree}((d_i)_{i\in[m]}) \to (h, \mathsf{st}_{\mathsf{MT}})$. Given a set of data blocks $(d_i)_{i\in[m]}$, this algorithm constructs the indexed Merkle tree and returns the Merkle root h along with the internal state $\mathsf{st}_{\mathsf{MT}}$ of the tree.

$\mathsf{MerkleProof}(i, \mathsf{st}_{\mathsf{MT}}) \to \mathsf{prf}_i$. Given an index i and the internal state $\mathsf{st}_{\mathsf{MT}}$ of the Merkle tree, this algorithm generates the Merkle proof prf_i for the data block at index i.

$\mathsf{MerkleVerify}(h, i, d_i, \mathsf{prf}_i) \to (\mathsf{true} \mid \mathsf{false})$. Given the Merkle root h, an index i, a data block d_i, and a Merkle proof prf_i, this algorithm verifies whether d_i is indeed part of the Merkle tree represented by h using the proof prf_i. It returns true if the verification succeeds and false otherwise.

4.3.2 Shamir's Secret Sharing

We use the *Shamir secret sharing* scheme [Sha79] as linear random secret sharing. Let $N < q$ be an integer such that $(i-j)$ is invertible over $\mathbb{Z}_q$ for distinct $i, j \in [N]$ and $S \subseteq [N]$ be a set of cardinality at least $t+1$. For a set S and $i \in S$, the Lagrange polynomial $L_{S,i}(Y)$ for $i \in S$ is defined by

$$L_{S,i}(Y) := \prod_{j\in S\setminus\{i\}} \frac{Y-j}{i-j}.$$

We denote the Lagrange coefficient $L_{S,i}(0)$ by $L_{S,i}$.

For a secret $s \in \mathbb{Z}$, a $(t+1)$-out-of-N secret sharing $[\![s]\!]^q = ([\![s]\!]^q_i)_{i\in[N]}$ over $\mathbb{Z}_q$ is generated as follows: First, a degree t polynomial $P \in \mathbb{Z}_q[Y]$ such that $P(0) = (s \bmod q)$ is randomly chosen. After that, each share $[\![s]\!]^q_i$ is provided by $[\![s]\!]^q_i = P(i)$. We will denote such sharing by $\mathsf{SSS.Share}_q(s, N, t)$.

Given any $t+1$ shares $([\![s]\!]^q_i)_{i\in S}$, by Lagrange interpolation, we can reconstruct $P(Y)$ and $(s \bmod q)$ as

$$P(Y) = \sum_{i\in S} L_{S,i}(Y) \cdot [\![s]\!]^q_i \bmod q,$$
$$(s \bmod q) = \sum_{i\in S} L_{S,i} \cdot [\![s]\!]^q_i \bmod q.$$

We denote $[\![s]\!]^q \bmod q'$ the sharing $([\![s]\!]^q_i \bmod q')_{i\in[N]}$ for $q' \mid q$.

Remark 4.3.1. We may omit q when it is clear from the context and denote the sharing simply by $[\![s]\!] = ([\![s]\!]_i)_{i\in[N]}$.

Decoding with Errors.

We will use two procedures for decoding Shamir shares in the presence of errors.

$\mathsf{SSS.Decode}_q(([\![s]\!]^q_i)_{i\in S}) \to (s \bmod q) \mid \perp$. Given shares indexed by a set S with $|S| > t+1$, this algorithm reconstructs the secret and checks consistency. It defines a subset $S' \subset S$ made of the first $t+1$ indices in S, reconstructs $P(X) := \sum_{i\in S'} L_{S',i}(X) \cdot [\![s]\!]^q_i \bmod q$, and verifies that $P(i) = [\![s]\!]^q_i$ for all $i \in S\setminus S'$. If the checks pass, it outputs $P(0)$; otherwise, it outputs $\perp$.

$\mathsf{SSS.ErrorCorrect}_q(([\![s]\!]^q_i)_{i\in S}) \to ([\![\hat{s}]\!]^q_i)_{i\in S}$. Viewing Shamir sharing as a Reed–Solomon code, this algorithm corrects up to e erroneous shares when $|S| \geqslant t+2e+1$. It outputs a corrected sharing that preserves the non-erroneous shares.

Remark 4.3.2. We may omit the subscript q when it is clear from the context.

Generalizations.

We can generalize Shamir's secret-sharing over the ring R_q and over vectors $\mathbf{s} \in R_q^k$ by interpreting $\mathbf{s}$ as a coefficient vector over $\mathbb{Z}_q^{kn}$, and generating shares for each coefficient independently. We denote the resulting sharing as $[\![\mathbf{s}]\!]^q = ([\![\mathbf{s}]\!]_i^q)_{i\in[N]}$, where each share $[\![\mathbf{s}]\!]_i^q \in R_q^k$.

For $a \in R_q$, $a \cdot [\![\mathbf{s}]\!]^q$ denotes the sharing obtained by multiplying each share of $[\![\mathbf{s}]\!]^q$ by a.

4.3.3 Chinese Remainder Theorem

Given two coprime integers q_1, q_2, the Chinese Remainder Theorem (CRT) states that there exists an isomorphism between $\mathbb{Z}_{q_1\cdot q_2}$ and $\mathbb{Z}_{q_1} \times \mathbb{Z}_{q_2}$. Formally, given $a \in \mathbb{Z}_{q_1\cdot q_2}$, we denote $\mathsf{CRT}(a)$ the function that outputs the pair $(a \bmod q_1, a \bmod q_2)$. Conversely, given two integers $a_1 \in \mathbb{Z}_{q_1}$ and $a_2 \in \mathbb{Z}_{q_2}$, we denote $\mathsf{CRT}^{-1}(a_1, a_2)$ the integer $a \in \mathbb{Z}_{q_1\cdot q_2}$ such that $a \equiv a_1 \bmod q_1$ and $a \equiv a_2 \bmod q_2$.

This extends to the rings $R_{q_1\cdot q_2}$ and $R_{q_1} \times R_{q_2}$, as well as to vectors over these rings, by applying CRT and CRT^{-1} coefficient-wise.

4.3.4 Gaussian Samples with Hints

We recall an extension of a useful lemma from [KLSS23], adapted to use matrices inside the hints. This lemma allows to show that for a Gaussian secret $\mathbf{s}$ and some noises $\mathbf{y}_i$, revealing hints of the form $(\mathbf{M}_i \cdot \mathbf{s} + \mathbf{y}_i)_i$ simply changes the parameters of the Gaussian distribution of $\mathbf{s}$, without changing its nature. The matrix adaptation is formally proven in the published version of this chapter [ENP24].

Lemma 4.3.3. *Let $Q > 0$ be an integer, and $\sigma_{\mathsf{sk}}, (\sigma_{\mathbf{y}}^{(i)})_{i\in[Q]}$ be reals > 0. Take matrices $\mathbf{M}_0, ..., \mathbf{M}_{Q-1} \in \mathbb{Z}^{n(\ell+k)\times n(\ell+k)}$. We define $\mathbf{\Sigma}_0 := (\frac{1}{\sigma_{\mathsf{sk}}^2} \cdot \mathbf{I} + \sum_{i\in[Q]} \frac{1}{(\sigma_{\mathbf{y}}^{(i)})^2} \cdot \mathbf{M}_i^\top \mathbf{M}_i)^{-1}$. Then the following two distributions over $\mathcal{R} \times \mathbb{Z}^{n(\ell+k)\cdot Q}$ are statistically identical.*

1. $\left\{(\mathbf{s}, \mathbf{z}_0, ..., \mathbf{z}_{Q-1}) \mid \mathbf{s} \leftarrow D_{\mathbb{Z}^{n(\ell+k)}, \sigma_{\mathsf{sk}}}, \mathbf{y}_i \leftarrow D_{\mathbb{Z}^{n(\ell+k)}, \sigma_{\mathbf{y}}^{(i)}}, \mathbf{z}_i = \mathbf{M}_i \cdot \mathbf{s} + \mathbf{y}_i\right\}$
2. $\left\{(\hat{\mathbf{s}}, \mathbf{z}_0, ..., \mathbf{z}_{m-1}) \;\middle|\; \begin{array}{c} \mathbf{s} \leftarrow D_{\mathbb{Z}^{n(\ell+k)}, \sigma_{\mathsf{sk}}}, \mathbf{y}_i \leftarrow D_{\mathbb{Z}^{n(\ell+k)}, \sigma_{\mathbf{y}}^{(i)}}, \mathbf{z}_i = \mathbf{M}_i \cdot \mathbf{s} + \mathbf{y}_i, \\ \mathbf{c} = \mathbf{\Sigma}_0 \cdot \sum_{i\in[Q]} \frac{1}{(\sigma_{\mathbf{y}}^{(i)})^2} \mathbf{M}_i^\top \mathbf{z}_i, \hat{\mathbf{s}} \leftarrow D_{\mathbb{Z}^{n(\ell+k)}, \mathbf{c}, \sqrt{\mathbf{\Sigma}_0}} \end{array}\right\}$

Remark 4.3.4. In [KLSS23], $\mathbf{M}_i$ is the matrix corresponding to the multiplication by a polynomial $c \in R$.

4.3.5 Threshold Signatures

We reuse the syntax for distributed key generation and threshold signature schemes from Section 3.4.

While some works studied the abstraction of DKGs security properties [KGS23] in the honest majority setting, they consider unbiased key generation protocols. This is however hard to achieve for lattice-based schemes, and in particular, our DKG will allow the adversary to partly bias the distribution of the secret key. We thus chose to work as for the rest of this thesis with security notions considering distributed key generation together with the signature protocol.

We rely on the security definitions from Section 3.4, specifically the correctness definition and the unforgeability definition from Fig. 3.4 in the synchronous communication model.

We additionally introduce a game-based definition for the robustness property of threshold signature schemes. We consider a similar game as for unforgeability, except that the adversary now tries to make the honest parties abort, or output an invalid signature.

Definition 4.3.5 (Robustness of Threshold Signatures). *A threshold signature scheme* TS *with key generation and signing in the synchronous communication model with reliable broadcast is* robust *against t corrupted parties if for all PPT adversaries $\mathcal{A}$, the advantage*

$$\mathsf{Adv}^{\mathsf{TS\text{-}RB}}_{\mathcal{A}}(1^\lambda, N, T, t) = \Pr\left[\mathsf{Game}^{\mathsf{TS\text{-}RB}}_{\mathcal{A}}(1^\lambda, N, T, t) = 1\right]$$

is negligible in λ, where the game $\mathsf{Game}^{\mathsf{TS\text{-}RB}}_{\mathcal{A}}(1^\lambda, N, T, t)$ is defined in Fig. 4.2.

4.4 VERIFIABLE SHORT SECRET SHARING

We start with our new proposal for verifiable (short) secret sharing. As presented in Section 4.2.1, we introduce a new *random submersion* technique allowing one to distribute and prove the shortness of a secret among N parties. This shortness is crucial in the context of lattice-based cryptographic schemes based on variants of the Learning With Errors (LWE) assumption, which is based on the shortness of secret elements. Importantly, our technique controls the leakage on the secret and can be leveraged to sample short secrets in threshold schemes, as will be formalized for a threshold robust signature scheme, and a distributed key generation protocol in Section 4.5.

We consider secrets in R_q^k for some integer $k \geqslant 1$, that is shared with Shamir's secret sharing.

4.4.1 Security Notions

We first define in Definition 4.4.1 the syntax of a verifiable short secret sharing for Shamir's secret sharing over R_q^k, as well as standard security properties in Definitions 4.4.2 and 4.4.3.

Definition 4.4.1 (Verifiable Short Secret Sharing). *Let $\chi_{\mathbf{x}}$ be a distribution (corresponding to* honestly generated secrets*), and a set V of valid short secrets in R_q^k. Let $N > t$ be two nonnegative integers. Assume that we have N parties communicating and yet another distinguished party called* dealer*. A Verifiable Short Secret Sharing (*V3S*) is defined as a tuple of algorithms:*

- $\mathsf{V3S.Share}(N, t, \mathbf{x}) \to (\llbracket \mathbf{x} \rrbracket^q, \pi, \pi_1, ..., \pi_N)$*, run by the dealer, and produces secret shares $\llbracket \mathbf{x} \rrbracket^q$, a global proof π and individual proofs π_i for each party i.*
- $\mathsf{V3S.Verify}_i(\llbracket \mathbf{x} \rrbracket_i^q, \pi, \pi_i) \to (\mathsf{true} \mid \mathsf{false})$*, run by the party i given its share and proofs, and outputs a bit.*
- $\mathsf{V3S.Reconstruct}((\llbracket \mathbf{x} \rrbracket_i^q)_{i \in S}) \to (\mathbf{x} \mid \perp)$*, given $|S| \geqslant t+1$ shares where $S \subseteq [N]$, it outputs a reconstructed secret $\mathbf{x}' \in R_q$ if successful, $\perp$ otherwise.*

Definition 4.4.2 (V3S **Correctness).** *A* V3S *scheme is said to be* correct *if when* V3S.Share() *is honestly executed with a secret $\mathbf{x}$ sampled from the honest distribution $\chi_{\mathbf{x}}$, then verification correctly passes for all parties with overwhelming probability, and* V3S.Reconstruct() *correctly recovers the secret* (mod q) *for any subset of at least $t+1$ shares.*

Definition 4.4.3 (V3S **Soundness).** *A* V3S *scheme (with access to a random oracle* H*) is said* sound *if for any $S_{\mathsf{snd}} \subseteq [N]$ of cardinal at least $t+1$, any sharing $\llbracket \mathbf{x} \rrbracket$, with corresponding proofs π, π_i, verification will fail with overwhelming probability if either:*

- *shares are* inconsistent *among S_{snd}, i.e. reconstruction over shares of S_{snd} returns $\perp$. Then, at least one party in S_{snd} will fail verification.*

$\mathsf{Game}^{\mathsf{TS\text{-}RB}}_{\mathcal{A}}(1^\lambda, N, T, t)$

$L_{\mathsf{sid}} := \varnothing$
$(\mathsf{CS}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathsf{H}}(1^\lambda, N, T, t)$ // $\mathcal{A}$ chooses corrupted parties
assert $\mathsf{CS} \subseteq [N] \wedge |\mathsf{CS}| \leqslant t$
$\mathsf{HS} := [N] \backslash \mathsf{CS}$
$(\mathsf{st}_i)_{i\in[N]} \leftarrow \mathsf{Setup}(1^\lambda, N, T, t)$
for $r \in [R_{\mathsf{KG}}]$ **do**
  for $i \in \mathsf{HS}$ **do**
    $\mathsf{pm}^i_r, \mathsf{st}_i \leftarrow \mathsf{ShareKeygen}(i, \mathsf{st}_i, \mathsf{pm}_{r-1})$
    if $\mathsf{pm}^i_r = \bot$ **then return** 1
  $((\mathsf{pm}^i_r)_{i\in\mathsf{CS}}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathsf{H}}((\mathsf{pm}^i_r)_{i\in\mathsf{HS}}, \mathsf{st}_{\mathcal{A}})$
$(\mathsf{vk}, \mathsf{aux}) := \mathsf{CombineKey}((\mathsf{pm}_r)_{r\in[R_{\mathsf{KG}}]})$
if $(\mathsf{vk}, \mathsf{aux}) = \bot$ **then return** 1
for $i \in \mathsf{HS}$ **do** $\mathsf{sk}_i := \mathsf{PartialSecret}(\mathsf{st}_i)$
$\mathsf{sid}, (\mathsf{pm}^i_{R_{\mathsf{sig}}})_{\mathsf{CS}\cap\mathsf{act}} \leftarrow \mathcal{A}^{\mathsf{H},(\mathsf{OPerformRound}_r)_{r\in[R_{\mathsf{sig}}]}}(\mathsf{vk}, \mathsf{aux}, \mathsf{st}_{\mathcal{A}})$
$\{r, \mathsf{act}, \mathsf{msg}, (\mathsf{pm}^i_r)_{i\in\mathsf{HS}\cap\mathsf{act}}\}$ **from** $L_{\mathsf{sid}}[\mathsf{sid}]$
assert $r = R_{\mathsf{sig}}$
$\mathsf{sig} := \mathsf{Combine}(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, (\mathsf{pm}_r)_{r\in[R_{\mathsf{sig}}]})$
return $\neg\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$

$\mathsf{OPerformRound}_1(\mathsf{sid}, \mathsf{act}, \mathsf{msg})$

require $\mathsf{sid} \notin L_{\mathsf{sid}} \wedge |\mathsf{act}| = T$
for $i \in \mathsf{HS} \cap \mathsf{act}$ **do**
  $(\mathsf{pm}^i_1, \mathsf{st}_i) := \mathsf{ShareSign}_1(\mathsf{vk}, \mathsf{sid}, \mathsf{act}, \mathsf{msg}, i, \mathsf{sk}_i, \mathsf{st}_i, \mathsf{msg}, \bot)$
$L_{\mathsf{sid}}[\mathsf{sid}] = \{1, \mathsf{act}, \mathsf{msg}, (\mathsf{pm}^i_1)_{i\in\mathsf{HS}\cap\mathsf{act}}\}$
return $(\mathsf{pm}^i_1)_{i\in\mathsf{HS}\cap\mathsf{act}}$

$\mathsf{OPerformRound}_r(\mathsf{sid}, (\mathsf{pm}^i_{r-1})_{i\in\mathsf{CS}\cap\mathsf{act}})$, for $r = 2, ..., R_{\mathsf{sig}}$

$\{r', \mathsf{act}, \mathsf{msg}, (\mathsf{pm}^i_{r-1})_{i\in\mathsf{HS}\cap\mathsf{act}}\}$ **from** $L_{\mathsf{sid}}[\mathsf{sid}]$
require $r' = r - 1$
$\mathsf{pm}_{r-1} := (\mathsf{pm}^i_{r-1})_{i\in\mathsf{act}}$
for $i \in \mathsf{HS} \cap \mathsf{act}$ **do**
  $(\mathsf{pm}^i_r, \mathsf{st}_i) := \mathsf{ShareSign}_r(\mathsf{vk}, \mathsf{sid}, \mathsf{act}, \mathsf{msg}, i, \mathsf{sk}_i, \mathsf{st}_i, \mathsf{msg}, \mathsf{pm}_{r-1})$
$L_{\mathsf{sid}}[\mathsf{sid}] = \{r, \mathsf{act}, \mathsf{msg}, (\mathsf{pm}^i_r)_{i\in\mathsf{HS}\cap\mathsf{act}} \cup (\mathsf{pm}^i_r)_{i\in\mathsf{HS}\cap\mathsf{act}}\}$
return $(\mathsf{pm}^i_r)_{i\in\mathsf{HS}\cap\mathsf{act}}$

Figure 4.2: Robustness game for a threshold signature with key generation and signing in the *synchronous model with reliable broadcast*. We assume access to a hash oracle H modeled as a random oracle. By convention, $\mathsf{pm}_0 = \bot$. The adversary $\mathcal{A}$ wins if the game TS-RB returns 1, i.e. if it made the honest parties abort, or the final signature is invalid.

- *or the secret is* invalid, *i.e. the secret* $\mathbf{x}'$ *reconstructed from* $(\llbracket \mathbf{x} \rrbracket_i)_{i \in S_{\mathsf{snd}}}$ *does not belong to* V. *Then verification will fail for at least one party in* S_{snd}.

This is formalized as requiring any efficient adversary $\mathcal{A}$ *to win Game* V3S-sound *defined in Figure 4.3 with negligible probability.*

$\mathsf{Game}_{\mathsf{V3S\text{-}sound}}$

$L_H \coloneqq \varnothing$
// The adversary $\mathcal{A}$ chooses a subset S of parties to target
$N, t, S_{\mathsf{snd}} \leftarrow \mathcal{A}()$
assert $S_{\mathsf{snd}} \subseteq [N] \wedge |S_{\mathsf{snd}}| \geqslant t+1$ // S_{snd} must be large enough to allow reconstruction
// $\mathcal{A}$ produces a $t+1$-sharing among N parties
$(\llbracket \mathbf{x} \rrbracket_i)_{i \in S_{\mathsf{snd}}}, \pi, (\pi_i)_{i \in S_{\mathsf{snd}}} \leftarrow \mathcal{A}^{\mathsf{H}}(N, T, S_{\mathsf{snd}})$
if $\forall i \in S_{\mathsf{snd}}, \mathsf{V3S.Verify}(\llbracket \mathbf{x} \rrbracket_i, \pi, \pi_i) = \mathsf{false}$ **then**
  return 0 // If a party in S_{snd} fails verification, $\mathcal{A}$ loses
$\mathbf{x} = \mathsf{V3S.Reconstruct}((\llbracket \mathbf{x} \rrbracket_i)_{i \in S_{\mathsf{snd}}})$
if $\mathbf{x} = \perp$ **then**
  return 1 // Verification passes but shares are inconsistent in S_{snd}
if $\mathbf{x} \notin V$ **then**
  return 1 // Verification passes but the secret is invalid
return 0 // The sharing chosen by the adversary is valid

Figure 4.3: Soundness game for a V3S. $\mathcal{A}$ wins if the game V3S-sound returns 1.

We additionally need to prove that V3S proofs only leak limited information on the shared secret $\mathbf{x}$. We introduce the fragmentary knowledge property to formalize this requirement, where we ask that an adversary distinguishes the scenario where it receives the secret and up to t shares generated honestly, from the scenario where it receives simulated shares and proofs, without access to the secret, and the secret is sampled a posteriori from a distribution dependent on the previously simulated values, thus capturing the impact of the leaked information on the secret.

Definition 4.4.4 (Computational/Statistical V3S Fragmentary Knowledge). *A* V3S *has the fragmentary knowledge property if there exists two PPT simulators* $\mathsf{SimProof}(S) \to ((\llbracket \mathbf{x} \rrbracket_i)_{i \in S}, \pi, (\pi_i)_{i \in S})$, *defined for subsets* $S \subset [N]$ *of cardinality at most* t, *and* $\mathsf{SimSecret}(\pi)$ *such that the output distribution* $(\mathbf{x}, (\llbracket \mathbf{x} \rrbracket_i)_{i \in S}, \pi, (\pi_i)_{i \in S})$ *of the two processes*

- $((\llbracket \mathbf{x} \rrbracket_i)_{i \in S}, \pi, (\pi_i)_{i \in S}) \leftarrow \mathsf{SimProof}(S)$, *and* $\mathbf{x} \leftarrow \mathsf{SimSecret}(\pi)$
- *or* $\mathbf{x} \leftarrow \chi_{\mathbf{x}}$, $((\llbracket \mathbf{x} \rrbracket_i)_{i \in [N]}, \pi, (\pi_i)_{i \in [N]}) \leftarrow \mathsf{V3S.Share}(N, t, \mathbf{x})$

are (computationally/statistically) indistinguishable. This is formalized as requiring the winning advantage of any efficient adversary $\mathcal{A}$ *to win Game* V3S-fk *defined in Figure 4.4 to be at a negligible distance from 1/2.*

4.4.2 Our V3S Construction

The protocol drafted in the technical overview can straightforwardly be translated to pseudo-code. However, for a tighter concrete instantiation, we introduce a modulus q_{V3S} such that $q \mid q_{\mathsf{V3S}}$. This modulus will be used internally in the V3S algorithms, while the secret shares and reconstructed secret remain defined modulo q. The reason can be found when concretely instantiating the submersion matrix $\mathbf{R}$ (c.f. Lemma 4.7.1), as the modulus of the revealed projection

$\mathsf{Game}_{\mathsf{V3S\text{-}fk}^b}$, with $b \in \{0,1\}$

$L_H := \varnothing$
// The adversary chooses a subset S_{fk} of parties to target
$N, T, S_{\mathsf{fk}} \leftarrow \mathcal{A}()$
assert $S_{\mathsf{fk}} \subset [N] \wedge |S_{\mathsf{fk}}| = T - 1$
if $b = 0$ **then**
  $\mathbf{x} \leftarrow \chi_{\mathbf{x}}$ // Honest sharing of $\mathbf{x}$
  $(\llbracket \mathbf{x} \rrbracket_i)_{i\in[N]}, \pi, (\pi_i)_{i\in[N]} \leftarrow \mathsf{V3S.Share}(\mathbf{x})$
else
  // Simulation of the sharing transcript
  $(\llbracket \mathbf{x} \rrbracket_i)_{i\in S_{\mathsf{fk}}}, \pi, (\pi_i)_{i\in S_{\mathsf{fk}}} \leftarrow \mathsf{SimProof}(S_{\mathsf{fk}})$
  $\mathbf{x} \leftarrow \mathsf{SimSecret}(\pi)$
$b' \leftarrow \mathcal{A}^{\mathsf{H}}(\mathbf{x}, (\llbracket \mathbf{x} \rrbracket_i)_{i\in S_{\mathsf{fk}}}, \pi, (\pi_i)_{i\in S_{\mathsf{fk}}})$
return b'

Figure 4.4: Fragmentary Knowledge game for a V3S. We consider the distinguishing advantage of an adversary $\mathcal{A}$ between the games with $b = 0$, and $b = 1$.

$\mathbf{R} \cdot \mathbf{x} + \mathbf{y} \bmod q_{\mathsf{V3S}}$ should be sufficiently larger than the proven bound on the secret $\mathbf{x}$ to prevent an adversary from using overflowing to produce invalid proofs.

We also add high-entropy seeds to the values stored in the Merkle leaves. This ensures that it is infeasible for an adversary to recover the secret shares from the Merkle tree root, even if the secret had a low entropy.

We will use the notations $\chi_{\mathbf{x}}, \chi_{\mathbf{y}}$ to denote the distributions over R^k of the secret and noise vectors, respectively, and $\llbracket \mathbf{x} \rrbracket^{q_{\mathsf{V3S}}}, \llbracket \mathbf{y} \rrbracket^{q_{\mathsf{V3S}}}$ to denote their corresponding sharings modulo q_{V3S}. We introduce the distribution $\chi_{\mathbf{R}}$ for the random submersion matrix $\mathbf{R}$.

We present our construction in Figure 4.5. The syntax and formalization are inspired by the recent work of [ABCP23], as well as the use of hash functions to commit on secret shares and derive a challenge for verification. We formalize the requirements on the submersion matrices which were hinted in Section 4.2.1.

Definition 4.4.5 (Property G). *A distribution of matrices* $\mathbf{R}$ *is said to satisfy the property* G_{p_1,p_2,p_3} *if there exists two bounds* B, B' *such that we have:*

SEPARATION *if* $(\mathbf{x}, \mathbf{y}) \neq (\mathbf{x}', \mathbf{y}')$, $\Pr_{\mathbf{R}\leftarrow\chi_{\mathbf{R}}}[\mathbf{R}\mathbf{x} + \mathbf{y} \bmod q_{\mathsf{V3S}} = \mathbf{R}\mathbf{x}' + \mathbf{y}' \bmod q_{\mathsf{V3S}}] = p_1$ *with* $p_1 = \mathsf{negl}(\lambda)$*: the matrices* $\mathbf{R}$ *send different secrets to different points.*

LARGE NORM DETECTION *if* $\|\mathbf{x}\| > B$*, for any* $\mathbf{y}$, $\Pr_{\mathbf{R}\leftarrow\chi_{\mathbf{R}}}[\|\mathbf{R}\mathbf{x} + \mathbf{y} \bmod q_{\mathsf{V3S}}\| > B'] = 1 - p_2$ *with* $p_2 = \mathsf{negl}(\lambda)$*: intuitively, if* $\mathbf{x}$ *is large, we want the challenge to also be large with overwhelming probability.*

HONEST EXECUTION $\Pr_{\mathbf{x}\leftarrow\chi_{\mathbf{x}},\mathbf{y}\leftarrow\chi_{\mathbf{y}},\mathbf{R}\leftarrow\chi_{\mathbf{R}}}[\|\mathbf{R}\mathbf{x} + \mathbf{y} \bmod q_{\mathsf{V3S}}\| \leqslant B'] = 1 - p_3$ *with* $p_3 = \mathsf{negl}(\lambda)$*: in case of honest generation of the secret, the challenge is small.*

Assuming this crucial property in the construction, we show that our V3S construction is secure for the notions introduced in Section 4.4.1, over Gaussian distributed secrets. We classically work in the ROM and assume that hash functions are modeled as random oracles with output size 2λ.

Theorem 4.4.6 (Security of V3S in the ROM). *Assume* $\chi_{\mathbf{x}} = D_{R^k,\sigma_{\mathbf{x}}}$ *and* $\chi_{\mathbf{y}} = D_{R^k,\sigma_{\mathbf{y}}}$ *are Gaussian distributions over* R^k*.*

Let $Q_{\mathsf{H_{MT}}}$ *(resp.* $Q_{\mathsf{H_R}}$*) be the maximum number of queries allowed to the random oracle* $\mathsf{H_{MT}}$ *(resp.* $\mathsf{H_R}$*). For the* V3S *in Fig. 4.5, taking as set of valid secrets* $V = \{\mathbf{x} \mid \|\mathbf{x}\| \leqslant B\}$*, if the distribution of matrices* $\mathbf{R}$ *satisfies property* G_{p_1,p_2,p_3}*, then we have:*

V3S.Share$(N, t, \mathbf{x})$

$\mathbf{y} \leftarrow \chi_{\mathbf{y}}; (\mathsf{seed}_i)_{i\in[N]} \xleftarrow{\$} \{0,1\}^{N\cdot 2\lambda}$
$[\![\mathbf{x}]\!]^{q_{\mathsf{V3S}}} = \mathsf{SSS.Share}_{q_{\mathsf{V3S}}}(\mathbf{x}, N, t)$ // Produce $(t+1)$-out-of-N sharings for $\mathbf{x}$ and $\mathbf{y}$
$[\![\mathbf{y}]\!]^{q_{\mathsf{V3S}}} = \mathsf{SSS.Share}_{q_{\mathsf{V3S}}}(\mathbf{y}, N, t)$
// Build a Merkle tree over the shares
$h, \mathsf{st}_{\mathsf{MT}} \leftarrow \mathsf{MerkleTree}((([\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}}, [\![\mathbf{y}]\!]_i^{q_{\mathsf{V3S}}}, \mathsf{seed}_i)_{i\in[N]})$
$[\![\mathbf{x}]\!]^{q}, [\![\mathbf{x}]\!]^{q_{\mathsf{V3S}}/q} := \mathsf{CRT}([\![\mathbf{x}]\!]^{q_{\mathsf{V3S}}})$
for $i \in [N]$ **do**
  $\mathsf{prf}_i := \mathsf{MerkleProof}(i, \mathsf{st}_{\mathsf{MT}})$ // Proof of inclusion of leaf i in the Merkle tree
  $\pi_i := ([\![\mathbf{x}]\!]^{q_{\mathsf{V3S}}/q}, [\![\mathbf{y}]\!]_i^{q_{\mathsf{V3S}}}, \mathsf{seed}_i, \mathsf{prf}_i)$
$\mathbf{R} := \mathsf{H}_{\mathbf{R}}(h)$ // Hash h to obtain a random matrix from $\chi_{\mathbf{R}}$
$[\![\mathbf{v}]\!]^{q_{\mathsf{V3S}}} := \mathbf{R} \cdot [\![\mathbf{x}]\!]^{q_{\mathsf{V3S}}} + [\![\mathbf{y}]\!]^{q_{\mathsf{V3S}}}$ // Johnson-Lindenstrauss
$\pi := (h, [\![\mathbf{v}]\!]^{q_{\mathsf{V3S}}})$ // Challenge polynomial and Merkle tree root
return $([\![\mathbf{x}]\!]^{q}, \pi, (\pi_i)_{i\in[N]})$

V3S.Verify$([\![\mathbf{x}]\!]_i^q, \pi, \pi_i)$

Parse $\pi := (h, [\![\mathbf{v}]\!]^{q_{\mathsf{V3S}}})$ and $\pi_i := ([\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}/q}, [\![\mathbf{y}]\!]_i^{q_{\mathsf{V3S}}}, \mathsf{seed}_i, \mathsf{prf}_i)$
$[\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}} := \mathsf{CRT}^{-1}([\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}/q}, [\![\mathbf{x}]\!]_i^{q})$ // Convert back to modulus q_{V3S}
$\mathbf{v} := \mathsf{SSS.Decode}([\![\mathbf{v}]\!]^{q_{\mathsf{V3S}}})$ // See Section 4.3.2
if $\neg\mathsf{MerkleVerify}(h, i, ([\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}}, [\![\mathbf{y}]\!]_i^{q_{\mathsf{V3S}}}, \mathsf{seed}_i), \mathsf{prf}_i)$ **then**
  return false
$\mathbf{R} := \mathsf{H}_{\mathbf{R}}(h)$
if $\left([\![\mathbf{v}]\!]_i^{q_{\mathsf{V3S}}} \neq \mathbf{R} \cdot [\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}} + [\![\mathbf{y}]\!]_i^{q_{\mathsf{V3S}}}\right) \vee (\|\mathbf{v}\| > B')$ **then**
  return false // Check the consistency of shares, and shortness of secret vector
return true

V3S.Reconstruct$((([\![\mathbf{x}]\!]_i^q)_{i\in I})$, with $|I| \geqslant t+1$

return $\mathsf{SSS.Decode}((([\![\mathbf{x}]\!]_i^q)_{i\in I})$ // See Section 4.3.2

Figure 4.5: Algorithms for our Verifiable Short Secret Sharing (V3S).

- ***Correctness*** *with probability* $1 - p_3$.
- ***Soundness*** *with an advantage of at most:* $\frac{Q_{\mathsf{H}_{\mathsf{MT}}}^2}{2^{2\lambda-1}} + Q_{\mathsf{H}_{\mathbf{R}}} \cdot (p_1 + p_2)$
- ***Fragmentary knowledge*** *with an advantage of at most* $N \cdot \frac{Q_{\mathsf{H}_{\mathsf{MT}}}}{2^{2\lambda}}$, *where for a set* $|S_{\mathsf{fk}}| = t$ *the simulators are defined in Fig. 4.6.*

$\mathsf{SimProof}^{\mathsf{H}_{\mathsf{MT}},\mathsf{H}_{\mathbf{R}}}(S_{\mathsf{fk}})$

1 : **for** $i \in S_{\mathsf{fk}}$ **do**
2 : $\quad [\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}}, [\![\mathbf{y}]\!]_i^{q_{\mathsf{V3S}}} \xleftarrow{\$} R^k_{q_{\mathsf{V3S}}}, \mathsf{seed}_i \xleftarrow{\$} \{0,1\}^{2\lambda}$
3 : $\quad [\![\mathbf{x}]\!]_i^{q}, [\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}/q} := \mathsf{CRT}([\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}})$
4 : Generate Merkle root h containing $([\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}}, [\![\mathbf{y}]\!]_i^{q_{\mathsf{V3S}}}, \mathsf{seed}_i)_{i \in S_{\mathsf{fk}}}$ and complete in the tree
5 : $\quad$ the hashes of missing shares $j \notin S_{\mathsf{fk}}$ by uniform i.i.d values in $\{0,1\}^{2\lambda}$.
6 : Produce prf_i for shares in S_{fk}.
7 : $\mathbf{x}, \mathbf{y} \leftarrow \chi_{\mathbf{x}}, \chi_{\mathbf{y}}$
8 : $\mathbf{R} = \mathsf{H}_{\mathbf{R}}(h)$
9 : Compute $[\![\mathbf{v}]\!]^{q_{\mathsf{V3S}}}$ such that:
10 : $\quad$ for $i \in S_{\mathsf{fk}}, [\![\mathbf{v}]\!]_i^{q_{\mathsf{V3S}}} = \mathbf{R} \cdot [\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}} + [\![\mathbf{y}]\!]_i^{q_{\mathsf{V3S}}}$
11 : $\quad [\![\mathbf{v}]\!]_0^{q_{\mathsf{V3S}}} = \mathbf{R} \cdot \mathbf{x} + \mathbf{y}$
12 : **return** $([\![\mathbf{x}]\!]_i^{q})_{i \in S_{\mathsf{fk}}}, \pi = (h, [\![\mathbf{v}]\!]^{q_{\mathsf{V3S}}}), (\pi_i)_{i \in S_{\mathsf{fk}}} = ([\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}/q}, [\![\mathbf{y}]\!]_i^{q_{\mathsf{V3S}}}, \mathsf{seed}_i, \mathsf{prf}_i)_{i \in S_{\mathsf{fk}}}$

$\mathsf{SimSecret}^{\mathsf{H}_{\mathbf{R}}}(\pi = (h, [\![\mathbf{v}]\!]^{q_{\mathsf{V3S}}}))$

1 : $\mathbf{R} = \mathsf{H}_{\mathbf{R}}(h)$
2 : $\mathbf{c} \equiv \Sigma_0 \cdot \frac{1}{\sigma_y^2} \cdot \mathbf{R}^\top \cdot \mathbf{z}$
3 : $\Sigma_0 \equiv \left(\frac{1}{\sigma_x^2} \cdot \mathbf{I} + \frac{1}{\sigma_y^2} \cdot \mathbf{R}^\top \mathbf{R} \right)^{-1}$
4 : **return** $\mathbf{x} \leftarrow \mathcal{D}_{\mathbb{Z}^n, \mathbf{c}, \sqrt{\Sigma_0}}$

Figure 4.6: Simulators for fragmentary knowledge property of our V3S

We demonstrate in Section 4.7 how to efficiently construct such a desirable distribution. We start with proof sketches below, before providing the full proofs.

CORRECTNESS. V3S.Share() correctly constructs a Shamir's sharing and a corresponding Merkle tree for secure sharing and verification. The V3S.Reconstruct function can accurately reconstruct the original value for any subset of shares of size at least $t + 1$, and the V3S.Verify() function consistently passes its checks by the *honest execution* property of the distribution of $\mathbf{R}$.

SOUNDNESS. The proof employs a hybrid argument approach, consisting of three main steps:

- The first game is the actual soundness game.
- Hybrid_2 is a tweak of the soundness game ensuring that the matrix $\mathbf{R}$ is sampled *after* the shares are chosen, which is the crux for applying *separation and large norm detection properties* of $\chi_{\mathbf{R}}$. This hybrid aims to maintain the integrity of the Merkle tree hashes and the random oracle's programming, penalizing the adversary for any inconsistency or attempts to exploit the hash function's collision or pre-image resistance.
- Hybrid_3: Modifies the conditions under which an adversary can win, specifically targeting inconsistencies and too large norms in the reconstructed secret during the verification process.

- Probability of Winning: Rely on the *separation and large norm detection* to limit the probability of an adversary's success.

FRAGMENTARY KNOWLEDGE. The proof also goes by three hybrids.

- Hybrid_1 : Adversary observes the real distribution of transcripts, corresponding to $b = 0$ in the V3S-fk game.
- Hybrid_2 : Introduces uniform random strings in place of the hashes of shares not in set S, arguing the adversary's view changes negligibly if they had not queried these shares (basically, the view of the adversary differs *only if* it did query the random oracle on one of the shares $(\llbracket \mathbf{x} \rrbracket_i, \llbracket \mathbf{y}_i \rrbracket, \mathsf{seed}_i)$ with $i \notin S$ in Hybrid_1)
- Hybrid_3 : Further modifies by replacing shares in S with uniformly random vectors and simulating the adversary's view under $b = 1$. It uses the correctness of the secret sharing and the indistinguishability proved in Lemma 4.3.3 to argue the adversary's advantage remains unchanged from Hybrid 2.

Proof of Correctness. By construction, the function V3S.Share() will sample a valid Shamir's sharing $\llbracket \mathbf{x} \rrbracket^{q_{\mathsf{V3S}}}$ and $\llbracket \mathbf{y} \rrbracket^{q_{\mathsf{V3S}}}$, in addition to a (valid) corresponding Merkle tree. Thus, V3S.Reconstruct reconstructs the expected value $\mathbf{x}$ for any subset of size $t+1$ by definition. Now remark that in V3S.Verify(), the Merkle tree proof verification, and equality check $\llbracket \mathbf{v} \rrbracket_i^{q_{\mathsf{V3S}}} = \mathbf{R} \cdot \llbracket \mathbf{x} \rrbracket_i^{q_{\mathsf{V3S}}} + \llbracket \mathbf{y} \rrbracket_i^{q_{\mathsf{V3S}}}$ always pass. Additionally, when the execution is honest, we have $\mathbf{v} = \mathbf{R} \cdot \mathbf{x} + \mathbf{y}$, where $\mathbf{R}, \mathbf{x}, \mathbf{y}$ are independently sampled from $\chi_{\mathbf{R}}, \chi_{\mathbf{x}}, \chi_{\mathbf{y}}$. By the *honest execution* property of the distribution $\chi_{\mathbf{R}}$, we have $\|\mathbf{v}\| = \|\mathbf{R} \cdot \mathbf{x} + \mathbf{y}\| \leqslant B'$ with a loss of at most $p_3 = \mathsf{negl}(\lambda)$. □

Proof of Soundness. We follow a hybrid proof approach where we define:

Hybrid_1. First hybrid corresponds to the real V3S-sound game of Figure 4.3.

Hybrid_2. This hybrid ensures – by checking the RO calls – that the matrix $\mathbf{R}$ is sampled *after* the shares $(\llbracket \mathbf{x} \rrbracket_j^{q_{\mathsf{V3S}}}, \llbracket \mathbf{y} \rrbracket_j^{q_{\mathsf{V3S}}})_{j \in S_{\mathsf{snd}}}$ are chosen. It is given in Figure 4.7. This will allow us to apply *separation* and *large norm detection* properties of $\chi_{\mathbf{R}}$ in the next hybrids as then $\mathbf{R}$ will be independent of the tested value. The added check makes the adversary lose in several cases:

- $(\llbracket \mathbf{x} \rrbracket_j^{q_{\mathsf{V3S}}}, \llbracket \mathbf{y} \rrbracket_j^{q_{\mathsf{V3S}}}, \mathsf{seed}_j)_{j \in S_{\mathsf{snd}}}$ do not correctly hash in the Merkle tree h. But then, at least one of the proofs prf_i fails, and one of the calls to V3S.Verify would have failed. Hence, this case does not introduce an advantage loss.
- The shares correctly hash to h but Programming$[h] = \perp$ at the time of programming of the random oracle, one of the intermediary hash of the Merkle tree hadn't been queried yet. This may only happen if the adversary is able to break the pre-image resistance of the hash function H_{MT}. This happens with probability at most $Q^2_{\mathsf{H}_{\mathsf{MT}}}/2^{2\lambda}$.
- The shares given by the adversary correctly hash to h but Programming$[h]$ is equal to a different set of shares. This can only happen if the adversary breaks the collision resistance of the hash function H_{MT}. This happens with probability at most $Q^2_{\mathsf{H}_{\mathsf{MT}}}/(2^{2\lambda})$.

Overall,

$$|\mathsf{Adv}^{\mathsf{Hybrid}_2}_{\mathcal{A}}(1^\lambda) - \mathsf{Adv}^{\mathsf{Hybrid}_1}_{\mathcal{A}}(1^\lambda)| \leqslant \frac{Q^2_{\mathsf{H}_{\mathsf{MT}}}}{2^{2\lambda-1}}$$

Hybrid_3. This hybrid ensures that in case the random oracle is programmed on the shares $(\llbracket \mathbf{x} \rrbracket_i^{q_{\mathsf{V3S}}}, \llbracket \mathbf{y} \rrbracket_i^{q_{\mathsf{V3S}}})_{i \in S_{\mathsf{snd}}}$, then if they are inconsistent or if the reconstructed secret is large, then $\mathbf{R} \cdot \mathbf{x} + \mathbf{y}$ will be inconsistent or large. In case one of these checks, we consider that the adversary wins, and remove the previous winning checks. This is formalized in Figure 4.8.

Hybrid$_2$

$L_{\mathsf{H_R}}$, Programmed$[\cdot] \coloneqq \varnothing$
// The adversary chooses a subset S_{snd} of parties to target
$N, T, S_{\mathsf{snd}} \leftarrow \mathcal{A}()$
assert $S_{\mathsf{snd}} \subset [N] \wedge |S_{\mathsf{snd}}| \geqslant T$ // S_{snd} must be large enough to allow reconstruction
// The adversary produces a T-sharing among N parties
$(\llbracket \mathbf{x} \rrbracket^q)_{i \in S_{\mathsf{snd}}}, \pi, (\pi_i)_{i \in S_{\mathsf{snd}}} \leftarrow \mathcal{A}^{\mathsf{H_{MT}}, \mathsf{H_R}}(N, T, S_{\mathsf{snd}})$
Parse $\pi = (h, \llbracket \mathbf{v} \rrbracket^{q_{\mathsf{V3S}}}), (\pi_i)_{i \in S_{\mathsf{snd}}} = (\llbracket \mathbf{x} \rrbracket_i^{q_{\mathsf{V3S}}/q}, \llbracket \mathbf{y} \rrbracket_i^{q_{\mathsf{V3S}}}, \mathsf{seed}_i, \mathsf{prf}_i)_{i \in S_{\mathsf{snd}}}$
$(\llbracket \mathbf{x} \rrbracket_i^{q_{\mathsf{V3S}}})_{i \in S_{\mathsf{snd}}} = \mathsf{CRT}^{-1}((\llbracket \mathbf{x} \rrbracket_i^q)_{i \in S_{\mathsf{snd}}}, (\llbracket \mathbf{x} \rrbracket_i^{q_{\mathsf{V3S}}/q})_{i \in S_{\mathsf{snd}}})$
if Programmed$[h] \neq (\llbracket \mathbf{x} \rrbracket_i^{q_{\mathsf{V3S}}}, \llbracket \mathbf{y} \rrbracket_i^{q_{\mathsf{V3S}}})_{i \in S_{\mathsf{snd}}}$ **then**
  return 0
if $\forall i \in S_{\mathsf{snd}}, \mathsf{V3S.Verify}(\llbracket \mathbf{x} \rrbracket_i^q, \pi, \pi_i) = \mathsf{false}$ **then**
  return 0 // In case a party in S_{snd} fails verification, the adversary loses
$\mathbf{x}' = \mathsf{V3S.Reconstruct}((\llbracket \mathbf{x} \rrbracket_i^q)_{i \in S_{\mathsf{snd}}})$
if $\mathbf{x}' = \perp$ **then**
  return 1 // Verification passes but shares are inconsistent in S_{snd}
if $\mathbf{x}' \notin V$ **then**
  return 1 // Verification passes but the secret is invalid
return 0 // The sharing chosen by the adversary is valid

$\mathsf{H_R}(h)$

if $\exists r.(h, r) \in L_{\mathsf{H_R}}$ **then**
  return r
else
  if $\exists (\llbracket \mathbf{x} \rrbracket_j^{q_{\mathsf{V3S}}}, \llbracket \mathbf{y} \rrbracket_j^{q_{\mathsf{V3S}}}, \mathsf{seed}_j)_{j \in S_{\mathsf{snd}}}$ s.t. they are in the Merkle tree described by h **then**
    $\mathbf{R} \leftarrow \chi_{\mathbf{R}}$
    Programmed$[h] = ((\llbracket \mathbf{x} \rrbracket_j^{q_{\mathsf{V3S}}}, \llbracket \mathbf{y} \rrbracket_j^{q_{\mathsf{V3S}}})_{j \in S_{\mathsf{snd}}})$
    $L_{\mathsf{H_R}} \coloneqq L_{\mathsf{H_R}} \cup \{(h, \mathbf{R})\}$
    return R
  else
    $\mathbf{R} \leftarrow \chi_{\mathbf{R}}$
    $L_{\mathsf{H_R}} \coloneqq L_{\mathsf{H_R}} \cup \{(h, \mathbf{R})\}$
    return R

Figure 4.7: The second hybrid of the security proof of the soundness of our V3S construction. Difference with the previous hybrid are highlighted.

These modifications allow the adversary to win in more cases. Indeed, in Hybrid_2, the adversary could win in two ways:

- If reconstruction over S_{snd} fails, but $\mathsf{V3S.Verify}()$ returns true over S_{snd}. But then, it means in particular that shares of $\mathbf{R} \cdot [\![\mathbf{x}]\!]^{q_{\mathsf{V3S}}} + [\![\mathbf{y}]\!]^{q_{\mathsf{V3S}}}$ were all consistent over S_{snd}, and that it correctly reconstructs. In Hybrid_3, this case is covered by the first assertion added in $\mathsf{H}_{\mathbf{R}}(h)$ that makes the adversary in that case.
- If the reconstructed secret $\mathbf{x}$ has a norm larger than B, but $\mathbf{R} \cdot \mathbf{x} + \mathbf{y}$ has a norm smaller than B'. Similarly, the second assertion added in $\mathsf{H}_{\mathbf{R}}$ would fail when called on h, and allow the adversary to win.

As such:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_2}(1^\lambda) \leqslant \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_3}(1^\lambda)$$

PROBABILITY OF WINNING Hybrid_3. First, consider the event where some call to $\mathsf{H}_{\mathbf{R}}$ works on inconsistent shares for $[\![\mathbf{x}]\!]^{q_{\mathsf{V3S}}}, [\![\mathbf{y}]\!]^{q_{\mathsf{V3S}}}$ over S_{snd} but $\mathbf{R} \cdot [\![\mathbf{x}]\!]^{q_{\mathsf{V3S}}} + [\![\mathbf{y}]\!]^{q_{\mathsf{V3S}}}$ are all consistent. Noting $(\mathbf{x}', \mathbf{y}') \neq (\mathbf{x}'', \mathbf{y}'')$ the secrets reconstructed from I, I' two subsets of S_{snd} of cardinal $t+1$, as $\mathbf{R}$ is independent of these vectors, we can apply the *separation* property of $\mathbf{R}$ to bound its probability:

$$\Pr\left[\mathbf{R}\mathbf{x}' + \mathbf{y}' \bmod q_{\mathsf{V3S}} = \mathbf{R}\mathbf{x}'' + \mathbf{y}'' \bmod q_{\mathsf{V3S}}\right] = p_1$$

By the union-bound the probability that the first assertion succeeds for all the calls to the random oracles is at most $Q_{\mathsf{H}_{\mathbf{R}}} \cdot p_1$. Similarly, for the second assertion, the matrix $\mathbf{R}$ is independent of the reconstructed value and the *large norm detection* property of $\chi_{\mathbf{R}}$ allows us to bound the probability of failure by $Q_{\mathsf{H}_{\mathbf{R}}} \cdot p_2$.

We can conclude:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_3}(1^\lambda) \leqslant Q_{\mathsf{H}_{\mathbf{R}}} \cdot (p_1 + p_2).$$

Summing all the intermediary advantage losses gives the final result. □

Proof of fragmentary knowledge. We proceed once again with a series of hybrids.

Hybrid_1. The first hybrid corresponds to the case $b = 0$ of the V3S-fk game, i.e. where the adversary observes the real distribution of transcripts.

Hybrid_2. The second hybrid replaces the hashes of shares not in S_{fk} in the Merkle tree by uniform strings. This change is depicted in Figure 4.9.

The view of the adversary differs *only if* it did query the random oracle H_{MT} on one of the shares $([\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}}, [\![\mathbf{y}_i]\!]^{q_{\mathsf{V3S}}}, \mathsf{seed}_i)$ with $i \notin S_{\mathsf{fk}}$ in Hybrid_1 and the Merkle tree hash does not correspond in Hybrid_2. However, since each share $([\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}}, [\![\mathbf{y}]\!]_i^{q_{\mathsf{V3S}}}, \mathsf{seed}_i)_{i \in S_{\mathsf{fk}}}$ has min-entropy at least $\mathcal{H}(\mathsf{seed}_i) = 2\lambda$, this happens for each share with probability at most $Q_{\mathsf{H}_{\mathsf{MT}}}/2^{2\lambda}$. By union bound for all shares we have:

$$|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_2}(1^\lambda) - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_1}(1^\lambda)| \leqslant N \cdot \frac{Q_{\mathsf{H}_{\mathsf{MT}}}}{2^{2\lambda}}$$

Hybrid_3. This hybrid replaces the shares of $\mathbf{x}$ in S_{fk} by uniformly drawn vectors of $R_{q_{\mathsf{V3S}}}^k$, and the $\mathbf{x}$ sent to the adversary by a Gaussian variable following $D_{R^k, \mathbf{c}, \sqrt{\Sigma_0}}$. This is formalized in 4.10. We observe that this corresponds to the case $b = 1$ of the V3S-fk game, and implements perfectly the simulators SimProof, SimSecret we previously described. The computation of $[\![\mathbf{v}]\!]^{q_{\mathsf{V3S}}}$ as a function of $\mathbf{R} \cdot x + \mathbf{y}$ and $(\mathbf{R} \cdot [\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}}, [\![\mathbf{y}]\!]_i^{q_{\mathsf{V3S}}})_{i \in S_{\mathsf{fk}}}$ is equivalent to the one done in Hybrid_2 due to

Hybrid_3

$L_{\mathsf{H_R}}, \mathrm{Programmed}[\cdot] := \varnothing$
$N, T, S_{\mathsf{snd}} \leftarrow \mathcal{A}()$ // The adversary chooses a subset S_{snd} of parties to target
assert $S_{\mathsf{snd}} \subset [N] \wedge |S_{\mathsf{snd}}| \geqslant T$ // S_{snd} must be large enough to allow reconstruction
// The adversary produces a T-sharing among N parties
$([\![\mathbf{x}]\!]_i^q)_{i\in[N]}, \pi, (\pi_i)_{i\in[N]} \leftarrow \mathcal{A}^{\mathsf{H_{MT}},\mathsf{H_R}}(N, T, S_{\mathsf{snd}})$
Parse $\pi = (h, [\![v]\!]^{q_{\mathsf{V3S}}}), (\pi_i)_{i\in S_{\mathsf{snd}}} = ([\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}/q}, [\![\mathbf{y}]\!]_i^{q_{\mathsf{V3S}}}, \mathsf{seed}_i, \mathsf{prf}_i)_{i\in S_{\mathsf{snd}}}$
$([\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}})_{i\in S_{\mathsf{snd}}} = \mathsf{CRT}^{-1}(([\![\mathbf{x}]\!]_i^q)_{i\in S_{\mathsf{snd}}}, ([\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}/q})_{i\in S_{\mathsf{snd}}})$
if $\mathrm{Programmed}[h] \neq ([\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}}, [\![\mathbf{y}]\!]_i^{q_{\mathsf{V3S}}})_{i\in S_{\mathsf{snd}}}$ **then**
  return 0
if $\forall i \in S_{\mathsf{snd}}, \mathsf{V3S.Verify}([\![\mathbf{x}]\!]_i^q, \pi, \pi_i) = \mathsf{false}$ **then**
  return 0 // In case a party in S_{snd} fails verification, the adversary loses
return 0 // The sharing chosen by the adversary is valid

$\mathsf{H_R}(h)$

if $\exists r.(h, r) \in L_{\mathsf{H_R}}$ **then**
  return r
else
  if $\exists([\![\mathbf{x}]\!]_j^{q_{\mathsf{V3S}}}, [\![\mathbf{y}]\!]_j^{q_{\mathsf{V3S}}})_{j\in S_{\mathsf{snd}}}$ s.t. they are in the Merkle tree described by h **then**
    $\mathbf{R} \leftarrow \chi_{\mathbf{R}}$
    $\mathrm{Programmed}[h] = (([\![\mathbf{x}]\!]_j^{q_{\mathsf{V3S}}}, [\![\mathbf{y}]\!]_j^{q_{\mathsf{V3S}}})_{j\in S_{\mathsf{snd}}})$
    if $\mathsf{V3S.Reconstruct}(([\![\mathbf{x}]\!]_j^{q_{\mathsf{V3S}}}, [\![\mathbf{y}]\!]_j^{q_{\mathsf{V3S}}})_{j\in S_{\mathsf{snd}}}) = \perp$ **then**
      // If shares of $\mathbf{x}, \mathbf{y}$ are inconsistent over S_{snd}
      // Then the shares of $\mathbf{R}\cdot\mathbf{x}+\mathbf{y}$ are also inconsistent
      assert $\mathsf{V3S.Reconstruct}((\mathbf{R}\cdot[\![\mathbf{x}]\!]_j^{q_{\mathsf{V3S}}} + [\![\mathbf{y}]\!]_j^{q_{\mathsf{V3S}}})_{j\in S_{\mathsf{snd}}}) = \perp$
    else
      // Reconstruct $\mathbf{x}, \mathbf{y}$ from the shares in S_{snd}
      $\mathbf{x}' = \mathsf{V3S.Reconstruct}(([\![\mathbf{x}]\!]_j^{q_{\mathsf{V3S}}})_{j\in S_{\mathsf{snd}}})$
      $\mathbf{y}' = \mathsf{V3S.Reconstruct}(([\![\mathbf{y}]\!]_j^{q_{\mathsf{V3S}}})_{j\in S_{\mathsf{snd}}})$
      if $\|\mathbf{x}'\| > B$ **then**
        assert $\|\mathbf{R}\cdot\mathbf{x}' + \mathbf{y}'\| > B'$
    $L_{\mathsf{H_R}} := L_{\mathsf{H_R}} \cup \{(h, \mathbf{R})\}$
    return $\mathbf{R}$
  else
    $\mathbf{R} \leftarrow \chi_{\mathbf{R}}$
    $L_{\mathsf{H_R}} := L_{\mathsf{H_R}} \cup \{(h, \mathbf{R})\}$
    return $\mathbf{R}$

Figure 4.8: The third hybrid of the security proof of the soundness of our V3S construction. Difference with the previous hybrid are highlighted. In case one of the assertions in $\mathsf{H_R}$ fails, consider that the adversary wins.

Hybrid$_2$

$L_H := \varnothing$
$N, T, S_{\mathsf{fk}} \leftarrow \mathcal{A}()$ // The adversary chooses a subset S_{fk} of parties to target
assert $S_{\mathsf{fk}} \subset [N] \wedge |S_{\mathsf{fk}}| = T - 1$
$\mathbf{x} \leftarrow \chi_{\mathbf{x}}$
// Content of V3S.Share()
$\mathbf{y} \leftarrow D_{R^k,\sigma_y}, (\mathsf{seed}_i) \xleftarrow{\$} \{0,1\}^{N \cdot 2\lambda}$
Sample random sharings $[\![\mathbf{x}]\!]^{q_{\mathsf{V3S}}}, [\![\mathbf{y}]\!]^{q_{\mathsf{V3S}}}$ of order T
$h :=$ hash of Merkle tree containing $([\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}}, [\![\mathbf{y}]\!]_i^{q_{\mathsf{V3S}}}, \mathsf{seed}_i)$ for $i \in S_{\mathsf{fk}}$,
  and replacing hashes of other shares by random values.
$([\![\mathbf{x}]\!]_i^{q}, [\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}/q})_{i \in S_{\mathsf{fk}}} := \mathsf{CRT}(([\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}})_{i \in S_{\mathsf{fk}}})$
for $i \in S_{\mathsf{fk}}$ **do**
  $\mathsf{prf}_i :=$ proof that $([\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}}, [\![\mathbf{y}]\!]_i^{q_{\mathsf{V3S}}}, \mathsf{seed}_i)$ is in Merkle tree h
  $\pi_i := ([\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}/q}, [\![\mathbf{y}]\!]_i^{q_{\mathsf{V3S}}}, \mathsf{seed}_i, \mathsf{prf}_i)$
$\mathbf{R} := \mathsf{H}_{\mathbf{R}}(h)$ // Hash h to obtain a random matrix from $\chi_{\mathbf{R}}$
$[\![\mathbf{v}]\!]^{q_{\mathsf{V3S}}} := \mathbf{R} \cdot [\![\mathbf{x}]\!]^{q_{\mathsf{V3S}}} + [\![\mathbf{y}^{q_{\mathsf{V3S}}}]\!]$
$\pi := (h, [\![\mathbf{v}]\!]^{q_{\mathsf{V3S}}})$ // Publish challenge polynomial and Merkle tree hash
$b' \leftarrow \mathcal{A}^{\mathsf{H}_{\mathsf{MT}}, \mathsf{H}_{\mathbf{R}}}(\mathbf{x}, ([\![x]\!]_i^{q})_{i \in S_{\mathsf{fk}}}, \pi, (\pi_i)_{i \in S_{\mathsf{fk}}})$
return b'

Figure 4.9: The second hybrid of the security proof of the fragmentary knowledge of our V3S construction. Difference with the previous hybrid are highlighted.

Hybrid$_3$

$L_H := \varnothing$
$N, T, S_{\mathsf{fk}} \leftarrow \mathcal{A}()$ // The adversary chooses a subset S_{fk} of parties to target
assert $S_{\mathsf{fk}} \subset [N] \wedge |S_{\mathsf{fk}}| = T - 1$
$\mathbf{x} \leftarrow \chi_{\mathbf{x}}$
// Content of V3S.Share()
$\mathbf{y} \leftarrow \chi_{\mathbf{y}}$
$[\![\mathbf{x}]\!]_{i \in S_{\mathsf{fk}}}, [\![\mathbf{y}]\!]_{i \in S_{\mathsf{fk}}} \xleftarrow{\$} R^k$ // Sample observed shares randomly
$h :=$ root of Merkle tree containing $([\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}}, [\![\mathbf{y}]\!]_i^{q_{\mathsf{V3S}}})$ for $i \in S_{\mathsf{fk}}$,
  and replacing hashes of other shares by random values.
$([\![\mathbf{x}]\!]_i^{q}, [\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}/q})_{i \in S_{\mathsf{fk}}} := \mathsf{CRT}(([\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}})_{i \in S_{\mathsf{fk}}})$
for $i \in S_{\mathsf{fk}}$ **do**
  $\mathsf{prf}_i :=$ proof that $([\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}}, [\![\mathbf{y}]\!]_i^{q_{\mathsf{V3S}}})$ is in Merkle tree h
  $\pi_i := ([\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}/q}, [\![\mathbf{y}]\!]_i^{q_{\mathsf{V3S}}}, \mathsf{prf}_i)$
$\mathbf{R} := \mathsf{H}_{\mathbf{R}}(h)$ // Hash h to obtain a random matrix from $\chi_{\mathbf{R}}$
Compute $[\![\mathbf{v}]\!]^{q_{\mathsf{V3S}}}$ s.t. $[\![\mathbf{v}]\!]_0^{q_{\mathsf{V3S}}} = \mathbf{R} \cdot \mathbf{x} + \mathbf{y}$, and for $i \in S$, $[\![\mathbf{v}]\!]_i^{q_{\mathsf{V3S}}} = \mathbf{R} \cdot [\![\mathbf{x}]\!]_i^{q_{\mathsf{V3S}}} + [\![\mathbf{y}]\!]_i^{q_{\mathsf{V3S}}}$
$\pi := (h, [\![\mathbf{v}]\!]^{q_{\mathsf{V3S}}})$ // Publish challenge polynomial and Merkle tree hash
$\hat{\mathbf{x}} \leftarrow D_{R^k, \mathbf{c}, \sqrt{\Sigma_0}}$
$b' \leftarrow \mathcal{A}^{\mathsf{H}, \mathsf{H}_{\mathbf{R}}}(\hat{\mathbf{x}}, ([\![x]\!]_i^{q_{\mathsf{V3S}}})_{i \in S_{\mathsf{fk}}}, \pi, (\pi_i)_{i \in S_{\mathsf{fk}}})$
return b'

Figure 4.10: The third hybrid of the security proof of the fragmentary knowledge of our V3S construction. Difference with the previous hybrid are highlighted.

the correctness of the sharing of $\mathbf{x}$, and as we only need $t+1$ points of $[\![\mathbf{v}]\!]^{q_{\text{V3S}}}$ to recover its full value. Hence, this change does not induce an advantage loss. We also note that the adversary observes outputs computed from exactly t shares of $[\![\mathbf{x}]\!]^{q_{\text{V3S}}}$ in Hybrid_3 and they are thus randomly distributed independently from $\mathbf{x}$. We can sample them directly with no advantage loss. Finally, applying Lemma 4.3.3, we have that $(\mathbf{x}, \mathbf{R}\cdot\mathbf{x}+\mathbf{y})$ follows the same distribution as $(\hat{\mathbf{x}}, \mathbf{R}\cdot\mathbf{x}+\mathbf{y})$. We conclude that: $\text{Adv}_{\mathcal{A}}^{\text{Hybrid}_3}(1^\lambda) = \text{Adv}_{\mathcal{A}}^{\text{Hybrid}_2}(1^\lambda)$. □

4.4.3 Toward Applications

Our definition of V3S is agnostic to the communication means used. We describe at a high level in this section two generic ways for using our V3S in protocols to obtain different guarantees and cost tradeoffs. Further, we demonstrate concrete usage and security proofs in Section 4.5.

V3S *protocol with detection of malicious behavior.*

The first protocol we introduce allows us to detect misbehavior during the protocol execution and abort in that case. It requires lightweight communication assumptions, i.e. confidential and authenticated pairwise channels, additionally reliable for correctness when all the parties are honest. However, the counterpart is that as soon as at least one of the parties is dishonest, the protocol may abort with no possibility of recovery as no specific malicious party can be identified. We assume from now on that there are at least $t+1$ honest parties. The protocol proceeds in several rounds:

1. In the first round, the dealer samples $\mathbf{x} \leftarrow \chi_{\mathbf{x}}$ and runs $([\![\mathbf{x}]\!]_i)_{i\in[N]}, \pi, (\pi_i)_{i\in[N]} \leftarrow \text{V3S.Share}(\mathbf{x})$. It then sends $([\![\mathbf{x}]\!]_i, \pi, \pi_i)$ to each other party i over the pairwise channel between them.
2. In the second round, everyone checks the data provided by the dealer. They run $b = \text{V3S.Verify}([\![\mathbf{x}]\!]_i, \pi, \pi_i)$. After that, they send b to each other party, along with a hash of π to prove they had the same common proof.
3. Finally, each party accepts the sharing only if it receives a positive response from all other parties, along with the same hash as theirs.

From the good properties of our V3S proposal, we can easily show that this protocol has also:

- *correctness*: in case all parties behave honestly, then the protocol terminates and all honest parties accept the sharing.
- *soundness*: in case the dealer sends inconsistent shares to any honest party, or reconstructing to an invalid secret, it will be detected with overwhelming probability and the protocol will abort.
- *fragmentary knowledge*: in case the dealer is honest, and the protocol terminates, the transcript is indistinguishable from the one simulated by the fragmentary knowledge property of our V3S.

Robust V3S ***protocol.***

We propose a second protocol providing guaranteed delivery in case the dealer is honest, and, in case the dealer is dishonest and misbehaves, all honest parties simultaneously abort. This protocol assumes the existence of an *authenticated reliable* (even for corrupted parties) *non-ordered broadcast channel*, and that a majority of the parties are honest.

It works with an IND-CPA symmetric encryption scheme SKE to implement pairwise communication. We assume that pairwise keys sk_i^{SKE} are shared prior to the protocol execution to communicate with the dealer. Our robust protocol works as follows:

1. The dealer samples $\mathbf{x}$, and runs V3S.Share($\mathbf{x}$). It broadcasts the proof π along with individual encryptions produced with the SKE for each other party of $[\![\mathbf{x}]\!]_i, \pi_i$.

2. In the second round, each party decrypts the message from the dealer and runs V3S.Verify() on its share. If decryption or verification fails, the party broadcasts a complaint against the original sender.

3. The remaining round consists of processing all the complaints. If any of the complaints is valid, it means that the dealer behaved dishonestly, if no complaint is valid, the sharing is valid, and the party can accept it.

We identify two common ways to integrate this complaint mechanism, trading off between round efficiency versus communication cost and complexity:

1. In the simplest version, parties broadcast complaints simply containing the identifier of the sender. Then this sender is required to resend its share in clear to all parties over the broadcast channel. If the share resent in clear remains invalid, the party is identified as malicious, otherwise the sender is exonerated, and the protocol continues. This approach however requires an additional round of communication to answer complaints.

2. Alternatively, to keep the protocol to three rounds, parties can include in their complaints the secret key of their communication channel with the dealer, allowing all parties to verify the complaint directly. To ensure the authenticity of the revealed key, each party also includes a signature signed by the dealer authorizing the reveal of the key: we assume that the dealer provides for each party i a signature $\mathsf{sig}_i = \mathsf{Sign}(\mathsf{sk}, (i, \mathsf{sk}_i^{\mathsf{SKE}}))$ authorizing the reveal of the key. This way, all parties can verify the complaint directly after the second round, and the dealer can be held accountable in case of a valid complaint. Note that an alternative to signatures would be to use publicly verifiable signcryption.

In this work, we opt for the second approach to keep the round complexity low, but we note that it may be harder to setup the signatures in practice, and when assuming a Public Key Infrastructure, the first approach may be preferable.

In this protocol, we achieve:

- *Unframeability:* if the dealer is honest, no corrupted party will be able to make honest parties reject. Indeed, as the signature scheme is unforgeable, only a complaint with the correct SKE key can be produced, and then the message sent by the dealer will be correctly decrypted.

- *Accountability:* if the dealer is corrupted, the use of a broadcast channel ensures that all honest parties will receive the same set of complaints and will conclude identically. If they accept the shares, the shares of all honest parties pass verification, which means that the shares are consistent, and the secret is valid.

4.5 RB-Raccoon: A ROBUST THRESHOLD SIGNATURE SCHEME

In this section, we present RB-Raccoon, a robust threshold signature scheme based on Raccoon.

As sketched in Section 4.2.2 and Section 4.2.3, our V3S can be used to build a robust distributed key generation (DKG) protocol, which in turn can be used to build a robust threshold signature scheme.

Note that while both protocols leverage our V3S design, they use it with different parameters and bounds as the shared secrets have different sizes in each protocol. In the DKG protocol, the shared secret is the private key of Raccoon, while in the signing protocol, the shared secret is the

noise used to flood the signature shares. In particular, the signing noise is much larger than the private key.

Before diving into our construction, we make the following assumptions in our communication model:

- **Synchronous communication with reliable broadcast.** We assume that all contributions by user i are broadcast on a reliable public channel, with authentication. This corresponds to the "synchronous model with reliable broadcast" introduced in Section 3.4.2 and for which we formalized unforgeability in Fig. 3.4. This could practically be implemented using a consensus protocol, or a trusted coordinator.

- **Signed pairwise keys.** For any ordered pair of users (i, j), i and j share a pairwise symmetric key $\mathsf{K}_{i\to j}$ that has been signed by i. This secret key is used by i to send encrypted data to i using an IND-CPA symmetric key encryption scheme $\mathsf{SKE} = \{\mathsf{Encrypt}, \mathsf{Decrypt}\}$.

  Encrypted messages are sent over the broadcast channel in order to authenticate the sender. When j files a complaint against i, they reveal the $\mathsf{K}_{i\to j}$ along with the signature. This binds i to the data they have sent to j, and in case of misbehavior, revealing $\mathsf{K}_{i\to j}$ allows other parties to acknowledge the misbehavior.

  Concretely, signed pairwise keys can be established in a setup phase. It suffices that i generates a KEM ciphertext i encapsulating a KEM symmetric key, and uses this KEM symmetric key to encrypt $\mathsf{K}_{i\to j}$ and a signature $\mathsf{sig}_{i\to j} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}_i, \{i, j, \mathsf{K}_{i\to j}\})$. Though, we acknowledge that it may be harder to tackle misbehavior during this setup phase. The alternative approach of adding an extra round to the protocols to allow parties to answer complaints by broadcasting the share of the plaintiff – as described in Section 4.4.3 – could be considered instead to avoid this issue, but we choose not to follow this path in this work to keep the protocols as round efficient as possible.

Our protocols are round-based, with one algorithm per round. Rounds are run sequentially and synchronously, with the returned value being posted to the broadcast channel.

4.5.1 Robust Distributed Key Generation (DKG)

We present a first application of our framework, allowing one to verifiably sample and share a short secret among parties, assuming that 2/3 of the participants are honest. We apply it to the key generation of RB-Raccoon and prove that the resulting scheme remains robust and unforgeable.

In environments where the key generation cannot be entitled to a single trusted entity, it is important to provide the possibility to distribute the key generation process among several actors.

Distributed key generation description

Our Distributed Key Generation was informally presented in Section 4.2.2. We now provide a formal description of its algorithms in Figure 4.11. We also formalize the setup corresponding to the creation of pairwise secure channels between all parties, with accountability for messages sent thanks to the signed keys.

Note that usual rounding techniques on the public key apply.

4.5.2 Robust Distributed Signing Procedure

We present as a second application of our result an efficient lattice-based robust threshold signature based on standard assumptions It provides additional guarantees over the existing threshold

Setup$(1^\lambda, N, T)$

1: **for** $i \in [N]$ **do**
2: $\quad \mathsf{sk}_i^{\mathsf{SIG}}, \mathsf{pk}_i^{\mathsf{SIG}} \leftarrow \mathsf{SIG.Keygen}(1^\lambda)$
3: $\quad$ **for** $j \in [N]$ **do**
4: $\quad\quad K_{i\to j}^{\mathsf{SKE}} \leftarrow \mathsf{SKE.Keygen}(1^\lambda)$
5: $\quad\quad \mathsf{sig}_{i\to j}^{\mathsf{SKE}} := \mathsf{SIG.Sign}(\mathsf{sk}_i^{\mathsf{SIG}}, \{i, j, K_{i\to j}^{\mathsf{SKE}}\})$
6: **for** $i \in [N]$ **do**
7: $\quad \mathsf{st}_i := \{(\mathsf{pk}_i^{\mathsf{SIG}})_{i\in[N]}, (K_{i'\to j'}^{\mathsf{SKE}}, \mathsf{sig}_{i'\to j'}^{\mathsf{SKE}})_{i'=i \vee j'=i}\}$
8: **return** $(\mathsf{st}_i)_{i\in[N]}$

RB-Raccoon.ShareKeygen$_1(\mathsf{st}_i)$

1: $\mathsf{salt}_i \xleftarrow{\$} \{0,1\}^\lambda$
2: $\mathbf{s}_i \leftarrow \chi_{\mathsf{sk}}$
3: $(\llbracket \mathbf{s}_i \rrbracket, \pi_i, (\pi_{i,j})_{j\in[N]}) \leftarrow \mathsf{V3S.Share}(N, T, \mathbf{s}_i)$ // Share $\llbracket \mathbf{s}_i \rrbracket$ and make proofs
4: **if** $\exists j$ s.t. $\mathsf{V3S.Verify}(\llbracket \mathbf{s}_i \rrbracket_j, \pi_i, \pi_{i\to j}) = \mathsf{false}$ **then restart** // Invalid proof
5: **for** $j \in [N]$ **do**
6: $\quad \mathsf{pt}_j := (\llbracket \mathbf{s}_i \rrbracket_j, \pi_{i,j})$
7: $\quad \mathsf{ct}_{i,j} \leftarrow \mathsf{SKE.Encrypt}(\mathsf{K}_{i\to j}, \mathsf{pt}_j)$ // Encrypt (share, proof) to each party
8: **return** $\mathsf{pm}_1[i] := \mathsf{seed}_i, \pi_i, (\mathsf{ct}_{i,j})_{j\in[N]}$

RB-Raccoon.ShareKeygen$_2(\mathsf{st}_i, \mathsf{pm}_1)$

1: $\mathsf{complaints}_i := \{\}$
2: **for** $(j \in \mathsf{pm}_1)$ **do** // Set of round 1 contributors
3: $\quad \mathsf{seed}_j, \pi_j, (\mathsf{ct}_{j\to k})_{k\in[N]} := \mathsf{pm}_1[j]$
4: $\quad \llbracket \mathbf{s}_j \rrbracket_i, \pi_{j\to i} := \mathsf{SKE.Decrypt}(\mathsf{K}_{j\to i}, \mathsf{ct}_{j\to i})$
5: $\quad$ **if** (SKE.Decrypt failed) **or** $(\mathsf{V3S.Verify}(\llbracket \mathbf{s}_j \rrbracket_i, \pi_j, \pi_{j\to i}) = \mathsf{false})$ **then**
6: $\quad\quad \mathsf{complaints}_i[j] = \{\mathsf{K}_{j\to i}, \mathsf{sig}_{j\to i}\}$ // i complains against j
7: $\mathsf{valid}_i = \{j \in \mathsf{pm}_1\} \backslash \mathsf{complaints}_i$
8: $\mathsf{st}_i.\mathsf{session.valid} := \mathsf{valid}_i$
9: $\mathsf{st}_i.\mathsf{session.pm}_1 := \mathsf{pm}_1$
10: $\mathsf{st}_i.\mathsf{session.shares} := (\llbracket \mathbf{s}_j \rrbracket_i)_{j\in\mathsf{valid}_i}$
11: **return** $\mathsf{pm}_2[i] := \mathsf{complaints}_i$

RB-Raccoon.ShareKeygen$_3(\mathsf{st}_i, \mathsf{pm}_2)$

1: $\mathsf{valid}_i := \mathsf{st}_i.\mathsf{session.valid} \cap \{j \in \mathsf{pm}_2\}$
2: **for** $j \in [N]$ **do**
3: $\quad$ **for** $k \in \mathsf{complaints}_j$ **do** // lines 4 to 8 study j's complaint against k
4: $\quad\quad \{\mathsf{K}_{k\to j}, \mathsf{sig}_{k\to j}\} := \mathsf{complaints}_j[k]$
5: $\quad\quad$ **if** $\mathsf{SIG.Verify}(\mathsf{pk}_k, \mathsf{sig}_{k\to j}, \{k, j, \mathsf{K}_{k\to j}\}) = \mathsf{false}$ **then continue**
6: $\quad\quad \llbracket \mathbf{s}_k \rrbracket_j, \pi_{k\to j} := \mathsf{SKE.Decrypt}(\mathsf{K}_{k\to j}, \mathsf{ct}_{k\to j})$
7: $\quad\quad$ **if** (SKE.Decrypt failed) **or** $\mathsf{V3S.Verify}(\llbracket \mathbf{s}_k \rrbracket_j, \pi_k, \pi_{k\to j}) = \mathsf{false}$ **then**
8: $\quad\quad\quad \mathsf{valid}_i = \mathsf{valid}_i \backslash \{k\}$ // If j's complaint against k is valid, invalidate j
9: $\mathsf{st}_i.\mathsf{session.seed} := \mathsf{H}_{\mathsf{seed}}((\mathsf{seed}_j)_{j\in\mathsf{valid}_i})$
10: // Derive the public matrix $\mathbf{A} \in R_q^{k\times\ell}$ from seed
11: $\mathbf{A} := \mathsf{H}_{\mathbf{A}}(\mathsf{st}_i.\mathsf{session.seed}) \in R_q^{k\times\ell}$
12: $\llbracket \mathbf{s} \rrbracket_i := \sum_{j\in\mathsf{valid}_i} \llbracket \mathbf{s}_j \rrbracket_i$ // $\mathbf{s} = \sum_{j\in\mathsf{valid}} \mathbf{s}_j$
13: $\llbracket \mathbf{b} \rrbracket_i := \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \llbracket \mathbf{s} \rrbracket_i$
14: Store $\llbracket \mathbf{s} \rrbracket_i$ in st_i
15: **return** $\mathsf{pm}_3[i] := (\mathsf{st}_i.\mathsf{session.seed}, \llbracket \mathbf{b} \rrbracket_i)$

Figure 4.11: Algorithms for the distributed Keygen. For conciseness, we omit the parsing of $(\mathsf{pm}_i)_{i\in\{1,2,3\}}$ in the algorithms.

RB-Raccoon.PartialSecret(st_i)

1 : **return** $[\![\mathbf{s}]\!]_i$

RB-Raccoon.CombineKey(pm_3)

1 : Retrieve seed from pm_3 // If contradictory contributions use majority vote
2 : $\mathbf{b} \coloneqq \mathsf{SSS.ErrorCorrect}([\![\mathbf{b}]\!])$
3 : **assert** $\mathbf{b} \neq \perp$
4 : $\mathbf{b}_\top \coloneqq \lfloor \mathbf{b} \rceil_{\nu_\mathbf{b}}$
5 : // We only return $\mathbf{b}$ as auxiliary value for the proof
6 : // to go through, but it is not actually used in the protocol.
7 : **return** $\mathsf{vk} \coloneqq (\mathsf{seed}, \mathbf{b}_\top), \mathsf{aux} \coloneqq \mathbf{b}$

Figure 4.12: Algorithms for the combination of the shares in the distributed Keygen.

schemes such as TRaccoon [DKMM+24] which provides unforgeability, but no guarantee of termination.

Assuming the existence of an authenticated reliable *broadcast channel*, and that at least 2/3 of the parties are honest, our protocol ensures that a valid signature will be produced.

Description of RB-Raccoon

RB-Raccoon was drafted in the Technical Overview, in Section 4.2.3. It relies on a symmetric encryption scheme SKE, a signature scheme SIG, and a Verifiable Short Secret Sharing V3S.

Our informal description of RB-Raccoon easily translates to formal algorithms, described in Figures 4.13 and 4.14.

4.6 SECURITY ANALYSIS OF RB-Raccoon

In this section, we prove the security of RB-Raccoon. We first introduce a leaky distributed key generation algorithm that captures the bias an adversary can introduce during the DKG protocol. We then prove the robustness and unforgeability of RB-Raccoon, relying on this leaky DKG.

4.6.1 Leaky Distributed Key Generation

We first wish to unify the treatment of our DKG for both the unforgeability and robustness security proof. To do so, we introduce the algorithm LeakyKeygen, described in Figure 4.15, which captures the security and bias of our distributed key generation protocol. Concretely, we will ensure that the shares generated by our protocol reconstruct to some secret $\mathbf{s} + \mathbf{s}'$, where $\mathbf{s}'$ is the "leaked" part, i.e. it can be biased and leaked to the adversary, and $\mathbf{s}$ is the "safe" part, sampled independently from $\mathbf{s}'$ from the distribution χ'_{sk}. The final security of the threshold signature scheme will rely on $\mathbf{s}$ as it remains hidden to the adversary.

We introduce intermediate games, called $\mathsf{Game}^{\mathsf{dkg\text{-}real}}$ and $\mathsf{Game}^{\mathsf{dkg\text{-}leaky}}$ in Figure 4.16, which show that the real DKG protocol execution can be simulated from a key generated by LeakyKeygen.

We define χ'_{sk} as the distribution of the sum of $|\mathsf{HS}|$ samples obtained with SimSecret (c.f. Fig. 4.6). Recall that each party samples a partial secret from χ_{sk} and produces a proof for it using our V3S from Section 4.4.2, partially revealing it to the adversary. SimSecret simulates the remaining uncertainty on this partial secret given the revealed proof information, and thus the sum of $|\mathsf{HS}|$ such samples corresponds to the safe part of the final secret key. We also introduce V', the set of all possible sums of at most t elements from V – the set of valid secrets of the V3S.

RB-Raccoon.ShareSign$_1$(vk, sid, act, msg, i, sk$_i$, st$_i$, msg)

1 : $\mathbf{r}_i \leftarrow \chi_{\mathbf{r}}$
2 : $[\![\mathbf{r}_i]\!], \pi_i, (\pi_{i\to j})_{j\in[N]} \leftarrow$ V3S.Share$(N, T, \mathbf{r}_i)$
3 : **for** $j \in [N]$ **do**
4 : $\quad \mathsf{ct}_{i\to j} \leftarrow$ SKE.Encrypt$(\mathsf{K}_{i\to j}, ([\![\mathbf{r}_i]\!]_j, \pi_{i\to j}))$
5 : st$_i$.session[sid] $:= [\![\mathbf{r}_i]\!]$
6 : **return** pm$_1[i] := (\pi_i, (\mathsf{ct}_{i\to j})_{j\in[N]})$

RB-Raccoon.ShareSign$_2$(vk, sid, act, msg, i, sk$_i$, st$_i$, msg, pm$_1$)

1 : Fetch $[\![\mathbf{r}_i]\!]$ from st$_i$.session[sid]
2 : complaints$_i := \{\}$
3 : **for** $(j \in \mathsf{pm}_1)$ **do**
4 : $\quad \pi_j, (\mathsf{ct}_{j\to k})_{k\in[N]} := \mathsf{pm}_1[j]$
5 : $\quad [\![\mathbf{r}_j]\!]_i, \pi_{j\to i} :=$ SKE.Decrypt$(\mathsf{K}_{j\to i}, \mathsf{ct}_{j\to i})$
6 : $\quad$ **if** (SKE.Decrypt failed) **or** (V3S.Verify$([\![\mathbf{r}_j]\!]_i, \pi_j, \pi_{j\to i}) =$ false) **then**
7 : $\quad\quad \mathsf{complaints}_i[j] = \{\mathsf{K}_{j\to i}, \mathsf{sig}_{j\to i}\}$ // j's ciphertext or proof is invalid
8 : valid$_i = \{j \in \mathsf{pm}_1\} \backslash \mathsf{complaints}_i$
9 : st$_i$.session[sid] $:= \{([\![\mathbf{r}_j]\!]_i)_{j\in\mathsf{valid}_i}, \mathsf{valid}_i, \mathsf{pm}_1\}$
10 : **return** pm$_2[i] :=$ complaints$_i$

RB-Raccoon.ShareSign$_3$(vk, sid, act, msg, i, sk$_i$, st$_i$, msg, pm$_2$)

1 : Fetch $(([\![\mathbf{r}_j]\!]_i)_{j\in\mathsf{valid}_i}, \mathsf{valid}_i, \mathsf{pm}_1)$ from st$_i$.session[sid]
2 : valid$_i :=$ valid$_i \cap \{j \in \mathsf{pm}_2\}$
3 : **for** $j \in [N]$ **do**
4 : $\quad$ **for** $k \in$ complaints$_j$ **do** // lines 5 to 10 study j's complaint against k
5 : $\quad\quad \{\mathsf{K}_{k\to j}, \mathsf{sig}_{k\to j}\} :=$ complaints$_j[k]$
6 : $\quad\quad$ **if** SIG.Verify$(\mathsf{pk}_k, \mathsf{sig}_{k\to j}, \{k, j, \mathsf{K}_{k\to j}\}) =$ false **then** // See Section 4.5
7 : $\quad\quad\quad$ **continue**
8 : $\quad\quad [\![\mathbf{r}_k]\!]_j, \pi_{k\to j} :=$ SKE.Decrypt$(\mathsf{K}_{k\to j}, \mathsf{ct}_{k\to j})$
9 : $\quad\quad$ **if** (SKE.Decrypt failed) **or** V3S.Verify$([\![\mathbf{r}_k]\!]_j, \pi_k, \pi_{k\to j}) =$ false **then**
10 : $\quad\quad\quad$ valid$_i =$ valid$_i \backslash \{k\}$
11 : $[\![\mathbf{r}]\!]_i := \sum_{j\in\mathsf{valid}_i} [\![\mathbf{r}_j]\!]_i$
12 : st$_i$.session[sid] $:= \{([\![\mathbf{r}_j]\!]_i)_{j\in\mathsf{valid}_i}, \mathsf{valid}_i, \mathsf{pm}_1\}$
13 : **return** pm$_3[i] := ([\![\mathbf{w}]\!]_i := [\mathbf{A} \quad \mathbf{I}] \cdot [\![\mathbf{r}]\!]_i)$

Figure 4.13: Algorithms for robust threshold signature, part 1/2. For conciseness, we omit the parsing of $(\mathsf{pm}_i)_{i\in\{1,2\}}$ in the algorithms.

RB-Raccoon.ShareSign$_4$(vk, sid, act, msg, i, sk$_i$, st$_i$, msg, pm$_3$)

1 : Fetch $((\llbracket \mathbf{r}_j \rrbracket_i)_{j\in\mathsf{valid}_i}, \mathsf{valid}_i, \mathsf{pm}_1)$ from st_i.session[sid]
2 : Parse $\mathsf{pm}_3 = (\llbracket \mathbf{w} \rrbracket_j)_j$
3 : // $\mathbf{w} = [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}$, where $\mathbf{r} = \sum_{j\in\mathsf{valid}_i} \mathbf{r}_j$
4 : $\mathbf{w} \coloneqq \mathsf{SSS.ErrorCorrect}((\llbracket \mathbf{w} \rrbracket_j)_{j\in\mathsf{valid}_i})$
5 : $c \coloneqq \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \lfloor \mathbf{w} \rceil_{\nu_\mathbf{w}}) \in \mathcal{C}$
6 : $\llbracket \mathbf{z} \rrbracket_i \coloneqq \llbracket \mathbf{r} \rrbracket_i + c \cdot \llbracket \mathbf{s} \rrbracket_i$
7 : $\llbracket \mathbf{z}^{(1)} \rrbracket_i, \llbracket \mathbf{z}^{(2)} \rrbracket_i \coloneqq \llbracket \mathbf{z} \rrbracket_i \in R_q^\ell \times R_q^k$
8 : **return** $\mathsf{pm}_4[i] = (\mathbf{w}, \llbracket \mathbf{z}^{(1)} \rrbracket_i)$

RB-Raccoon.Combine(vk, msg, pm$_4$)

1 : Parse $\mathsf{pm}_4 = (\mathbf{w}, \llbracket \mathbf{z}^{(1)} \rrbracket_j)_{j\in\mathsf{act}}$
2 : // If inconsistent $\mathbf{w}$: majority vote
3 : $\mathbf{z}^{(1)} \coloneqq \mathsf{SSS.ErrorCorrect}((\llbracket \mathbf{z}^{(1)} \rrbracket_j)_{j\in\mathsf{pm}_4})$
4 : $c \coloneqq \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \lfloor \mathbf{w} \rceil_{\nu_\mathbf{w}})$
5 : $\mathbf{u} \leftarrow \left\lfloor \mathbf{A} \cdot \mathbf{z}^{(1)} - 2^{\nu_\mathbf{b}} \cdot c \cdot \mathbf{b}_\top \right\rceil_{\nu_\mathbf{w}}$
6 : $\mathbf{h} \leftarrow \mathbf{w} - \mathbf{u}$ // Hint
7 : **return** $\mathsf{sig} \coloneqq (c, \mathbf{z}^{(1)}, \mathbf{h})$

Verify(vk = (seed, $\mathbf{b}_\top$), msg, sig)

1 : Parse $\mathsf{sig} = (c, \mathbf{z}^{(1)}, \mathbf{h})$
2 : $\mathbf{A} = \mathsf{H}_\mathbf{A}(\mathsf{seed}) \in R_q^{k\times\ell}$
3 : $c' \leftarrow \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \left\lfloor \mathbf{A} \cdot \mathbf{z}^{(1)} - 2^{\nu_\mathbf{b}} \cdot c \cdot \mathbf{b}_\top \right\rceil_{\nu_\mathbf{w}} + \mathbf{h})$
4 : **if** $c = c' \wedge \left\| (\mathbf{z}^{(1)}, 2^{\nu_\mathbf{w}} \cdot \mathbf{h}) \right\| \leqslant \beta$ **then**
5 : **return** true
6 : **return** false

Figure 4.14: Algorithms for robust threshold signature, part 2/2. For conciseness, we omit the parsing of $(\mathsf{pm}_i)_{i\in\{1,3,4\}}$, vk and sig in the algorithms. Challenges c are sampled from the challenge space $\mathcal{C} = \{c \in R_q \mid \|c\|_\infty = 1 \wedge \|c\|_1 = \omega\}$.

V' contains the part of the secret that can be provided by the corrupted parties during the DKG protocol.

Lemma 4.6.1. *Assume that $T \geqslant 3t+1$. There exist efficient simulators SimDKG_1, SimDKG_2 such that for any polynomial-time adversary $\mathcal{A}$, we can derive adversaries $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4$ against respectively the* V3S-sound, EUF-CMA, IND-CPA, *and* V3S-fk *security notions, running in time $\mathsf{Time}(\mathcal{B}_i) \approx \mathsf{Time}(\mathcal{A})$ for $i \in \{1,...,4\}$, and overwhelming probabilities $1-p_1, 1-p_2$ – corresponding to the correctness of the signature scheme and of the* V3S– *such that*

- $\left|\Pr\left[\mathsf{Game}_{\mathcal{A}}^{\text{dkg-real}}(1^\lambda, N, T, t) = 1\right] - \Pr\left[\mathsf{Game}_{\mathcal{A}}^{\text{dkg-leaky}}(1^\lambda, N, T, t) = 1\right]\right| \leqslant \delta(1^\lambda)$
- $\Pr\left[\mathsf{Game}_{\mathcal{A}}^{\text{dkg-leaky}}(1^\lambda, N, T, t) = \bot\right] \leqslant \delta(1^\lambda)$

where

$$\begin{aligned}\delta(1^\lambda) :=& \mathsf{Adv}_{\mathcal{B}_1}^{\text{V3S-sound}}(1^\lambda) + N \cdot t \cdot (1-p_1) + \mathsf{Adv}_{\mathcal{B}_2}^{\text{EUF-CMA}}(1^\lambda) \\ &+ N \cdot (1-p_2) + N^2 \mathsf{Adv}_{\mathcal{B}_3}^{\text{IND-CPA}}(1^\lambda) + N \cdot \mathsf{Adv}_{\mathcal{B}_4}^{\text{V3S-fk}}(1^\lambda)\end{aligned}$$

$\mathsf{RB\text{-}Raccoon.LeakyKeygen}^{\mathsf{H_A}}(N, T, t, \mathsf{bias} = (\mathsf{seed}, [\![\mathbf{s}_{\mathsf{CS}}]\!], ([\![\mathbf{s}_{\mathsf{HS}}]\!]_i)_{i\in\mathsf{CS}'}, (\pi_i)_{i\in\mathsf{HS}}))$

$\mathbf{s}_{\mathsf{CS}} \coloneqq \mathsf{V3S.Reconstruct}(([\![\mathbf{s}_{\mathsf{CS}}]\!]_i)_i)$
if $\mathbf{s}_{\mathsf{CS}} = \bot \vee \mathbf{s}_{\mathsf{CS}} \notin V'$ **then return** $\bot, \varnothing$
$\mathbf{s}_{\mathsf{HS}} \leftarrow \chi'_{\mathsf{sk}} \coloneqq \sum_{i\in\mathsf{HS}} \mathsf{SimSecret}(\pi_i)$
Interpolate $P \in R_q^2[X]$ of degree $\leqslant t$ such that $P(0) = \mathbf{s}_{\mathsf{HS}} \wedge \forall i \in \mathsf{CS}', P(i) = [\![\mathbf{s}_{\mathsf{HS}}]\!]_i$
$\mathbf{A} \coloneqq \mathsf{H}_{\mathbf{A}}(\mathsf{seed})$
$\mathbf{b} \coloneqq [\mathbf{A} \quad \mathbf{I}] \cdot (\mathbf{s}_{\mathsf{CS}} + \mathbf{s}_{\mathsf{HS}}) \bmod q$
$\mathbf{b}_\top \coloneqq \lfloor \mathbf{b} \rceil_{\nu_{\mathbf{b}}}$
return $\mathsf{vk} = (\mathsf{seed}, \mathbf{b}_\top), \mathsf{aux} = \mathbf{b}, (\mathsf{sk}_i \coloneqq (P(i) + [\![\mathbf{s}_{\mathsf{CS}}]\!]_i))_{i\in[N]}$

Figure 4.15: Algorithms for the leaky key generation of RB-Raccoon, capturing the bias an adversary introduces during the distributed key generation. V' contains all the elements of V convoluted with itself at most t times. CS' is any set of size t containing CS.

Proof. We proceed via a sequence of hybrids, starting from $\mathsf{Game}_{\mathcal{A}}^{\text{dkg-real}}$ and ending at $\mathsf{Game}_{\mathcal{A}}^{\text{dkg-leaky}}$. The proof is composed of two main parts, as we first enforce that the contributions of corrupted parties are valid sharings (hybrid 2), and then simulate the honest parties' contributions (hybrids 3 to 6). We will denote by ε_i the probability that the i-th hybrid returns 1. We will show that for each hybrid i, $|\varepsilon_{i+1} - \varepsilon_i|$ is negligible, and that the final hybrid corresponds to $\mathsf{Game}_{\mathcal{A}}^{\text{dkg-leaky}}$.

We will introduce game aborts in the hybrids, e.g. the entire game returns 0 if some condition is not met. We will ensure that the probability of these aborts is negligible, so that they do not impact the overall advantage of the adversary.

The simulators SimDKG_1 and SimDKG_2 will be defined at the end of the proof, based on the final hybrid. Aborts in intermediary hybrids will be handled by returning an arbitrary output from the simulators, and defining an adversary $\mathcal{B}$ that returns 0 in these cases.

Hybrid_1. This hybrid corresponds to $\mathsf{Game}_{\mathcal{A}}^{\text{dkg-real}}$.

Hybrid_2. In this hybrid, we assert for each corrupted party i that we are always in one of the following cases:

- If no honest party complains against it during round 2, the honest parties receive from user i consistent shares $[\![\mathbf{s}_i]\!]_j$ of some secret $\mathbf{s}_i$, that belongs to V.

$\mathsf{Game}^{\mathsf{dkg\text{-}real}}_{\mathcal{A}}(1^\lambda, N, T, t)$

$\mathsf{H} := \mathsf{H}_{\mathsf{seed}}, \mathsf{H}_{\mathbf{A}}, \mathsf{H}_{\mathbf{R}}, \mathsf{H}_{\mathsf{MT}}$
$(\mathsf{CS}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathsf{H}}(1^\lambda, N, T, t)$ // $\mathcal{A}$ chooses corrupted parties
assert $\mathsf{CS} \subseteq [N] \wedge |\mathsf{CS}| \leqslant t$
$\mathsf{HS} := [N] \backslash \mathsf{CS}$
$(\mathsf{st}_i)_{i \in [N]} \leftarrow \mathsf{Setup}(1^\lambda, N, T, t)$
$\mathsf{st}_{\mathcal{A}} \leftarrow \mathcal{A}((\mathsf{st}_i)_{i \in \mathsf{CS}}, \mathsf{st}_{\mathcal{A}})$
for $r \in [R_{\mathsf{KG}}]$ **do**
  for $i \in \mathsf{HS}$ **do**
    $\mathsf{pm}^i_r, \mathsf{st}_i \leftarrow \mathsf{ShareKeygen}(i, \mathsf{st}_i, \mathsf{pm}_{r-1})$
    if $\mathsf{pm}^i_r = \perp$ **then return** 0
  $((\mathsf{pm}^i_r)_{i \in \mathsf{CS}}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathsf{H}}((\mathsf{pm}^i_r)_{i \in \mathsf{HS}}, \mathsf{st}_{\mathcal{A}})$
for $i \in \mathsf{HS}$ **do** $\mathsf{sk}_i := \mathsf{PartialSecret}(\mathsf{st}_i)$
$(\mathsf{vk}, \mathsf{aux}) \leftarrow \mathsf{CombineKey}((\mathsf{pm}_r)_{r \in [R_{\mathsf{KG}}]})$
$\mathsf{info} := ((\mathsf{pk}^{\mathsf{SIG}}_i)_{i \in [N]}, (K^{\mathsf{SKE}}_{i \to j}, \mathsf{sig}^{\mathsf{SKE}}_{i \to j})_{(i,j) \in \mathsf{CS} \times \mathsf{HS} \vee (i,j) \in \mathsf{HS} \times \mathsf{CS}})$
$b \leftarrow \mathcal{A}^{\mathsf{H}, (\mathsf{SKE.Encrypt}(\mathsf{K}_{i \to j}, \cdot))_{i,j \in \mathsf{HS}}}(\mathsf{info}, \mathsf{vk}, \mathsf{aux}, (\mathsf{sk}_i)_{i \in \mathsf{HS}}, \mathsf{st}_{\mathcal{A}})$
return $b = 1$

$\mathsf{Game}^{\mathsf{dkg\text{-}leaky}}_{\mathcal{A}}(1^\lambda, N, T, t)$

$\mathsf{H} := \mathsf{H}_{\mathsf{seed}}, \mathsf{H}_{\mathbf{A}}, \mathsf{H}_{\mathbf{R}}, \mathsf{H}_{\mathsf{MT}}$
$(\mathsf{CS}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathsf{H}}(1^\lambda, N, T, t)$
assert $\mathsf{CS} \subseteq [N] \wedge |\mathsf{CS}| \leqslant t$
$\mathsf{HS} := [N] \backslash \mathsf{CS}$
$(\mathsf{st}_i)_{i \in [N]} \leftarrow \mathsf{Setup}(1^\lambda, N, T, t)$
$\mathsf{st}_{\mathcal{A}} \leftarrow \mathcal{A}((\mathsf{st}_i)_{i \in \mathsf{CS}}, \mathsf{st}_{\mathcal{A}})$
$\mathsf{info} := ((\mathsf{pk}^{\mathsf{SIG}}_i)_{i \in [N]}, (K^{\mathsf{SKE}}_{i \to j}, \mathsf{sig}^{\mathsf{SKE}}_{i \to j})_{(i,j) \in \mathsf{CS} \times \mathsf{HS} \vee (i,j) \in \mathsf{HS} \times \mathsf{CS}})$
$\mathsf{bias}, \mathsf{st}_{\mathcal{A}} \leftarrow \mathsf{SimDKG}_1^{\mathsf{H}, (\mathsf{SKE.Encrypt}(\mathsf{K}_{i \to j}, \cdot))_{i,j \in \mathsf{HS}}}(\mathsf{info}, \mathcal{A}, \mathsf{st}_{\mathcal{A}})$
$\mathsf{vk}, \mathsf{aux}, (\mathsf{sk}_i)_{i \in [N]} \leftarrow \mathsf{LeakyKeygen}^{\mathsf{H}_{\mathbf{A}}}(N, T, t, \mathsf{bias})$
if $\mathsf{vk} = \perp$ **then return** $\perp$
$\mathsf{st}_{\mathcal{A}} \leftarrow \mathsf{SimDKG}_2^{\mathsf{H}, (\mathsf{SKE.Encrypt}(\mathsf{K}_{i \to j}, \cdot))_{i,j \in \mathsf{HS}}}(\mathsf{vk}, \mathsf{aux}, \mathsf{info}, \mathcal{A}, \mathsf{st}_{\mathcal{A}})$
$b \leftarrow \mathcal{A}^{\mathsf{H}, (\mathsf{SKE.Encrypt}(\mathsf{K}_{i \to j}, \cdot))_{i,j \in \mathsf{HS}}}(\mathsf{info}, \mathsf{vk}, \mathsf{aux}, (\mathsf{sk}_i)_{i \in \mathsf{HS}}, \mathsf{st}_{\mathcal{A}})$
return $b = 1$

Figure 4.16: Games capturing the real and simulated leaky distributed key generation of RB-Raccoon.

- Otherwise, at least one complaint against i is accepted by all honest users during round 3 of the key generation protocol, and the contribution of the corrupted user i will be discarded.

This holds by the soundness of V3S and the correctness of the signature scheme SIG used to sign the pairwise keys.

First, a honest user j will complain in round 2 against a corrupted user i if it sends an invalid encryption or a share that does not pass the V3S verification. By the correctness of SIG, this complaint will be approved by other honest users as they will then run the same checks as j and determine that i misbehaved.

Otherwise, when no honest user complains against a corrupted user i, it means that they all receive shares $[\![\mathbf{s}_i]\!]_j$ that pass the verification of the V3S. As there are at least $T - t \geqslant t + 1$ honest users checking their shares, by the soundness of V3S we are guaranteed that the shares $([\![\mathbf{s}_i]\!]_j)_{j\in\mathsf{HS}}$ are consistent and that they reconstruct to a valid secret $\mathbf{s}_i \in V$, except with negligible probability.

We conclude that there exists an adversary $\mathcal{B}_1$ against the soundness of V3S such that:

$$|\varepsilon_2 - \varepsilon_1| \leqslant \mathsf{Adv}^{\mathsf{V3S\text{-}sound}}_{\mathcal{B}_1}(1^\lambda) + N \cdot t \cdot (1 - p_1)$$

where $1 - p_1 = \mathsf{negl}(\lambda)$ is the probability that the signature produced by SIG is invalid.

Remark 4.6.2. The soundness game applies to all corrupted users at once as the adversary $\mathcal{B}_1$ can determine which corrupted user breaks soundness and return only its contribution. However, we need to apply a union bound for the correctness of the signature scheme.

Hybrid_3. In this hybrid, we assert that (i) no complaint against a honest user is raised by another honest user, and (ii) no complaint against a honest user from a corrupted user is accepted.

This holds thanks to the correctness of the V3S which guarantees that honest users produce sharings that pass verification with overwhelming probability, and due to the unforgeability of SIG which ensures that corrupted users cannot invalidly frame a honest user with an invalid encryption key.

First, by the correctness of V3S, each honest user i will send shares that pass verification to all other users with overwhelming probability $1 - N \cdot \mathsf{negl}(\lambda)$.

In this case, honest users will never raise a complaint against other honest users as all checks will pass. As for corrupted users, since we use a broadcast channel, the only way they could frame a honest user in their complaint is if it could convince other users that the honest user i encrypted its messages with a different key, hence breaking the verification. However, by the unforgeability of SIG and as the adversary does not have access to its key, this happens with negligible probability.

We conclude that there exists an adversary $\mathcal{B}_2$ against the EUF-CMA security of SIG such that:

$$|\varepsilon_3 - \varepsilon_2| \leqslant \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{B}_2}(1^\lambda) + N \cdot (1 - p_2)$$

where $1 - p_2 = \mathsf{negl}(\lambda)$ is the probability that V3S produces a sharing that does not pass verification for all users.

Hybrid_4. In this hybrid, we replace the encryptions between honest users (e.g. of $([\![\mathbf{s}_i]\!]_j, \pi_{i,j})$, for $i, j \in \mathsf{HS}$) by encryptions of zero. We instead store those shares in a table and retrieve them from the table in round 3. This ensures that before round 3 completes, the view of the adversary only depends on honest secrets through the proofs broadcasted by honest users and the corrupted shares it receives.

This change is indistinguishable for the adversary due to the IND-CPA security of SKE. Indeed, at this point the adversary never gets direct access to $\mathsf{K}_{i\to j}$ for $i, j \in \mathsf{HS}$ and only has an encryption oracle, corresponding to the view in the IND-CPA game. Thus, it cannot distinguish whether a given ciphertext encrypts the value $([\![\mathbf{s}_i]\!]_j, \pi_{i,j})$ or 0.

We apply it to all encryptions between honest users. There exists an adversary $\mathcal{B}_3$ against the IND-CPA security of SKE such that:

$$|\varepsilon_4 - \varepsilon_3| \leqslant N^2 \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{B}_3}(1^\lambda)$$

Hybrid_5. In this hybrid, we define a set CS' of cardinality t, such that $\mathsf{CS} \subseteq \mathsf{CS}'$. We change the conceptual order in which the variables defining the honest users' contributions are sampled. Instead of sampling a random polynomial P_i (defining $\mathbf{s}_i$ and all shares), we first sample the secret $\mathbf{s}_i$ and the shares for the set CS' $((\llbracket \mathbf{s}_i \rrbracket_j)_{j\in\mathsf{CS}'})$ uniformly at random, and define the polynomial P_i via Lagrange interpolation from these $t+1$ points.

The resulting distribution of shares and secrets remains identical to the previous hybrid. This is because in a $t+1$-out-of-N Shamir secret sharing scheme, there is a bijection between a polynomial of degree t and the set of images consisting of the secret (evaluation at 0) and any t shares (evaluation at $x \in \mathsf{CS}'$).

Thus,

$$\varepsilon_5 = \varepsilon_4$$

Hybrid_6. In this hybrid, we modify how the proofs and the shares for CS' are generated. We apply the fragmentary knowledge property of V3S to generate the view of the adversary.

Specifically, for each honest user i, instead of sampling $\mathbf{s}_i$ and $(\llbracket \mathbf{s}_i \rrbracket_j)_{j\in\mathsf{CS}'}$ and then computing the proof π_i, we:

1. Run $(\pi_i, (\llbracket \mathbf{s}_i \rrbracket_j)_{j\in\mathsf{CS}'}, \pi_{i\to j}) \leftarrow \mathsf{SimProof}(\mathsf{CS}')$ to generate the proof and the shares destined for CS'.
2. Defer the sampling of the secret $\mathbf{s}_i$ to round 3, then obtained with $\mathbf{s}_i \leftarrow \mathsf{SimSecret}(\pi_i)$.

This change is indistinguishable to the adversary. The V3S-fk security definition guarantees that the joint distribution of the proof and the shares for any set of size t produced by SimProof is computationally indistinguishable from those generated by the real V3S.Share. Furthermore, SimSecret ensures that the secret $\mathbf{s}_i$ sampled later is consistent with the proof π_i revealed earlier.

We can do so as, at this point (due to Hybrid_4), the view of the adversary in Rounds 1 and 2 depends only on the proofs π_i and the shares for CS'. The shares for honest users (which would require the full polynomial) are masked by encryption of zero and are not needed until Round 3.

We apply this reasoning to each honest user, and conclude that there exists an adversary $\mathcal{B}_4$ against the V3S-fk security of V3S such that:

$$|\varepsilon_6 - \varepsilon_5| \leqslant N \cdot \mathsf{Adv}^{\mathsf{V3S\text{-}fk}}_{\mathcal{B}_4}(1^\lambda)$$

CONCLUSION. We can now define the simulators SimDKG_1 and SimDKG_2 used in $\mathsf{Game}^{\mathsf{dkg\text{-}leaky}}_{\mathcal{A}}$. SimDKG_1 simulates the view of the adversary in Hybrid_6 until the beginning of round 3, and constructs the bias information required by LeakyKeygen, i.e. the seed, the sharing of the secret sampled by corrupted users, the proofs of honest users, and the shares of honest users for users in CS'. It defines $\llbracket \mathbf{s}_{\mathsf{CS}} \rrbracket$ as the sum of the corrupted secrets that have been accepted during round 3, and $\llbracket \mathbf{s}_{\mathsf{HS}} \rrbracket_i$ as the shares of honest users for $i \in \mathsf{CS}'$. SimDKG_2 simulates the rest of the execution of Hybrid_6 after the public key and honest secrets have been sampled by LeakyKeygen: honest users broadcast shares of $\llbracket \mathbf{b} \rrbracket$ computed from the key shares returned by LeakyKeygen.

While some aborts were introduced in the hybrids, they all occur before honest secrets are sampled in round 3 of the key generation protocol, e.g. in the part handled by SimDKG_1. Thus, we handle them by returning an arbitrary bias from SimDKG_1 such that LeakyKeygen will abort. We will later see that this is the only case where $\mathsf{Game}^{\mathsf{dkg\text{-}leaky}}_{\mathcal{B}}$ returns $\perp$.

First, we can observe that the only case where $\mathsf{Game}_{\mathcal{A}}^{\mathsf{dkg\text{-}leaky}}$ returns $\perp$ is when LeakyKeygen returns $\perp$. Previous hybrids added assertions that $[\![\mathbf{s}_{\mathsf{CS}}]\!]$ was consistent, hence, this can only happen when one of such assertions failed in Hybrid_6. Applying the same transitions between hybrids as above and summing all the bounds, the probability of this event is bounded by $\delta(1^\lambda)$.

Then, we wish to show that the probability that Hybrid_6 and $\mathsf{Game}_{\mathcal{A}}^{\mathsf{dkg\text{-}leaky}}$ return 1 is identical. The case where an abort occurred in Hybrid_6 is not important here as both games return a value different from 1. When no abort occurs, SimDKG_1 directly executes the same steps as in Hybrid_6 until the beginning of round 3. At this point, we can observe that LeakyKeygen samples the honest secrets using SimSecret based on the proofs of honest users, exactly as in Hybrid_6, and reconstructs the final secret key shares consistently with Hybrid_6 using the corrupted shares accepted by honest users. SimDKG_2 then simulates the rest of the execution of Hybrid_6. CombineKey in Hybrid_6 and LeakyKeygen reconstruct the same final public key, to be consistent with the final secret key shares of honest users. This is because shares accepted from corrupted users must be consistent and as $N \geqslant 3t+1$, at least $2t+1$ honest users use shares of the same secret at the end of round 3 to compute the final secret key shares and $[\![\mathbf{b}]\!]_i$. As we use error correction, it ensures that CombineKey will reconstruct the same final public key as the one corresponding to the secrets accepted by honest users.

Thus $\Pr\left[\mathsf{Game}_{\mathcal{A}}^{\mathsf{dkg\text{-}leaky}}(1^\lambda, N, T, t) = 1\right] = \varepsilon_6$.

We conclude the proof by summing all the bounds between hybrids:

$$\begin{aligned}
&\left|\Pr\left[\mathsf{Game}_{\mathcal{A}}^{\mathsf{dkg\text{-}real}}(1^\lambda, N, T, t) = 1\right] - \Pr\left[\mathsf{Game}_{\mathcal{A}}^{\mathsf{dkg\text{-}leaky}}(1^\lambda, N, T, t) = 1\right]\right| \\
&\leqslant \delta(1^\lambda) := \mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{V3S\text{-}sound}}(1^\lambda) + N \cdot t \cdot (1-p_1) + \mathsf{Adv}_{\mathcal{B}_2}^{\mathsf{EUF\text{-}CMA}}(1^\lambda) + N \cdot (1-p_2) \\
&\qquad + N^2 \mathsf{Adv}_{\mathcal{B}_3}^{\mathsf{IND\text{-}CPA}}(1^\lambda) + N \cdot \mathsf{Adv}_{\mathcal{B}_4}^{\mathsf{V3S\text{-}fk}}(1^\lambda)
\end{aligned}$$

□

4.6.2 Unforgeability

In order for the unforgeability proof to go through, we need to make several assumptions on the parameters of RB-Raccoon. We will need to aggregate the secrets and noises of honest users, and ensure that the commitments used during signing have enough min-entropy.

We require the following conditions on the parameters of RB-Raccoon:

- $q_{\nu_{\mathbf{b}}}$ and $q_{\nu_{\mathbf{w}}}$ (as defined in Section 3.6.1) verify $q_{\nu_{\mathbf{b}}} = \lceil q/2^{\nu_{\mathbf{b}}} \rceil$ and $q_{\nu_{\mathbf{w}}} = \lceil q/2^{\nu_{\mathbf{w}}} \rceil$.
- $T \geqslant 3t+1$, as in Lemma 4.6.1.
- We consider Gaussian distributions $\chi_{\mathsf{sk}} := D_{\sigma_{\mathsf{sk}}}$ and $\chi_{\mathbf{r}} := D_{\sigma_{\mathbf{r}}}$.
- The noise $\mathbf{y}$ used for masking in the V3S is sampled from $\chi_{\mathbf{y}} := D_{\sigma_{\mathbf{y}}}$ for key generation, and from $\chi_{\mathbf{y}} := D_{\sigma'_{\mathbf{y}}}$ during signing.
- We define $B_{\mathbf{R}}$ such that the matrices $\mathbf{R}$ used by our V3S verify $\|\mathbf{R}\|_2 \leqslant B_{\mathbf{R}}$ with overwhelming probability.
- $\sigma'^{-2}_{\mathsf{sk}} := 2 \cdot \left(\sigma_{\mathsf{sk}}^{-2} + \frac{B_{\mathbf{R}}^2}{\sigma_{\mathbf{y}}^2}\right)$ verifies $\sigma'_{\mathsf{sk}} \geqslant \sqrt{2}\eta'_\varepsilon(\mathbb{Z}^{n(k+\ell)})$ for some negligible ε.
- $\sigma'^{-2}_{\mathbf{r}} := 2 \cdot \left(\sigma_{\mathbf{r}}^{-2} + \frac{B_{\mathbf{R}}^2}{\sigma_{\mathbf{y}}'^2}\right)$ verifies $\sigma'_{\mathbf{r}} \geqslant \sqrt{2}\eta'_\varepsilon(\mathbb{Z}^{n(k+\ell)})$ for some negligible ε.
- $\sqrt{2t} \cdot \sigma'_{\mathbf{r}} \geqslant 2n \cdot q^{\frac{1}{k+\ell} + \frac{2}{n\ell}}$ and $\nu_{\mathbf{w}} < \log(q) - 2$.

We now prove the unforgeability of RB-Raccoon. Recall that we consider a synchronous network with a broadcast channel during signing, as introduced in Fig. 3.4.

Theorem 4.6.3. *Under the parameter constraints above,* RB-Raccoon *is* SYNC-TS-UF *secure in the random oracle model.*

Formally, for any adversary $\mathcal{A}$ against the SYNC-TS-UF *security of* RB-Raccoon *making at most Q_s signing queries, and Q_{H_c}, $Q_{\mathsf{H}_\mathbf{A}}$, and $Q_{\mathsf{H}_\mathbf{R}}$ calls respectively to the random oracle H_c, $\mathsf{H}_\mathbf{R}$, and $\mathsf{H}_\mathbf{A}$, there exists an adversary $\mathcal{B}_1$ against the* V3S-sound *security of* V3S*, an adversary $\mathcal{B}_2$ against the* EUF-CMA *security of* SIG*, and an adversary $\mathcal{B}_3$ against the* IND-CPA *security of* SKE*, an adversary $\mathcal{B}_4$ against the* V3S-fk *security of* V3S*, an adversary $\mathcal{B}_5$ against the $\mathsf{Hint\text{-}MLWE}_{R_q,k,\ell,Q_s,\sqrt{2t}\sigma'_{\mathsf{sk}},\sqrt{2t}\sigma'_{\mathbf{r}},\mathcal{C}}$ problem, and an adversary $\mathcal{B}_6$ against the $\mathsf{SelfTargetMSIS}_{R_q,k,\ell+1,\beta_{\mathsf{stmsis}}}$ problem with $\beta_{\mathsf{stmsis}} = \beta + \sqrt{\omega} + (\omega \cdot 2^{\nu_\mathbf{b}-1} + 2^{\nu_\mathbf{w}-1}) \cdot \sqrt{nk}$ such that:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{SYNC\text{-}TS\text{-}UF}}_{\mathcal{A}}(\lambda) \leqslant\ & \delta(1^\lambda) + \mathsf{Adv}^{\mathsf{V3S\text{-}sound}}_{\mathcal{B}_1}(1^\lambda) + \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{B}_2}(1^\lambda) \\ & + Q_s \cdot N^2 \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{B}_3}(1^\lambda) + Q_s \cdot N \cdot \mathsf{Adv}^{\mathsf{V3S\text{-}fk}}_{\mathcal{B}_4}(1^\lambda) \\ & + Q_{\mathsf{H}_\mathbf{A}} \cdot \left(\mathsf{Adv}^{\mathsf{Hint\text{-}MLWE}}_{\mathcal{B}_5}(1^\lambda) + \mathsf{Adv}^{\mathsf{SelfTargetMSIS}}_{\mathcal{B}_6}(1^\lambda)\right) \\ & + Q_s \cdot N \cdot t \cdot (1-p_1) + Q_s \cdot N \cdot (1-p_2) + 2(Q_s+1) \cdot N \cdot 8\varepsilon \\ & + Q_{\mathsf{H}_\mathbf{R}} \cdot p_\mathbf{R} + Q_s \cdot (Q_{\mathsf{H}_c} + Q_s) \cdot 2^{-n+2}\end{aligned}$$

where p_1, p_2 are the probabilities of success of SIG *and* V3S *respectively, $p_\mathbf{R}$ is the probability that any of the sampled matrices $\mathbf{R}$ during the execution has $\|\mathbf{R}\|_2 > B_\mathbf{R}$, and $\delta(1^\lambda)$ is negligible and defined in Lemma 4.6.1.*

Proof. Before proving the main theorem, we recall useful lemmas.

We will need to extract and aggregate isotropic Gaussian samples from the samples generated by SimSecret from Fig. 4.6. To do so, we recall the following convolution lemma from [Pei10], and a useful corollary proven in [KLSS23].

Lemma 4.6.4 (Simplified convolution lemma [Pei10, Thm. 4.5]). *Let positive reals σ_1, σ_2 such that $\sigma_1^{-2} + \sigma_2^{-2} \leqslant \eta'_\varepsilon(\mathbb{Z}^m)^{-2}$ for $0 < \varepsilon < 1/2$. Then the distribution $D_{\mathbb{Z}^m,\sigma_1} + D_{\mathbb{Z}^m,\sigma_2}$ is within statistical distance 8ε of $D_{\mathbb{Z}^m,\sqrt{\sigma_1^2+\sigma_2^2}}$.*

Lemma 4.6.5 (Adapted from the proof of [KLSS23, Theorem 1]). *For any $\sigma \geqslant \eta'_\varepsilon(\mathbb{Z}^m)$ and positive definite matrix $\mathbf{\Sigma}$ such that $\left\|\mathbf{\Sigma}^{-1}\right\|_2 \leqslant \frac{1}{2\sigma^2}$, there exists $\mathbf{\Sigma}'$ such that the two following distributions are within statistical distance 8ε:*

- $\mathbf{r} \leftarrow D_{\mathbb{Z}^m,\sqrt{\mathbf{\Sigma}},\mathbf{c}}$
- $\mathbf{r} \leftarrow D_{\mathbb{Z}^m,\sigma} + D_{\mathbb{Z}^m,\sqrt{\mathbf{\Sigma}'},\mathbf{c}}$

We will also need a bound on the min-entropy of rounded MLWE samples, straightforwardly adapted from [DKMM+24] to accept a shift $\mathbf{c}$.

Definition 4.6.6. *For any matrix $\mathbf{A} \in R_q^{k\times\ell}$, let $\mathcal{D}^{\mathsf{r-MLWE}}_{q,\ell,k,\sigma,\mathbf{c},\nu}(\mathbf{A})$ be the distribution defined by sampling $\mathbf{r} \leftarrow D_{R^{k+\ell},\sigma}$ and outputting $\mathbf{w} = \left\lfloor \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{r} + \mathbf{c} \right\rceil_\nu$.*

Lemma 4.6.7. *For any $\sigma \geqslant 2n \cdot q^{\frac{1}{k+\ell}+\frac{2}{n\ell}}$ and $\nu < \log(q) - 2$, we have:*

$$\Pr_{\mathbf{A}\leftarrow R_q^{k\times\ell}}\left[H_\infty\left(\mathcal{D}^{\mathsf{r-MLWE}}_{q,\ell,k,\sigma,\mathbf{c},\nu}(\mathbf{A})\right) \geqslant n-1\right] \geqslant 1 - 2^{-n+1}$$

We now prove the unforgeability of RB-Raccoon in the synchronous setting. We proceed via a sequence of hybrids, starting from the real SYNC-TS-UF game and ending at a game that we reduce to the SelfTargetMSIS problem. We will denote by ε_i the probability that the i-th hybrid returns 1.

At a high level, hybrids 2 to 7 apply the V3S properties to the key generation and signing protocols, and determine the part of the secrets and signing noise that remains unknown to the adversary after broadcasting proofs.

Hybrid_1. This hybrid corresponds to the real SYNC-TS-UF game from Fig. 3.4.

Hybrid_2 AND Hybrid_3. In these hybrids, we apply the same reasoning as in Hybrid_2 and Hybrid_3 of the leaky DKG proof to ensure that the contributions of corrupted parties are valid sharings, and that no complaint against honest users is raised or accepted.

We apply the arguments for each signing query made by the adversary, leading to a multiplicative factor Q_s in the bounds.

We conclude that there exist adversaries $\mathcal{B}_1, \mathcal{B}_2$ against respectively the V3S-sound and EUF-CMA problems, and overwhelming probabilities $1-p_1, 1-p_2$ – corresponding to the correctness of the signature scheme and of the V3S– such that

$$\begin{aligned}|\varepsilon_3 - \varepsilon_1| \leqslant & \mathsf{Adv}^{\mathsf{V3S\text{-}sound}}_{\mathcal{B}_1}(1^\lambda) + Q_s \cdot N \cdot t \cdot (1-p_1) \\ & + \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{B}_2}(1^\lambda) + Q_s \cdot N \cdot (1-p_2)\end{aligned}$$

Hybrid_4. In this hybrid, we replace the key generation protocol by the leaky DKG from Fig. 4.15, using the simulators SimDKG_1 and SimDKG_2 defined in the proof of the leaky DKG Lemma 4.6.1.

This is possible as Hybrid_3 ensures that the game never tries to decrypt messages between honest users. Hence, access to the encryption oracle for honest users in the games of Fig. 4.16 is sufficient to simulate the view of the adversary. Furthermore, the games provide access to the values needed during signing queries, e.g.:

- The verification key vk.
- The final shares of honest users $(\mathsf{sk}_i)_{i\in\mathsf{HS}}$.
- The pairwise keys and their associated signatures between corrupted and honest users $(K^{\mathsf{SKE}}_{i\to j}, \mathsf{sig}^{\mathsf{SKE}}_{i\to j})_{(i,j)\in\mathsf{CS}\times\mathsf{HS}\vee(i,j)\in\mathsf{HS}\times\mathsf{CS}}$. These keys are used (i) to encrypt (resp. decrypt) shares to (resp. from) corrupted users in $\mathsf{ShareSign}_1$ (resp $\mathsf{ShareSign}_3$), and (ii) to complain against misbehaving users in $\mathsf{ShareSign}_2$.

We conclude that

$$|\varepsilon_4 - \varepsilon_3| \leqslant \delta(1^\lambda)$$

where $\delta(1^\lambda)$ is defined in Lemma 4.6.1.

Hybrid_5. In this hybrid, we proceed as in Hybrid_4 of the leaky DKG proof, replacing encryptions between honest users by encryptions of zero for the signing protocol, and we store the corresponding values in a table.

Previous hybrids ensure that the key generation and signing queries never try to decrypt messages between honest users. Hence, access to the encryption oracle of the IND-CPA game is sufficient to simulate the view of the adversary.

There exists an adversary $\mathcal{B}_3$ against the IND-CPA security of SKE such that:

$$|\varepsilon_5 - \varepsilon_4| \leqslant Q_s \cdot N^2 \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{B}_3}(1^\lambda)$$

Hybrid_6. In this hybrid, we use the same set CS' as in the leaky DKG proof, such that $\mathsf{CS} \subseteq \mathsf{CS}'$, and change the order in which honest users' contributions are sampled. This is done as in Hybrid_5 of the leaky DKG proof.

Concretely, for each honest user i, we first sample $\mathbf{r}_i$ and $(\llbracket \mathbf{r}_i \rrbracket_j)_{j\in\mathsf{CS}'}$ uniformly at random, and define the polynomial P_i via Lagrange interpolation from these $t+1$ points.

The resulting distribution of shares and secrets remains identical to the previous hybrid, and thus

$$\varepsilon_6 = \varepsilon_5$$

Hybrid_7. In this hybrid, we modify how the proofs and the shares for CS' are generated during signing. We apply the fragmentary knowledge property of V3S to generate the view of the adversary, as in Hybrid_6 of the leaky DKG proof.

Specifically, for each honest user i, instead of sampling $\mathbf{r}_i$ and $(\llbracket \mathbf{r}_i \rrbracket_j)_{j\in\mathsf{CS}'}$ and then computing the proof π_i, we:

1. Run $(\pi_i, (\llbracket \mathbf{r}_i \rrbracket_j)_{j\in\mathsf{CS}'}, \pi_{i\to j}) \leftarrow \mathsf{SimProof}(\mathsf{CS}')$ to generate the proof and the shares destined for CS'.
2. Defer the sampling of the secret $\mathbf{r}_i$ to round 3 of signing queries – after checking complaints, then execute $\mathbf{r}_i \leftarrow \mathsf{SimSecret}(\pi_i)$.

This change is indistinguishable to the adversary. The V3S-fk security definition guarantees that the joint distribution of the proof and the shares for any set of size t produced by SimProof is computationally indistinguishable from those generated by the real V3S.Share. Furthermore, SimSecret ensures that the secret $\mathbf{r}_i$ sampled later is consistent with the proof π_i revealed earlier.

Hence, there exists an adversary $\mathcal{B}_4$ against the V3S-fk security of V3S such that:

$$|\varepsilon_7 - \varepsilon_6| \leqslant Q_s \cdot N \cdot \mathsf{Adv}^{\mathsf{V3S\text{-}fk}}_{\mathcal{B}_4}(1^\lambda)$$

Hybrid_8. In this hybrid, we sample the matrices $\mathbf{R}$ that will be output by the random oracle $\mathsf{H}_{\mathbf{R}}$ at the beginning of the game, and assert that they all verify $\|\mathbf{R}\|_2 \leqslant B_{\mathbf{R}}$.

This holds with overwhelming probability by our assumption on $B_{\mathbf{R}}$. Thus,

$$|\varepsilon_8 - \varepsilon_7| \leqslant Q_{\mathsf{H}_{\mathbf{R}}} \cdot p_{\mathbf{R}}$$

where $p_{\mathbf{R}} = \mathsf{negl}(\lambda)$ is the probability that a matrix $\mathbf{R}$ does not verify $\|\mathbf{R}\|_2 \leqslant B_{\mathbf{R}}$.

Hybrid_9. In this hybrid, we extract isotropic Gaussian samples from the secrets and noises generated by SimSecret during key generation and signing queries, using Lemma 4.6.5.

Specifically, for each honest user i, during key generation we replace the sampling of $\mathbf{s}_i \leftarrow \mathsf{SimSecret}(\pi_i)$ by:

- Sample $\mathbf{s}'_i \leftarrow D_{\mathbb{Z}^{n(k+\ell)}, \sigma'_{\mathsf{sk}}}$.
- Sample $\mathbf{s}''_i \leftarrow D_{\mathbb{Z}^{n(k+\ell)}, \sqrt{\Sigma'_{\mathsf{sk}} - \mathbf{c}}}$ where $\mathbf{\Sigma}'_{\mathsf{sk}}$ is defined as in Lemma 4.6.5.
- Define $\mathbf{s}_i := \mathbf{s}'_i + \mathbf{s}''_i$.

During signing queries, we replace the sampling of $\mathbf{r}_i \leftarrow \mathsf{SimSecret}(\pi_i)$ by:

- Sample $\mathbf{r}'_i \leftarrow D_{\mathbb{Z}^{n(k+\ell)}, \sigma'_{\mathbf{r}}}$.
- Sample $\mathbf{r}''_i \leftarrow D_{\mathbb{Z}^{n(k+\ell)}, \sqrt{\Sigma'_{\mathbf{r}} - \mathbf{c}}}$ where $\mathbf{\Sigma}'_{\mathbf{r}}$ is defined as in Lemma 4.6.5.
- Define $\mathbf{r}_i := \mathbf{r}'_i + \mathbf{r}''_i$.

This change is statistically close to the previous hybrid by Lemma 4.6.5, as our assumptions on σ'_{sk} and $\sigma'_{\mathbf{r}}$ ensure that the conditions of the lemma are met. Indeed, the matrix $\mathbf{\Sigma}^{-1}_{\mathsf{sk}} = \sigma^{-2}_{\mathsf{sk}}\mathbf{I} + \frac{\mathbf{R}^\top\mathbf{R}}{\sigma^2_{\mathbf{y}}}$ used during key generation verifies $\left\|\mathbf{\Sigma}^{-1}_{\mathsf{sk}}\right\|_2 \leqslant \frac{1}{2\sigma'^2_{\mathsf{sk}}} = \sigma^{-2}_{\mathsf{sk}} + \frac{B^2_{\mathbf{R}}}{\sigma^2_{\mathbf{y}}}$ by our assumption $\sigma'_{\mathsf{sk}} \geqslant \sqrt{2}\eta'_\varepsilon(\mathbb{Z}^{n(k+\ell)})$, and similarly for $\mathbf{\Sigma}_{\mathbf{r}}$ used during signing.

Thus,

$$|\varepsilon_9 - \varepsilon_8| \leqslant (Q_s + 1) \cdot N \cdot 8\varepsilon$$

where ε is the negligible parameter from the assumptions on σ'_{sk} and $\sigma'_{\mathbf{r}}$.

Hybrid_{10}. In this hybrid, we aggregate the secrets $\mathbf{s}'_i$ and noises $\mathbf{r}'_i$ of $2t$ honest users during key generation and signing queries.

Specifically, when round 3 of key generation is executed, we define a set HS' of $2t$ honest users and sample directly $\mathbf{s}' := \sum_{i\in\mathsf{HS}'} \mathbf{s}'_i$ from $D_{\mathbb{Z}^{n(k+\ell)},\sqrt{2t}\cdot\sigma'_{\mathsf{sk}}}$, and we define in LeakyKeygen $\mathbf{s}_{\mathsf{HS}} = \sum_{i\in\mathsf{HS}} \mathbf{s}''_i + \sum_{i\in\mathsf{HS}\setminus\mathsf{HS}'} \mathbf{s}'_i + \mathbf{s}'$.

During signing queries, when round 3 of signing is executed, we define a set HS'_i of $2t$ honest users during the signing session and sample directly $\mathbf{r}' := \sum_{i\in\mathsf{HS}'_i} \mathbf{r}'_i$ from $D_{\mathbb{Z}^{n(k+\ell)},\sqrt{2t}\cdot\sigma'_{\mathbf{r}}}$, and we define in $\mathsf{ShareSign}_3$ $\mathbf{r}_{\mathsf{HS}} = \sum_{i\in\mathsf{HS}} \mathbf{r}''_i + \sum_{i\in\mathsf{HS}\setminus\mathsf{HS}'_i} \mathbf{r}'_i + \mathbf{r}'$.

By the convolution lemma 4.6.4, the resulting distribution of secrets and noises remains statistically close to the previous hybrid, as our assumptions on σ'_{sk} and $\sigma'_{\mathbf{r}}$ ensure that the conditions of the lemma are met, i.e. $\sigma'_{\mathsf{sk}} \geqslant \eta'_\varepsilon(\mathbb{Z}^{n(k+\ell)})$ and $\sigma'_{\mathbf{r}} \geqslant \eta'_\varepsilon(\mathbb{Z}^{n(k+\ell)})$.

Thus,

$$|\varepsilon_{10} - \varepsilon_9| \leqslant (Q_s + 1) \cdot N \cdot 8\varepsilon$$

where ε is the negligible parameter from the assumptions on σ'_{sk} and $\sigma'_{\mathbf{r}}$.

Hybrid_{11}. In this hybrid, we choose the signing challenge c in $\mathsf{ShareSign}_3$ uniformly at random from the set of possible challenges, instead of computing it as $c \leftarrow \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \lfloor\mathbf{w}\rceil_{\nu_{\mathbf{w}}})$. We then program the random oracle H_c to return this value when queried on $(\mathsf{vk}, \mathsf{msg}, \lfloor\mathbf{w}\rceil_{\nu_{\mathbf{w}}})$, and assert that the same challenge is used in round 4.

This change is statistically close to the previous hybrid due to the min-entropy of $\lfloor\mathbf{w}\rceil_{\nu_{\mathbf{w}}}$, ensured by Lemma 4.6.7 under our assumptions on $\sigma'_{\mathbf{r}}$ and $\nu_{\mathbf{w}}$.

Indeed, for each signing query, $\mathbf{w}$ is computed as $\mathbf{w} = \left\lfloor \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{r}_{\mathsf{HS}} + \mathbf{c}' \right\rceil_\nu$ for some $\mathbf{c}'$, where $\mathbf{r}_{\mathsf{HS}}$ contains a sample from $D_{\mathbb{Z}^{n(k+\ell)},\sqrt{|\mathsf{HS}|}\cdot\sigma'_{\mathbf{r}}}$ independent of $\mathbf{c}'$. And we are guaranteed that this $\mathbf{w}$ will be used in round 4 as we have $T \geqslant 2t+1$ honest users using consistent shares of the same $\mathbf{r}$ during signing. As $\sqrt{|\mathsf{HS}|}\cdot\sigma'_{\mathbf{r}} \geqslant \sqrt{2t}\cdot\sigma'_{\mathbf{r}} \geqslant 2n \cdot q^{\frac{1}{k+\ell}+\frac{2}{n\ell}}$, we can apply Lemma 4.6.7 to obtain that $\lfloor\mathbf{w}\rceil_{\nu_{\mathbf{w}}}$ has min-entropy at least $n-1$ with probability at least $1-2^{-n+1}$ over the choice of $\mathbf{A}$. This guarantees that the programming of H_c succeeds with probability at least $1-(Q_{\mathsf{H}_c}+Q_s)2^{-n+2}$ for each signing query.

We conclude that

$$|\varepsilon_{11} - \varepsilon_{10}| \leqslant Q_s(Q_{\mathsf{H}_c} + Q_s) \cdot 2^{-n+2}$$

Hybrid_{12}. In this hybrid, in round 3, we compute the aggregated signature $\mathbf{z} = c\cdot\mathbf{s} + \mathbf{r}$ first, and then recover $\mathbf{w} = \left\lfloor \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{z} - c\cdot\mathbf{b} \right\rceil_\nu$ from it.

Recall that $\mathbf{b}$ is provided as auxiliary value by the LeakyKeygen during key generation.

This is perfectly indistinguishable from the previous hybrid, as the distribution of $\mathbf{w}$ remains unchanged. Thus,

$$\varepsilon_{12} = \varepsilon_{11}$$

Hybrid_{13}. In this hybrid, we guess in advance which output of $\mathsf{H}_{\mathbf{A}}$ will be used as matrix $\mathbf{A}$ in the verification key vk, and replace it by a uniformly random matrix sampled from $R_q^{k\times\ell}$.

The chance of guessing correctly is $\frac{1}{Q_{\mathsf{H}_{\mathbf{A}}}}$. Thus, we have:

$$\varepsilon_{13} \geqslant \frac{\varepsilon_{12}}{Q_{\mathsf{H}_{\mathbf{A}}}}$$

Hybrid_{14}. In this hybrid, we replace the verification key vk by the rounding of a uniformly random vector $\mathbf{b} \leftarrow R_q^k$.

Observe that in Hybrid_{12}, the public key is computed as $\mathsf{vk} = \lfloor\mathbf{A}\cdot\mathbf{s}' + \mathbf{c}\rceil_\nu$ where $\mathbf{s}'$ contains a sample from $D_{\mathbb{Z}^{n(k+\ell)},\sqrt{|\mathsf{HS}|}\cdot\sigma'_{\mathsf{sk}}}$ independent of $\mathbf{c}$. Furthermore, the only other place where $\mathbf{s}'$ is used is in the computation of $\mathbf{z} = c\cdot\mathbf{s} + \mathbf{r}$ during signing queries, where $\mathbf{s}$ contains $\mathbf{s}'$. We can however see $\mathbf{z}$ as containing a hint $c\cdot\mathbf{s}' + \mathbf{r}'$ on $\mathbf{s}'$.

Thus, there exists an adversary $\mathcal{B}_5$ against the Hint-MLWE problem with Q_s hints of the form $c \cdot \mathbf{s}' + \mathbf{r}'$, such that:

$$|\varepsilon_{14} - \varepsilon_{13}| \leqslant \mathsf{Adv}^{\mathsf{Hint\text{-}MLWE}}_{\mathcal{B}_5}(1^\lambda)$$

CONCLUSION. We finally show how we can build an efficient adversary $\mathcal{B}_6$ against the problem $\mathsf{SelfTargetMSIS}_{R_q,k,\ell+1,\beta_{\mathsf{stmsis}}}$ from any adversary $\mathcal{A}$ against Hybrid_{14}. The proof is analogous to the final step of the unforgeability proof of [DKMM+24, Lemma C.4], with adjustments to account for the different rounding (c.f. Remark 3.6.1). Let $\mathcal{B}_6$ be given an instance of the SelfTargetMSIS problem, i.e., a matrix $\mathbf{M} = [-\mathbf{b} \mid \mathbf{A}] \in R_q^{k\times(\ell+1)}$ where $\mathbf{b}$ is its first column and $\mathbf{A}$ the remaining ℓ columns. $\mathcal{B}_6$ sets the verification key $\mathsf{vk} = (\mathbf{A}, \mathbf{b}_\top = \lfloor \mathbf{b} \rceil_{\nu_\mathbf{b}})$ and simulates the signing challenger for $\mathcal{A}$. Whenever the random oracle H_c is invoked on an input $(\mathsf{vk}, \mathsf{msg}, \mathbf{y})$, $\mathcal{B}_6$ queries its own oracle $G(2^{\nu_\mathbf{w}} \cdot \mathbf{y} \bmod q, \mathsf{msg})$ and returns the result as the challenge. Since the map $\mathbf{y} \mapsto 2^{\nu_\mathbf{w}} \cdot \mathbf{y} \bmod q$ is injective over $\mathbb{Z}_q$, the random oracle is perfectly simulated. Furthermore, as the challenger in Hybrid_{14} no longer uses the signing key, $\mathcal{B}_6$ can perfectly simulate the signing oracle by using G instead of H_c when programming the challenges.

Upon $\mathcal{A}$ outputting a forgery $(c^*, \mathbf{z}^*, \mathbf{h}^*)$ for a message msg^*, which must satisfy:

$$\|(\mathbf{z}^*, 2^{\nu_\mathbf{w}} \cdot \mathbf{h}^* \bmod q)\|_2 \leqslant \beta. \tag{1}$$

Let $\mathbf{y}^* = \lfloor \mathbf{A} \cdot \mathbf{z}^* - 2^{\nu_\mathbf{b}} \cdot c^* \cdot \mathbf{b}_\top \rceil_{\nu_\mathbf{w}} + \mathbf{h}^*$. Due to the simulation of the random oracle, we have $c^* = G(2^{\nu_\mathbf{w}} \cdot \mathbf{y}^* \bmod q, \mathsf{msg}^*)$.

Writing $2^{\nu_\mathbf{b}} \cdot \mathbf{b}_\top = \mathbf{b} - \mathbf{b}_\perp \pmod q$ with $\|\mathbf{b}_\perp\|_\infty \leqslant 2^{\nu_\mathbf{b}-1}$, and using the properties of rounding, we have:

$$\begin{aligned}
2^{\nu_\mathbf{w}} \cdot \mathbf{y}^* \bmod q &= 2^{\nu_\mathbf{w}} \left(\lfloor \mathbf{A} \cdot \mathbf{z}^* - c^* \cdot (\mathbf{b} - \mathbf{b}_\perp) \rceil_{\nu_\mathbf{w}} + \mathbf{h}^* \right) \pmod q \\
&= \mathbf{A} \cdot \mathbf{z}^* - c^* \cdot (\mathbf{b} - \mathbf{b}_\perp) + 2^{\nu_\mathbf{w}} \cdot \mathbf{h}^* + \boldsymbol{\delta}_\mathbf{w} \pmod q \\
&= \mathbf{M} \cdot \begin{bmatrix} c^* \\ \mathbf{z}^* \end{bmatrix} + c^* \cdot \mathbf{b}_\perp + 2^{\nu_\mathbf{w}} \cdot \mathbf{h}^* + \boldsymbol{\delta}_\mathbf{w} \pmod q \\
&= [\mathbf{M} \mid \mathbf{I}] \cdot \mathbf{z}_{\mathrm{sol}} \pmod q
\end{aligned}$$

where $\mathbf{z}_{\mathrm{sol}} = (c^*, \mathbf{z}^*, c^* \cdot \mathbf{b}_\perp + 2^{\nu_\mathbf{w}} \cdot \mathbf{h}^* + \boldsymbol{\delta}_\mathbf{w})^\top$ and $\|\boldsymbol{\delta}_\mathbf{w}\|_\infty \leqslant 2^{\nu_\mathbf{w}-1}$. Finally, since c^* is the first entry of $\mathbf{z}_{\mathrm{sol}}$, $\mathcal{B}_6$ outputs $(\mathbf{z}_{\mathrm{sol}}, \mathsf{msg}^*)$. One can check that $\|\mathbf{z}_{\mathrm{sol}}\|_2 \leqslant \beta_{\mathsf{stmsis}} = \beta + \sqrt{\omega} + (\omega \cdot 2^{\nu_\mathbf{b}-1} + 2^{\nu_\mathbf{w}-1}) \cdot \sqrt{nk}$ following the definition of β_{stmsis} in theorem 4.6.3. Thus, $(\mathbf{z}_{\mathrm{sol}}, \mathsf{msg}^*)$ is a valid solution to the SelfTargetMSIS problem.

We conclude that:

$$\varepsilon_{14} \leqslant \mathsf{Adv}^{\mathsf{SelfTargetMSIS}}_{\mathcal{B}_6}(1^\lambda)$$

We obtain the final bound by summing all the bounds between hybrids. □

4.6.3 Robustness

We now prove the robustness of RB-Raccoon in the synchronous setting. We reuse the parameter constraints from the unforgeability section, and add new ones on the bounds checked in the scheme to ensure that signatures produced are valid:

- We define B_{sk} and $B_\mathbf{r}$ the bounds guaranteed by the V3S soundness (even against malicious adversaries) respectively on the partial secrets during key generation and on the partial noises during signing.
- Assume that $\nu_\mathbf{b}, \nu_\mathbf{w}$ verify the conditions of Lemma 3.7.4.
- We assume that $\beta \geqslant \omega \cdot (T \cdot B_{\mathsf{sk}} + \tau \cdot \sqrt{N-t} \cdot \sigma_{\mathsf{sk}}) + (T \cdot B_\mathbf{r} + \tau \cdot \sqrt{T-t} \cdot \sigma_\mathbf{r}) + \sqrt{nk}(3 \cdot 2^{\nu_\mathbf{w}-1} + \omega \cdot 2^{\nu_\mathbf{b}-1})$, where τ verifies $\left(\tau \geqslant 1 + \frac{\lambda \log 2}{3d}\right)$ or alternatively $\left(\tau \geqslant 1 + \sqrt{\frac{4\lambda \log 2}{d}}\right)$.

Theorem 4.6.8. *Under the parameter constraints above,* RB-Raccoon *is* TS-RB *secure in the random oracle model.*

Formally, for any adversary $\mathcal{A}$ against the TS-RB *security of* RB-Raccoon *making at most Q_s signing queries, there exists an adversary $\mathcal{B}_1$ against the* V3S-sound *security of* V3S*, an adversary $\mathcal{B}_2$ against the* EUF-CMA *security of* SIG*, and an adversary $\mathcal{B}_3$ against the* V3S-fk *security of* V3S *such that:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{TS\text{-}RB}}_{\mathcal{A}}(\lambda) \leqslant \delta(1^\lambda) &+ \mathsf{Adv}^{\mathsf{V3S\text{-}sound}}_{\mathcal{B}_1}(1^\lambda) + \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{B}_2}(1^\lambda) \\ &+ Q_s \cdot N \cdot \mathsf{Adv}^{\mathsf{V3S\text{-}fk}}_{\mathcal{B}_3}(1^\lambda) + (1+Q_s)\cdot(N\cdot 8\varepsilon + 2^{-\lambda}) \\ &+ Q_s \cdot N \cdot t \cdot (1-p_1) + Q_s \cdot N \cdot (1-p_2)\end{aligned}$$

where p_1, p_2 are the probabilities of success of SIG *and* V3S *respectively, and $\delta(1^\lambda)$ is negligible and defined in Lemma 4.6.1.*

Proof. We first recall a useful lemma.

Lemma 4.6.9 (Lemma 4.4 in [Lyu12], extended). *For any $\tau > 1$ and any lattice $\Lambda \subseteq \mathbb{R}^d$,*

$$\mathbb{P}\left[\|\mathbf{z}\| > \tau\sigma\sqrt{d}; \mathbf{z} \leftarrow D_{\Lambda,\sigma}\right] \leqslant C^d. \tag{2}$$

where $C = \tau \cdot e^{\frac{1}{2}(1-\tau^2)} < 1$. As a corollary, if $\left(\tau \geqslant 1 + \frac{\lambda \log 2}{3d}\right)$ or if $\left(\tau \geqslant 1 + \sqrt{\frac{4\lambda \log 2}{d}}\right)$, we note that $C^d \leqslant 2^{-\lambda}$.

We proceed via a sequence of hybrids, starting from the real TS-RB game and ending at a game where the adversary cannot prevent honest users from producing valid signatures.

Hybrid_1. This hybrid corresponds to the real TS-RB game from Fig. 4.2.

Hybrid_2 TO Hybrid_3. In these hybrids, we apply the same reasoning as in Hybrid_2 to Hybrid_3 of the unforgeability proof to ensure that the contributions of corrupted parties are valid sharings, and that no complaint against honest users is raised or accepted.

We conclude that there exist adversaries $\mathcal{B}_1, \mathcal{B}_2$ against respectively the V3S-sound and EUF-CMA problems, and overwhelming probabilities $1-p_1, 1-p_2$ – corresponding to the correctness of the signature scheme and of the V3S– such that

$$\begin{aligned}|\varepsilon_4 - \varepsilon_1| \leqslant &\mathsf{Adv}^{\mathsf{V3S\text{-}sound}}_{\mathcal{B}_1}(1^\lambda) + Q_s \cdot N \cdot t \cdot (1-p_1) \\ &+ \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{B}_2}(1^\lambda) + Q_s \cdot N \cdot (1-p_2)\end{aligned}$$

Hybrid_4. In this hybrid, we replace the key generation protocol by the leaky DKG from Fig. 4.15, using the simulators SimDKG_1 and SimDKG_2 defined in the proof of the leaky DKG Lemma 4.6.1. We further add an assertion that key generation succeeds, i.e. that $\mathsf{vk} \neq \perp$ in $\mathsf{Game}^{\mathsf{dkg\text{-}leaky}}$ game.

Lemma 4.6.1 guarantees that replacing the key generation protocol by the leaky DKG introduces a distinguishing advantage at most $\delta(1^\lambda)$. Furthermore, the probability that key generation fails is bounded by $\delta(1^\lambda)$ by the same lemma. Thus,

$$|\varepsilon_4 - \varepsilon_3| \leqslant 2 \cdot \delta(1^\lambda)$$

where $\delta(1^\lambda)$ is defined in Lemma 4.6.1.

Hybrid_5. In this hybrid, we aggregate the secrets and noises of honest users during key generation and signing queries, as in Hybrid_{10} of the unforgeability proof.

By the convolution lemma 4.6.4, the resulting distribution of secrets and noises remains statistically close to the previous hybrid, as our assumptions on σ'_{sk} and $\sigma'_{\mathbf{r}}$ ensure that the conditions of the lemma are met, i.e. $\sigma_{\mathsf{sk}} \geqslant \sigma'_{\mathsf{sk}} \geqslant \eta'_\varepsilon(\mathbb{Z}^{n(k+\ell)})$ and $\sigma_{\mathbf{r}} \geqslant \sigma'_{\mathbf{r}} \geqslant \eta'_\varepsilon(\mathbb{Z}^{n(k+\ell)})$.

Thus,

$$|\varepsilon_5 - \varepsilon_4| \leqslant (Q_s + 1) \cdot N \cdot 8\varepsilon$$

where ε is the negligible parameter from the assumptions on σ'_{sk} and $\sigma'_{\mathbf{r}}$.

CONCLUSION. We now evaluate the probability of success of an adversary in Hybrid_5. At this point, the only way for the adversary to win is to prevent the signing protocol from producing a valid signature.

However, since $T \geqslant 3t + 1$, there are at least $2t + 1$ honest users during signing queries using consistent shares of the same $\mathbf{r}$ and $\mathbf{s}$ (guaranteed by the previous hybrids), so the protocol will always produce a $\mathbf{z}, \cdot = c \cdot \mathbf{s} + \mathbf{r}$ where $\mathbf{s}$ and $\mathbf{r}$ are consistent with the shares of honest parties, and $c = \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \lfloor \mathbf{w} \rceil_{\nu_{\mathbf{w}}})$ with $\mathbf{w} = \left\lfloor \begin{bmatrix} \mathbf{A} & \mathbf{I} \end{bmatrix} \cdot \mathbf{r} \right\rceil_{\nu_{\mathbf{w}}}$. It remains to check that the signature $(c, \mathbf{z}^{(1),\mathbf{h}})$ derived from these values is valid.

By the previous hybrids, we know that the secret and noise used during signing contain at most t contributions from corrupted users that are in the set of valid secrets and noises of the V3S scheme, and that the contribution of honest users is sampled as a Gaussian.

Hence, we deduce that $\mathbf{s}$ and $\mathbf{r}$ have a two norm bounded by $T \cdot B_{\mathsf{sk}} + \tau \cdot \sqrt{N - t} \cdot \sigma_{\mathsf{sk}}$ and $T \cdot B_{\mathbf{r}} + \tau \cdot \sqrt{T - t} \cdot \sigma_{\mathbf{r}}$ respectively, with overwhelming probability $1 - (1 + Q_s) \cdot 2^{-\lambda}$ by Lemma 4.6.9.

Since the challenge c has a Hamming weight bounded by ω, we have that $\|\mathbf{z}\|_2 \leqslant \omega \cdot \|\mathbf{s}\|_2 + \|\mathbf{r}\|_2$. Thus, we have that $\mathbf{z}$ satisfies $\|\mathbf{z}\|_\infty \leqslant B_{\mathbf{z}} := \omega \cdot (T \cdot B_{\mathsf{sk}} + \tau \cdot \sqrt{N - t} \cdot \sigma_{\mathsf{sk}}) + (T \cdot B_{\mathbf{r}} + \tau \cdot \sqrt{T - t} \cdot \sigma_{\mathbf{r}})$ with overwhelming probability.

Then, we apply Lemma 3.7.4 to bound $(\mathbf{z}^{(1)}, 2^{\nu_{\mathbf{w}}} \cdot \mathbf{h})$:

$$\begin{aligned} \left\| (\mathbf{z}^{(1)}, 2^{\nu_{\mathbf{w}}} \cdot \mathbf{h}) \right\|_2 &\leqslant \|\mathbf{z}\|_2 + \left\| 2^{\nu_{\mathbf{w}}} \cdot \mathbf{h} - \mathbf{z}^{(2)} \right\|_2 \\ &\leqslant B_{\mathbf{z}} + \sqrt{nk}(3 \cdot 2^{\nu_{\mathbf{w}} - 1} + \omega \cdot 2^{\nu_{\mathbf{b}} - 1}) \end{aligned}$$

This concludes the proof as we assume that β is larger than the above bound, ensuring that the signature is valid with overwhelming probability. □

4.7 PARAMETER SELECTION AND INSTANTIATION

We now move on to instantiating our constructions. The main component required is a proper distribution of matrices $\mathbf{R}$ verifying property G from Definition 4.4.5. We chose to leverage matrices $\mathbf{R} \in \{0, \pm 1\}^{256 \times 2n}$ where each coefficient of $\mathbf{R}$ is 0 with probability 1/2, and ± 1 with probability 1/4. They have strong distribution properties, and have been already successfully applied in [GHL22; Ngu22]. We denote this distribution $\chi_{\mathbf{R}}$. We will rely on Lemmas 4.7.1 to 4.7.3.

Lemma 4.7.1 (Large Norm Detection, Lemma 3.2.5 from [Ngu22]). *Fix $n, q \in \mathbb{N}$, and a bound $b \leqslant q/(41n(k + \ell))$, and let $\mathbf{s} \in [\pm q/2]^{n(k+\ell)}$, with $\|\mathbf{s}\|_2 \geqslant b$. Let $\mathbf{y} \in [\pm q/2]^{256}$. Then we have $\Pr_{\mathbf{R} \leftarrow \chi_{\mathbf{R}}} \left[\|\mathbf{R} \cdot \mathbf{s} + \mathbf{y} \bmod q\|_2 < \frac{1}{2} b \sqrt{26} \right] < 2^{-128}$.*

Lemma 4.7.2 (separation). *Fix $n, q \in \mathbb{N}$, and $(\mathbf{s}, \mathbf{y}) \neq (\mathbf{s}', \mathbf{y}') \in [\pm q/2]^{n(k+\ell)+256}$. Then we have $\Pr_{\mathbf{R} \leftarrow \chi_{\mathbf{R}}} [\mathbf{R} \cdot \mathbf{s} + \mathbf{y} = \mathbf{R} \cdot \mathbf{s}' + \mathbf{y}' \bmod q] \leqslant 2^{-256}$.*

Lemma 4.7.3 (small spectral norm). *Fix $n, q \in \mathbb{N}$ and $\mathbf{R} \in \{0, \pm 1\}^{256 \times n(k+\ell)}$. $\|\mathbf{R}^\top \mathbf{R}\|_2 \leqslant 256 \cdot n(k + \ell)$. For fixed values of n, k, ℓ, we can obtain better average bounds. For $n(k + \ell) = 4096$,*

$$\Pr_{\mathbf{R} \leftarrow \chi_{\mathbf{R}}} \left[\|\mathbf{R}^\top \mathbf{R}\|_2 < 20096 \right] \geqslant 1 - 2^{-142}$$

Proof of Lemma 4.7.2.

- Case 1: $\mathbf{s} = \mathbf{s}'$ and $\mathbf{y} \neq \mathbf{y}'$.

  Then, $(\mathbf{R}\mathbf{s}' + \mathbf{y}') - (\mathbf{R}\mathbf{s} + \mathbf{y}') = (\mathbf{y} - \mathbf{y}') \neq 0 \bmod q$.

  Equality never happens in that case.

- Case 2: $\mathbf{s} \neq \mathbf{s}'$. Note i such that $s_i \neq s'_i$.

  Then, for each line $\mathbf{r}_j$ of $\mathbf{R}$, we have

$$\begin{aligned} &\langle \mathbf{r}_j, \mathbf{s} \rangle = \langle \mathbf{r}_j, \mathbf{s}' \rangle \bmod q \\ \iff &r_{j,i}(s_i - s'_i) = \sum_{i' \neq i} r_{j,i'}(s'_{i'} - s_{i'}) \bmod q \end{aligned}$$

  The term on the left is independent of the term on the right, and since $s_i - s'_i \neq 0$, the above equality happens for at most one of $r_{j,i} \in \{0, \pm 1\}$. Hence, it happens with probability at most $1/2$.

  Thus the probability that all the coefficients $\langle \mathbf{r}_j, \mathbf{s} \rangle$ are equal to $\langle \mathbf{r}_j, \mathbf{s}' \rangle$ is at most 2^{-256}.

□

Proof of Lemma 4.7.3. We want to evaluate $\|\mathbf{R}^\top \mathbf{R}\|_2 = \|\mathbf{R}\mathbf{R}^\top\|_2$. For $j \in [256]$, let us note $\mathbf{r}_j$ the i-th row of $\mathbf{R}_i$.

We have $\mathbf{R}\mathbf{R}^\top = (\langle \mathbf{r}_j, \mathbf{r}_k \rangle)_{j,k \in [256]}$. Since $\mathbf{R}\mathbf{R}^\top$ is symmetric, its largest eigenvalue is also its spectral norm. For any $j, k \in [256]$, we have $|\langle \mathbf{r}_j, \mathbf{r}_j \rangle| \leqslant n(k + \ell)$. Therefore via Gershgorin's disc theorem, $\|\mathbf{R}\mathbf{R}^\top\|_2 \leqslant 256 \cdot n(k + \ell)$.

As noted in the theorem's statement this analysis is very coarse, and when fixing n we can perform a more refined average analysis.

One method is to bound $\sum_{j \in [256] \wedge j \neq i_0} |\langle \mathbf{r}_{i_0}, \mathbf{r}_j \rangle|$ with overwhelming probability for any i_0. We observe that when $\mathbf{r}_{i_0}$ is fixed, the worse case is for $\mathbf{r}_{i_0}$ having no zero coordinate. We can actually prove that the distribution of $\sum_{j \in [256] \wedge j \neq i_0} |\langle \mathbf{r}_{i_0}, \mathbf{r}_j \rangle|$ conditioned on the number of zeros k in $\mathbf{r}_{i_0}$ only disperses toward higher value when k increases. It is thus sufficient to overwhelmingly bound $\sum_{j \neq i_0} |\langle \mathbf{1}, \mathbf{r}_j \rangle|$.

The latter is easier to perform as it is a sum of independent variables. For $n = 2048$, a software evaluation of the distribution $\sum_{j \neq i_0} |\langle \mathbf{1}, \mathbf{r}_j \rangle|$ using the security scripts for Kyber[4] tells us that it is lower than 16000 with probability at least $1 - 2^{-142}$. It follows with Gershgorin's disc theorem that $\|\mathbf{R}_i \mathbf{R}_i^\top\|_2 \leqslant 16000 + n(k + \ell) = 20096$ with overwhelming probability. □

4.7.1 Robust Encoding of Signatures

So far RB-Raccoon has been described with mathematical objects, and we have not discussed how to encode signatures and public keys in a compact way. While public keys can simply be encoded as a byte array for the seed component, and a value in $[0, \lceil q/2^{\nu_b} \rceil]^{nk}$ for the $\mathbf{b}_\top$ component, it is less clear how to encode signatures while leveraging the smaller size of $(\mathbf{z}^{(1)}, \mathbf{h})$ compared to the modulus q.

Raccoon [dEKM+23] reused the encoding proposed in Falcon [PFHK+22] to leverage the Gaussian distribution of $\mathbf{z}^{(1)}$ and encode it with a Huffman-style code: they encode each coefficient of $\mathbf{z}^{(1)}$ in two parts, the ν least significant bits are encoding using ν bits, while the remaining most significant bits are encoded using a unary code, e.g. the value 0 is encoded as 0, the value 1 is encoded as 10, etc. The sign of each coefficient is encoded on a separate bit. This encoding produces signatures with size close to the Shannon entropy of the distribution of $\mathbf{z}^{(1)}$. Formally, for a coefficient z_i of $\mathbf{z}^{(1)}$ such that $|z_i| = 2^\nu h + l$ with $l \in [0, 2^\nu)$, we encode z_i as the concatenation of the unary encoding of h (denoted $\mathsf{EncodeUnary}(h)$), the ν-bit encoding of l (denoted

4 https://github.com/pq-crystals/security-estimates

$\mathsf{EncodeBits}_\nu(l)$), and a bit for the sign of z_i. We show in Fig. 4.17 a visual representation of this encoding.

For the hint $\mathbf{h}$, we simply encode each coefficient with the unary encoding of its absolute value, and a bit for its sign. This is sufficient to achieve a compact encoding of $\mathbf{h}$ as well since it is a rounded value of $\mathbf{w}$ and thus has a small magnitude.

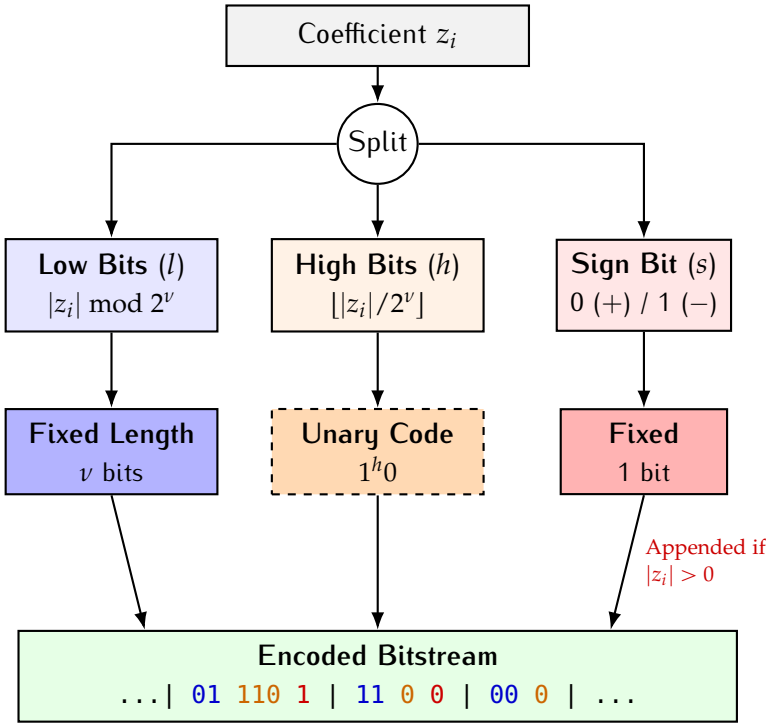


Figure 4.17: Falcon-style encoding visualization. The ν low bits are encoded first, followed by the high bits encoded in unary. The sign bit is only appended if the combined value is non-zero (i.e., at least one low or high bit is set).

We leverage the same encoding in this work, however since we want to achieve robustness, we need to tackle the case where $(\mathbf{z}^{(1)}, \mathbf{h})$ is not distributed according to the expected distribution, e.g. due to the contributions of malicious signers. To this end, we upper-bound the length of the encoding of $(\mathbf{z}^{(1)}, \mathbf{h})$ in relation to their proven two norms.

Lemma 4.7.4. *Let $\mathbf{z}^{(1)} \in R_q^\ell$ be such that $\left\|\mathbf{z}^{(1)}\right\| \leqslant \beta$, then the encoding of $\mathbf{z}^{(1)}$ with the Huffman-style code with cutoff ν described above has size at most $n\ell \cdot (\nu + 2) + \sqrt{n\ell}\beta/2^\nu$ bits.*

Proof. First, there will be at most $n\ell \cdot (\nu + 1)$ bits for the ν least significant bits and the sign of each coefficient.

Second, we can see that the length of the unary encoding of h is of length exactly $h + 1 \leqslant |z_i|/2^\nu + 1$. Thus, the total length of the unary encoding of all coefficients of $\mathbf{z}^{(1)}$ is at most $\sum_{i=1}^{n\ell}(|z_i^{(1)}|/2^\nu + 1) \leqslant n\ell + \sqrt{n\ell}\beta/2^\nu$ by the Cauchy-Schwarz inequality. □

Lemma 4.7.5. *Let $\mathbf{z}^{(1)} \in R_q^\ell$ and $\mathbf{z}^{(2)} \in R_q^k$ such that $\left\|\mathbf{z}^{(2)}\right\| \leqslant \beta$. Let $\mathbf{h} = \mathbf{w} - \mathbf{u}$, where we define $\mathbf{w} = \left\lfloor \mathbf{A} \cdot \mathbf{z}^{(1)} + \mathbf{z}^{(2)} - c \cdot \mathbf{b} \right\rceil_{\nu_\mathbf{w}}$ and $\mathbf{u} = \left\lfloor \mathbf{A} \cdot \mathbf{z}^{(1)} - 2^{\nu_\mathbf{b}} \cdot c \cdot \mathbf{b}_\top \right\rceil_{\nu_\mathbf{w}}$, then the unary encoding and the sign of $\mathbf{h}$ has size at most $nk \cdot (1 + 3/2) + nk \cdot (\beta + \omega \cdot \sqrt{nk} 2^{\nu_\mathbf{b} - 1})/2^{\nu_\mathbf{w}}$ bits.*

Proof. By Lemma 3.7.3, we have $\|\mathbf{h}\| = \|\mathbf{w} - \mathbf{u}\| \leqslant \|\delta\|/2^{\nu_\mathbf{w}} + \sqrt{nk} \cdot 3/2$, where $\delta = \mathbf{z}^{(2)} - c \cdot \mathbf{b} + 2^{\nu_\mathbf{b}} \cdot c \cdot \mathbf{b}_\top$.

By the triangle inequality and Lemma 3.7.1, we deduce $\|\delta\| \leqslant \beta + \omega \cdot \sqrt{nk} 2^{\nu_\mathbf{b} - 1}$.

With the same reasoning as in Lemma 4.7.4, we can upper bound the size of the encoding of $\mathbf{h}$ by upper bounding its norm one. It is thus bounded by:

$$nk \cdot (1 + 3/2) + \sqrt{nk} \cdot (\beta + \omega \cdot \sqrt{nk} 2^{\nu_b - 1})/2^{\nu_w}$$

□

Remark 4.7.6. The above formulas do not leverage the potential absence of the sign bit, which is only appended if $|z_i| > 0$. In practice, this can lead to a significant reduction of the signature size, especially when ν is large and reduces most of the coefficients to zero.

4.7.2 Parameter Selection for RB-Raccoon

Parameter selection for RB-Raccoon is quite involved since we need to take into account the leakage from the V3S, as well as the slack in the norm bound it proves. Note that a parameter set defined for a threshold T also supports any $1 \leqslant T' \leqslant T$ with the same security and verification procedure. We thus limit our analysis to an upper bound T on the size of the signing set supported. It is achieved by increasing σ_{sk} and $\sigma_{\mathbf{p}}$ by a factor $\sqrt{T/T'}$.

Recall we require that the number of corrupted users t verifies $3t + 1 \leqslant T$ (c.f. Section 4.6.2). For simplicity, we require that $3t + 1 = T$, as this maximizes the number of corrupted users tolerated for a given T.

ROBUSTNESS. Let us first consider the robustness constraints. As per Section 4.6.3, we need to ensure that $\beta \geqslant \omega \cdot (T \cdot B_{\mathsf{sk}} + \tau \cdot \sqrt{N - t} \cdot \sigma_{\mathsf{sk}}) + (T \cdot B_{\mathbf{r}} + \tau \cdot \sqrt{T - t} \cdot \sigma_{\mathbf{r}}) + \sqrt{nk}(3 \cdot 2^{\nu_w - 1} + \omega \cdot 2^{\nu_b - 1})$, where B_{sk} and $B_{\mathbf{r}}$ are the bounds guaranteed by the V3S soundness. We wish to evaluate B_{sk} and $B_{\mathbf{r}}$.

Lemma 4.7.1 tells us that $\frac{\sqrt{26}}{2}\|\mathbf{r}_i\| \leqslant \|\mathbf{v}_i\|$ with overwhelming probability for $\mathbf{v}_i = \mathbf{R}_i \cdot \mathbf{r}_i + \mathbf{y}_i$ the value revealed in the V3S. On the other hand, [Ngu22, Lemma 3.2.4] tells us:

$$\|\mathbf{v}_i\| \leqslant \|\mathbf{R}_i \cdot \mathbf{r}_i + \mathbf{y}_i\| \leqslant \sqrt{337} \cdot \|\mathbf{r}_i\| + \|\mathbf{y}_i\|, \tag{3}$$

where τ is a value such that $\mathbb{P}_{\mathbf{x} \leftarrow \mathcal{D}^{256}_{\sigma_{\mathbf{r}}}}\left[\|\mathbf{x}\| \geqslant \tau\sigma_{\mathbf{r}}\sqrt{256}\right] \leqslant 2^{-128}$, here $\tau \leqslant 1.4$. If we set $\sigma_{\mathbf{y}} = \sqrt{337} \cdot \sigma_{\mathbf{r}}$ and note $\mathsf{Slack}_{\mathsf{V3S}} = \tau\sqrt{337}\frac{2}{\sqrt{26}} \approx 10$, our V3S proves:

$$\|\mathbf{r}_i\| \leqslant B_{\mathbf{r}} := \mathsf{Slack}_{\mathsf{V3S}} \cdot (\sigma_{\mathbf{r}}\sqrt{(k + \ell)\, n}) \tag{4}$$

There is a gap $\mathsf{Slack}_{\mathsf{V3S}}$ between the expected norm of $\mathbf{r}_i$, which is $\sigma_{\mathbf{r}}\sqrt{(k + \ell)\, n}$, and the norm that is actually proven, which is the one in Eq. (4). While $\mathsf{Slack}_{\mathsf{V3S}}$ is constant due to the lemmas we use, increasing the security parameter λ will also increase $\mathsf{Slack}_{\mathsf{V3S}}$. In order to have robustness, we need to use the latter norm in our verification procedure.

Similarly for the secret keys, we set:

$$\|\mathbf{s}_i\| \leqslant B_{\mathsf{sk}} := \mathsf{Slack}_{\mathsf{V3S}} \cdot (\sigma_{\mathsf{sk}}\sqrt{(k + \ell)\, n}) \tag{5}$$

UNFORGEABILITY. Now, for the unforgeability to hold, we need to take into account the leakage from the V3S on the perturbations $\mathbf{r}_i$ and secrets $\mathbf{s}_i$. Recall as per the discussion in Section 4.6.2, the unforgeability of RB-Raccoon relies on the hardness of the $\mathsf{Hint\text{-}MLWE}_{R_q,k,\ell,Q_s,\sqrt{2t}\sigma'_{\mathsf{sk}},\sqrt{2t}\sigma'_{\mathbf{r}},\mathcal{C}}$ and $\mathsf{SelfTargetMSIS}_{R_q,k,\ell+1,\beta_{\mathsf{stmsis}}}$ problems, where σ'_{sk} and $\sigma'_{\mathbf{r}}$ are the usable parts of the secrets and perturbations after leakage from the V3S.

We have $\sigma'^{-2}_{\mathsf{sk}} := 2 \cdot \left(\sigma^{-2}_{\mathsf{sk}} + \frac{B^2_{\mathbf{R}}}{\sigma^2_{\mathbf{y}}}\right)$ and $\sigma'^{-2}_{\mathbf{r}} := 2 \cdot \left(\sigma^{-2}_{\mathbf{r}} + \frac{B^2_{\mathbf{R}}}{\sigma^2_{\mathbf{y}}}\right)$ for $B^2_{\mathbf{R}}$ the spectral norm of $\mathbf{R}_i^\top \mathbf{R}_i$.

According to Lemma 4.7.3, for $n(k+\ell) = 4096$, we have $\|\mathbf{R}_i^\top \mathbf{R}_i\|_2 < 20096$ with overwhelming probability. Combining this with the above definition of $\sigma'_{\mathbf{r}}$ and the fact that $\sigma_{\mathbf{y}} = \sqrt{337} \cdot \sigma_{\mathbf{r}}$, the usable part of perturbations $\mathbf{r}_i$ for $n(k+\ell) = 4096$ has a standard deviation $\sigma'_{\mathbf{r}} = \Theta(\sigma_{\mathbf{r}})$ since:

$$\sigma'_{\mathbf{r}} := \left(\sigma_{\mathbf{r}}^{-2} + \sigma_{\mathbf{y}}^{-2} \cdot 20096\right)^{-1/2} \approx \sqrt{\frac{337}{20096}} \cdot \sigma_{\mathbf{r}} \approx \frac{1}{\sqrt{60}} \cdot \sigma_{\mathbf{r}} \tag{6}$$

Similarly, we have $\sigma'_{\mathsf{sk}} \approx \frac{1}{\sqrt{60}} \cdot \sigma_{\mathsf{sk}}$.

4.7.3 Selected Parameter Sets

We rely on the lattice estimator [APS15] – an open-source tool available at `https://github.com/malb/lattice-estimator` – for estimating the concrete hardness of Hint-MLWE, relying on the reduction from Hint-MLWE to MLWE from Theorem 3.2.12, and for estimating the concrete hardness of SelfTargetMSIS.

As explained in Section 3.2.3, we evaluate the hardness of SelfTargetMSIS based on the best known attacks, which are either to break the second preimage resistance of the underlying hash function, or to solve an instance of the (inhomogeneous) MSIS problem.

We select parameters so that both problems offer at least 128 bits of security.

Remark 4.7.7. We choose $q_{\mathsf{V3S}} > q$ for use in each V3S such that $b \leqslant q_{\mathsf{V3S}}/(41n(k+\ell))$, with b the bound on the norms of secrets and perturbations used in the V3S.

We provide signature sizes using the robust encoding described in Section 4.7.1, selecting the ν parameter to minimize the signature size. We express the communication cost per party in round 1 as a function of T. It increases linearly with T as our V3S broadcasts an entire sharing. The total communication cost of our protocol hence evolves quadratically with T.

Level	Q_s	q	n	k	ℓ	σ_{sk}	$\sigma_{\mathbf{r}}$	$\nu_{\mathbf{b}}$	$\nu_{\mathbf{w}}$	$\lvert\mathsf{vk}\rvert$	$\lvert\mathsf{sig}\rvert$	Hint-MLWE C/Q	SelfTargetMSIS C/Q	comm. per party rnd_1	rnd_3	rnd_4
I	2^{59}	2^{49}	256	8	9	2^{14}	2^{29}	31	39	4.6	14.8	121/106	117/103	$34T$	12.5	15.1

Table 4.1: Parameter sets for RB-Raccoon up to $T = 1024$ parties. We select $\omega = 19$ as in the original Raccoon scheme [dEKM+23]. All sizes are in kilobytes (kB). We indicate the core-SVP hardness of Hint-MLWE and SelfTargetMSIS, a metric which ignores several polynomial factors. We also give approximate communication cost per participant in each round of key generation and threshold signature, excluding authentication of broadcast communication. Note that round 4 happens only for signature generation.

5 SHORT SECRET SHARINGS AND EFFICIENT IDENTIFIABLE ABORTS

While *robustness* is a property of interest, it can be costly to achieve. It requires an honest majority of signing parties, which weakens the security guarantees of the scheme. The construction we presented in the previous chapter to match the most compact lattice-based threshold signatures also requires a synchronous setting with broadcast even for unforgeability, and incurs communication linear in T for each signing party, which can be prohibitive when T is large.

In some applications, rather than achieving robustness, it can be acceptable to abort the signing process when some parties misbehave, and follow such an event with a procedure that identifies the misbehaving parties, which can for instance then be excluded from future signing sessions. This leads to the notion of *identifiable aborts* (IA) in threshold signature schemes, which we study in this chapter.

We introduce simple yet efficient lattice-based threshold signatures with identifiable aborts, secure under the MLWE assumption. Central to our construction are novel Distributed Key Generation with Short Shares (sDKG) protocols over lattices, ensuring short shares, small reconstruction coefficients, and linear evaluation of honest shares. This uniquely realizes the "threshold designer's dream": *signature shares double as valid signatures under the corresponding secret key shares*. With two concrete instantiations (replicated and Vandermonde secret sharings), our schemes match TRaccoon (del Pino et al., EUROCRYPT 2024)'s compact ∼10kB signature size.

5.1 INTRODUCTION

While the baseline security notion for threshold signature schemes is unforgeability, practical applications often require stronger guarantees. The approach studied in Chapter 4 is *robustness*: the protocol must output a valid signature even in the presence of malicious parties, at the cost of requiring an honest majority and significant communication overhead. An alternative is *identifiable aborts (IA)*: when the protocol fails, it identifies the misbehaving parties, allowing their exclusion from future runs. IA provides a strong deterrent against misbehavior and allows the system to recover over time. In practice, schemes achieving IA are typically more efficient than robust schemes, and they do not require an honest majority.

As surveyed in Section 3.5, classical threshold schemes like FROST [KG20] or Sparkle [CKM23] (for Schnorr) naturally support identifiable aborts due to their algebraic structure. In these settings, partial signatures are essentially valid signatures for the corresponding key shares. However, achieving this property in the lattice setting is non-trivial: for security reasons, partial signatures are typically uniformly distributed in a large space, while valid signatures must be short, preventing individual verification. TRaccoon [DKMM+24] (Fig. 3.8) relies on fresh zero-shares masking for each signing, making it impossible to verify the validity of partial signatures without proving the correctness of the masking, which is a complex task. [CTZ24] replaces the zero-shares masking with a linear secret sharing scheme with small reconstruction coefficients, although at the cost of larger parameters. However, since the shares are still uniformly distributed, partial signatures remain large and one cannot simply verify their validity with the usual signature verification algorithm.

The line of work exploiting (fully) homomorphic encryption [ASY22; GKS24; GHKS+25] in the lattice setting suggests achieving IA by adding relatively straightforward Non-Interactive Zero-

Knowledge (NIZKs) proofs to prove the well-formedness of encrypted values, but this approach remains quite inefficient for real-world applications.

More recently, in an independent work [PKNR+25], we proposed a mechanism for identifying aborts in the TRaccoon signature scheme using zero-knowledge proofs, while avoiding proving correct PRF evaluations for the zero-share masking. Although more efficient than the homomorphic encryption approach, it still relies on complex cryptographic machinery and incurs significant overhead. [PKNR+25] preserves the original TRaccoon design and introduces a deferred abort identification relying on cryptographic zero-knowledge proofs rather than simple *non-interactive* partial signature verification. That construction may scale to larger thresholds than the one presented in this chapter, but it comes at the cost of an additional interactive protocol, increased communication (about $60 + 6T$ additional kB per party) and complex implementation. Additionally, it seems hard to distribute its key generation due to the need for commitments on secret shares, while this is built into the solution presented in this thesis.

5.1.1 Our Contributions

In this chapter, we address the challenge of constructing *efficient* threshold signatures with identifiable aborts, notably without resorting to Non-Interactive Zero-Knowledge proofs (NIZKs). We propose an elegantly simple solution, consisting in this design rationale:

"Signature shares are valid signatures for their corresponding key shares".

This inherent property makes honesty verification essentially free by simply *validating individual shares against the respective public key share*. It is satisfied by the classical Schnorr-based threshold signatures Sparkle [CKM23] and FROST [KG20]; however, it was not clear how to achieve this property in the lattice setting, where valid signatures must be short whereas the shares and thus partial signatures are typically uniformly distributed over the full space. To formalize this, our framework requires a secret sharing/key generation with three critical properties:

1. The output must consist of *short shares*.
2. The reconstruction coefficients must be *small*.
3. It must enable *functional evaluation* over the honest shares.

In particular, this provides a general framework for generating secret sharing schemes with short shares, a critical feature for applications such as lattice trapdoor sampling and advanced properties like Identifiable Aborts (IA), where the shortness property is paramount. Our scheme notably provides more comprehensive properties than approaches based on linear secret sharing with small reconstruction coefficients alone (e.g., [CTZ24]), which achieve only small reconstruction coefficients. We achieve *both* short shares and small reconstruction coefficients. As such, we showcase this construction in the context of *lattice-based threshold signature schemes (TSS) with identifiable aborts (IA)*, where the shortness of shares in turn enables the generation of short partial signatures, which are valid signatures for the corresponding public key shares. Crucially, our methodology circumvents the need for cumbersome cryptographic machinery such as general multiparty computation (MPC), NIZKs, or homomorphic encryption (HE). To our knowledge, this work introduces *the most efficient lattice-based TSS with IA*. We instantiate this framework in two ways:

1. The first construction uses *replicated* secret sharing and demonstrates efficiency for small values of N and T (typically, $N \leqslant 16$). It has the advantage of being conceptually simple, and will serve as a stepping stone to understand our second construction. Additionally, we

will see in Chapter 6 that this construction appears as the most fit for rejection-sampling based TSS.

2. The second construction, based on *Vandermonde* sharing, can scale to larger values of N and T (e.g., $N \leqslant 64$) while maintaining short shares and small reconstruction coefficients. It produces shares with larger norms than the replicated scheme for small N and T, but this is offset by the noise flooding technique we employ for signature generation, which employs noise much larger than the share size.

5.2 TECHNICAL OVERVIEW

5.2.1 From Noise-Flooding to Identifiable Aborts

Noise-flooding-based threshold signatures, exemplified by TRaccoon [DKMM+24] and our robust construction in Chapter 4, use Shamir's secret sharing to support T-out-of-N thresholds with practical efficiency. However, both suffer from fundamental limitations for identifiable aborts: TRaccoon uses Shamir sharing combined with zero-shares to randomize partial signatures for security, making them effectively uniformly distributed and preventing individual verification. Our robust construction in Chapter 4 removes the zero-shares to enable robust signing, but still relies on Shamir sharing, which results in large Lagrange reconstruction coefficients that prevent individual verification of partial signatures, and relies on a Verifiable Secret Sharing in the *super-majority* setting ($T \geqslant 3t + 1$) to detect misbehavior.

In this chapter, we overcome both limitations by introducing an alternative approach to build threshold signatures from the Raccoon signature scheme that enables identifiable aborts. The key idea is to use alternative secret sharing mechanisms where both shares and reconstruction coefficients remain small, allowing us to verify partial signatures under individual key shares and immediately detect misbehavior. As illustrated in Fig. 5.1, the choice of secret sharing mechanism for both the secret key and the signing noise determines which properties can be achieved.

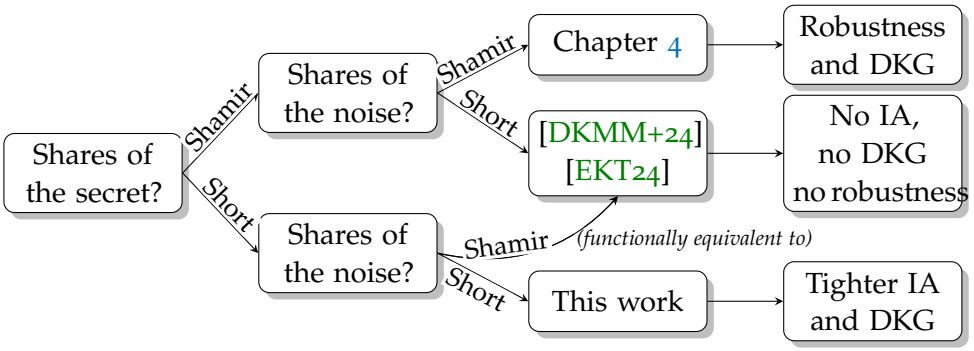


Figure 5.1: Concurrent T-out-of-N threshold Fiat-Shamir signatures on lattices classified by whether their secret and noise are shared using either short secret sharing or Shamir secret sharing. The choice of secret sharing for the secret key and the noise has a significant impact on the resulting properties.

5.2.2 Warm-up: N-out-of-N TSS with Additive Sharing

To build intuition, we first consider the simplest case: an N-out-of-N threshold scheme using additive secret sharing. Individual key pairs $(\mathbf{s}_i, \mathbf{b}_i)$ are generated for each party $i \in [N]$, where $\mathbf{s}_i$ is sampled from a discrete Gaussian. These are aggregated to form the global keys: $\mathbf{s} = \sum_{i\in[N]} \mathbf{s}_i$ and $\mathbf{b} = \left\lfloor \sum_{i\in[N]} \mathbf{b}_i \right\rceil_{\nu_\mathbf{b}}$.

To sign a message msg, parties generate individual commitments $\mathbf{w}_i = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{r}_i$ in parallel, aggregate them into $\mathbf{w} = \left\lfloor \sum_{i\in[N]} \mathbf{w}_i \right\rceil_{\nu_\mathbf{w}}$, derive a common challenge $c = \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$, and compute individual responses $\mathbf{z}_i = c \cdot \mathbf{s}_i + \mathbf{r}_i$ where $\mathbf{r}_i$ is some local signing randomness. By linearity, the final signature is $(c, \mathbf{z} = \sum_{i\in[N]} \mathbf{z}_i, \mathbf{h})$.

The key observation enabling IA is that each partial signature $\mathbf{z}_i$ can be verified as a valid signature under the individual keypair $(\mathbf{s}_i, \mathbf{b}_i)$. Misbehavior is immediately detectable by running the usual signature verification on these partial contributions. This is possible because both the shares $\mathbf{s}_i$ and reconstruction coefficients (all equal to 1) are small, ensuring $\mathbf{z}_i$ remains small, and transitively the aggregated signature $\mathbf{z}$ is also small.

While efficient, this construction is limited to the N-out-of-N setting. The challenge is to extend it to T-out-of-N while preserving the IA property—which requires maintaining shortness of both shares and reconstruction coefficients.

5.2.3 Two T-out-of-N TSS Constructions from Short Secret-Sharing

With Replicated Secret Sharing.

A first way to circumvent this issue is through (additive) T-out-of-N *replicated* secret sharing. Introduced in [ISN87], multiple threshold schemes have used replicated secret sharing due to its purely additive structure. To our knowledge, the threshold decryption scheme of [BD10] is the only previous scheme that used replicated secret sharing with small shares; it is, however, worth noting that additional difficulties arise when considering signatures rather than decryption since adding small shares can reveal secret information. Each user i knows a tuple of short secrets $(\mathbf{s}_1^i, \ldots, \mathbf{s}_n^i)$ such that: (i) any set S of T users can reconstruct the secret $\mathbf{s}$ as a subset-sum of the secrets $(\mathbf{s}_j^i)_{j\in S, i\in[n]}$, and (ii) no set of less than T users can reconstruct $\mathbf{s}$. Given such a secret sharing and a function $\mathcal{F}$ that maps sets of T users to the appropriate subset of partial secrets (i.e., $\mathcal{F}(S, i) = \{i_1, \ldots, i_k\}$ corresponds to the indices that party $i \in S$ adds to the subset sum to recover $\mathbf{s}$), one can bootstrap our previous construction into a T-out-of-N TSS by replacing $\mathbf{s}_i$ with $\mathbf{s}_i = \sum_{j\in\mathcal{F}(S,i)} \mathbf{s}_j^i$.

Such a suitable secret sharing can be constructed with $\binom{N}{T-1}$ secrets in total, and this number is optimal. While it may seem prohibitive, our construction is very efficient for small parameters, such as $N = 16$. Indeed, while the *computational* complexity of the signature is linear in the number of secrets, the public key is unchanged, and the secret keys can be compressed using seeds and trees. Perhaps counterintuitively, adding up to $\binom{N}{T-1}$ secrets to constructing the partial verification key $\mathbf{s}_i$ has *virtually no impact* on the signature size. This is thanks to our use of the Hint-MLWE assumption allowing us to largely decorrelate the signature size from the secret key size when setting parameters.

With Vandermonde Secret Sharing.

Our second approach adapts the secret sharing from [DDB95] to produce short shares. This sharing is essentially an algorithmic interpretation of Vandermonde's identity and based on the same combinatorial decomposition which states that for $0 \leqslant b \leqslant N$:

$$\binom{N}{T} = \sum_{k=0}^{T} \binom{b}{k} \cdot \binom{N-b}{T-k} \tag{1}$$

Intuitively, this means that for any set of T signing parties among N parties, when we split the set of parties into two halves (taking $b = \lfloor N/2 \rfloor$ to minimize complexity), one half will have k signing parties for some $0 \leqslant k \leqslant T$, while the other half will have exactly $T - k$ signing parties. Thus, the core idea of this sharing is, for each possible $0 \leqslant k \leqslant T$, to share the secret x into two

sub-secrets x_0 and x_1 s.t. $x_0 + x_1 = x$, one for each half, with thresholds k and $T - k$ respectively, and recursively share these sub-secrets within each half. The base case $T = 1$ is trivial as we can duplicate the secret among all parties. The possibility to reconstruct the secret given the shares of any T users follows from the existence of an appropriate k at each step, while security is guaranteed by the fact that either x_0 or x_1 will be shared among less corrupted parties than its supported threshold (to corrupt both secrets the adversary would need to corrupt at least $k + (T - k) = T$ parties).

This sharing can be made short by sampling the sub-secrets from a discrete Gaussian distribution. Interestingly, this sharing can efficiently scale to larger values of N and T, and support up to $N = 64$ parties with reasonable parameters.

Remark 5.2.1. The preprint version of this work contained another construction based on ramp secret sharing and scaling to high thresholds. This construction has been replaced by Vandermonde secret sharing in this version, as it achieves the standard notion of security while still scaling well. We refer the interested reader to the full version of the paper [dENP25] for details.

5.2.4 Abstracting the Requirements: Short DKG with Leaks

Both solutions require a secret sharing scheme with specific properties, modelable as an enhanced *Distributed Key Generation* (DKG) protocol. To ensure the IA property and security, shares must remain *short* and reconstruction coefficients *small*. *Linear evaluation* over honest shares is crucial for functional correctness, and the secret must be protected despite statistical leaks from corrupted shares. In the interest of generality, we wish to abstract these requirements into generic DKG properties that are sufficient to build T-out-of-N TSS with IA, and can be of independent interest for other applications.

Simulation-based security notions require that the execution of the DKG protocol can be simulated from the output of a trusted dealer. However, this requires gaining access to corrupted shares in the simulation, which is costly to achieve in the dishonest majority setting. Moreover, short shares inherently leak information about the secret, and we won't be able to prove that the reconstructed secret is as secure as if it was generated by a trusted dealer. The other approach is to always consider the security of both the key generation and signing together, however this approach is inflexible and requires re-proving security for each new DKG.

Instead, we introduce a weaker security notion tailored to lattice-based constructions, capturing the security requirements on the keypair and the shortness of shares. First, the public key must be pseudo-random despite leaks from corrupted shares. Further, TSS building on *noise-flooding* signatures partially leak the secret through their partial signatures. While schemes with a trusted setup such as TRaccoon can handle this leak directly by expressing signatures as a function of the secret key and corrupted shares, this is not possible in our setting as the secret key itself and corrupted shares are not known during simulation. Instead, we introduce an extension for functional evaluation with partial leakage, leveraging the *fragmentary knowledge* property introduced in Chapter 4 for *Verifiable Short Secret Sharing* schemes. At a high level, this property captures the leakage coming from producing leaky signatures over honest shares, and ensures public key security despite small leakages. Both together, this allows to reduce the security of *noise-flooding*-based TSS to standard lattice assumptions relying on uniformity such as SelfTargetMSIS. Identifiable aborts compatibility is then obtained by providing a mechanism to detect malicious parties when the DKG fails, and when it succeeds by guaranteeing the consistency and shortness of honest shares. We coin this key generation protocol *Distributed Key Generation with Short Shares*, or sDKG (see Section 5.4).

5.3 PRELIMINARIES

We reuse the notations from Chapter 3 and additionally introduce new notations for dictionaries, specific to this chapter, and for abort identification within Threshold Signature Schemes.

5.3.1 Dictionaries

We recall that a dictionary, or associative array, is a collection of ordered (key, value) pairs with unique keys. We denote $\{:\}$ the empty dictionary, and $\mathsf{Dict}[k]$ the value of Dict corresponding to the key k. We note $\mathsf{Dict}_1 \cup \mathsf{Dict}_2$ the union of two dictionaries. When a key k is present in both Dict_1 and Dict_2 and $\mathsf{Dict}_1[k], \mathsf{Dict}_2[k]$ are either both sets or both dictionaries, the new value $(\mathsf{Dict}_1 \cup \mathsf{Dict}_2)[k]$ sets as $\mathsf{Dict}_1[k] \cup \mathsf{Dict}_2[k]$.

5.3.2 Abort Identification in Threshold Signature Schemes

We extend the syntax of threshold signatures to support abort identification (IA). We adapt the notion from [RRJS+22] – targeting partially non-interactive signatures – to the multi round setting. We add new algorithm KGIdentifyAbort, IdentifyAbort that allows to identify misbehaving users based on their contributions respectively during key generation and signing.

$\mathsf{KGIdentifyAbort}((\mathsf{pm}_r)_{r\in[R_{\mathsf{KG}}]}) \to \mathcal{P}$: takes as input contributions $(\mathsf{pm}_r)_{r\in[R_{\mathsf{KG}}]}$ from each key generation round. It outputs a set of users $\mathcal{P} \subseteq [N]$ identified as misbehaving during key generation.

$\mathsf{IdentifyAbort}(\mathsf{vk}, \mathsf{aux}, \mathsf{act}, \mathsf{msg}, (\mathsf{pm}_r)_{r\in[R_{\mathsf{sig}}]}) \to \mathcal{P}$: takes as input a verification key vk, auxiliary information aux, a signer set act, a message msg and a tuple of contributions $(\mathsf{pm}_r)_{r\in[R_{\mathsf{sig}}]}$ from each signing round. It outputs a set of users $\mathcal{P} \subseteq \mathsf{act}$ identified as misbehaving during signing.

Definition 5.3.1 (Security with Identifiable Aborts). *A Threshold Signature Scheme* TS*is* secure with identifiable aborts *if for any user index* $i^* \in [N]$ *and any PPT adversary* $\mathcal{A}$*, the advantage*

$$\mathsf{Adv}^{\mathsf{TS\text{-}IA}}_{\mathsf{TS},\mathcal{A}}(1^\lambda, N, T, i^*) = \left[\mathsf{Game}^{\mathsf{TS\text{-}IA}}_{\mathsf{TS},\mathcal{A}}(1^\lambda, N, T, i^*) = 1\right]$$

is negligible in the security parameter λ*, where the game* $\mathsf{Game}^{\mathsf{TS\text{-}IA}}_{\mathsf{TS},\mathcal{A}}(1^\lambda, N, T, i^*)$ *is defined in Fig. 5.2.*

5.4 DISTRIBUTED KEY GENERATION WITH SHORT SHARES

In this section, we introduce our notion of Distributed Key Generation with Short Shares (sDKG). This refers to an extension to the notion of Distributed Key Generation (DKG) that constrains the output of DKG to: (i) have short shares, (ii) have small reconstruction coefficients, and (iii) allow one to perform functional evaluation over the honest shares. To allow for these constrained shares, we allow a partial leakage of the secret.

5.4.1 Definition

We will use the syntax for Distributed Key Generation (DKG) as introduced in Section 3.4. In Section 3.4, we chose to define safety properties of threshold signatures together with the DKG that generates their keys, since, in the dishonest majority setting, it is hard to define standalone

security properties of DKG schemes allowing for drop-in replacement of a trusted key generation in the security games.

However, it can be cumbersome to repeat the entire proof for each DKG. It is thus of interest to define properties that are sufficient to guarantee the security of a threshold signature scheme when using a given DKG. We also wish to extend this to capture abort identification, and show that there exists a mechanism to detect misbehaving parties when the DKG produces bad shares, or inconsistent auxiliary information that would normally help the signing process identify misbehavior.

As explained in the introduction of this chapter (Section 5.2.4), we first want to capture the public key security despite leakages induced by the short shares provided to corrupted parties, and the functional evaluations over honest shares inherent to *noise-flooding*-based schemes.

We identify two requirements to capture the leakage resilience of a sDKG: (i) the public key should remain secure and appear as sampled from an ideal distribution for an adversary corrupting up to $T-1$ parties, and (ii) it is later possible to simulate the evaluation of a function over the honest shares without impacting the security of the public key. We cover this by introducing *functional simulatability*, which simulates an adversary's transcript in the sDKG, as well as evaluations over honest shares only from an honestly generated public key.

Definition 5.4.1 (sDKG **functional simulatability).** *We say that a (distributed)* sDKG *is T-simulatable for a target key distribution χ and a function $f : (\mathsf{vk}, \mathsf{aux}, (\mathsf{sk}_i)_{i\in\mathsf{HS}}, \mathsf{in}) \to V$ if there exist two functions* SimKeygen, SimF *such that any adversary $\mathcal{A}$ has advantages in the games $(\mathsf{Game}^{\text{sDKG-SIM}}_{\text{sDKG},b})_{b\in\{0,1\}}$ that are negligibly close for sets of up to $T-1$ corrupted users, where:*

- *A simulator* $\mathsf{SimKeygen} : (\mathcal{A}, \mathsf{CS}, \mathsf{vk}) \to (\mathsf{aux}, \mathsf{st}, 0 \mid 1)$ *that simulates the* sDKG *protocol execution for the adversary given a public key sampled from χ, auxiliary information, and computes bias information in* st *for the functional evaluation. It also simulates key generation aborts with the last parameter, returning* 0 *in case of abort,* 1 *otherwise. Typically, the public key is sampled from a perfect uniform distribution, while the private shares are potentially biased by the adversary.*
- $\mathsf{SimF} : (\mathsf{vk}, \mathsf{aux}, \mathsf{st}, \mathsf{in}) \mapsto V$ *simulates the evaluation of the function f with public key information, the bias information from* st *and input* in.

Note: in the random oracle model, we allow the simulator to program the random oracle.

Now, we also want to capture the identifiable aborts (IA) property of a DKG. We extend the syntax of DKG to capture linear reconstruction, and will later introduce a verification algorithm to check the consistency of the shares and auxiliary information produced by the DKG, and the shortness requirements. Assume that secret keys sk_i contain a dictionary of shares, accessed as $\mathsf{sk}_i[\mathsf{idx}]$ for some index idx. We then introduce the following reconstruction algorithm:

$\mathsf{DKG.VandRecover}(\mathsf{vk}, \mathsf{aux}, \mathsf{act}) \to \mathsf{Dict}[\mathsf{Dict}]$. This algorithm provides coefficients allowing the reconstruction of a common secret from individual shares for any given set act of at least T parties. For each party $i \in \mathsf{act}$, it provides a dictionary of coefficients indexed by the share index idx, i.e., $\nu_i = \mathsf{DKG.VandRecover}(\mathsf{vk}, \mathsf{aux}, \mathsf{act})[i]$ is a dictionary such that for each share index idx, $\nu_i[\mathsf{idx}]$ is the coefficient to apply to share $\mathsf{sk}_i[\mathsf{idx}]$.

We can then define the identifiable aborts property for a DKG with short shares (sDKG). We make it relative to a verification algorithm $\mathsf{SharesVerify}(\mathsf{vk}, \mathsf{aux}, \mathsf{sk}_{i^*}) \to \{0,1\}$ for any user index $i^* \in [N]$. We will instantiate it in the next section to check the shortness of the shares, and their consistency with the public key and auxiliary information.

Definition 5.4.2 (sDKG **Identifiable Aborts).** *We say that a* sDKG *can identify aborts relative to* SharesVerify *if there exists an efficient algorithm* $\mathsf{KGIdentifyAbort}((\mathsf{pm}_i)_{i\in[R_{\mathsf{KG}}]}) \to \mathcal{P}$ *such that any adversary $\mathcal{A}$ has negligible advantage in the game $\mathsf{Game}^{\text{sDKG-IA}}_{\text{sDKG},\mathcal{A}}$ defined in Fig. 5.4 for any user index $i^* \in [N]$.*

5.5 INSTANTIATING sDKG FOR LATTICE-BASED SCHEMES

The notion of sDKG especially targets lattice-based schemes where shortness is crucial notably for trapdoor sampling, and in which a bias or partial leakage of the private keys does not necessarily lead to a security issue. Short shares also help achieve practical schemes with more advanced properties. Specifically, we are interested in a sDKG that samples public keys perfectly from a target distribution, despite of bias and partial leakage of its (short) private keys. Functions of honest shares (with partial leakage) also appear of interest as will be illustrated in our application section, by allowing one to evaluate over honest partial private keys with leakage without compromising the perfect sampling of the public key. We start by detailing the properties required for a sDKG crafted for lattice-based schemes and then provide two possible instantiations.

5.5.1 Properties of a sDKG for Lattice-based Schemes

For lattice-based schemes, the public key typically consists in a $\mathsf{seed} \in \{0,1\}^\lambda$ used to derive a public matrix $\mathbf{A} \in R_q^{k \times \ell}$ and a polynomial $\mathbf{b}_\top = \lfloor \mathbf{b} \rceil_{\nu_\mathbf{b}} \in R_{q_{\nu_\mathbf{b}}}^k$, where $\mathbf{b} \in R_q^k$ is an (indistinguishable from) uniform vector for which the $\nu_\mathbf{b}$ lower bits are rounded. Schemes practically rely on computational assumptions such as MLWE or NTRU to sample a short trapdoor $\mathbf{s}$ and compute a vector $\mathbf{b}$, function of $\mathbf{s}$, appearing uniform in R_q^k. We focus here on the MLWE case, where $\mathbf{b} = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{s}$. Interestingly, this assumption is resilient to a number of biases:

- It is possible to reveal public hints on the secret vector $\mathbf{s}$ of the form $c \cdot \mathbf{s} + \mathbf{r}$ (when $\mathbf{s}$ and $\mathbf{r}$ are Gaussian with proper parameters) without compromising the uniformity of $\mathbf{b}$. This is captured by the Hint-MLWE assumption, recalled in Section 3.2.4.
- An adversary can introduce an offset in $\mathbf{b}$, and $\mathbf{b}$ still appears uniformly random as long as the offset is independent of $\mathbf{s}$ (conditioned the hints).

Building on this, we show that it is possible to build DKG protocols sampling short shares—potentially biased and leaking to the adversary—but for which the public key remains provably pseudorandom. For concrete usability within noise-flooded schemes, since we want to directly use generated shares within partial signatures (without the zero-share randomization technique of TRaccoon [DKMM+24], c.f. [**fig:sparkle-vs-traccoon**]), we will also require the sDKG to allow the evaluation of hints of the shares, and thus partially leaking the honest shares. Specifically, we want to instantiate sDKG verifying the below properties:

- **Functional simulatability.** i.e. its output public key $(\mathsf{seed}, \mathbf{b}_\top = \lfloor \mathbf{b} \rceil_{\nu_\mathbf{b}})$ appears as if $(\mathsf{seed}, \mathbf{b})$ are uniformly sampled from $\{0,1\}^\lambda \times R_q^k$ when up to $T-1$ parties are corrupted, and even under evaluations of hints of the honest shares. We thus select as target public key distribution $\chi := \chi_\mathsf{seed} \times \chi_\mathbf{b}$ where $\chi_\mathsf{seed} := \mathcal{U}(\{0,1\}^\lambda)$ and $\chi_\mathbf{b} := \left\lfloor \mathcal{U}(R_q^k) \right\rceil_{\nu_\mathbf{b}}$ for $\nu_\mathbf{b}$ parameterizing the public key rounding.

  We define the hint function f over the honest shares – with a noise distribution $\chi_\mathbf{r}$ over $R^{\ell+k}$, for any set act of T signers:

$f(\mathsf{vk}, \mathsf{aux}, (\mathsf{sk}_j)_{j \in \mathsf{HS}}, \mathsf{in} := (\mathsf{act}, i))$

1: $\nu := \mathsf{sDKG.VandRecover}(\mathsf{vk}, \mathsf{aux}, \mathsf{act})$
2: $c \leftarrow \mathcal{C}$ ▷ Sample a challenge c
3: $\mathbf{r}_i \leftarrow \chi_\mathbf{r}$ ▷ Sample noise in $R_q^{\ell+k}$
4: **return** $c \cdot \sum_{\mathsf{idx} \in \nu[i]} \nu[i][\mathsf{idx}] \cdot \mathsf{sk}_i[\mathsf{idx}] + \mathbf{r}_i$

- **Identifiable Aborts.** We also want a sDKG that achieves identifiable aborts (IA), i.e. honest parties simultaneously accept or abort, and the resulting public key, auxiliary information and honest shares are consistent, and shares are short.

  For our sDKG, the auxiliary information shall consist of partial public keys corresponding to all the shares: $\mathbf{b}_i[\mathsf{idx}] := \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathsf{sk}_i[\mathsf{idx}]$. The shares verification algorithm SharesVerify is parameterized by $K, \eta_1, \eta_2 \in \mathbb{N}$, corresponding respectively to the maximum number of shares per party, and bounds on the reconstruction coefficients and shares, respectively. We explicit below this verification algorithm.

  $\mathsf{SharesVerify}_{K,\eta_1,\eta_2}(\mathsf{vk} = (\mathsf{seed}, \mathbf{b}_\top), \mathsf{aux} = (\mathbf{b}_i[\mathsf{idx}])_{i\in[N],\mathsf{idx}}, \mathsf{sk}_{i*})$

  1: **for** $\mathsf{idx} \in [|S_{i*}|]$ **do**
  2: **assert**{ $\|S_{i*}[\mathsf{idx}]\| \leqslant \eta_1$ }
  3: **assert**{ $\mathbf{b}_{i*}[\mathsf{idx}] = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathsf{sk}_{i*}[\mathsf{idx}]$ }
  4: **for** $\mathsf{act} \subseteq [N]$ s.t. $|\mathsf{act}| = T$ **do**
  5: $\nu := \mathsf{sDKG.VandRecover}(\mathsf{vk}, \mathsf{aux}, \mathsf{act})$
  6: **assert**{ $|\{\mathsf{idx} \mid \nu[i][\mathsf{idx}] \neq 0\}| \leqslant K$ }
  7: **assert**{ $\max_{i\in\mathsf{act},\mathsf{idx}\in\mathsf{sk}_i} |\nu[i][\mathsf{idx}]| \leqslant \eta_2$ }
  8: **assert**{ $\mathbf{b}_\top = \left\lfloor \sum_{i\in\mathsf{act}} \sum_{\mathsf{idx}\in\mathsf{sk}_i} \nu[i][\mathsf{idx}] \cdot \mathbf{b}_i[\mathsf{idx}] \right\rceil_{\nu_\mathbf{b}}$ }

  Remark 5.5.1. We assume that SharesVerify is efficiently implementable for the DKG instantiations we consider. Even though the number of sets act to consider is $\binom{N}{T}$, in our instantiations, we can leverage the structure of the sharing to efficiently verify all these conditions without iterating over all sets act.

In both of our sDKG, we will require a Symmetric Key Encryption algorithm (SKE) to implement non-repudiable pairwise communication. We assume that pairwise keys $\mathsf{sk}_i^{\mathsf{SKE}}$ are shared prior to the protocol execution to communicate with the dealer during a Setup phase, along with a signature $\mathsf{sig}_i = \mathsf{Sign}(\mathsf{sk}, (i, \mathsf{sk}_{i,j}))$ signed by the dealer for a verified reveal of the key, similarly to what was done in Chapter 4. We again choose this approach to keep the protocols as round efficient as possible, but the alternative approach of adding an extra round to the protocols to allow parties to answer complaints by broadcasting the share of the plaintiff – as described in Section 4.4.3 – could be considered instead to avoid the issue of detecting potential misbehavior during this setup phase.

5.5.2 Replicated Secret Sharing

We propose a first sDKG construction that builds on the replicated secret sharing technique used for Multi-Party Computation (MPC). The core idea of this technique lies in distributing a pool of secrets among the parties so that (i) any set of T parties collectively possesses all the secrets, and (ii) any set of less than T parties misses at least one secret. We extend it by using a pool of *short* secrets and introduce new techniques to sample (in a distributive way) such sharing securely with identifiable aborts.

In detail, our sDKG relies on a pool of secrets S, and individual sets of secrets $S_i \subset S$, for $i \in [N]$. The sDKG secret is the sum of all the secrets in S. We can implement properties (i) and (ii) by taking a pool of $\binom{N}{N-T+1} = \binom{N}{T-1}$ secrets. Each secret is sampled from a distribution χ_{sk}, and has a related set $I \subset [N]$ of cardinality $N - T + 1$, we note this secret $\mathbf{s}_I$. Secret $\mathbf{s}_I$ is simply distributed to all parties $i \in I$.

The secrets distribution verifies (i): given a set act of T parties, then act $\cap\, I$ must be nonempty for any set I, so at least one party in act knows $\mathbf{s}_I$, and (ii): given a set CS of cardinality less than

T, there exists at least one set I such that $\mathsf{CS} \subset [N] \backslash I$, i.e. no corrupted party is in I, and thus knows $\mathbf{s}_I$.

To help the reader build an intuition, we first present Repl.TrustedKeygen which samples such a sDKG relying on a trusted party.

In order to reconstruct the sDKG secret $\mathbf{s} = \sum_I \mathbf{s}_I$, all users in the acting set act need to agree on a partition of the pool of secrets S among T parties, so that each secret is manipulated by exactly one party in act. line 1 provides such a (non-interactive) consensus mechanism.

DISTRIBUTING THE KEY GENERATION. It is possible to distribute our key generation simply by having one party in I sample $\mathbf{s}_I$ for each set I. By accompanying it with shortness checks, and consistency verifications of the partial public key reconstructions, we additionally implement identifiable aborts within our sDKG.

At a high level, our sDKG works as follows:

1. Each party i samples a seed $\mathsf{seed}_i \leftarrow \{0,1\}^\lambda$. It then sends to other parties a hash $\mathsf{cmt}_i := \mathsf{H}_{\mathsf{cmt}}(\mathsf{seed}_i)$.
2. In a second round, party i reveals seed_i.
3. Then, each party i computes $\mathsf{seed} = \mathsf{H}_{\mathsf{seed}}((\mathsf{seed}_i)_{i\in[N]})$, and a public matrix $\mathbf{A} = \mathsf{H}_{\mathbf{A}}(\mathsf{seed})$. It samples secrets $\mathbf{s}_I \leftarrow \chi_{\mathsf{sk}}$ for each set I of cardinality $N - T + 1$ such that $i = \max(I)$, and computes the partial public key $\mathbf{b}_I = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{s}_I$. It sends $\mathbf{s}_I$ over a private channel to all parties $j \in I$, and publicly a commitment $\mathsf{cmt}'_I := \mathsf{H}_{\mathsf{cmt}}(\mathbf{b}_I)$ for each $\mathbf{b}_I$.
4. Finally, all the parties reveal the $\mathbf{b}_I$ they sampled in the previous round. The parties also verify the consistency of the $\mathbf{s}_I$ they received with the hashes cmt'_I, in case of inconsistency, they complain against the party that sent it, i.e. they reveal the value they received so that the inconsistency can be publicly verified.

Key aggregation consists in choosing as public key $\mathsf{vk} := \mathbf{b}_\top := \lfloor \sum_I \mathbf{b}_I \rceil_{\nu_\mathbf{b}}$, and the auxiliary information is the set of all the partial keys $\mathsf{aux} := (\mathbf{b}_I)_I$. In case there were complaints in the last round, aggregation verifies them and in case they are valid, malicious parties are identified. We provide a formal construction in Fig. 5.6. Key aggregation and abort identification is formalized in Fig. 5.7.

SECURITY PROPERTIES. We now prove that this sDKG is functionally simulatable, and correctly identifies aborts.

Theorem 5.5.2. *Consider the* $\mathsf{Hint\text{-}MLWE}_{R_q,k,\ell,Q_f,\chi_{\mathsf{sk}},(\chi_{\mathsf{sk}})_i,\mathcal{C}^{Q_f}}$ *assumption with* Q_f *hints of the form* $c \cdot \mathbf{s} + \mathbf{r}$, *where the noises are sampled from* $\chi_{\mathbf{r}}$. *Assume that* η_1 *is an overwhelming bound on honest samples from* χ_{sk}.

Formally, there exist simulators SimKeygen, SimF, *such that for any adversary* $\mathcal{A}$ *against the functional simulatability of* Repl sDKG *(with key distribution* χ *and function* f *from Section 5.5.1) making at most* Q_f *calls to the evaluation oracle* OEval, *and* Q_H *random oracle queries, we can derive adversaries* $\mathcal{B}_1$ *against the* IND-CPA *security of the* SKE, $\mathcal{B}_2$ *against* $\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathsf{sk}}}$, *and* $\mathcal{B}_3$ *against* $\mathsf{Hint\text{-}MLWE}_{R_q,k,\ell,Q_f,\chi_{\mathsf{sk}},(\chi_{\mathbf{r}})_i,\mathcal{C}^{Q_f}}$ *running in time* $\forall i, \mathsf{Time}(\mathcal{B}_i) \approx \mathsf{Time}(\mathcal{A})$, *we have:*

$$|\mathsf{Adv}^{\mathsf{Game}_1^{\mathsf{sDKG\text{-}SIM}}}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Game}_0^{\mathsf{sDKG\text{-}SIM}}}_{\mathcal{A}}| \leqslant \frac{N+2}{2^\lambda} + \frac{Q_H}{q^{nk}} + \frac{2(Q_H+N)Q_H}{2^{2\lambda}} + \mathsf{negl}(\lambda)$$
$$+ N^2 \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{B}_1} + \mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}_2} + \mathsf{Adv}^{\mathsf{Hint\text{-}MLWE}}_{\mathcal{B}_3}$$

The intuition of this proof lies in the fact that replicated secret sharing ensures that at least one of the secrets $\mathbf{s}_I$ remains hidden from the adversary. The only leakage on this safe secret is through the evaluation oracle of the hint function f. This safe secret ensures by relying on MLWE that the public key vk is properly uniformly sampled.

Proof of Theorem 5.5.2. We prove the functional simulatability of our Repl sDKG with a series of hybrid games, starting from the case $b = 0$ (real key distribution) of the $\mathsf{Game}^{\mathsf{sDKG\text{-}SIM}}$ game in the random oracle model, and finishing with the case $b = 1$ of the $\mathsf{Game}^{\mathsf{sDKG\text{-}SIM}}$ game (simulated key distribution). Let $\mathcal{A}$ be an adversary against the $\mathsf{Game}_{\mathsf{sDKG\text{-}SIM}}$ security game.

Hybrid_1. This is the original case $b = 0$ of the $\mathsf{Game}^{\mathsf{sDKG\text{-}SIM}}$ game from Fig. 5.3, where key distribution is executed using real algorithms $(\mathsf{ShareKeygen}_i)_i$.

Hybrid_2. In this hybrid, we replace the hash commitment on seed_i in the first round of honest parties by random values $\mathsf{cmt}_i \xleftarrow{\$} \{0,1\}^{2\lambda}$. The random oracle is later programmed in round 2 to remain consistent. We also program $\mathsf{H}_{\mathsf{seed}}$ and $\mathsf{H}_{\mathbf{A}}$ to ensure true random sampling of $\mathsf{seed} \leftarrow \chi_{\mathsf{seed}}$ and of the public matrix $\mathbf{A} \xleftarrow{\$} R_q^{k\times\ell}$. This is formalized in Fig. 5.8.

The view of the adversary differs in several cases:

- If $\mathcal{A}$ queries the random oracle on one of $\mathsf{H}_{\mathsf{cmt}}(\mathsf{seed}_i)$, $\mathsf{H}_{\mathsf{seed}}((\mathsf{seed}_i)_i)$, or $\mathsf{H}_{\mathbf{A}}(\mathsf{seed})$ before their programming. This happens with probability at most:
$$N \cdot 2^{-\lambda} + 2^{-\lambda} + 2^{-\lambda} = (N+2) \cdot 2^{-\lambda}$$
- If the adversary finds a pre-image for one of its commitment $(\mathsf{cmt}_i)_{i\in\mathsf{CS}}$ after round 2 is performed. This happens with probability at most $N \cdot Q_{\mathsf{H}} \cdot 2^{-2\lambda}$.
- If the adversary calls round 3 with a different sets of $(\mathsf{cmt}_i)_{i\in\mathsf{CS}}$ than what was used for programming in round 2, i.e. if $\mathcal{A}$ finds a collision in $\mathsf{H}_{\mathsf{cmt}}$. This happens with probability bounded by $Q_{\mathsf{H}}^2 \cdot 2^{-2\lambda}$.

Thus, the advantage difference with Hybrid_1 is:

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_1} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_2}\right| \leqslant (N+2) \cdot 2^{-\lambda} + (Q_{\mathsf{H}} + N)Q_{\mathsf{H}} \cdot 2^{-2\lambda}$$

Hybrid_3. In this hybrid, we ensure that no complaints are raised by honest parties against honest parties. In particular, it ensures that the private keys of SKE remain hidden. This is formalized in Fig. 5.9.

The only case where an honest party may complain about another is when a share is larger than the bound η_1. However, this bound is chosen so that this event happens with negligible probability.

Thus,

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_2} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_3}\right| = \mathsf{negl}(\lambda)$$

Hybrid_4. In this hybrid, we replace SKE encryptions to other honest parties, with encryptions of 0, ensuring that honest shares do not leak on transit to the adversary. This is formalized in Fig. 5.10.

As we made sure no decryptions are performed using the private key of the channel between $i, j \in \mathsf{HS}$, and this private key is only used for the encryption of the shares $\mathbf{s}_I$, we can reduce the advantage loss of Hybrid_4 to the IND-CPA security of the SKE. We need to replace up to N^2 encryptions, adding a factor loss N^2.

Formally, there exists an adversary $\mathcal{B}_1$ against the IND-CPA security of the SKE such that:

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_3} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Hybrid}_4}\right| \leqslant N^2 \cdot \mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{IND\text{-}CPA}}$$

Hybrid_5. In this hybrid, we fix an index I_0 such that $I_0 \subseteq \mathsf{HS}$, i.e. the secret $\mathbf{s}_{I_0}$ is not shared with the adversary. We defer the sampling of $\mathbf{b}_{I_0}$ to the fourth round by programming the random oracle to tackle rushing adversaries. This is formalized in Fig. 5.11.

First, such an index I_0 exists by the *threshold security* property of replicated secret sharing, as we assume that at most $T-1$ parties are corrupted.

Then, similarly to Hybrid_2, several bad events may happen that make the view of $\mathcal{A}$ differ:

- If $\mathcal{A}$ queries $\mathsf{H}_{\mathsf{cmt}}$ on $\mathbf{b}_{I_0}$ before it is programmed in round 4. Consider two probabilities: (i) p_1 the probability that this event occurs when $\mathbf{b}_{I_0} = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{s}_{I_0}$, and (ii) p_2 that it occurs if we had instead sampled $\mathbf{b}_{I_0}$ at random in R_q^k. We can design an adversary $\mathcal{B}_2$ against the problem $\mathsf{MLWE}_{R_q,k,\ell,\sigma_{\mathbf{s}}}$ that aborts when round 4 starts (before $\mathbf{s}_{I_0}$ is used anywhere), and tries to distinguish the two above scenarios. Then $|p_1 - p_2| \leqslant \mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}_2}$.

  Additionally, we have $p_2 \leqslant Q_H \cdot q^{-nk}$. Hence the advantage loss for this case is $\mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}_2} + Q_H \cdot q^{-nk}$.

- If $\mathcal{A}$ breaks the pre-image or collision resistance of the function $\mathsf{H}_{\mathsf{cmt}}$, this happens with probability $Q_H(Q_H+N)\cdot 2^{-2\lambda}$.

We deduce that the advantage difference is,

$$\left|\mathsf{Adv}^{\mathsf{Hybrid}_4}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Hybrid}_5}_{\mathcal{A}}\right| \leqslant \mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}_2} + Q_H \cdot q^{-k} + Q_H(Q_H+N)\cdot 2^{-2\lambda}$$

Hybrid_6. In this hybrid, we rewrite the function f so that it can be evaluated only from hints on $\mathbf{s}_{I_0}$ and from $(\mathbf{s}_I)_{I\neq I_0}$. This is formalized in Fig. 5.12.

This expression is equivalent and there is no advantage difference.

$$\mathsf{Adv}^{\mathsf{Hybrid}_5}_{\mathcal{A}} = \mathsf{Adv}^{\mathsf{Hybrid}_6}_{\mathcal{A}}$$

Hybrid_7. In this final hybrid, we replace the public key vector $\mathbf{b}_\top$ by a sample from the rounded distribution $\chi_{\mathbf{b}}$. We also sample $\mathbf{b} \xleftarrow{\$} R_q^k$ conditioned on $\mathbf{b}_\top = \lfloor \mathbf{b} \rceil_{\nu_{\mathbf{b}}}$, and express $\mathbf{b}_{I_0}$ as a function of $\mathbf{b}$ and $(\mathbf{b}_I)_{I\neq I_0}$. This is formalized in Fig. 5.13.

Observe that when $\mathbf{b}_\top$ is sampled from $\chi_{\mathbf{b}}$, then $\mathbf{b}$ is uniformly sampled from R_q^k. This implies that $\mathbf{b}_{I_0}$ is uniformly distributed in R_q^k in Hybrid_7. As $\mathbf{s}_{I_0}$ is only used to compute $\mathbf{b}_{I_0}$ and for hints in OEval, we deduce that distinguishing Hybrid_7 from Hybrid_6 reduces to the problem $\mathsf{Hint\text{-}MLWE}_{R_q,k,\ell,Q_f,\chi_{\mathsf{sk}},\chi_{\mathbf{r}},\mathcal{C}^{Q_f}}$, where we assume up to Q_f hints on the secret $\mathbf{s}_{I_0} \leftarrow \chi_{\mathsf{sk}}$ of the form $c\cdot\mathbf{s}_{I_0} + \mathbf{r}$, for $\mathbf{r} \leftarrow \chi_{\mathbf{r}}$.

Formally, there exists an adversary $\mathcal{B}_3$ against the $\mathsf{Hint\text{-}MLWE}_{R_q,k,\ell,Q_f,\chi_{\mathsf{sk}},\chi_{\mathbf{r}},\mathcal{C}^{Q_f}}$ problem such that:

$$\left|\mathsf{Adv}^{\mathsf{Hybrid}_6}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Hybrid}_7}_{\mathcal{A}}\right| \leqslant \mathsf{Adv}^{\mathsf{Hint\text{-}MLWE}}_{\mathcal{B}_3}$$

We observe that this last hybrid corresponds to the case $b=1$ of $\mathsf{Game}_{\mathsf{sDKG\text{-}SIM}}$, where we define the simulators $\mathsf{SimKeygen}, \mathsf{SimF}$ based on the hybrid.

We conclude the proof of Theorem 5.5.2 by summing all the advantage differences. □

Theorem 5.5.3. *Assuming the unforgeability of* Sign, *our* Repl sDKG *can identify aborts for the verification* $\mathsf{SharesVerify}_{K,\eta_1,\eta_2}$, *with* $K = \binom{N}{T-1}$, $\eta_1 = \tau\cdot\sigma_{\mathbf{s}}\sqrt{n(\ell+k)}$, $\eta_2 = 1$ *when we have* $\chi_{\mathsf{sk}} = \mathcal{D}_{\sigma_{\mathbf{s}}}$, *and* $\left(\tau \geqslant 1 + \frac{\lambda\log 2}{3n}\right)$ *or* $\left(\tau \geqslant 1 + \sqrt{\frac{4\lambda\log 2}{n}}\right)$.

Proof. The intuition for this proof is that the honest party i^* is never marked as malicious as it sends correctly built messages to other parties, secrets $\mathbf{s}_I$ smaller than η_1 with overwhelming probability by Lemma 4.6.9, and malicious parties cannot forge a valid complaint against them

thanks to the unforgeability of Sign. In case a malicious party sends an invalid secret $\mathbf{s}_I$ to an honest party, the honest party will detect it and complain to other parties. Finally, by construction of the key aggregation algorithm, the partial public keys always reconstruct to the public key, and reconstruction coefficients are binary. □

5.5.3 Vandermonde Secret Sharing

In this section, we revisit the secret sharing scheme from Desmedt, Di Crescenzo, and Burmester [DDB95] to build an *efficient secret sharing* scheme with *short* shares and reconstruction coefficients, that efficiently scales to a large number of parties $N = 64$. The secret sharing from [DDB95] is essentially an algorithmic interpretation of Vandermonde's identity and based on the same combinatorial decomposition which states that for $0 \leqslant b \leqslant N$:

$$\binom{N}{T} = \sum_{k=0}^{T} \binom{b}{k} \cdot \binom{N-b}{T-k} \tag{2}$$

Intuitively, Eq. (2) means that for any set of T selected parties among N parties, when we split the set of parties into two halves (taking $b = \lfloor N/2 \rfloor$ in order to minimize complexity), one half will have k selected parties for some $0 \leqslant k \leqslant T$, while the other half will have exactly $T - k$ selected parties. Thus, the core idea of this sharing is, for each possible $0 \leqslant k \leqslant T$, to share the secret x into two sub-secrets x_L and x_R s.t. $x_L + x_R = x$, one for each half, with thresholds k and $T - k$ respectively, and recursively share these sub-secrets within each half. The base case $T = 1$ is trivial as we can duplicate the secret among all parties. The possibility to reconstruct the secret given the shares of any T users follows from the existence of an appropriate k at each step, while security is guaranteed by the fact that either x_L or x_R will be shared among less corrupted parties than its supported threshold (to corrupt both secrets the adversary would need to corrupt at least $k + (T - k) = T$ parties). We illustrate this sharing in Fig. 5.14 for $N = 4$ and $T = 2$.

This scheme has been recently brought back into the spotlight by Battagliola et al. [BBCM+25]. In that work, the initial purpose was achieving a multiplicative secret sharing of an element in a (non-abelian) group, thus with strong limitations on the operations allowed.

We now show how we can revisit this secret sharing to *shortly* share a secret. Our core modification is simple in hindsight and consists in sharing the secret x in two halves x_L and $x - x_L$ in the recursion step, with x_L being sampled from a *short* distribution instead of the classical uniform one. Then, we can prove recursively that the shares will preserve a short norm (see Lemma 5.5.4), and the reconstruction coefficients are in $\{0, 1\}$ (terminology from [CTZ24]). A formal description is given in Vand.Share, Vand.Recover.

The main difficulty then lies in making sure that the shares produced by this scheme remain secure despite being sampled from a short distribution, which intrinsically leaks information on the secret. At a high level, we observe that the partial secrets $x - x_L$ in the recursion can be interpreted as hints on x. We then need to carefully account for the recursive leakages on x_L as well, and we prove that the total leakage can be captured by the leakage of several hints $x + x_L$ on the original secret x.

We formalize this sharing in Figs. 5.15 and 5.16, and show how it can be used in the presence of a trusted dealer in Vand.TrustedKeygen.

This key generation can be distributed and turned into a sDKG using the same blueprint of complaints and consistency checks as for replicated secret sharing in Fig. 5.6. However, the distribution strategy must differ because of the Vandermonde scheme's *recursive structure*. In replicated sharing, the secret assignment is simple: each secret $\mathbf{s}_I$ is assigned to a single party i (the one with the highest index in I) and directly sampled and sent by it to the other parties in I. In Vandermonde, the same secret must be split *multiple times* through recursive decomposition: when Vand.Share$(\mathbf{s}, [N], T, \ldots)$ is called, it recursively splits $\mathbf{s}$ into sub-secrets, which are further

split at each level. However, no single party has access to the full secret $\mathbf{s}$ and could thus sample the shares it needs in all of those recursive sharings simultaneously. To handle this, we use a *distributive evaluation*: each party i samples a single partial secret $\mathbf{s}_i$, then it evaluates $\mathsf{Vand.Share}(\mathbf{s}_i, [N], T, \ldots)$ locally. The algorithm explicitly computes which shares belong to each party (see the Share structure in $\mathsf{ShareKeygen}_3$), and party i sends these to the respective parties. Each party j collects the shares from all N parties' Vandermonde evaluations and sums them into its secret share sk_j. Consistency checks and complaints are performed on each received share.

SECURITY PROPERTIES. We can prove that this sDKG is functionally simulatable, and correctly identifies aborts.

Let us start by proving a bound on the norm of the shares produced by this scheme, that we later define as η_1^{ind}.

Lemma 5.5.4. *Assume $\chi_{\mathsf{sk}} = D_{\mathcal{R}^{k+\ell},\sigma_{\mathsf{sk}}}$ with $\sigma_{\mathsf{sk}} \geqslant \sqrt{2} \cdot \eta'_\varepsilon(\mathbb{Z}^{n(k+\ell)})$ for $\varepsilon = \mathsf{negl}(\lambda)$, then any share produced by our Vandermonde sharing has a norm bounded with overwhelming probability by $\mu = \tau\sigma\sqrt{n(k+\ell)\cdot T}$, where τ verifies $(\tau \geqslant 1 + \frac{\lambda \log 2}{3n})$ or $(\tau \geqslant 1 + \sqrt{\frac{4\lambda \log 2}{n}})$.*

Proof of Lemma 5.5.4. We first observe that our sharing algorithm works by adding Gaussian noises sampled from χ_{sk} to the input secret value. Hence, in the end, all shares are the sum of Gaussian samples.

We show by induction on the number of parties N that the maximal number of samples that compose a share (except for the input secret) is bounded by $T-1$.

1. In the case $T = 1$, the shares are equal to the secret, and no additional sample is added.
2. In the case $T = N$, all shares are one Gaussian sample, except for the last, which is the sum of $N - 1 = T - 1$ samples added to the secret.
3. In the case $T = 2$, all shares have at most one new Gaussian sample.
4. In the general case, let k be as in line 9 of Line 1. For $k = 0$ or $k = T$ no sample is added and we conclude by induction. Otherwise, we add at most one new Gaussian sample per share before sharing recursively. We can conclude using the induction hypothesis since we have that $k, T - k < T$.

Applying Lemmas 4.6.4 and 4.6.9, we deduce that any share has a norm bounded by the value $\tau\sigma\sqrt{n(k+\ell)\cdot T}$ with overwhelming probability. □

Theorem 5.5.5. *Consider the $\mathsf{Hint\text{-}MLWE}_{R_q,k,\ell,Q_f,\chi_{\mathsf{sk}}}$ assumption with Q_v (defined in Remark 5.5.6) hints of the form $\mathbf{s} + \mathbf{r}$ with $\mathbf{r} \leftarrow \chi_{\mathsf{sk}}$, and Q_f hints of the form $c \cdot \mathbf{s} + \mathbf{r}$ with $\mathbf{r} \leftarrow \chi_{\mathbf{r}}$. Assume that η_1^{ind} is an overwhelming bound on honest shares produced by $\mathsf{Vand.Share}$.*

Formally, there exist simulators $\mathsf{SimKeygen}, \mathsf{SimF}$, such that for any adversary $\mathcal{A}$ against the functional simulatability of our Vandermonde-based sDKG (with key distribution χ_{sk} and function f from Section 5.5.1) making at most Q_f calls to the evaluation oracle OEval, and Q_H random oracle queries, we can derive adversaries $\mathcal{B}_1$ against the $\mathsf{IND\text{-}CPA}$ security of the SKE, and $\mathcal{B}_2$ against the problem $\mathsf{Hint\text{-}MLWE}_{R_q,k,\ell,Q_f,\chi_{\mathsf{sk}},(\chi_{\mathbf{r}})_i,\mathcal{C}^{Q_f}}$ running in time $\forall i, \mathsf{Time}(\mathcal{B}_i) \approx \mathsf{Time}(\mathcal{A})$, we have:

$$|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_1^{\mathsf{sDKG\text{-}SIM}}} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_0^{\mathsf{sDKG\text{-}SIM}}}| \leqslant \frac{N+2}{2^\lambda} + \frac{Q_v(N,T)\cdot Q_H}{q^{nk}} + \frac{2(Q_H+N)Q_H}{2^{2\lambda}} + N^2 \cdot \mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{IND\text{-}CPA}} + \mathsf{Adv}_{\mathcal{B}_2}^{\mathsf{Hint\text{-}MLWE}} + \mathsf{negl}(\lambda)$$

Proof. The proof follows the same structure as Theorem 5.5.2, with the main difference being in the handling of the leakage due to the short shares. We carefully analyze the recursive structure of the Vandermonde secret sharing and show that the total leakage can be bounded by the

leakage of several hints on the original secret, which is captured by the Hint-MLWE assumption. The rest of the proof proceeds similarly to that of Theorem 5.5.2, leveraging the security of the underlying SKE and Hint-MLWE assumptions.

We provide a proof overview below.

Hybrid_1. This is the original case $b = 0$ of the $\mathsf{Game}^{\mathsf{sDKG\text{-}SIM}}$ game from Fig. 5.3, where key distribution is executed using real algorithms $(\mathsf{ShareKeygen}_i)_i$.

Hybrid_2. In this hybrid, we replace the hash commitment of honest parties on seed_i in the first round by random values $\mathsf{cmt}_i \leftarrow \{0,1\}^{2\lambda}$. The difference in advantage is bounded by $(N+2) \cdot 2^{-\lambda} + (Q_\mathsf{H} + N)Q_\mathsf{H} \cdot 2^{-2\lambda}$.

Hybrid_3. In this hybrid, we ensure that no complaints are raised by honest parties against honest parties. In particular it ensures that the private keys of SKE remain hidden. This is guaranteed by the assumption that η_1^{ind} is an overwhelming bound on honest shares produced by Vand.Share. The difference in advantage is negligible.

Hybrid_4. In this hybrid, we replace SKE ciphertexts sent by honest parties with encryptions of zero. The difference in advantage is bounded by $N^2 \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{B}_1}$.

Hybrid_5. This hybrid is the crux of the proof. In this hybrid, we fix the index i^* of a honest user and we wish to capture the leakage on its secret share sk_{i^*} due to the Vandermonde secret sharing. We show that given fresh hints of the form $h_i := \mathbf{s}_{i^*} + \mathbf{r}_i$ with $\mathbf{r} \leftarrow \chi_{\mathsf{sk}}$, we can equivalently sample all the shares of $\mathbf{s}_{i^*}$ such that:

- Corrupted shares are expressed as linear combinations of the hints h_i.
- For any honest share $\mathsf{sk}_{i^*}[\mathsf{idx}]$, either $\mathsf{sk}_{i^*}[\mathsf{idx}]$ or $\mathbf{s}_{i^*} - \mathsf{sk}_{i^*}[\mathsf{idx}]$ is expressed as a linear combination of the hints h_i.

Concretely, we proceed by induction on the size N of the set of parties $\mathcal{P}$ – assuming that at most $T-1$ parties in $\mathcal{P}$ are corrupted – to show that in every recursive call to the function $\mathsf{Vand.Share}(\mathbf{x}, \mathcal{P}, T, \mathsf{idx})$, we can simulate the shares produced for party i^* using hints on $\mathbf{x}$ of the form $\mathbf{x} + \mathbf{r}$ with $\mathbf{r} \leftarrow \chi_{\mathsf{sk}}$: corrupted shares are linear functions of these hints, and for every honest share $S_i[\mathsf{idx}]$ either $S_i[\mathsf{idx}]$ or $\mathbf{x} - S_i[\mathsf{idx}]$ is a linear function of these hints.

Consider a set of parties $\mathcal{P} = \{p_1, \ldots, p_N\}$ such that $T < N$. We prove our induction hypothesis on the base case $T = 1$, $T = 2$ and $T = N$. Then we show it to be true for the general case using our induction hypothesis on $\mathcal{P}_L$ and $\mathcal{P}_R$.

CASE $T = 1$. In this case, there are no corrupted parties, and thus no leaked share. Additionally, each share is directly $\mathbf{x}$, thus $\mathbf{x} - S_i[\mathsf{idx}] = \mathbf{0}$ is known.

GENERAL CASE. Let k be one of the values taken by the for statement (Line 1, line 9) of the general case. If $k = 0$ or $k = T$, we simply conclude by the induction hypothesis applied to $\mathcal{P}_L$ or $\mathcal{P}_R$. Otherwise, either (i) $|\mathcal{P}_L \cap \mathsf{CS}| \leqslant k-1$ (i.e. $\mathbf{x}_L$ can not be recovered) or (ii) $|\mathcal{P}_R \cap \mathsf{CS}| \leqslant T-k-1$ (i.e. $\mathbf{x}_R$ can not be recovered).

First, assume (i). Then, we consider that we leak $\mathbf{x}_R$ as an hint of the form $\mathbf{x} + \mathbf{r}$, and Dict_R can be sampled directly from $\mathbf{x}_R$ and the shares all leaked. For Dict_L, we apply the induction hypothesis to the call $\mathsf{Vand.Share}(\mathbf{x}_L, \mathcal{P}_L, k, \mathsf{idx}_L)$, ensuring that leaked shares can be simulated using hints on $\mathbf{x}_L$ and that unleaked shares $S_i[\mathsf{idx}] \in \mathsf{Dict}_L$ verify $\mathbf{x}_L - S_i[\mathsf{idx}]$ is a function of leaked shares. The core observation is that we can replace $\mathbf{x}_L$ by $\mathbf{x} - \mathbf{x}_R$ in these statements, and as $\mathbf{x}_R$ is leaked it is a linear function of leaked shares, thus also $\mathbf{x} - S_i[\mathsf{idx}]$. This concludes this induction step as $\mathbf{x}_R$ is itself a hint on $\mathbf{x}$.

The case (ii) is analogous, and we instead apply the induction hypothesis to Dict_R and consider that $\mathbf{x}_L$ is leaked, allowing to directly depend on $\mathbf{x}$ observing that $\mathbf{x}_R = \mathbf{x} - \mathbf{x}_L$.

Remark 5.5.6. Note that the number of necessary hints $Q_v(T, N)$ can be computed by unfolding the above induction proof. This is done using the following recursive function: $Q_v(T, N)$:

1. If $T = 1$, return 0.
2. Otherwise, return the sum for $\max(0, T - N + b) \leqslant k \leqslant \min(b, T)$ – with $b = \lfloor N/2 \rfloor$ – of
   a) $Q_v(k, b)$ when $k = T$
   b) $Q_v(T - k, N - b)$ when $k = 0$
   c) $\max(1 + Q_v(k, b), Q_v(T - k, N - b))$ otherwise

This hybrid does not change the advantage of $\mathcal{A}$.

Hybrid_6. In this hybrid, we rewrite the output of the oracle OEval using additional hints on $\mathbf{s}_{i*}$ of the form $c \cdot \mathbf{s}_{i*} + \mathbf{r}$ with $\mathbf{r} \leftarrow \chi_{\mathbf{r}}$. We distinguish two cases:

- If the honest share involved in the evaluation was directly expressed as a linear function of the hints h_i, we do not need any additional hint.
- Otherwise, we observe that $\mathbf{s}_{i*} - S_{i*}[\mathsf{idx}]$ is expressed as a linear function of the hints h_i. Thus, we can express $c \cdot S_{i*}[\mathsf{idx}] + \mathbf{r}_i$ from a hint of the form $c \cdot \mathbf{s}_{i*} + \mathbf{r}$ with $\mathbf{r} \leftarrow \chi_{\mathbf{r}}$, to which we add a linear function of the hints h_i.

This hybrid does not change the advantage of $\mathcal{A}$.

Hybrid_7. In this hybrid, we sample a vector $\mathbf{b}_{i*}$ uniformly at random in R_q^k, and use it to compute the partial public keys of party i^* by replacing $\begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \mathbf{s}_{i*}$ by $\mathbf{b}_{i*}$ in the computations.

At this stage, all computations can be expressed as a function of $\mathbf{b}_{i*}$ and hints on $\mathbf{s}_{i*}$, and we can apply the Hint-MLWE assumption to replace $\mathbf{b}_{i*}$ by a uniform sample. Hence, there exists an adversary $\mathcal{B}_2$ against the Hint-MLWE assumption with $Q_v(T, N)$ hints of the form $\mathbf{s}_{i*} + \mathbf{r}$ with $\mathbf{r} \leftarrow \chi_{\mathsf{sk}}$, and Q_f hints of the form $c \cdot \mathbf{s}_{i*} + \mathbf{r}$ with $\mathbf{r} \leftarrow \chi_{\mathbf{r}}$, such that the difference in advantage is bounded by $\mathsf{Adv}^{\mathsf{Hint\text{-}MLWE}}_{\mathcal{B}_2}$.

Hybrid_8. In this hybrid, we sample the vector $\mathbf{b}$ corresponding to the final public key uniformly at random in R_q^k, and compute the partial public key of i^* to be consistent with $\mathbf{b}$.

This can be achieved by replacing commitments on $\mathbf{b}^{(i^*)}_{j,\mathsf{idx}}$ (when they contain $\mathbf{b}_{i*}$) by random values in round 3, and then in round 4, extracting $\mathbf{b}^{(j)}_{k,\mathsf{idx}}$ from the commitments of other parties j. Finally, we program the random oracle $\mathsf{H}_{\mathsf{cmt}}$ to ensure that $\mathbf{b}^{(j)}_{i*,\mathsf{idx}}$ is consistent with $\mathbf{b}$.

The argument is similar to that of Hybrid_5 in Theorem 5.5.2. We argue that the adversary cannot cheat on his commitments (i.e. a collision, or find a preimage after round 4). And, we can hide $\mathbf{b}_{i*}$ until after we extract all other shares $\mathbf{b}^{(j)}_{k,\mathsf{idx}}$ for $j \neq i^*$. Indeed, since $\mathbf{b}_{i*}$ is uniformly sampled, and $\mathbf{b}^{(i^*)}_{j,\mathsf{idx}}$ has the same min-entropy, we deduce that the adversary cannot guess $\mathbf{b}^{(i^*)}_{j,\mathsf{idx}}$ before we program the random oracle. Thus, the difference in advantage is negligible.

CONCLUSION. We can now conclude the proof by observing that in Hybrid_8, the view of $\mathcal{A}$ can be simulated using only the public key. We hence define $\mathsf{SimKeygen}$ and SimF to output the view of Hybrid_8.

Finally, by summing up all differences in advantage between consecutive hybrids, we obtain the bound stated in the theorem.

□

We now move to the proof of abort identification.

Theorem 5.5.7. *Assuming the unforgeability of* Sign*, our Vandermonde-based* sDKG *can identify aborts for* $\mathsf{SharesVerify}_{K,\eta_1,\eta_2}$ *with* $K = 1$, $\eta_1 = N \cdot \eta_1^{\mathsf{ind}}$ *as in Lemma 5.5.4, and* $\eta_2 = 1$.

Proof. The honest user i^* will not be identified as malicious with overwhelming probability as it always follows the protocol, and since the adversary cannot forge a signature to complain against i^* with a wrongful secret key.

Then, we can see that any final share of i^* will be the sum of at most N shares approved during the key generation and hence of norm at most η_1^{ind}. By triangular inequality, any share of i^* will hence be of norm at most $N \cdot \eta_1^{\mathsf{ind}}$.

For K and η_2, we can simply see that Vandermonde sharing always uses a single share per party for reconstruction, and with a reconstruction coefficient 1. □

5.5.4 Comparison of our Two Sharing Schemes

Replicated secret sharing has a number of shares that grows with $\binom{N}{T-1}$. On the other hand, Vandermonde secret sharing has been analyzed in [BBCM+25], where the authors provide estimates on the total and per party number of shares, showing a superpolynomial growth in T and $\log N$.

We plot in Figure 5.19 the number of shares per party for both schemes, for N up to 64 and T ranging from 1 to N. We estimate that replicated secret sharing remains practical for $N \leqslant 16$, while Vandermonde secret sharing remains practical as high as $N \leqslant 64$.

In terms of scalability to large N and T, Vandermonde secret sharing is clearly the best choice. However, replicated secret sharing is conceptually simpler, and also has the advantage for small values of N and T: for those cases, replicated secret sharing remains more efficient in terms of number of shares. Furthermore, the shares produced by replicated secret sharing have a smaller norm, as they simply consist of MLWE secrets, while Vandermonde shares contain noisy hints on a main secret, which are larger. For noise flooded schemes, the acceptable size of the shares has a certain slack, as the signing noise remains dominant. However, for schemes based on rejection sampling, minimizing this norm is paramount to limit the rejection rate, and replicated secret sharing appears like the best choice. This will be exposed in Chapter 6, where replicated secret sharing is leveraged for Threshold ML-DSA with small thresholds.

Now let us compare the Vandermonde secret sharing with other secret sharing schemes that could be used in our setting. Other sharing schemes with similar features have been proposed; however, for the range $N \leqslant 64$, the scheme from [DDB95] appears to be the most efficient. In the range of interest ($N \leqslant 64$), it outperforms:

- The Replicated Secret Sharing approach (see Figure 5.19), which performs well only for small N and T.
- The generic construction from [BL90] to Boppana's monotone formula [Bop85] (see also [DT06]), used in [CTZ24], in which the number of shares grow as $O(\min(T, N-T)^{4.3} N \log N)$.

"Near-threshold" or "ramp" secret sharing schemes are sharings for which the privacy threshold T_p is different from the reconstruction threshold T_r. For a large number of parties, these schemes can provide better efficiency than the perfect threshold schemes. Some of these schemes have the potential to be used in our setting, but, to our knowledge, none of them is fully compatible or outperforms the scheme previously proposed:

- In [dENP25], a ramp secret sharing scheme based on the Coupon Collector Problem is considered; however, the ratio T_r/T_p is quite large. For n such that $T_p = \Theta(n \log n)$, we have:

$$\frac{T_r}{T_p} \approx 1 + O\left(\frac{\lambda/m + \log(\lambda/p)}{\log n}\right). \tag{3}$$

Setting $m = \omega(\lambda \log n)$ and $p = \omega(\lambda/n)$ gives a ratio $\frac{T_r}{T_p} = 1 + o(1)$ at the cost of $m \cdot p$ shares per party. This gap remains large for concrete instantiations [dENP25, Figure 8]. In addition, in the adaptive corruption setting (as opposed to static), the privacy threshold gets downgraded to $T_p = n - 1$. This is because this scheme relies on probabilistic sharing to avoid sending too many shares to the set of corrupted parties, but an adaptive adversary can simply target a set of n parties that received shares of interest after the sharing phase.

- In [ANP23], Applebaum et al. show how to instantiate a ramp secret achieving any gap T_r/T_p arbitrarily close to 1. Moreover, the sharing and reconstruction can be performed with $O(N)$ additions. However, to our knowledge, there is no way to simulate the shares using hints from Hint-MLWE. Hence, it cannot be used in our setting (an equivalent privacy related problem with this sharing is noted in [CTZ24]).

5.6 IA-Raccoon: THRESHOLD SIGNATURES WITH IDENTIFIABLE ABORTS

In this section, we introduce a new design for TRaccoon [DKMM+24] relying on the sDKG abstraction introduced in Section 5.5 to provide IA. We coin our construction IA-Raccoon.

At a high level, during each signing session party i individually signs the message using its partial secret $\sum_{u \in [|S_i|]} \nu_{i,u} \cdot S_i[u]$. The sDKG ensures the shortness of this partial secret key which in turn ensures the shortness of partial signatures and allows one to verify honest behavior. To optimize signature size, we consider that sDKG secrets $S_i[u]$ are divided in two parts $\mathbf{s}^{(1)}[u] \in R^\ell$ and $\mathbf{s}^{(2)}[u] \in R^k$ for $i \in [N], u \in [|S_i|]$. We provide a formal description in Fig. 5.20. Verification procedures are described separately in IdentifyAbort and Verify. Parameter sets are given in Table 5.1.

Parameter	Explanation
R_q	Polynomial ring $R_q = \mathbb{Z}_q[X]/(X^n + 1)$
(k, ℓ)	Dimension of public matrix $\mathbf{A} \in R_q^{k \times \ell}$
$(\chi_{\mathsf{sk}}, \sigma_{\mathbf{s}})$	Gaussian distribution with width $\sigma_{\mathbf{s}}$ used in the sDKG for sampling the secret
$\nu_{\mathbf{b}}$	Amount of bit dropping performed on verification key in the sDKG
(K, η_1, η_2)	Shortness properties of sDKG (see Definition 5.4.2)
$(\chi_{\mathbf{r}}, \sigma_{\mathbf{r}})$	Gaussian distribution with width $\sigma_{\mathbf{r}}$ used for the commitment $\mathbf{w}$
$\nu_{\mathbf{w}}$	Amount of bit dropping performed on (aggregated) commitment
$(\mathcal{C} \subset R_q, \omega)$	Challenge set $\{c \in R_q \mid \|c\|_\infty = 1 \wedge \|c\|_1 = \omega\}$ s.t. $\|\mathcal{C}\| \geqslant 2^\lambda$
β^{ind}	Two-norm bound for partial verification $\beta^{\mathsf{ind}} = \omega \cdot K\eta_1\eta_2 + \tau\sqrt{n(\ell + k)} \cdot \sigma_{\mathbf{r}}$
β	Two-norm bound on the signature $\beta = T \cdot \beta^{\mathsf{ind}} + (2^{\nu_{\mathbf{w}}} + \omega \cdot 2^{\nu_{\mathbf{b}}-1}) \cdot \sqrt{nk}$

Table 5.1: Overview of parameters used in our TSS. We assume τ is such that $\left(\tau \geqslant 1 + \frac{\lambda \log 2}{3d}\right)$ or $\left(\tau \geqslant 1 + \sqrt{\frac{4\lambda \log 2}{d}}\right)$.

5.6.1 Unforgeability

We first prove the unforgeability of our IA-Raccoon scheme. We target the notion of *unforgeability* in the asynchronous setting (ASYNC-TS-UF) as defined in Fig. 3.5. Before proceeding to the proof, we recall a useful result from [DKMM+24].

Definition 5.6.1. *For* $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{k\times\ell}$*, and any* $\mathbf{u} \in \mathcal{R}_q^k$*, let* $\mathcal{D}^{\mathsf{bd-MLWE}}_{q,l,\ell,\sigma,\nu}(\mathbf{A},\mathbf{u})$ *be the distribution defined as* $\{\lfloor \mathbf{A}\cdot\mathbf{s}+\mathbf{e}+\mathbf{u}\rceil_\nu \mid (\mathbf{s},\mathbf{e}) \leftarrow \mathcal{D}_\sigma^\ell \times \mathcal{D}_\sigma^k\}$*. That is, it samples an* $\mathsf{MLWE}_{q,k,\ell}$ *instance shifted by* $\mathbf{u}$ *with* ν *bits dropped.*

Lemma 5.6.2. *For any* $\sigma > 2n\cdot q^{\frac{1}{\ell+k}+\frac{2}{n\ell}}$ *and* $\nu < \log(q)-2$*, we have:*

$$\Pr_{\mathbf{A}\xleftarrow{\$}\mathcal{R}^{k\times\ell}}\left[H_\infty(\mathcal{D}^{\mathsf{bd-MLWE}}_{q,l,\ell,\sigma,\nu}(\mathbf{A},\mathbf{u})) \geqslant n-1\right] \geqslant 1-2^{-n+1}$$

We refer to [DKMM+24, Lemma 3.8] for the proof of Lemma 5.6.2.

Theorem 5.6.3 (Unforgeability of IA-Raccoon). *Assume* $\sigma > 2n\cdot q^{\frac{1}{k+\ell}+\frac{2}{n\ell}}$ *and* $\nu_{\mathbf{w}} < \log(q)-2$*. Define* $\beta_{\mathsf{stmsis}} = \beta + \sqrt{\omega} + (\omega\cdot 2^{\nu_{\mathbf{b}}-1} + 2^{\nu_{\mathbf{w}}-1})\cdot\sqrt{nk}$*.*

Formally, let $\mathcal{A}$ *be an adversary against the* TS-UF *security of our threshold scheme making* Q_s *calls to signing oracles, and at most* $Q_{\mathsf{H}_{\mathsf{cmt}}}, Q_{\mathsf{H}_c}$ *queries respectively to the random oracles* $\mathsf{H}_{\mathsf{cmt}}, \mathsf{H}_c$*. There exist adversaries* $\mathcal{B}_1$ *against the* sDKG-SIM *security of our short secret sharing,* $\mathcal{B}_2$ *against the* $\mathsf{SelfTargetMSIS}_{R_q,k,\ell+1,\mathcal{C},\beta_{\mathsf{stmsis}}}$ *problem running in time* $\mathsf{Time}(\mathcal{B}_1) \approx \mathsf{Time}(\mathcal{B}_2) \approx \mathsf{Time}(\mathcal{A})$ *such that:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ASYNC\text{-}TS\text{-}UF}}_{\mathcal{A}} \leqslant \mathsf{Adv}^{\mathsf{sDKG\text{-}SIM}}_{\mathcal{B}_1} + \mathsf{Adv}^{\mathsf{SelfTargetMSIS}}_{\mathcal{B}_2} + Q_s\cdot(Q_{\mathsf{H}_{\mathsf{cmt}}}+Q_{\mathsf{H}_c})\cdot 2^{-n+2} \\ + \frac{Q_{\mathsf{H}_{\mathsf{cmt}}}\cdot T\cdot Q_s + (Q_{\mathsf{H}_{\mathsf{cmt}}}+Q_s)^2}{2^{2\lambda}}\end{aligned}$$

Proof. We proceed with a series of hybrids starting from the ASYNC-TS-UF game from Fig. 3.5. Throughout this proof, we denote the advantage of an adversary $\mathcal{A}$ against a search game G by $\mathsf{Adv}^G_{\mathcal{A}} := \Pr[G(\mathcal{A}) \to 1]$ – in particular, we omit passing λ as input.

Game_1. This is the ASYNC-TS-UF game from Fig. 3.5.

Game_2. In this hybrid we defer the computation of the honest $\mathbf{w}_i$ to the second round of the protocol, and program the random oracle to be consistent. We also introduce a flag f_{fail} to explicitly mark additional cases where the adversary loses. Notably, f_{fail} is set to $\top$ when the random oracle is programmed on a previously queried value. This is the same as Game_2 of the Threshold ML-DSA proof in Fig. 6.8.

The view of the adversary $\mathcal{A}$ differs if they called the random oracle of one of the $\mathbf{w}_i$ before round 2 was executed, i.e.

$$\left|\mathsf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}}\right| \leqslant \Pr[E]$$

where E is the event "$\mathsf{H}_{\mathsf{cmt}}$ was queried on a honest $\mathbf{w}_i$ before round 2 in Game_1".

By union-bound, we can reduce this to a single call to $\mathcal{O}_{\mathsf{ShareSign}_1}$.

$$\left|\mathsf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}}\right| \leqslant \sum_{s=1}^{Q_s}\Pr[E'_s] \tag{4}$$

where E'_s is the event "The $\mathbf{w}_i$ produced by the s-th call to $\mathcal{O}_{\mathsf{ShareSign}_1}$ is queried on $\mathsf{H}_{\mathsf{cmt}}$ before round 2 in Game_1".

Lemma 5.6.2 tells us that the min-entropy of a single $\mathbf{w}_i$ is at least $n-1$ except with probability at most 2^{-n+1}. As there at most $Q_{\mathsf{H}_{\mathsf{cmt}}}$, we deduce that $\Pr[E'_s]$ is bounded by $Q_{\mathsf{H}_{\mathsf{cmt}}}\cdot 2^{-n+2}$.

Summing the above inequality over $s \in [Q_s]$, we obtain:

$$\left|\mathsf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}}\right| \leqslant Q_s\cdot Q_{\mathsf{H}_{\mathsf{cmt}}}\cdot 2^{-n+2}$$

Remark 5.6.4. As a useful remark for the following hybrids, we note that a given $\mathbf{w}_i$ cannot be used twice by the same party after Game_2 as then the random oracle programming would fail, and the adversary loses.

Game_3. In this hybrid, we sample a challenge c in advance during round 2. When the last honest hash cmt_i obtains a corresponding $\mathbf{w}_i$ programmed, we program the values $\mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w}) = c$. This is analogous to Game_3 of Theorem 6.4.4 in Fig. 6.10.

The proof for this hybrid is analogous to the previous one. First, these changes do not bias H_c as the sampled challenges c are used at most once for programming H_c, and are not provided to the adversary before programming. The view of the adversary hence differs only if it queried the random oracle H_c on some tuple $(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$, where $\mathbf{w}$ are the high bits of $\hat{\mathbf{w}} = \sum_{i \in \mathsf{act}} \mathbf{w}_i$ before it is programmed in round 2.

Again, we can bound this probability due to the high min-entropy of the last honest $\mathbf{w}_i$ that is sampled. As it is independent of the other $\mathbf{w}_j$, $j \neq i$, we can capture their sum in the vector $\mathbf{u}$ in Lemma 5.6.2 and deduce that $\mathbf{w}$ has a high min-entropy, except with probability 2^{-n+1}. We apply the union-bound over the Q_s programmation, and Q_{H_c} calls to H_c and deduce

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_2} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_3}\right| \leqslant Q_s \cdot Q_{\mathsf{H}_c} \cdot 2^{-n+1}$$

Game_4. In this hybrid, we ensure that $\mathsf{H}_{\mathsf{cmt}}$ has no collisions and that the adversary cannot find preimages of hashes cmt_i after passing them to the second signing oracle. We introduce a table Cmt that records every cmt sampled in $\mathsf{H}_{\mathsf{cmt}}$ or in $\mathcal{O}_{\mathsf{ShareSign}_1}$, as well as those passed to the second signing oracle. Whenever a new hash cmt is sampled, we ensure that it was not previously added to Cmt, or the challenger sets the flag f_{fail} to $\top$ in order to abort the game. This is the same as Game_4 of the proof of Theorem 6.4.4 in Fig. 6.12.

The view of the adversary differs in Game_4 if it was previously able to (i) find a pre-image of a cmt_i after passing it to $\mathcal{O}_{\mathsf{ShareSign}_2}$, or (ii) if a given cmt was sampled twice. We can bound the probability of (i) due to the pre-image resistance of the hash function $\mathsf{H}_{\mathsf{cmt}}$, and with an union bound over all the commits passed by the adversary to $\mathcal{O}_{\mathsf{ShareSign}_2}$ which gives a bound $Q_{\mathsf{H}_{\mathsf{cmt}}} \cdot T \cdot Q_s \cdot 2^{-2\lambda}$.

As for (ii), we rely on the collision resistance of $\mathsf{H}_{\mathsf{cmt}}$. Since the commitments are sampled uniformly from $\{0,1\}^{2\lambda}$, we can bound the probability of (ii) by $\frac{(Q_{\mathsf{H}_{\mathsf{cmt}}}+Q_s)^2}{2^{2\lambda}}$.

We conclude that,

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_3} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_4}\right| \leqslant \frac{Q_{\mathsf{H}_{\mathsf{cmt}}} \cdot T \cdot Q_s + (Q_{\mathsf{H}_{\mathsf{cmt}}} + Q_s)^2}{2^{2\lambda}}$$

Game_5. In this hybrid, we ensure that the challenge used in round 3 is the same as the one sampled in round 2, and we precompute the responses $\mathbf{z}_i$ in advance. This is the same as Game_6 in the proof Theorem 6.4.4 except that we sample responses as hints here instead of doing rejection sampling. It was formalized in Fig. 6.14.

We want to show that responses are identically distributed in Game_5. This is the case if programmed responses use the same c in both round 2 and round 3, and if they are used at most once.

Now, we can first observe that if the hash checks in round 3 pass, then all the $\mathsf{cmt}_i := \mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, i, \mathbf{w}_i)$ are correctly defined. Recall that Game_4 ensured that $\mathsf{H}_{\mathsf{cmt}}$ has no collisions, and that pre-images cannot be found after round 2 is called. Hence, if these checks pass, then it means that in round 2, all the hashes cmt_i either: (i) already had pre-images, or (ii) were produced by an honest party in $\mathcal{O}_{\mathsf{ShareSign}_1}$ and were waiting for programming. Eventually, all the missing cmt_i from (ii) must thus have been programmed in $\mathcal{O}_{\mathsf{ShareSign}_2}$ and as the challenger programs H_c as soon as all the $\mathbf{w}_i$ are defined, then it ensures that the same c is used in round 2 and in round 3.

Additionally, responses are used at most once by a given party due to the collision resistance of $\mathsf{H}_{\mathsf{cmt}}$ and Remark 6.4.6 which ensures that party i will accept cmt_i at most once.

All in all, we conclude that the adversary's view is identically distributed in Game_4 and Game_5 so,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_5} = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_4}$$

Game_6. In this hybrid, we reverse the sampling of $\mathbf{w}_i$ and $\mathbf{z}_i$. We start by sampling $\mathbf{r}_i \leftarrow \chi_{\mathbf{r}}, c \leftarrow \mathcal{C}$ and $\mathbf{z}_i = \sum_{\mathsf{idx}\in\nu[i]} c \cdot \nu[i][\mathsf{idx}] \cdot \mathsf{sk}_i[\mathsf{idx}] + \mathbf{r}_i$. Then, we compute $\mathbf{w}_i$ as $\begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{z}_i - \sum_{\mathsf{idx}\in\nu[i]} \mathsf{aux}[i][\mathsf{idx}]$, where $\mathsf{aux}[i][\mathsf{idx}] = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathsf{sk}_i[\mathsf{idx}]$ is the partial public key constructed from $\mathsf{sk}_i[\mathsf{idx}]$. This is analogous to what is done in Game_7 in Theorem 6.4.4.

We can see that the view of the adversary is identically distributed. Indeed, in Game_6 $\begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{z}_i = c \cdot \underbrace{\begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \sum_{\mathsf{idx}\in\nu[i]} \nu[i][\mathsf{idx}] \cdot \mathsf{sk}_i[\mathsf{idx}]}_{\sum_{\mathsf{idx}\in\nu[i]} \mathsf{aux}[i][\mathsf{idx}]} + \underbrace{\begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{r}_i}_{\mathbf{w}_i}$ for $\mathbf{z}_i = \mathbf{r}_i + \sum_{\mathsf{idx}\in\nu[i]} c \cdot \nu[i][\mathsf{idx}] \cdot \mathsf{sk}_i[\mathsf{idx}]$. We can thus equivalently write $\mathbf{w}_i = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{z}_i - \sum_{\mathsf{idx}\in\nu[i]} c \cdot \nu[i][\mathsf{idx}] \cdot \mathsf{aux}[i][\mathsf{idx}]$ and sample $\mathbf{z}_i$ first. So,

$$\mathsf{Adv}^{\mathsf{Game}_6}_{\mathcal{A}} = \mathsf{Adv}^{\mathsf{Game}_5}_{\mathcal{A}}$$

Game_7. In this hybrid, we replace the public key by the rounding of a uniform value, i.e. $\lfloor \mathbf{b} \rceil_{\nu_\mathbf{b}}$ where $\mathbf{b} \xleftarrow{\$} R_q^k$, and we replace the key generation and hints used for generating partial signatures by the *functional simulatability* simulators of our Vandermonde secret sharing. This is formalized in Fig. 6.15.

Formally, there exists an adversary $\mathcal{B}_1$ against the sDKG-SIM property of our secret sharing:

$$\left|\mathsf{Adv}^{\mathsf{Game}_6}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Game}_7}_{\mathcal{A}}\right| \leqslant \mathsf{Adv}^{\mathsf{sDKG\text{-}SIM}}_{\mathcal{B}_1}$$

CONCLUSION. At this stage the verification key of the signature scheme is distributed randomly, and signatures no longer depend on a secret value.

The same reduction used for the final step of the unforgeability proof of RB-Raccoon in Theorem 4.6.3 applies here.

Formally, there exists an adversary $\mathcal{B}_2$ against the $\mathsf{SelfTargetMSIS}_{R_q,k,\ell+1,\mathcal{C},\beta_{\mathsf{stmsis}}}$ problem with execution time $\mathsf{Time}(\mathcal{B}_2) \approx \mathsf{Time}(\mathcal{A})$ such that

$$\mathsf{Adv}^{\mathsf{Game}_7}_{\mathcal{A}} \leqslant \mathsf{Adv}^{\mathsf{SelfTargetMSIS}}_{\mathcal{B}_2}$$

We obtain the theorem statement by summing the different advantage differences obtained at each step of the proof. □

5.6.2 Identifiable Aborts

We now prove the Identifiable Abort property (Theorem 5.6.5) of our TSS scheme.

Theorem 5.6.5. *Our TSS identifies abort if the* sDKG *identifies abort for* $\mathsf{SharesVerify}_{K,\eta_1,\eta_2}$*, with parameters as in Table 5.1.*

Let $\mathcal{A}$ be a PPT adversary against $\mathsf{Game}_{\mathsf{TS\text{-}ID}}$ *of Fig. 5.2. There exists an adversary $\mathcal{B}_1$ against the* sDKG-IA *security of* sDKG *with* $\mathsf{Time}(\mathcal{B}_1) \approx \mathsf{Time}(\mathcal{A})$ *such that:*

$$\mathsf{Adv}^{\mathsf{TS\text{-}ID}}_{\mathcal{A}}(1^\lambda) \leqslant \mathsf{Adv}^{\mathsf{sDKG\text{-}IA}}_{\mathcal{B}_1} + Q_s \cdot 2^{-\lambda}$$

Though it is not made explicit in IdentifyAbort (for concision), we consider that when the parsing of messages fails, the sending party is identified as malicious.

Proof of Theorem 5.6.5. In order to prove the theorem, we analyze the probability of each winning condition of the game in Fig. 5.2.

KEY GENERATION. We first tackle bad events during key generation. We can move to a game where the winning conditions Lines 10 and 12 in Fig. 5.2 always return 0 and add an assertion that when the key generation succeeds, the generation key, auxiliary information, and share sk_{i*} verify $\mathsf{SharesVerify}_{K,\eta_1,\eta_2}$.

The difference in winning probability between the two games is upper bounded by the advantage of an adversary against the sDKG-IA security of sDKG. Indeed, if the adversary $\mathcal{A}$ wins the game because of one of these two conditions or causes the key generation to output invalid shares, we can build an adversary $\mathcal{B}_1$ against the sDKG-IA security of sDKG that simulates $\mathcal{A}$ until the end of key generation. The advantage of $\mathcal{B}_1$ is exactly the difference in winning probability between the two games.

SIGNING. We now analyze the signing phase.

At this point, we can assume that the key generation phase always succeeded and that the share sk_{i*} is valid with respect to $\mathsf{SharesVerify}_{K,\eta_1,\eta_2}$.

First, we can show that i^* will not be identified as malicious by IdentifyAbort except with negligible probability. i^* behaves honestly in the session considered and the only ways for it to be identified as malicious are: (i) if $\mathbf{z}$ is not short enough, or (ii) the $\mathbf{z}^{(2)}$ recovered in IdentifyAbort is different from the one computed by i^* during round 3 of signing.

For (i), we have that $\mathbf{z}_{i*} = c \cdot \mathbf{s}_{i*}^{\mathsf{part}} + \mathbf{r}_{i*}$, where $\mathbf{s}_{i*}^{\mathsf{part}} = \sum_{\mathsf{idx} \in \nu[i^*]} \nu[i^*][\mathsf{idx}] \cdot \mathsf{sk}_{i*}[\mathsf{idx}]$. Due to the bounds enforced by $\mathsf{SharesVerify}_{K,\eta_1,\eta_2}$, we have $|\nu_{i*,u}| \leqslant \eta_2$, $\|\mathsf{sk}_{i*}\| \leqslant \eta_1$, and there are at most K secrets sk_{i*}. We thus deduce that $\left\|\mathbf{s}_{i*}^{\mathsf{part}}\right\| \leqslant K\eta_1\eta_2$. Besides, c has Hamming weight ω and $r_{i*} \leftarrow \chi_{\mathbf{r}}$, hence by Lemma 4.6.9 we have that $\|\mathbf{z}_{i*}\| \leqslant \tau\sqrt{n(\ell+k)}\sigma_{\mathbf{r}}$ with probability $1 - 2^{-\lambda}$ for each signing query. By a union bound the probability that i^* is identified as malicious because of (i) is at most $Q_s \cdot 2^{-\lambda}$.

As for (ii), $\mathsf{SharesVerify}_{K,\eta_1,\eta_2}$ ensures the consistency of the partial secret key $\mathbf{s}_{i*}^{\mathsf{part}}$ used in round 3 of signing with the partial public key $\mathbf{b}_{i*}^{\mathsf{part}}$ used in abort identification. Hence, the value $\mathbf{z}^{(2)}$ computed by i^* during round 3 of signing is exactly the same as the one recomputed by IdentifyAbort.

Now, it remains to show that when IdentifyAbort returns an empty set, the protocol produces a valid signature. First, we can observe that $\mathbf{z}^{(1)} = \sum_{i \in \mathsf{act}} \mathbf{z}_i^{(1)}$ and $\mathbf{z}^{(2)} = \sum_{i \in \mathsf{act}} \mathbf{z}_i^{(2)}$ (as computed in IdentifyAbort) verify $\mathbf{A} \cdot \mathbf{z}_i^{(1)} + \mathbf{z}_i^{(2)} = c \cdot \mathbf{b} + \sum_{i \in \mathsf{act}} \mathbf{w}_i$ because of the consistency of the partial public keys $\mathbf{b}_i^{\mathsf{part}}$ with the global public key $\mathbf{b}$ ensured by $\mathsf{SharesVerify}_{K,\eta_1,\eta_2}$. Furthermore, we can bound $\|(\mathbf{z}^{(1)}, \mathbf{z}^{(2)})\|$ with $T \cdot \beta^{\mathsf{ind}}$ with a simple triangular inequality.

We then relate this to the signature $(c, \mathbf{z}^{(1)}, \mathbf{h})$ output by Combine. Remark that by construction the correct challenge c is recomputed during the verification of the signature. We then apply Lemma 3.7.4 to deduce that $(\mathbf{z}^{(1)}, 2^{\nu_{\mathsf{w}}} \cdot \mathbf{h})$ has a small norm, specifically $\|(\mathbf{z}^{(1)}, 2^{\nu_{\mathsf{w}}} \cdot \mathbf{h})\| \leqslant \|(\mathbf{z}^{(1)}, \mathbf{z}^{(2)}) + \sqrt{nk}(3 \cdot 2^{\nu_{\mathsf{w}}-1} + \omega \cdot 2^{\nu_{\mathsf{b}}-1}) \leqslant T \cdot \beta^{\mathsf{ind}} + \sqrt{nk}(3 \cdot 2^{\nu_{\mathsf{w}}-1} + \omega \cdot 2^{\nu_{\mathsf{b}}-1}) = \beta$. This concludes the proof.

□

5.6.3 Parameter Selection

We derive parameters for our TSS with identifiable aborts for both of the sDKG we introduced. Interestingly, we can use the same parameter sets for both, the number of secrets and size of shares having a limited impact on parameters due to the unusually large randomness used for noise flooding. We obtain sizes competitive with the state-of-the-art TRaccoon [DKMM+24]. We provide concrete parameters in Table 5.2 and explain our selection strategy below.

Parameter selection.

Parameter sets are provided in Table 5.2. Parameter selection is fairly standard and detailed thereafter. For the signature size, we rely on the robust encoding described in Section 4.7.1 to upper bound the size of a signature that passed partial verification, selecting the ν parameter to minimize its size.

Table 5.2: Parameter sets for IA-Raccoon for NIST security levels I and V. They are valid for both of our sDKG, up to $N = 16$ for the replicated one, and up to $N = 64$ for the Vandermonde one. Sizes are given in kB.

Level	Q_s	q	n	k	ℓ	σ_{sk}	$\sigma_{\mathbf{r}}$	$\nu_{\mathbf{b}}$	$\nu_{\mathbf{w}}$	$\lvert\mathsf{vk}\rvert$	$\lvert\mathsf{sig}\rvert$	Hint-MLWE C/Q	SelfTargetMSIS C/Q
I	2^{59}	2^{49}	256	8	9	2^{14}	2^{31}	32	40	4.3	12.3	123/108	126/110
V	2^{60}	2^{53}	512	8	7	2^{14}	2^{37}	39	46	7.2	22.1	257/225	254/223

Unforgeability.

The unforgeability of our scheme relies on the SelfTargetMSIS assumption, and on the functional simulatability of the underlying sDKG. The latter reduces to the Hint-MLWE assumption for both of the sDKG we introduced. We study both assumptions separately.

Hint-MLWE.

Both of our sDKG are functionally simulatable if the Hint-MLWE problem is hard, for a secret $\mathbf{s} \leftarrow \mathcal{D}_{\sigma_{\mathbf{s}}}$, and up to Q_s hints of the form $c \cdot \mathbf{s} + \mathbf{r}$, plus $Q_v(N, T)$ hints of the form $\mathbf{s} + \mathbf{s}'$ for the Vandermonde sDKG, where $\mathbf{r} \leftarrow \mathcal{D}_{\sigma_{\mathbf{r}}}, \mathbf{s}' \leftarrow \mathcal{D}_{\sigma_{\mathsf{sk}}}$. By a direct computation (see Remark 5.5.6 for the formula) we can verify that for $T \leqslant N \leqslant 64$ the number of hints $Q_v(T, N)$ in Theorem 5.6.3 is bounded by $30327 \approx 2^{15}$. Following Theorem 3.2.12, this problem is at least as hard as $\mathsf{MLWE}_{R_q,k,\ell,\sigma_{\mathsf{red}}}$, for:

$$\frac{1}{\sigma_{\mathsf{red}}^2} = 2 \cdot \left(\frac{1}{\sigma_{\mathbf{s}}^2} + (1 + o(1)) \cdot \frac{Q_s \cdot \omega}{\sigma_{\mathbf{r}}^2} \right) \sim \frac{2 \cdot Q_s \cdot \omega}{\sigma_{\mathbf{r}}^2}, \tag{5}$$

where the $\sim$ asymptotic holds when $\sigma_{\mathbf{s}} \sqrt{Q_s \cdot \omega} = \omega(\sigma_{\mathbf{r}})$. We note that the factor 2 in the formula is a proof artifact of the reduction, and we ignore it in our parameter selection, as often done in the literature. We then quantify the hardness of $\mathsf{MLWE}_{R_q,k,\ell,\sigma_{\mathsf{red}}}$ using the open source Lattice Estimator [APS15].

Self-Target MSIS.

In Theorem 5.6.3, $\mathsf{Adv}^{\mathsf{SelfTargetMSIS}}_{\mathcal{B}_2}$ corresponds to the hardness of $\mathsf{SelfTargetMSIS}_{R_q,k,\ell+1,\mathcal{C},\beta_{\mathsf{stmsis}}}$. We proceed as in Section 3.2.3 and evaluate the hardness of this problem based on the best known attacks, following the footsteps of Dilithium [LDKL+22, §C.3] and Raccoon [dEKM+23, §4.3.5]. Specifically, the best known attacks boil down to either (i) breaking the second preimage resistance of the underlying hash function, or (ii) breaking the related inhomogeneous MSIS problem $\mathsf{MSIS}_{R_q,k,\ell,\beta_{\mathsf{stmsis}}}$.

Correctness and Identifiable aborts.

Correctness is mainly subsumed to the bounds provided in Theorem 5.6.5. Interestingly, due to the noise flooding technique, there exists a large gap between the secret standard deviations $\sigma_{\mathbf{s}}$ and $\sigma_{\mathbf{r}}$ to guarantee unforgeability. Looking at the Hint-MLWE reduction, we can select this gap to be as high as $\sqrt{Q_s}$ with no security loss. This ensures for both of our sDKG that the number and size of shares has a small influence on the signature size and $B_2 \approx T \cdot \tau\sqrt{n(\ell+k)} + (2^{\nu_{\mathbf{w}}} + \omega \cdot 2^{\nu_{\mathbf{b}}-1}) \cdot \sqrt{nk}$.

$\mathsf{Game}^{\mathsf{TS\text{-}IA}}_{\mathsf{TS},\mathcal{A}}(1^\lambda, N, T, i^*)$

1 : $\mathsf{sid} \leftarrow 0$
2 : $\mathsf{SignStates} \leftarrow \varnothing$
3 : $(\mathsf{vk}, \mathsf{aux}, (\mathsf{sk}_i)_{i\in[N]}) \leftarrow \mathsf{Keygen}(1^\lambda, N, T, t)$
4 : $(\mathsf{st}_i)_{i\in[N]} \leftarrow \mathsf{Setup}(1^\lambda, N, T, t)$
5 : $\mathsf{st}_{\mathcal{A}} \leftarrow \mathcal{A}((\mathsf{st}_i)_{i\in[N]}, \mathsf{st}_{\mathcal{A}})$
6 : **for** $r \in [R_{\mathsf{KG}}]$ **do**
7 : $\quad \mathsf{st}_{i*}, \mathsf{pm}_r^{i^*} \leftarrow \mathsf{ShareKeygen}(i^*, \mathsf{st}_{i*}, \mathsf{pm}_{r-1})$
8 : $\quad ((\mathsf{pm}_r^i)_{i\in[N]\setminus\{i^*\}}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathsf{H}}(\mathsf{pm}_r^{i^*}, \mathsf{st}_{\mathcal{A}})$
9 : $U := \mathsf{KGIdentifyAbort}((\mathsf{pm}_r)_{r\in[R_{\mathsf{KG}}]})$
10 : **if** $U \neq \varnothing$ **then return** $i^* \notin U$
11 : $(\mathsf{vk}, \mathsf{aux}) := \mathsf{CombineKey}((\mathsf{pm}_r)_{r\in[R_{\mathsf{KG}}]})$
12 : **if** $(\mathsf{vk}, \mathsf{aux}) = \perp$ **then return** 1
13 : $\mathsf{sk}_{i*} := \mathsf{PartialSecret}(\mathsf{st}_i)$
14 : $O := \mathsf{H}, (\mathcal{O}_{\mathsf{TS.Sign}_r})_{r\in[R_{\mathsf{sig}}]}$
15 : $(\mathsf{sid}, (\mathsf{pm}^i_{R_{\mathsf{sig}}})_{i\in\mathsf{act}'}) \leftarrow \mathcal{A}(\mathsf{vk}, \mathsf{aux}, \mathsf{sk}_{i*})$
16 : $(r, \mathsf{act}, \mathsf{msg}, (\mathsf{pm}_r)_{r\in[r-1]} \cup \{\mathsf{pm}_r^{i^*}\}) \leftarrow \mathsf{SignStates}[\mathsf{sid}]$
17 : **require** $r = R_{\mathsf{sig}} \wedge \mathsf{act} = \mathsf{act}' \sqcup \{i^*\}$
18 : $U := \mathsf{IdentifyAbort}(\mathsf{vk}, \mathsf{aux}, \mathsf{act}, \mathsf{msg}, (\mathsf{pm}_r)_{r\in[R_{\mathsf{sig}}]})$
19 : **if** $U \neq \varnothing$ **then return** $i^* \notin U$
20 : $\mathsf{sig} \leftarrow \mathsf{TS.Combine}(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, (\mathsf{pm}_r)_{r\in[R_{\mathsf{sig}}]})$
21 : **return** $\neg\mathsf{TS.Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$

$\mathcal{O}_{\mathsf{TS.Sign}_1}(\mathsf{act}, \mathsf{msg})$

1 : **require** $\mathsf{act} \subset [N] \wedge |\mathsf{act}| = T \wedge i^* \in \mathsf{act}$
2 : $\mathsf{sid} \leftarrow \mathsf{sid} + 1$
3 : $(\mathsf{pm}_1^{i^*}, \mathsf{st}_{i*}) \leftarrow \mathsf{TS.Sign}_1(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, i^*, \mathsf{sk}_{i*}, \mathsf{st}_{i*}, \varnothing)$
4 : $\mathsf{SignStates}[\mathsf{sid}] \leftarrow \left(1, \mathsf{act}, \mathsf{msg}, \mathsf{pm}_1^{i^*}\right)$
5 : **return** $\mathsf{pm}_1^{i^*}$

$\mathcal{O}_{\mathsf{TS.Sign}_r}(\mathsf{sid}, (\mathsf{pm}^j_{r-1})_{j\in\mathsf{act}\setminus\{i^*\}})$ for $r \in [2, R_{\mathsf{sig}}]$

1 : **require** $\mathsf{SignStates}[\mathsf{sid}] \neq \perp$
2 : $(r', \mathsf{act}, \mathsf{msg}, (\mathsf{pm}_m^{i^*})_{m\in[r-1]}) \leftarrow \mathsf{SignStates}[\mathsf{sid}]$
3 : **require** $r' = r$
4 : $(\mathsf{st}_{i*}, \mathsf{sig}_{i*}) \leftarrow \mathsf{TS.Sign}_r(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, i^*, \mathsf{sk}_{i*}, \mathsf{st}_{i*}, \mathsf{pm}_{r-1})$
5 : $\mathsf{SignStates}[\mathsf{sid}] \leftarrow \left(r, \mathsf{act}, \mathsf{msg}, (\mathsf{pm}_m)_{m\in[r-1]} \cup \{\mathsf{pm}_r^{i^*}\}\right)$
6 : **return** sig_{i*}

Figure 5.2: Abort identification game for a threshold signature scheme. Dashed boxes correspond to the centralized key generation setting, and solid boxes to the distributed key generation. By convention, $\mathsf{pm}_0 = \perp$. The adversary $\mathcal{A}$ wins if the game returns 1, i. e. if it successfully frames the honest user i^* or produces an invalid signature without any misbehaving user in the signing set.

$\mathsf{Game}^{\mathsf{sDKG\text{-}SIM}}_{\mathsf{sDKG},b}(1^\lambda, N, T)$ for $b \in \{0,1\}$

$(\mathsf{CS}, \mathsf{st}_\mathcal{A}) \leftarrow \mathcal{A}(1^\lambda)$ // $\mathcal{A}$ chooses corrupted parties
$b_\perp \xleftarrow{\$} \{0,1\}$
if $\neg\,(\mathsf{CS} \subseteq [N] \wedge |\mathsf{CS}| \leqslant T-1)$ **then return** $b_\perp$
$\mathsf{HS} := [N] \backslash \mathsf{CS}$
if $b = 0$ **then** // Real sharing
  $(\mathsf{st}_i)_{i\in[N]} \leftarrow \mathsf{Setup}(1^\lambda, N, T, t)$
  for $r \in [R_{\mathsf{KG}}]$ **do**
    for $i \in \mathsf{HS}$ **do**
      $\mathsf{pm}_r^i, \mathsf{st}_i \leftarrow \mathsf{ShareKeygen}(i, \mathsf{st}_i, \mathsf{pm}_{r-1})$
      if $\mathsf{pm}_r^i = \perp$ **then return** $b_\perp$
    $((\mathsf{pm}_r^i)_{i\in\mathsf{CS}}, \mathsf{st}_\mathcal{A}) \leftarrow \mathcal{A}^{\mathsf{H}}((\mathsf{pm}_r^i)_{i\in\mathsf{HS}}, \mathsf{st}_\mathcal{A})$
  $(\mathsf{vk}, \mathsf{aux}) := \mathsf{CombineKey}((\mathsf{pm}_r)_{r\in[R_{\mathsf{KG}}]})$
  if $(\mathsf{vk}, \mathsf{aux}) = \perp$ **then return** $b_\perp$
  for $i \in \mathsf{HS}$ **do** $\mathsf{sk}_i := \mathsf{PartialSecret}(\mathsf{st}_i)$
else // Simulated sharing
  $\mathsf{vk} \leftarrow \chi$
  $\mathsf{aux}, \mathsf{st}, \mathsf{status} \leftarrow \mathsf{SimKeygen}(\mathcal{A}, \mathsf{CS}, \mathsf{vk})$
  if $\mathsf{status} = 0$ **then return** $b_\perp$
$b' \leftarrow \mathcal{A}^{\mathsf{OEval}}(\mathsf{vk}, \mathsf{aux}, \mathsf{st}_\mathcal{A})$
return $b' = 0$

$\mathsf{OEval}(\mathsf{in})$

if $b = 0$ **then**
  return $f(\mathsf{vk}, \mathsf{aux}, (\mathsf{sk}_i)_{i\in\mathsf{HS}}, \mathsf{in})$
else
  return $\mathsf{SimF}(\mathsf{vk}, \mathsf{aux}, \mathsf{st}, \mathsf{in})$

Figure 5.3: Simulatability game for a sDKG scheme. The adversary $\mathcal{A}$ wins if the game sDKG-SIM returns 1, i.e. if it is guessed correctly whether it is provided with real or simulated shares.

$\mathsf{Game}^{\mathsf{sDKG\text{-}IA}}_{\mathsf{sDKG},\mathcal{A}}(1^\lambda, N, T, i^*)$

$(\mathsf{st}_i)_{i\in[N]} \leftarrow \mathsf{Setup}(1^\lambda, N, T, t)$
$\mathsf{st}_\mathcal{A} \leftarrow \mathcal{A}((\mathsf{st}_i)_{i\in[N]}, \mathsf{st}_\mathcal{A})$
for $r \in [R_{\mathsf{KG}}]$ **do**
  $\mathsf{st}_{i*}, \mathsf{pm}_r^{i*} \leftarrow \mathsf{ShareKeygen}(i^*, \mathsf{st}_{i*}, \mathsf{pm}_{r-1})$
  $((\mathsf{pm}_r^i)_{i\in[N]\backslash\{i*\}}, \mathsf{st}_\mathcal{A}) \leftarrow \mathcal{A}^{\mathsf{H}}(\mathsf{pm}_r^{i*}, \mathsf{st}_\mathcal{A})$
$U := \mathsf{KGIdentifyAbort}((\mathsf{pm}_r)_{r\in[R_{\mathsf{KG}}]})$
if $U \neq \varnothing$ **then return** $i^* \notin U$
$(\mathsf{vk}, \mathsf{aux}) := \mathsf{CombineKey}((\mathsf{pm}_r)_{r\in[R_{\mathsf{KG}}]})$
if $(\mathsf{vk}, \mathsf{aux}) = \perp$ **then return** 1
$\mathsf{sk}_{i*} := \mathsf{PartialSecret}(\mathsf{st}_i)$
return $\neg\mathsf{SharesVerify}(\mathsf{vk}, \mathsf{aux}, \mathsf{sk}_{i*}) = 0$

Figure 5.4: Abort identification game for a sDKG. By convention, $\mathsf{pm}_0 = \perp$. The adversary $\mathcal{A}$ wins if the game returns 1, i.e. if it successfully frames the honest user i^* or produces an invalid sharing without any misbehaving user detected.

$\mathsf{Repl.TrustedKeygen}(N, T) \to (\mathsf{vk}, \mathsf{aux}, \vec{\mathsf{sk}})$

$\mathsf{seed} \leftarrow \{0,1\}^\lambda$
$\mathbf{A} := \mathsf{H}_{\mathbf{A}}(\mathsf{seed})$
// List sets of $N - T + 1$ parties
for $I \subset [N]$ s.t. $|I| = N - T + 1$ **do**
  $\mathbf{s}_I \leftarrow \chi_{\mathsf{sk}}$
  $\mathbf{b}_I := [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{s}_I$
  for $i \in I$ **do**
    $\mathsf{sk}_i = \mathsf{sk}_i \cup \{\mathbf{s}_I\}$
$\mathbf{b}_\top := \left\lfloor \sum_I \mathbf{b}_I \right\rceil_{\nu_{\mathbf{b}}}$
$\mathsf{vk} := (\mathsf{seed}, \mathbf{b}_\top)$ // Verification key
$\mathsf{aux} := (\mathbf{b}_I)_I$ // Auxiliary key
$\vec{\mathsf{sk}} := (\mathsf{sk}_i)_{i \in [N]}$
return $(\mathsf{vk}, \mathsf{aux}, \vec{\mathsf{sk}})$

$\mathsf{Repl.VandRecover}(\mathsf{act}) \to \{0,1\}^{S_1 \times S_2 \times \ldots \times S_N}$

for $i \in \mathsf{act}$ **do**
  for $I \subset [N]$ of cardinality $N - T + 1$ such that $i \in I$ **do**
    if $i = \max(I \cap \mathsf{act})$ **then**
      $\nu_{i,I} = 1$
    else
      $\nu_{i,I} = 0$
return $\mathbf{m} := (m_i)_{i \in \mathsf{act}, I \text{ s.t. } i \in I}$

Figure 5.5: Trusted key generation and coefficient recovery for replicated secret sharing.

Setup$(1^\lambda, N, T)$

for $i \in [N]$ **do**
  $\mathsf{sk}_i^{\mathsf{SIG}}, \mathsf{pk}_i^{\mathsf{SIG}} \leftarrow \mathsf{SIG.Keygen}(1^\lambda)$
  for $j \in [N]$ **do**
    $K_{i\to j}^{\mathsf{SKE}} \leftarrow \mathsf{SKE.Keygen}(1^\lambda)$
    $\mathsf{sig}_{i\to j}^{\mathsf{SKE}} := \mathsf{SIG.Sign}(\mathsf{sk}_i^{\mathsf{SIG}}, \{i, j, K_{i\to j}^{\mathsf{SKE}}\})$
for $i \in [N]$ **do**
  $\mathsf{st}_i := \{(\mathsf{pk}_i^{\mathsf{SIG}})_{i\in[N]}, (K_{i'\to j'}^{\mathsf{SKE}}, \mathsf{sig}_{i'\to j'}^{\mathsf{SKE}})_{i'=i \vee j'=i}\}$
return $(\mathsf{st}_i)_{i\in[N]}$

ShareKeygen$_1(i, \mathsf{st}_i)$

$\mathsf{seed}_i \xleftarrow{\$} \{0,1\}^\lambda$
$\mathsf{st}_i.\mathsf{session} := \{\mathsf{seed}_i\}$
$\mathsf{cmt}_i = \mathsf{H}_{\mathsf{cmt}}(\mathsf{seed}_i)$
return $\mathsf{pm}_1^i := \mathsf{cmt}_i$

ShareKeygen$_2(i, \mathsf{st}_i, \mathsf{pm}_1)$

Fetch (seed_i) from $\mathsf{st}_i.\mathsf{session}$
$\mathsf{st}_i.\mathsf{session} := \{\mathsf{pm}_1\}$
return $\mathsf{pm}_2^i := \mathsf{seed}_i$

ShareKeygen$_3(i, \mathsf{st}_i, \mathsf{pm}_2)$

Fetch (pm_1) from $\mathsf{st}_i.\mathsf{session}[\mathsf{sid}]$
Parse $\mathsf{pm}_1 = (\mathsf{cmt}_j)_j, \mathsf{pm}_2 = (\mathsf{seed}_j)_j$
for $j \in [N]$ **do**
  if $\mathsf{cmt}_j \neq \mathsf{H}_{\mathsf{cmt}}(\mathsf{seed}_j)$ **then**
    return $\perp$ // Check hash commit.
$\mathsf{seed} := \mathsf{H}_{\mathsf{seed}}((\mathsf{seed}_j)_j)$
$\mathbf{A} := \mathsf{H}_{\mathbf{A}}(\mathsf{seed})$
$\mathsf{msgs}_i = \varnothing$
for $|I| = N - T + 1 \wedge i = \max(I)$ **do**
  $\mathbf{s}_I \leftarrow \chi_{\mathsf{sk}}$ // Sample secret share
  $\mathbf{b}_I \leftarrow [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{s}_I$
  $\mathsf{cmt}'_I := \mathsf{H}_{\mathsf{cmt}}(\mathbf{b}_I)$
  for $j \in I \backslash \{i\}$ **do**
    $\mathsf{msgs}_i := \mathsf{msgs}_i \cup \{(I, j, \mathsf{SKE.Encrypt}(K_{i\to j}^{\mathsf{SKE}}, \mathbf{s}_I))\}$
$\mathsf{st}_i.\mathsf{session} := \{\mathsf{seed}, (\mathbf{s}_I)_I\}$
return $\mathsf{pm}_3^i := (\mathsf{msgs}_i, (\mathsf{cmt}'_I)_I)$

ShareKeygen$_4(i, \mathsf{st}_i, \mathsf{pm}_3)$

Fetch $(\mathsf{seed}, (\mathbf{s}_I)_I)$ from $\mathsf{st}_i.\mathsf{session}$
Parse $\mathsf{pm}_3 = (\mathsf{msgs}_j, (\mathsf{cmt}'_I)_{I \text{ s.t. } \max(I)=j})_j$
$\mathsf{seed} := \mathsf{H}_{\mathsf{seed}}((\mathsf{seed}_i)_i)$
$\mathbf{A} := \mathsf{H}_{\mathbf{A}}(\mathsf{seed})$
$\mathsf{complaints}_i = \varnothing$
for $|I| = N - T + 1 \wedge i \neq \max(I)$ **do**
  $\mathbf{s}_I := \mathsf{SKE.Decrypt}$ from $\mathsf{msgs}_{\max(I)}$ // Recover $\mathbf{s}_I$
  $\mathbf{b}_I \leftarrow [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{s}_I$
  if $\|\mathbf{s}_I\| > \eta_1 \vee \mathsf{cmt}'_I \neq \mathsf{H}_{\mathsf{cmt}}(\mathbf{b}_I)$ **then**
    $\mathsf{complaints}_i := \mathsf{complaints}_i \cup \{I, j := \max(I), K_{j\to i}^{\mathsf{SKE}}, \mathsf{sig}_{j\to i}^{\mathsf{SKE}}\}$
return $\mathsf{pm}_4^i := (\mathsf{complaints}_i, (\mathbf{b}_I)_I)$

Figure 5.6: Algorithms for our sDKG relying on replicated secret sharing. For conciseness, we omit aborts due to parsing errors: if an unexpected message is received, parties abort by returning $\perp$.

CombineKey($\mathsf{pm}_2, \mathsf{pm}_3, \mathsf{pm}_4$)

Parse $\mathsf{pm}_2 = (\mathsf{seed}_i)_i, \mathsf{pm}_3 = (\mathsf{msgs}_i, \cdot)_i$, and $\mathsf{pm}_4 = (\mathsf{complaints}_i, (\mathbf{b}_I)_I)_i$
for set I of cardinality $N - T + 1$ **do**
  if $\mathsf{H}_{\mathsf{cmt}}(\mathbf{b}_I) \neq \mathsf{cmt}'_I$ **then return** $\perp$
for $i \in [N]$ **do**
  for $(I, j, K^{\mathsf{SKE}}_{j\to i}, \mathsf{sig}^{\mathsf{SKE}}_{j\to i}) \in \mathsf{complaints}_i$ **do**
    if $\neg\mathsf{SIG.Verify}(\mathsf{sig}^{\mathsf{SKE}}_{j\to i}, K^{\mathsf{SKE}}_{j\to i})$ **then**
      return $\perp$
    else
      $\mathbf{s}_I :=$ SKE.Decrypt using $K^{\mathsf{SKE}}_{i\to j}$ from msgs_j; $\mathbf{b}_I \leftarrow [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{s}_I$
      if $\|\mathbf{s}_I\| > \eta_1 \vee \mathsf{cmt}'_I \neq \mathsf{H}_{\mathsf{cmt}}(\mathbf{b}_I)$ **then return** $\perp$
$\mathsf{seed} := \mathsf{H}_{\mathsf{seed}}((\mathsf{seed}_i)_i)$
$\mathbf{b}_\top := \left\lfloor \sum_I \mathbf{b}_I \right\rceil_{\nu_{\mathbf{b}}}$
return $\mathsf{vk} := (\mathsf{seed}, \mathbf{b}_\top), \mathsf{aux} := (\mathbf{b}_I)_I$

KGIdentifyAbort($(\mathsf{pm}_i)_{i\in[R_{\mathsf{KG}}]}$)

Parse $\mathsf{pm}_1 = (\mathsf{cmt}_i)_i, \mathsf{pm}_2 = (\mathsf{seed}_i)_i, \mathsf{pm}_3 = (\mathsf{msgs}_i, (\mathsf{cmt}'_I)_I)_i$, and $\mathsf{pm}_4 = (\mathsf{complaints}_i, (\mathbf{b}_I)_I)_i$
// Check round 3
for $j \in [N]$ **do**
  if $\mathsf{cmt}_j \neq \mathsf{H}_{\mathsf{cmt}}(\mathsf{seed}_j)$ **then**
    return $\{j\}$ // Check hash commit.
// Check round 4
for set I of cardinality $N - T + 1$ **do**
  if $\mathsf{H}_{\mathsf{cmt}}(\mathbf{b}_I) \neq \mathsf{cmt}'_I$ **then return** $\{\max(I)\}$
$\mathsf{invalid} = \varnothing$
for $i \in [N]$ **do**
  for $(I, j, K^{\mathsf{SKE}}_{j\to i}, \mathsf{sig}^{\mathsf{SKE}}_{j\to i}) \in \mathsf{complaints}_i$ **do**
    if $\neg\mathsf{SIG.Verify}(\mathsf{sig}^{\mathsf{SKE}}_{j\to i}, K^{\mathsf{SKE}}_{j\to i})$ **then** $\mathsf{invalid} := \mathsf{invalid} \cup \{i\}$
    else
      $\mathbf{s}_I :=$ SKE.Decrypt using $K^{\mathsf{SKE}}_{i\to j}$ from msgs_j; $\mathbf{b}_I \leftarrow [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{s}_I$
      if $\|\mathbf{s}_I\| > \eta_1 \vee \mathsf{cmt}'_I \neq \mathsf{H}_{\mathsf{cmt}}(\mathbf{b}_I)$ **then** $\mathsf{invalid} := \mathsf{invalid} \cup \{j\}$
return invalid

Figure 5.7: Key aggregation and abort identification for our sDKG relying on replicated secret sharing. We omit aborts due to parsing errors for conciseness.

Hybrid_2

1 : $\mathsf{ProgrammedSeed} = \bot$
2 : // Lines 1-4 are identical
3 : **for** $i \in \mathsf{HS}$ **do**
4 : $\quad \mathsf{cmt}_i \xleftarrow{\$} \{0,1\}^{2\lambda}$
5 : // Round 1: identical
6 : // Round 2: updated
7 : **for** $i \in \mathsf{HS}$ **do**
8 : $\quad \mathsf{seed}_i \xleftarrow{\$} \{0,1\}^{\lambda}$
9 : $\quad \mathsf{H}_{\mathsf{cmt}}(\mathsf{seed}_i) := \mathsf{cmt}_i$ // Program random oracle
10 : **if** $\forall j \in \mathsf{CS}, \exists \mathsf{seed}_j, \mathsf{cmt}_j = H(\mathsf{seed}_j)$ **then**
11 : $\quad \mathsf{seed} \xleftarrow{\$} \chi_{\mathsf{seed}}, \mathbf{A} \xleftarrow{\$} R_q^{k \times \ell}$
12 : $\quad \mathsf{H}_{\mathsf{seed}}((\mathsf{seed}_i)_{i \in [N]}) := \mathsf{seed}$
13 : $\quad \mathsf{H}_{\mathbf{A}}(\mathsf{seed}) := \mathbf{A}$ // Program random oracle
14 : $\quad \mathsf{ProgrammedSeed} = (\mathsf{seed}_i)_{i \in [N]}$
15 : $(\mathsf{pm}_2^i)_{i \in \mathsf{HS}} = (\mathsf{seed}_i)_{i \in \mathsf{HS}}$
16 : // Rest is identical
17 : // Round 3: updated
18 : **for** $i \in \mathsf{HS}$ **do**
19 : $\quad$ Parse $\mathsf{pm}_1 = (\mathsf{cmt}_j)_j, \mathsf{pm}_2 = (\mathsf{seed}_j)_j$
20 : $\quad$ **for** $j \in [N]$ **do**
21 : $\quad\quad$ **if** $\mathsf{cmt}_j \neq \mathsf{H}_{\mathsf{cmt}}(\mathsf{seed}_j)$ **then return** $\bot$
22 : **assert** $\mathsf{ProgrammedSeed} = (\mathsf{seed}_i)_{i \in [N]}$
23 : // Round 4 and remaining lines are left unchanged

Figure 5.8: The second hybrid of the security proof of the functional simulatability of our Repl sDKG scheme. Difference with the previous hybrid are highlighted. We omit state updates for conciseness. In case the game programs the random oracle on an existing index, we consider that the adversary loses the game.

Hybrid_3

// Round 4: updated
for $i \in \mathsf{HS}$ **do**
  Fetch $(\mathsf{seed}, (\mathbf{s}_I)_I)$ from $\mathsf{st}_i.\mathsf{session}$
  Parse $\mathsf{pm}_3 = (\mathsf{msgs}_j, (\mathsf{cmt}'_I)_{I \text{ s.t. } \max(I)=j})_j$
  $\mathsf{seed} := \mathsf{H}_{\mathsf{seed}}((\mathsf{seed}_i)_i); \mathbf{A} := \mathsf{H}_{\mathbf{A}}(\mathsf{seed})$
  $\mathsf{complaints}_i = \varnothing$
  for I s.t. $|I| = N - T + 1 \wedge i \neq \max(I)$ **do**
    if $\max(I) \in \mathsf{HS}$ **then**
      continue // Do not complain against honest senders
    $\mathbf{s}_I := \mathsf{SKE.Decrypt}$ from $\mathsf{msgs}_{\max(I)}$
    $\mathbf{b}_I \leftarrow [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{s}_I$
    if $\|\mathbf{s}_I\| > \eta_1 \vee \mathsf{cmt}'_I \neq \mathsf{H}_{\mathsf{cmt}}(\mathbf{b}_I)$ **then**
      $\mathsf{complaints}_i := \mathsf{complaints}_i \cup \{I, j := \max(I), K^{\mathsf{SKE}}_{j \to i}, \mathsf{sig}^{\mathsf{SKE}}_{j \to j}\}$
  $\mathsf{pm}^i_4 := (\mathsf{complaints}_i, (\mathbf{b}_I)_I)$

Figure 5.9: The third hybrid of the security proof of the functional simulatability of our Repl sDKG scheme. Difference with the previous hybrid are highlighted. We omit state updates for conciseness.

Hybrid_4

// Round 3: updated
Retrieve $(\mathsf{seed}_j)_{j \in \mathsf{HS}}$
for $i \in \mathsf{HS}$ **do**
  Parse $\mathsf{pm}_1 = (\mathsf{cmt}_j)_j, \mathsf{pm}_2 = (\mathsf{seed}_j)_j$
  for $j \in [N]$ **do**
    if $\mathsf{cmt}_j \neq \mathsf{H}_{\mathsf{cmt}}(\mathsf{seed}_j)$ **then return** $\perp$
assert $\mathsf{ProgrammedSeed} = (\mathsf{seed}_i)_{i \in [N]}$
for $i \in \mathsf{HS}$ **do**
  $\mathsf{seed} := \mathsf{H}_{\mathsf{seed}}((\mathsf{seed}_j)_j)$
  $\mathbf{A} := \mathsf{H}_{\mathbf{A}}(\mathsf{seed})$
  $\mathsf{msgs}_i = \varnothing$
  for I s.t. $|I| = N - T + 1 \wedge i = \max(I)$ **do**
    $\mathbf{s}_I \leftarrow \chi_{\mathsf{sk}}$ // Sample secret share
    $\mathbf{b}_I \leftarrow [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{s}_I$
    $\mathsf{cmt}'_I := \mathsf{H}_{\mathsf{cmt}}(\mathbf{b}_I)$
    for $j \in I \backslash \{i\}$ **do**
      if $j \in \mathsf{HS}$ **then**
        $\mathsf{msgs}_i := \mathsf{msgs}_i \cup \{(I, j, \mathsf{SKE.Encrypt}(K^{\mathsf{SKE}}_{i \to j}, \mathbf{0}))\}$
      else
        $\mathsf{msgs}_i := \mathsf{msgs}_i \cup \{(I, j, \mathsf{SKE.Encrypt}(K^{\mathsf{SKE}}_{i \to j}, \mathbf{s}_I))\}$
  $\mathsf{st}_i.\mathsf{session} := \{\mathsf{seed}, (\mathbf{s}_I)_I\}$
  $\mathsf{pm}^i_3 := (\mathsf{msgs}_i, (\mathsf{cmt}'_I)_I)$
return $(\mathsf{pm}^i_3)_{i \in \mathsf{HS}}$

Figure 5.10: The fourth hybrid of the security proof of the functional simulatability of our Repl sDKG scheme. Difference with the previous hybrid are highlighted. We omit state updates for conciseness.

Hybrid_5

$\mathsf{ProgrammedB} = \bot$
// Round 3: updated
Retrieve $(\mathsf{seed}_j)_{j \in \mathsf{HS}}$
for $i \in \mathsf{HS}$ **do**
  Parse $\mathsf{pm}_1 = (\mathsf{cmt}_j)_j, \mathsf{pm}_2 = (\mathsf{seed}_j)_j$
  for $j \in [N]$ **do**
    if $\mathsf{cmt}_j \neq \mathsf{H}_{\mathsf{cmt}}(\mathsf{seed}_j)$ **then return** $\bot$
assert $\mathsf{ProgrammedSeed} = (\mathsf{seed}_i)_{i \in [N]}$
Fix index I_0 such that $I_0 \subset \mathsf{HS}$
$\mathsf{cmt}'_{I_0} \xleftarrow{\$} \{0,1\}^{2\lambda}$
for $i \in \mathsf{HS}$ **do**
  $\mathsf{seed} := \mathsf{H}_{\mathsf{seed}}((\mathsf{seed}_j)_j)$
  $\mathbf{A} := \mathsf{H}_{\mathbf{A}}(\mathsf{seed})$
  $\mathsf{msgs}_i = \varnothing$
  for I s.t. $|I| = N - T + 1 \wedge i = \max(I) \wedge I \neq I_0$ **do**
// Rest is identical
// Round 4: updated
Parse $\mathsf{pm}_3 = (\mathsf{msgs}_j, (\mathsf{cmt}'_I)_{I \text{ s.t. } \max(I)=j})_j$
if $\forall I \neq I_0, \exists \mathsf{H}_{\mathsf{cmt}}(\mathbf{b}_I) = \mathsf{cmt}'_I$ **then**
  $\mathsf{ProgrammedB} = (\mathbf{b}_I)_{I \neq I_0}$
$\mathbf{s}_{I_0} \leftarrow \chi_{\mathsf{sk}}$
$\mathbf{b}_{I_0} := [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{s}_{I_0}$
$\mathsf{H}(\mathbf{b}_{I_0}) := \mathsf{cmt}'_{I_0}$ // Program random oracle
for $i \in \mathsf{HS}$ **do**
// Other lines are identical

$\mathsf{CombineKey}(\mathsf{pm}_2, \mathsf{pm}_3, \mathsf{pm}_4)$

Parse $\mathsf{pm}_2 = (\mathsf{seed}_i)_i, \mathsf{pm}_3 = (\mathsf{msgs}_i, \cdot)_i$, and $\mathsf{pm}_4 = (\mathsf{complaints}_i, (\mathbf{b}_I)_I)_i$
for set I of cardinality $N - T + 1$ **do**
  if $\mathbf{b}_I \neq \mathsf{cmt}'_I$ **then**
    return $\bot$
assert $\mathsf{ProgrammedB} = (\mathbf{b}_I)_{I \neq I_0}$
// Other lines are identical

Figure 5.11: The fifth hybrid of the security proof of the functional simulatability of our Repl sDKG scheme. Difference with the previous hybrid are highlighted . We omit state updates for conciseness.

Hybrid_6

1 : // Identical

$\mathsf{OEval}(\mathsf{in} := (\mathsf{act}, i))$

1 : $(\nu_{j,I})_{j \in \mathsf{act}, I \text{ s.t. } j \in I} := \mathsf{Repl.VandRecover}(\mathsf{vk}, \mathsf{aux}, \mathsf{act})$
2 : $(\mathbf{r}_j)_{j \in \mathsf{HS} \cap \mathsf{act}} \leftarrow \mathcal{D}_{\mathbf{r}}^{(\ell+k) \cdot |\mathsf{HS} \cap \mathsf{act}|}$
3 : $c \leftarrow \mathcal{C}$
4 : **if** $i = \max(I_0 \cap \mathsf{act})$ **then**
5 : $\quad \mathbf{r}_i \leftarrow \chi_{\mathbf{r}}$
6 : $\quad \mathbf{h}_i := c \cdot \mathbf{s}_{I_0} + \mathbf{r}_i$
7 : $\quad$ **return** $\mathbf{h}_i + c \cdot \sum_{I \neq I_0} \nu_{i,I}$
8 : **else**
9 : $\quad \mathbf{r}_i \leftarrow \chi_{\mathbf{r}}$
10 : $\quad$ **return** $c \cdot \sum_I \nu_{i,I} \cdot S_i[I] + \mathbf{r}_i$

Figure 5.12: The sixth hybrid of the security proof of the functional simulatability of our Repl sDKG scheme. Difference with the previous hybrid are highlighted. We omit state updates for conciseness.

Hybrid_7

1 : // Hybrid 4: updated
2 : Parse $\mathsf{pm}_3 = (\mathsf{msgs}_j, (\mathsf{cmt}'_I)_{I \text{ s.t. } \max(I)=j})_j$
3 : **if** $\forall I \neq I_0, \exists \mathsf{H}_{\mathsf{cmt}}(\mathbf{b}_I) = \mathsf{cmt}'_I$ **then**
4 : $\quad \mathsf{ProgrammedB} = (\mathbf{b}_I)_{I \neq I_0}$
5 : $\mathbf{s}_{I_0} \leftarrow \chi_{\mathsf{sk}}$
6 : $\mathbf{b}_\top \leftarrow \chi_{\mathbf{b}}$
7 : Sample $\mathbf{b} \xleftarrow{\$} R_q^k$ conditioned on $\mathbf{b}_\top = \lfloor \mathbf{b} \rceil_{\nu_{\mathbf{b}}}$
8 : $\mathbf{b}_{I_0} := \mathbf{b} - \sum_{I \neq I_0} \mathbf{b}_I$
9 :
10 : $H(\mathbf{b}_{I_0}) := \mathsf{cmt}'_{I_0}$ // Program random oracle
11 : **for** $i \in \mathsf{HS}$ **do**
12 : // Other lines are identical

Figure 5.13: The seventh hybrid of the security proof of the functional simulatability of our Repl sDKG scheme. Difference with the previous hybrid are highlighted. We omit state updates for conciseness.

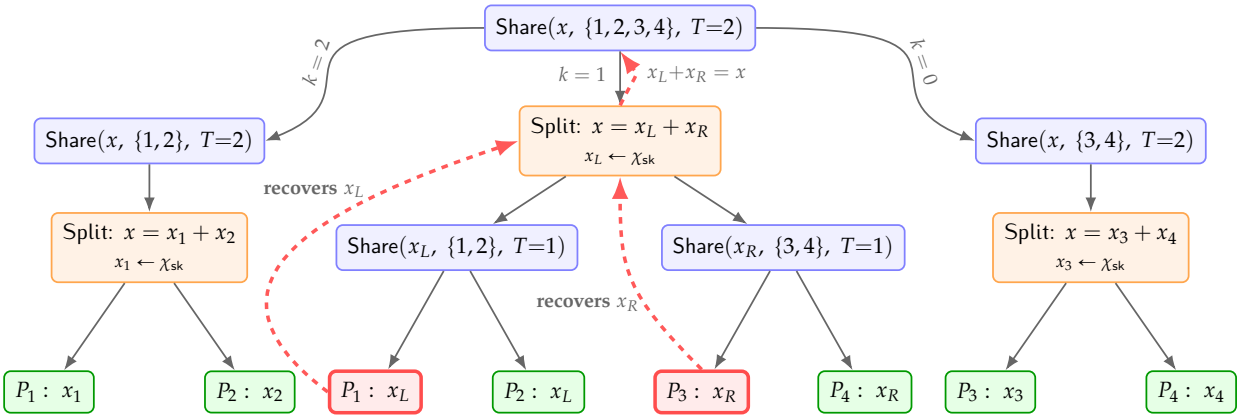


Figure 5.14: Vandermonde sharing of a secret x among $N = 4$ parties with threshold $T = 2$. Blue rectangles correspond to recursive sharing, orange rectangles correspond to the splitting of secrets into two sub-secrets, and green or red boxes represent the shares given to parties. The sharing is based on going over all the possible values k of selected parties in the left half $\{P_1, P_2\}$, which corresponds to the combinatorial decomposition of $\binom{4}{2}$ into $\binom{2}{2}\cdot\binom{2}{0} + \binom{2}{1}\cdot\binom{2}{1} + \binom{2}{0}\cdot\binom{2}{2}$. The **red** leaves and dashed arrows highlight the recovery path for the set of selected parties $\{P_1, P_3\}$ (one party per half, i.e. $k = 1$).

$\mathsf{Vand.TrustedKeygen}(N, T) \to (\mathsf{vk}, \mathsf{aux}, \vec{\mathsf{sk}})$

$\mathsf{seed} \leftarrow \{0,1\}^\lambda$
$\mathbf{A} := \mathsf{H}_{\mathbf{A}}(\mathsf{seed})$
$\mathbf{s} \leftarrow \chi_{\mathsf{sk}}$
$\mathbf{b} := [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{s}$
$\mathsf{Share} \leftarrow \mathsf{Vand.Share}(\mathbf{s}, [N], T, \texttt{""}, \mathsf{Share} = \{:\})$
$\mathsf{aux} = \{:\}$
for $i \in [N]$ **do**
  for idx s.t. $\mathsf{Share}[i][\mathsf{idx}] \neq \bot$ **do**
    $\mathbf{b}_{i,\mathsf{idx}} := [\mathbf{A} \quad \mathbf{I}] \cdot \mathsf{Share}[i][\mathsf{idx}]$
    $\mathsf{aux}[i][\mathsf{idx}] := \mathsf{aux}[i][\mathsf{idx}] \cup \{\mathbf{b}_{i,\mathsf{idx}}\}$
$\vec{\mathsf{sk}} := (\mathsf{sk}_i := \mathsf{Share}[i])_{i \in [N]}$
return $(\mathsf{vk}, \mathsf{aux}, \vec{\mathsf{sk}})$

$\mathsf{Vand.Share}(x, \mathcal{P}, T, \mathsf{idx}, \mathsf{Share})$

$N \leftarrow |\mathcal{P}|$
if $T = 1$ **then**
  for $i \in \mathcal{P}$ **do** $\mathsf{Share}[i][\mathsf{idx}] \leftarrow x$ // 1-out-of-N
else
  $b \leftarrow \lfloor N/2 \rfloor$
  $\mathcal{P}_L \leftarrow \mathcal{P}[1 \ldots b]$; $\mathcal{P}_R \leftarrow \mathcal{P}[b+1 \ldots N]$ // Divide $\mathcal{P}$ into two disjoint sets
  $\min T_L \leftarrow \max(0, T - (N - b))$ // Minimum threshold for $\mathcal{P}_L$
  $\max T_L \leftarrow \min(T, b)$ // Maximum threshold for $\mathcal{P}_L$
  for $k \in [\min T_L, \max T_L]$ **do**
    $\mathsf{idx}_L \leftarrow \mathsf{idx} \parallel \texttt{":L:"} \parallel k$, $\mathsf{idx}_R \leftarrow \mathsf{idx} \parallel \texttt{":R:"} \parallel (T - k)$
    if $k = 0$ **then**
      $\mathsf{Share} \leftarrow \mathsf{Vand.Share}(x, \mathcal{P}_R, T, \mathsf{idx}_R, \mathsf{Share})$ // T-out-of-$(N-b)$
    else if $k = T$ **then**
      $\mathsf{Share} \leftarrow \mathsf{Vand.Share}(x, \mathcal{P}_L, T, \mathsf{idx}_L, \mathsf{Share})$ // T-out-of-b
    else
      $x_L \leftarrow \chi_{\mathsf{sk}}; x_R \leftarrow (x - x_L) \bmod q$ // This guarantees short shares
      $\mathsf{Share} \leftarrow \mathsf{Vand.Share}(x_L, \mathcal{P}_L, k, \mathsf{idx}_L, \mathsf{Share})$ // k-out-of-b
      $\mathsf{Share} \leftarrow \mathsf{Vand.Share}(x_R, \mathcal{P}_R, T - k, \mathsf{idx}_R, \mathsf{Share})$ // $(T-k)$-out-of-$(N-b)$
return Share

Figure 5.15: Algorithms (part 1/2) for our short Vandermonde secret sharing. The list $\mathsf{Share}[\cdot][\cdot]$ is initialized to $\bot$. We assume that $\mathcal{P}$ is always lexicographically ordered.

```
Vand.Recover(𝒫, act, idx, Index)
1 : N ← |𝒫|; T ← |act|
2 : if T = 1 then
3 :   for i ∈ 𝒫 do Index[i] ← idx
4 : else
5 :   b ← ⌊N/2⌋
6 :   𝒫_L ← 𝒫[1...b]; 𝒫_R ← 𝒫[b+1...N]
7 :   act_L ← act ∩ 𝒫_L; act_R ← act ∩ 𝒫_R
8 :   k ← |act_L|
9 :   idx_L ← idx ‖ ":L:" ‖ k, idx_R ← idx ‖ ":R:" ‖ (T − k)
10 :  if k = 0 then
11 :    Index ← Vand.Recover(𝒫_R, act_R, idx_R, Index)
12 :  else if k = T then
13 :    Index ← Vand.Recover(𝒫_L, act_L, idx_L, Index)
14 :  else
15 :    Index ← Vand.Recover(𝒫_L, act_L, idx_L, Index)
16 :    Index ← Vand.Recover(𝒫_R, act_R, idx_R, Index)
17 : return Index
```

Figure 5.16: Algorithms (part 2/2) for our short Vandermonde secret sharing. The list $\mathsf{Index}[\cdot]$ is initialized to $\perp$. We assume that $\mathcal{P}$ and act is always lexicographically ordered.

ShareKeygen$_3(i, \mathsf{st}_i, \mathsf{pm}_2)$

Fetch (pm_1) from $\mathsf{st}_i.\mathsf{session}[\mathsf{sid}]$
Parse $\mathsf{pm}_1 = (\mathsf{cmt}_j)_j, \mathsf{pm}_2 = (\mathsf{seed}_j)_j$
for $j \in [N]$ **do**
  if $\mathsf{cmt}_j \neq \mathsf{H}_{\mathsf{cmt}}(\mathsf{seed}_j)$ **then**
    return $\perp$ // Check hash commit.
$\mathsf{seed} := \mathsf{H}_{\mathsf{seed}}((\mathsf{seed}_j)_j)$
$\mathbf{A} := \mathsf{H}_{\mathbf{A}}(\mathsf{seed})$
$\mathbf{s}_i \leftarrow \chi_{\mathsf{sk}}$
$\mathsf{Share} := \mathsf{Vand.Share}(\mathbf{s}_i, [N], T, \text{""}, \mathsf{Share} = \{:\})$
$\mathsf{msgs}_i = \varnothing$
for $j \in [N] \backslash \{i\}$ **do**
  for idx s.t. $\mathsf{Share}[j][\mathsf{idx}] \neq \perp$ **do**
    $\mathbf{b}^{(i)}_{j,\mathsf{idx}} := [\mathbf{A} \quad \mathbf{I}] \cdot \mathsf{Share}[j][\mathsf{idx}]$
    $\mathsf{cmt}'^{(i)}_{j,\mathsf{idx}} := \mathsf{H}_{\mathsf{cmt}}(\mathbf{b}^{(i)}_{j,\mathsf{idx}})$
  $\mathsf{msgs}_i := \mathsf{msgs}_i \cup \{(j, \mathsf{SKE.Encrypt}(K^{\mathsf{SKE}}_{i \to j}, \mathsf{Share}[j]))\}$
$\mathsf{st}_i.\mathsf{session} := \{\mathsf{seed}, (\mathbf{b}^{(i)}_j)_j, \mathsf{Share}\}$
return $\mathsf{pm}^i_3 := (\mathsf{msgs}_i, (\mathsf{cmt}'^{(i)}_{j,\mathsf{idx}})_{j,\mathsf{idx}})$

ShareKeygen$_4(i, \mathsf{st}_i, \mathsf{pm}_3)$

Fetch $(\mathsf{seed}, (\mathbf{b}^{(i)}_j)_j, \mathsf{Share})$ from $\mathsf{st}_i.\mathsf{session}$
Parse $\mathsf{pm}_3 = (\mathsf{msgs}_j, (\mathsf{cmt}'^{(j)}_{u,\mathsf{idx}})_{u,\mathsf{idx}})_j$
$\mathsf{seed} := \mathsf{H}_{\mathsf{seed}}((\mathsf{seed}_i)_i)$
$\mathbf{A} := \mathsf{H}_{\mathbf{A}}(\mathsf{seed})$
$\mathsf{complaints}_i = \varnothing$
for $j \in [N] \backslash \{i\}$ **do**
  $\mathsf{Share}_j := \mathsf{SKE.Decrypt}$ from msgs_j
  for $\mathsf{idx} \in \mathsf{Share}[i]$ **do**
    $\mathbf{b}'^{(j)}_{i,\mathsf{idx}} \leftarrow [\mathbf{A} \quad \mathbf{I}] \cdot \mathsf{Share}_j[i][\mathsf{idx}]$
    if $\|\mathsf{Share}_j[i][\mathsf{idx}]\| > \eta^{\mathsf{ind}}_1 \vee \mathsf{cmt}'^{(j)}_{i,\mathsf{idx}} \neq \mathsf{H}_{\mathsf{cmt}}(\mathbf{b}'^{(j)}_{i,\mathsf{idx}})$ **then**
      $\mathsf{complaints}_i := \mathsf{complaints}_i \cup \{j, K^{\mathsf{SKE}}_{j \to i}, \mathsf{sig}^{\mathsf{SKE}}_{j \to i}\}$
return $\mathsf{pm}^i_4 := (\mathsf{complaints}_i, (\mathbf{b}^{(i)}_j)_j)$

Figure 5.17: Modified algorithms for our sDKG relying on Vandermonde secret sharing. For conciseness, we omit aborts due to parsing errors: if an unexpected message is received, parties abort by returning $\perp$.

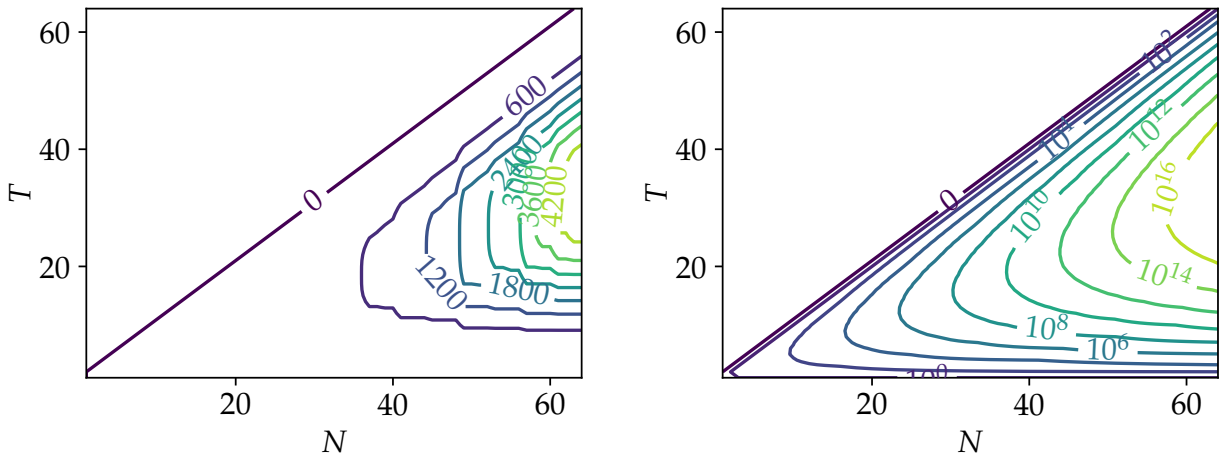
$\mathsf{CombineKey}(\mathsf{pm}_2, \mathsf{pm}_3, \mathsf{pm}_4)$

1 : // Reconstruct final key
2 : $\mathsf{Index} := \mathsf{Vand.Recover}([N], [T], \texttt{""}, \mathsf{Index} = \{:\})$
3 : $\mathbf{b} = \sum_{i\in[N]} \sum_{j\in[T]} \mathbf{b}^{(i)}_{j,\mathsf{Index}[j]}$
4 : ... // If reconstruction of $\mathbf{b}$ with different signing set yields different result, abort
5 : ... // Check complaints and abort if any is valid as in Fig. 5.7
6 : $\mathsf{seed} := \mathsf{H}_{\mathsf{seed}}((\mathsf{seed}_i)_i)$
7 : $\mathbf{b}_\top := [\mathbf{b}]_{\nu_\mathbf{b}}$
8 : **return** $\mathsf{vk} := (\mathsf{seed}, \mathbf{b}_\top), \mathsf{aux} := (\mathbf{b}_I)_I$

$\mathsf{KGIdentifyAbort}((\mathsf{pm}_i)_{i\in[R_{\mathsf{KG}}]})$

1 : ... // Check complaints and identify aborts as in Fig. 5.7

Figure 5.18: Key aggregation and abort identification for our sDKG relying on Vandermonde secret sharing. We omit aborts due to parsing errors for conciseness.



(a) Vandermonde

(b) Replicated

Figure 5.19: Contour plots of the maximum number of shares per party, as a function of N and T (undefined for $T > N$).

$\mathsf{ShareSign}_1(\mathsf{vk}, i, \mathsf{sk}_i, \mathsf{st}_i)$

1 : **require** $\nexists\, \mathsf{st}_i.\mathsf{session}[\mathsf{sid}]$
2 : $\mathbf{r}_i \leftarrow \chi_{\mathbf{r}}$
3 : $\mathbf{w}_i := [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}_i$
4 : $\mathsf{cmt}_i = \mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, i, \mathbf{w}_i)$
5 : $\mathsf{st}_i := \mathsf{st}_i \cup \{(\mathsf{cmt}_i, \mathbf{w}_i, \mathbf{r}_i)\}$
6 : **return** $\mathsf{pm}_1^i := \mathsf{cmt}_i$

$\mathsf{ShareSign}_3(\mathsf{vk}, i, \mathsf{sk}_i, \mathsf{st}_i, \mathsf{pm}_2)$

1 : **require** $(\cdot, \mathsf{pm}_2^i, \cdot) \in \mathsf{st}_i$
2 : Pick $(\mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, \mathbf{w}_i, \mathbf{r}_i)$ from st_i with $\mathsf{pm}_2^i = \mathbf{w}_i$
3 : Parse $\mathsf{pm}_1 = (\mathsf{cmt}_j)_j$ and $\mathsf{pm}_2 = (\mathbf{w}_j)_{j \neq i}$
4 : **for** $j \in \mathsf{act}$ **do**
5 : **if** $\mathsf{cmt}_j \neq \mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, j, \mathbf{w}_j)$ **then**
6 : **return** $\perp$
7 : $\mathsf{st}_i := \mathsf{st}_i \backslash \{(\mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, \mathbf{w}_i, \mathbf{r}_i)\}$
8 : $\mathbf{w} := \left\lfloor \sum_{j \in \mathsf{act}} \mathbf{w}_i \right\rceil_{\nu_{\mathbf{w}}}$
9 : $c := \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$
10 : $\nu := \mathsf{VandRecover}(\mathsf{vk}, \mathsf{aux}, \mathsf{act})$
11 : // Individual response in R_q^ℓ
12 : $\mathbf{z}_i^{(1)}, \mathbf{z}_i^{(2)} := c \cdot \sum_{\mathsf{idx} \in \nu[i]} \nu[i][\mathsf{idx}] \cdot \mathsf{sk}_i[\mathsf{idx}] + \mathbf{r}_i$
13 : **return** $\mathsf{pm}_3^i := \mathbf{z}_i^{(1)}$

$\mathsf{ShareSign}_2(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, i, \mathsf{sk}_i, \mathsf{st}_i, \mathsf{pm}_1)$

1 : **require** $\mathsf{act} \subseteq [N] \wedge i \in \mathsf{act}$
2 : **require** $(\mathsf{pm}_1^i, \cdot, \cdot) \in \mathsf{st}_i$
3 : Pick $(\mathsf{cmt}_i, \mathbf{w}_i, \mathbf{r}_i)$ from st_i with $\mathsf{pm}_1^i = \mathsf{cmt}_i$
4 : $\mathsf{st}_i := \mathsf{st}_i \backslash \{(\mathsf{cmt}_i, \mathbf{w}_i, \mathbf{r}_i)\}$
5 : $\mathsf{st}_i := \mathsf{st}_i \cup \{(\mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, \mathbf{w}_i, \mathbf{r}_i)\}$
6 : **return** $\mathsf{pm}_2^i := \mathbf{w}_i$

Figure 5.20: Algorithms (part 1/2) for a distributed signing with IA. For conciseness, we omit aborts due to parsing errors: if an unexpected message is received, parties abort.

$\mathsf{Combine}(\mathsf{vk} = (\mathsf{seed}, \mathbf{b}), \mathsf{act}, \mathsf{aux}, \mathsf{msg}, (\mathsf{pm}_k)_{k\in\{1,2,3\}})$

1 : Parse $\mathsf{pm}_2 = (\mathbf{w}_i)_i$ and $\mathsf{pm}_3 = (\mathbf{z}_i^{(1)})_i$
2 : $\mathbf{w} := \left\lfloor \sum_{j\in\mathsf{act}} \mathbf{w}_i \right\rceil_{\nu_\mathbf{w}}$
3 : $c := \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$
4 : $\mathbf{z}^{(1)} = \sum_{i\in[N]} \mathbf{z}_i^{(1)}$
5 : $\mathbf{u} := \left\lfloor \mathbf{A}\cdot\mathbf{z}^{(1)} - 2^{\nu_\mathbf{b}}\cdot c\cdot\mathbf{b} \right\rceil_{\nu_\mathbf{w}}$
6 : $\mathbf{h} := \mathbf{w} - \mathbf{u}$ // Hint
7 : **return** $\mathsf{sig} := (c, \mathbf{z}^{(1)}, \mathbf{h})$

$\mathsf{Verify}(\mathsf{vk} = (\mathsf{seed}, \mathbf{b}_\top), \mathsf{msg}, \mathsf{sig})$

1 : Parse $\mathsf{sig} = (c, \mathbf{z}^{(1)}, \mathbf{h})$
2 : $\mathbf{A} = \mathsf{H}_\mathbf{A}(\mathsf{seed}) \in R_q^{k\times\ell}$
3 : $c' \leftarrow \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \left\lfloor \mathbf{A}\cdot\mathbf{z}^{(1)} - 2^{\nu_\mathbf{b}}\cdot c\cdot\mathbf{b}_\top \right\rceil_{\nu_\mathbf{w}} + \mathbf{h})$
4 : **if** $c = c' \wedge \left\| (\mathbf{z}^{(1)}, 2^{\nu_\mathbf{w}}\cdot\mathbf{h}) \right\| \leqslant \beta$ **then**
5 : **return** true
6 : **return** false

$\mathsf{IdentifyAbort}(\mathsf{vk}, \mathsf{aux}, \mathsf{act}, \mathsf{msg}, (\mathsf{pm}_i)_{i\in[R_\mathsf{sig}]})$

1 : Parse $\mathsf{pm}_1 = (\mathsf{cmt}_i)_{i\in\mathsf{act}}, \mathsf{pm}_2 = (\mathbf{w}_i)_{i\in\mathsf{act}}, \mathsf{pm}_3 = (\mathbf{z}_i^{(1)})_{i\in\mathsf{act}}$
2 : // Check commitments hash.
3 : $\mathsf{invalid} = \{i \in [N] \mid \mathsf{cmt}_i \neq \mathsf{H}_\mathsf{cmt}(\mathsf{vk}, i, \mathbf{w}_i)\}$
4 : **if** $\mathsf{invalid} \neq \varnothing$ **then**
5 : **return** invalid
6 : $c := \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \left\lfloor \sum_{i\in\mathsf{act}} \mathbf{w}_i \right\rceil_{\nu_\mathbf{w}})$
7 : $\nu := \mathsf{VandRecover}(\mathsf{vk}, \mathsf{aux}, \mathsf{act})$
8 : **for** $i \in \mathsf{act}$ **do**
9 : $\mathbf{b}_i^{\mathsf{part}} := \sum_{\mathsf{idx}\in\nu[i]} \nu[i][\mathsf{idx}]\cdot\mathbf{b}_i[\mathsf{idx}]$
10 : $\mathbf{z}_i^{(2)} := \left(\mathbf{A}\cdot\mathbf{z}_i^{(1)} - c\cdot\mathbf{b}_i^{\mathsf{part}}\right) - \mathbf{w}_i$
11 : **if** $\{\left\| (\mathbf{z}_i^{(1)}, \mathbf{z}_i^{(2)}) \right\| > \beta^{\mathsf{ind}}\}$ **then**
12 : $\mathsf{invalid} := \mathsf{invalid} \sqcup \{i\}$
13 : **return** invalid

Figure 5.21: Algorithms (part 2/2) for a distributed signing with IA. For conciseness, we omit aborts due to parsing errors: if an unexpected message is received, parties that produced the unexpected message are included in the set of invalid parties.

6 REJECTION-SAMPLING BASED TSS AND THRESHOLD ML-DSA

The previous chapters focused on lattice-based threshold signature schemes (TSS) built with the *noise flooding* technique, and achieving advanced properties for them. While this approach has led to efficient constructions, it results in relatively large signatures, and most importantly, it does not directly indicate how to distribute the lattice-based NIST standards FN-DSA and ML-DSA. Compatibility with standards is often crucial for practical adoption and interoperability with existing systems.

In this chapter, we address the problem of building efficient TSS leveraging *rejection sampling*, breaking the remaining barrier toward more compact lattice-based TSS. As a breakthrough application, we present the first threshold signature scheme that is *fully compatible* with ML-DSA, supporting secure and efficient signing for a few parties.

This construction is enabled by a new simulation result for rejection sampling, allowing us to simulate Fiat–Shamir commitments even when the final signature is rejected, whereas centralized security proofs typically rely on these commitments remaining hidden. Our construction leverages short secret sharing techniques and integrates optimized rejection sampling to achieve practical performance.

We implement our threshold ML-DSA construction in Go and evaluate its performance across local, LAN, and WAN network settings.

6.1 INTRODUCTION

As organizations migrate to post-quantum cryptography, they are adopting the Module-Lattice-based Digital Signature Algorithm (ML-DSA), recently finalized by NIST [Nat24]. However, this transition leaves an important gap: modern systems rely on threshold signature schemes (TS) to eliminate single points of failure and distribute trust, but there is currently no practical threshold construction for ML-DSA. This creates a trade-off between achieving quantum resistance and preserving the operational resilience provided by mature classical threshold schemes for RSA, Schnorr, and ECDSA. Systems that migrate to ML-DSA today therefore must forfeit the threshold guarantees they rely on.

The absence of a threshold ML-DSA scheme is not merely an oversight but a deep technical challenge. The core of ML-DSA's design – the *Fiat-Shamir-with-aborts* paradigm – is fundamentally at odds with multi-party computation. This paradigm relies on a rejection sampling procedure where a signature attempt is "aborted" and retried. Rejected signature candidates leak information about the secret key if revealed, and must therefore be kept secret. In a multi-party setting, this requires multiple parties to collaboratively decide whether to abort without revealing the candidate to each other, nor to an adversary. Even whether the randomness commitment used in the Fiat-Shamir-with-aborts paradigm can be safely revealed in case of rejection is an open question, and previous attempts to prove this claim relied on assumptions that do not yield competitive parameters [BTT22; DFPS23; ADP24].

More precisely, [BTT22; DFPS23] give two orthogonal proofs that the distribution of $\begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{r}$ conditioned on the rejection sampling step aborting is close to uniform.

1. The first uses a leftover-hash-lemma or regularity lemma on $\begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{r}$. However, the leftover-hash lemma does not apply to polynomial rings, and regularity lemmas for polynomial rings only work for Gaussian distributions with large standard deviations.

2. The second approach consists in reducing to the MLWE hardness over $\chi_{\mathbf{r}}$ assuming that the distribution of accepted $\mathbf{z}_i$ is at a negligible statistical distance from the ideal distribution. However, their proof relies on non-tight search-to-decision and decision-to-search LWE reductions, and cannot be exploited for setting any parameter.

Naïve protocols to securely distribute Fiat-Shamir-with-aborts signatures would require the multi-party computation of a hash function with secret inputs [CS19], which is prohibitively inefficient. The threshold solutions [DOTT22; BTT22; Che23] surveyed in Section 3.5 instead rely on revealing the input of the hash publicly while attempting to hide potential leaks from rejected commitments. Provable approaches include the reliance on trapdoor commitments [DOTT21] or the combination of several commitments together with trapdoor sampling [BTT22]. They then rely on local rejection sampling in the N-out-of-N setting to ensure that revealed partial signatures do not leak information. These constructions, however, introduce significant overhead, leaving them behind *noise flooding* based schemes in terms of efficiency and compactness, and even further away from compatibility with ML-DSA, with the state-of-the-art multisignature [Che23] generating signatures of 31 kB for 1024 users. [DKLS25] assumes heuristically that this leakage is safe, and proposes an MPC protocol for ML-DSA under this assumption, though it remains rather inefficient and assumes a trusted party to generate correlated randomness, which falls outside standard security models.

6.1.1 Our Contribution: The First Efficient Threshold ML-DSA

We introduce the first practical threshold signature scheme for the ML-DSA standard. We assume a trusted key generation[1]. It supports signing with up to 6 parties. While a threshold of six may appear small at first glance, it represents a significant step forward and is already more than sufficient for most industrial applications (which usually require 2-out-of-2 and 2-out-of-3 settings).

We resolve the main tension that prevented previous works for efficiently distributing Fiat-Shamir with Aborts signatures: the need to keep rejected commitments secret, and we also leverage short secret sharing techniques from the previous chapter to support T-out-of-N settings. Finally, we introduce ML-DSA-specific optimizations for tight parameters and practical performance.

DISTRIBUTING FIAT-SHAMIR WITH ABORTS SIGNATURES. The first major challenge we address is how to securely distribute the Fiat-Shamir-with-aborts signing process. Our key insight is that *rejected commitments can be safely revealed without leaking secret information*. This enables an efficient 3-round distributed signing protocol following the initial design of [DOTT21; BTT22], where parties compute locally a Fiat-Shamir-with-aborts commitment and a partial signature. Parties can reveal their Fiat-Shamir commitment in clear and only keep the final partial signature secret in case of rejection. This allows an aggregated commitment to be computed publicly and hashed to derive the challenge, without any parameter increase. To support this claim, we present in Lemma 6.3.2 a new proof that the distribution of commitments in case of abort remains pseudo-uniform, which is much simpler than previous attempts and directly relies on the hardness of MLWE for standard distributions.

Further, we upgrade this basic protocol to support T-out-of-N settings by leveraging short secret sharing techniques from the previous chapter.

1 While it is possible to distribute the key generation, it requires more care than in Chapter 5 as our bound analysis (c.f. Section 6.4.5) relies on the honest sampling of the shares, and we refer to the full version of this work for a discussion and a compatible distributed key generation protocol.

THRESHOLD ML-DSA. Our key contribution is a distributed protocol for ML-DSA. It is a tailored combination of (i) short replicated secret sharing, and (ii) optimized per-party rejection sampling, specifically designed for ML-DSA's unique constraints. Our protocol includes a two-stage rejection sampling: a per-party rejection, to ensure the good shape of all partial signatures, and that their reveal does not leak any secret; and a global one, at the time of combination, which ensures that the final combined signature fits in the sizes of ML-DSA signatures. Note that while the first one acts as in ML-DSA as is required for security, the second one is only there for size compatibility. Even though this idea seems simple, it requires several new pieces of technology to make it work effectively:

- **Unbalanced Hyperball Rejection Sampling.** We replace the rejection sampling technique from ML-DSA with one based on hyperballs. This significantly reduces abort probability when multiple parties must simultaneously satisfy the rejection criteria. Further, we fine-tune this new rejection sampling by using simultaneously different norms.
- **Optimized Share Reconstruction (Section 6.4.2).** The probability of signing success is directly tied to the size of the secret shares. We designed a tailored reconstruction procedure that minimizes this size by modeling the problem as a variant of the B-matching problem, which we solve efficiently using a maximum flow algorithm.
- **Parallel Protocol Instances.** To ensure high signing success rates at reconstruction time, our protocol runs multiple instances in parallel. This approach guarantees fast and reliable signing without compromising security.

As this is not the main focus of this chapter, we present only a distributed signing process, and refer to the published version of this work [CPEN+26] for a discussion and a compatible distributed key generation protocol.

FORMAL SECURITY (SECTION 6.4.3) We provide a full formal analysis of our scheme, showing that it preserves the security guarantees of ML-DSA.

PRACTICAL DEPLOYMENT (SECTION 6.5) We implement our scheme in Go and benchmark its performance in local settings. Our experiments show that a signing attempt completes in milliseconds in local deployments. Average communication costs to produce a signature range from 21 kB to 1050 kB per party depending on the threshold (T, N) and up to 6 parties, which is competitive with established classical TS. Crucially, the output signatures are fully compliant with the standard ML-DSA format, enabling drop-in integration with existing systems.

COMPARISON WITH OTHER WORKS ON THRESHOLD ML-DSA. Several works pursued direct thresholdization of ML-DSA, through generic MPC techniques to address its rejection sampling behavior, as summarized in Table 6.1. First, the generic MPC-based approach of Cozzo and Smart [CS19] is secure in theory but suffers from high computational and communication costs, making practical deployment infeasible. The "Trilithium" scheme by Dufka et al. [DKLS25] supports two-party signing, but it assumes a *trusted party* to generate correlated randomness: an assumption that falls outside standard adversarial models and incurs high communication overhead. Moreover, its security analysis lacks a full proof of unforgeability, relying instead on heuristic assumptions about the distribution of rejected ML-DSA transcripts. Most recently, and concurrently with this work, Bienstock et al. [BCEP+25] proposed a threshold ML-DSA variant using MPC under an honest-majority assumption. Their construction achieves either 78 rounds with 705 kB of *online* communication per party or 23 rounds with 1.5 MB, depending on parameter choices (c.f., [BCEP+25, Table 5]). To this must be added the cost of the offline phase (between 136 and 37 rounds on average), which in particular includes correlated randomness generation, which is known to be expensive computational and/or communication-wise. We note

Table 6.1: Comparison of approaches to thresholdize ML-DSA-44, with per party metrics averaged over successful signing attempts.

Scheme	# Parties (bound)	# Rounds Off + Online	Communication Off + Online (MB)	Computation	Model	Security
This work	6	2+4	10^{-5} + 0.021 to 1.05	Lightweight	Game-based	Dishonest majority
Bienstock et al. [BCEP+25]	∞	136* + 79 37* + 23	> 0.87* + 0.70 > 1.9* + 1.5	Online lightweight*	UC	Honest majority
Trilithium [DKLS25]	2	1†+60	234† + 0.40	Heavy	UC	Trusted party†
Generic MPC [CS19]	∞	High	High	Impractical	UC	Dishonest majority

Rounds and communication costs are reported *as per-party averages*, measured over successful signing attempts for threshold Threshold ML-DSA-44. These reflect the impact of ML-DSA's inherent probabilistic abort mechanism. For [BCEP+25], we include the costs for 5 parties, with 1 and 8 parallel protocol executions. *Offline costs for [BCEP+25] do not include the generation of correlated randomness (e.g., Beaver triples, edaBits, etc.). †Requires 234 MB of correlated randomness to be generated by a trusted party.

that [BCEP+25] leverages the proof techniques we introduce in this work to show that rejected commitments can be safely revealed, but they still rely on a costly MPC protocol to securely compute the partial signatures, which results in high communication and round complexity.

Our scheme departs from generic MPC paradigms by tailoring the design to ML-DSA's design and leveraging new insights for distributing Fiat-Shamir with Aborts signatures. It supports up to 6 parties in the strongest adversarial model (dishonest majority), retains compatibility, and achieves practical efficiency.

6.2 PRELIMINARIES

6.2.1 Additional Notations

SET AND DISTRIBUTIONS. We recall that the union $S \cup T$ of two sets is denoted by $S \sqcup T$ when disjoint. For a real parameter $0 \leqslant p \leqslant 1$, the *Bernoulli distribution* $\mathsf{Ber}(p)$ outputs 1 with probability p and 0 otherwise. The *Binomial distribution* $\mathsf{Bin}(m, p)$ is defined as the sum of m independent Bernoulli trials sampled from $\mathsf{Ber}(p)$.

6.2.2 Threshold Signatures

We reuse the syntax and definitions from Section 3.4. In particular, we will prove the unforgeability of our Threshold ML-DSA scheme in the asynchronous communication model.

However, in this chapter, we focus on Fiat-Shamir with Aborts (FSwA) signature schemes. In the threshold setting, this translates to having signature protocols that may *abort* with some probability even when all parties behave honestly.

Hence, the standard notion of *correctness* is unfit for such schemes. Instead, we consider a notion of *correctness* tolerating a certain probability of abort. We use the same game as in Fig. 3.2, but we tolerate some probability $1 - p$ that it returns $\perp$, e.g. that the parties decide to abort the signing protocol.

Definition 6.2.1 (Correctness of TS with aborts). *We say that a* TS *with aborts p-terminates if:*

$$\Pr\left[\mathsf{Game}_{\mathsf{TS}}^{\text{TS-CORR}}(1^\lambda, N, T, \mathsf{act}, \mathsf{msg}) \neq \perp\right] \geqslant p \tag{1}$$

$$\Pr\left[\mathsf{Game}_{\mathsf{TS}}^{\text{TS-CORR}}(1^\lambda, N, T, \mathsf{act}, \mathsf{msg}) = 0\right] = \mathsf{negl}(\lambda) \tag{2}$$

where $\mathsf{Game}_{\mathsf{TS}}^{\text{TS-CORR}}$ *is defined in Figure 3.2.*

6.2.3 Modulus Rounding in ML-DSA

We recall helper functions over $\mathbb{Z}_q$, used in ML-DSA. These functions are applied coefficient-wise when applied to polynomials in R_q to optimize the sizes of keys and signatures.

$\mathsf{Power2Round}_q(r)$: rounds an integer $r \in \mathbb{Z}_q$. It outputs a pair (r_0, r_1), where $r_0 = r \bmod^{\pm} 2^d$ and $r_1 = (r - r_0)/2^d$.

$\mathsf{HighBits}(r, \alpha)$: returns the high-order bits r_1 of r, defined as $\mathsf{HighBits}(r, \alpha) = \frac{r - (r \bmod^{\pm} \alpha)}{\alpha}$ if $r - (r \bmod^{\pm} \alpha) \neq q - 1$ and 0 otherwise.

$\mathsf{MakeHint}_q(z, r, \alpha)$: produces a binary hint $h \in \{0, 1\}$ indicating that the high bits of z and $z + r$ differ. The hint mechanism ensures the correct reconstruction of high bits during verification.

$\mathsf{UseHint}_q(h, r, \alpha)$: takes a hint bit $h \in \{0, 1\}$ and an integer $r \in \mathbb{Z}_q$. Based on the hint, it corrects the computation of $\mathsf{HighBits}(r, 2\gamma_2)$. The goal is to recover $\mathsf{HighBits}(r + z, 2\gamma_2)$. If $h = 1$, a carry occurred, and the high bits need adjustment: the high bits by $+1$ if $r - \alpha \cdot r_1 > 0$, and otherwise by -1.

Lemma 6.2.2 ([KLS18]). *For $q, \alpha > 0$ with $q > 2\alpha$, $q = 1 \bmod \alpha$ and α is even. Let $\mathbf{r}, \mathbf{z} \in R_q$ where $\|\mathbf{z}\|_\infty \leqslant \alpha/2$. Then:*

$$\mathsf{UseHint}_q(\mathsf{MakeHint}_q(\mathbf{z}, \mathbf{r}, \alpha), \mathbf{r}, \alpha) = \mathsf{HighBits}_q(\mathbf{r} + \mathbf{z}, \alpha).$$

6.2.4 The Rényi Divergence

Following Devevey et al. [DFPS22], we rely on the *Rényi divergence of infinite order* R_∞ as well as the *smooth Rényi divergence* R_∞^ε to measure the distance between two distributions.

Definition 6.2.3. *Let $\mathcal{P}, \mathcal{Q}$ two distributions such that $\mathcal{P}$ is absolutely continuous with respect to $\mathcal{Q}$. The Rényi divergence of infinite order is $R_\infty(\mathcal{P}||\mathcal{Q}) = \mathsf{ess\ sup} \frac{\partial \mathcal{P}}{\partial \mathcal{Q}}(x)$.*

We additionally recall a relaxed version of the Rényi divergence from [DFPS22, Def. 2.1], called *smooth Rényi divergence*, where one can remove a few problematic points from the support, including those that may lie in $\mathsf{Supp}(\mathcal{P}) \backslash \mathsf{Supp}(\mathcal{Q})$.

Definition 6.2.4. *Let $\varepsilon > 0$. Let $\mathcal{P}, \mathcal{Q}$ two probability distributions such that $\int_{\mathsf{Supp}(\mathcal{Q})} \mathcal{P}(x) \partial\mu(x) \geqslant 1 - \varepsilon$ for the measure μ. Their ε-smooth Rényi divergence is $R_\infty^\varepsilon(\mathcal{P}||\mathcal{Q}) = \inf\{M > 0 \mid \Pr_{x \leftarrow \mathcal{P}}(\mathcal{P}(x) \leqslant M\mathcal{Q}(x)) \geqslant 1 - \varepsilon\}$.*

Lemma 6.2.5 ([BLLS+15] and [HPRR20]). *For two distributions $\mathcal{P}, \mathcal{Q}$, the Rényi divergence satisfies the following properties:*

- ***Data processing inequality.*** *$R_\infty(\mathcal{P}^f||\mathcal{Q}^f) \leqslant R_\infty(\mathcal{P}||\mathcal{Q})$ for any function f, where $\mathcal{P}^f$ (resp. $\mathcal{Q}^f$) is the distribution of $f(y)$ induced by sampling $y \leftarrow \mathcal{P}$ (resp. $y \leftarrow \mathcal{Q}$).*

- ***Multiplicativity.*** *Assume that $\mathcal{P}$ and $\mathcal{Q}$ are the joint distributions of random variables $(x_i)_i$. Note $\mathcal{P}_{i,|x_{<i}=v}$, $\mathcal{Q}_{i,|x_{<i}=v}$, the conditional distributions of x_i given $x_{<i} = v_{<i}$. Assume $R_\infty(\mathcal{P}_{i,|x_{<i}=v_{<i}} || \mathcal{Q}_{i,|x_{<i}=v_{<i}}) \leqslant r_i$ for any possible $v_{<i}$. Then, $R_\infty(\mathcal{P}||\mathcal{Q}) \leqslant \prod_i r_i$.*
- ***Probability preservation.*** *for any event $E \subseteq \mathsf{Supp}(\mathcal{Q})$, $\mathcal{Q}(E) \geqslant \mathcal{P}(E)/R_\infty(\mathcal{P}||\mathcal{Q})$.*

6.2.5 Rejection Sampling

ML-DSA is a Fiat-Shamir with Aborts signature scheme. It heavily relies on the rejection sampling technique introduced by Lyubashevsky [Lyu09], which we recall with slightly different notations.

$\mathsf{Rej}(\mathbf{v}, \chi_{\mathbf{z}}, \chi_{\mathbf{r}}, M) \to \mathbf{z} \in R^{\ell+k} \cup \{\perp\}$

1 : $\mathbf{r} \leftarrow \chi_{\mathbf{r}}$
2 : $\mathbf{z} := \mathbf{v} + \mathbf{r}$
3 : $b \leftarrow \mathsf{Ber}\left(\min\left(\frac{\chi_{\mathbf{z}}(\mathbf{z})}{M\chi_{\mathbf{r}}(\mathbf{r})}, 1\right)\right)$
4 : **if** $b = 0$ **then** $\mathbf{z} = \perp$
5 : **return** $\mathbf{z}$

$\mathsf{Ideal}(\chi_{\mathbf{z}}, M) \to \mathbf{z} \in R^{\ell+k} \cup \{\perp\}$

1 : $\mathbf{z} \leftarrow \chi_{\mathbf{z}}$
2 : $b \leftarrow \mathsf{Ber}\left(\frac{1}{M}\right)$
3 : **if** $b = 0$ **then** $\mathbf{z} = \perp$
4 : **return** $\mathbf{z}$

Figure 6.1: Rejection algorithm and ideal algorithm.

We extend the algorithm Rej to $\mathsf{Rej}(\mathbf{v}, \chi_{\mathbf{z}}, \chi_{\mathbf{r}}, M; \mathbf{r})$ to parse the randomness $\mathbf{r} \leftarrow \chi_{\mathbf{r}}$ as an input. We will also write $(\mathbf{z}|\mathsf{acc})_{\mathbf{v}}$ to denote the distribution of $\mathbf{z} := \mathbf{v} + \mathbf{r}$ conditioned on $b = 1$, and $(\mathbf{z}|\mathsf{rej})_{\mathbf{v}}$ to denote the distribution of $\mathbf{z} := \mathbf{v} + \mathbf{r}$ conditioned on $b = 0$ (note that in that case $\mathbf{z} = \perp$ but the distribution we are interested in is the one of $\mathbf{v} + \mathbf{r}$ so we abuse this notation).

Rejection sampling aims to map a distribution $\mathbf{v} + \chi_{\mathbf{r}}$ – that depends on the sensitive value $\mathbf{v}$ – to a distribution $\chi_{\mathbf{z}}$ that does not. We recall the following lemma that bounds the Rényi divergence between the real and ideal distributions above.

Lemma 6.2.6 (Lemma 4.1, [DFPS22]). *Let $M > 1$ and $\varepsilon < 1$ such that $R^{\varepsilon}_{\infty}(\chi_{\mathbf{z}}||\mathbf{v} + \chi_{\mathbf{r}}) \leqslant M$. Then, $R_\infty(\mathsf{Rej}||\mathsf{Ideal}) \leqslant 1 + \frac{\varepsilon}{M-1}$.*

6.2.6 Rejection Sampling over Hyperballs

Devevey et al. [DFPS22] showed that hyperball-based rejection sampling achieves the most compact parameters, and outperforms uniforms [Lyu09] and polytopes as used by Bambury et al. [BBRS24]. Hyperballs achieve the same compactness as using Gaussians, but have the advantage of a simpler rejection condition in the form of a norm two bound check.

We extend the polynomial ring used in this thesis to the real numbers $R_{\mathbb{R}} = \mathbb{R}[X]/(X^n + 1)$, and define hyperballs over $R_{\mathbb{R}}$ as follows.

Definition 6.2.7 (Continuous hyperball). *We denote:*

$$\mathcal{B}_{R,\ell}(r, \mathbf{c}) = \left\{\mathbf{x} \in R^{\ell}_{\mathbb{R}} \mid \|x - \mathbf{c}\|_2 \leqslant r\right\}$$

the continuous hyperball with center $\mathbf{c} \in R_{\mathbb{R}}^{\ell}$ *and radius* $r > 0$ *in dimension* $\ell > 0$. *When* $\mathbf{c} = 0$, *we omit it.*

To bound the Rényi divergence we rely on the followings.

Definition 6.2.8 (Regularized Incomplete Beta Function). *The incomplete beta function is defined over* $[0,1] \times \mathbb{R}^+ \times \mathbb{R}^+$ *as* $B : (x;a,b) \mapsto \int_0^x t^{a-1}(1-t)^{b-1}\partial t$. *We also define* $I_x(a,b) = B(x;a,b)/B(1;a,b)$.

Lemma 6.2.9 ([DFPS22, Lemma 5.1]). *Let* $\ell \geqslant 1$ *and* $\mathbf{v} \in \mathbb{R}^{n\ell}$. *Let* $\varepsilon \in [0,1/2)$ *and* $\phi \geqslant 1$ *be such that* $2\varepsilon = I_{1-1/\phi^2}\left(\frac{n\ell+1}{2}, \frac{1}{2}\right)$. *Let* $r, r' > 0$ *such that* $r'^2 \geqslant r^2 + \|\mathbf{v}\|_2^2 + 2r\|\mathbf{v}\|_2/\phi$. *It holds that:*

$$R_{\infty}^{\varepsilon}\left(\mathcal{U}(\mathcal{B}_{R,\ell}(r)) \| \mathcal{U}(\mathcal{B}_{R,\ell}(r', \mathbf{v}))\right) = \left(\frac{r'}{r}\right)^{n\ell} \tag{3}$$

Let $M > 1$. *If* $r \geqslant \|\mathbf{v}\|_2 \cdot \frac{\frac{1}{\phi} + \sqrt{\frac{1}{\phi^2} + M^{2/(n\ell)} - 1}}{M^{2/(n\ell)} - 1}$ *and* $r' = M^{1/(n\ell)} r$, *then the value in Eq. (3) is upper-bounded by* M.

Lemma 6.2.10 (from [DFPS22, Section A.6]). *For* $n > 1, \phi > 1$, *we have* $I_{1-1/\phi^2}(\frac{n+1}{2}, \frac{1}{2}) < \left(1 - \frac{1}{\phi^2}\right)^{n-1} \cdot n \cdot \left(1 - \frac{1}{\phi}\right)$.

We describe in Fig. 6.1 the imbalanced rejection sampling that we will use for Threshold-ML-DSA.

$\mathsf{HRej}(\mathbf{v}, r, r', \nu, M) \to \mathbf{z} \in R^{\ell+k} \cup \{\perp\}$

1 : $\mathbf{r} \xleftarrow{\$} \mathcal{B}_{R,\ell+k}(r')$
2 : $(\mathbf{v}^{(1)}, \mathbf{v}^{(2)}) := \mathbf{v} \in R_{\mathbb{R}}^{\ell} \times R_{\mathbb{R}}^{k}$
3 : $\mathbf{z} := (\mathbf{v}^{(1)} / \nu, \mathbf{v}^{(2)}) + \mathbf{r}$
4 : **if** $\|\mathbf{z}\| > r$ **then** $\mathbf{z} = \perp$
5 : $(\mathbf{z}^{(1)}, \mathbf{z}^{(2)}) := \mathbf{z} \in R_{\mathbb{R}}^{\ell} \times R_{\mathbb{R}}^{k}$
6 : **return** $\lfloor \mathbf{z}^{(1)} \cdot \nu, \mathbf{z}^{(2)} \rceil \in R^{\ell+k}$

Figure 6.2: Imbalanced rejection sampling over hyperballs. Major differences with Rej (Fig. 6.1) are highlighted.

6.2.7 Σ-protocols with Aborts

We recall the notion of Σ-protocols with aborts that are often used to model rejection sampling based protocols, and its associated zero-knowledge properties. Although we don't use them to formalize our threshold signature, these notions can provide a broader understanding of the potential applications of our simulation result Lemma 6.3.2.

Definition 6.2.11 (Σ-protocol with aborts). *Let* $\mathcal{X}, \mathcal{Y}$ *be two finite sets. A* Σ*-protocol for a relation* $R \subset \mathcal{X} \times \mathcal{Y}$ *with commitment set* $\mathcal{W}$, *challenge set* $\mathcal{C}$ *and response set* $\mathcal{Z}$ *is a 3-round interactive proof system between a prover written as* $\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2)$ *and a verifier* $\mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2)$ *with the following specifications:*

- $\mathsf{P}_1 : (x, y) \to (w, \mathsf{st})$ *is a PPT algorithm that takes as input a pair of strings in* $\mathcal{X} \times \mathcal{Y}$ *and outputs a commitment* $w \in \mathcal{W}$ *and a state* $\mathsf{st} \in \{0,1\}^*$
- $\mathsf{V}_1 : (x, w) \to c$ *is a PPT algorithm that takes as inputs a string* $x \in \mathcal{X}$ *and a commitment* $w \in \mathcal{W}$ *and outputs a challenge* $c \in \mathcal{C}$

- $\mathsf{P}_2 : (x, y, w, c, \mathsf{st}) \to z$ *is a PPT algorithm that takes as inputs a pair of strings in* $\mathcal{X} \times \mathcal{Y}$*, a commitment* $w \in \mathcal{W}$*, a challenge* $c \in \mathcal{C}$*, and a state* st *and outputs a response* $z \in \mathcal{Z} \cup \{\perp\}$ *(we say that* P_2 *aborts if it outputs* $\perp$*);*
- $\mathsf{V}_2 : (x, w, c, z) \to b \in \{0, 1\}$ *is a deterministic polynomial-time algorithm that takes as inputs a string* $x \in \mathcal{X}$ *, a commitment* $w \in \mathcal{W}$*, a challenge* $c \in \mathcal{C}$*, and a response* $z \in \mathcal{Z}$ *and outputs a bit* b *which represents acceptance or rejection; in the case that* $z = \perp$*, it returns* o*.*

The literature considers two differents notions of zero-knowledge for Σ-protocols with aborts, depending on whether aborted transcripts need to be simulated: Honest-Verifier Zero-Knowledge (HVZK), and its counterpart no-abort Honest-Verifier Zero-Knowledge (naHVZK).

Definition 6.2.12 (Computational Honest-Verifier Zero-Knowledge). *A* Σ*-protocol* $((\mathsf{P}_1, \mathsf{P}_2), (\mathsf{V}_1, \mathsf{V}_2))$ *for a relation* R *is HVZK if there exists a PPT simulator* Sim *such that for all PPT algorithm* $\mathcal{A}$ *and all* $(x, y) \in R$*, the following advantage is negligible:*

$$\mathsf{Adv}(\mathcal{A}) := \left| \Pr\left[\mathcal{A}((w,c,z),y) = 1 \;\middle|\; \begin{array}{c} (w, \mathsf{st}) \leftarrow \mathsf{P}_1(x,y) \\ c \leftarrow \mathsf{V}_1(x,w) \\ z \leftarrow \mathsf{P}_2(x,y,c,w,\mathsf{st}) \end{array} \right] - \Pr\left[\mathcal{A}((w,c,z),y) = 1 \middle| (w,c,z) \leftarrow \mathsf{Sim}(x) \right] \right|$$

Definition 6.2.13 (Computational no-abort Honest-Verifier Zero-Knowledge). *A* Σ*-protocol for a relation* R *is naHVZK if there exists a PPT simulator* Sim *such that for all PPT algorithm* $\mathcal{A}$ *and all* $(x, y) \in R$*, the distributions of simulated transcripts* (w, c, z) *and* y *are indistinguishable from the real distribution produced by* Σ*, conditioned on the event* $z \neq \perp$*.*

6.3 SIMULATING REJECTED TRANSCRIPTS IN FIAT-SHAMIR WITH ABORTS

In this section, we introduce a core result for distributing signature schemes based on the Fiat-Shamir with Aborts paradigm, namely that commitments can be safely revealed, even in case of rejected signatures, allowing the safe derivation of the Fiat-Shamir challenge in clear.

We start by recalling a high-level description of the Fiat-Shamir with Aborts technique to construct signatures. Let χ_{sk}, $\chi_{\mathbf{r}}$, $\chi_{\mathbf{z}}$ be three distributions over $R^{\ell+k}$ for some ring R. The secret key for our signature scheme will be $\mathbf{s} \leftarrow \chi_{\mathsf{sk}}$ and the corresponding public key will be the MLWE sample: $\mathbf{b} = \begin{bmatrix} \mathbf{A} & \mathbf{I} \end{bmatrix} \cdot \mathbf{s}$.

To sign a message msg the signer proceeds as follows:

- Sample $\mathbf{r} \leftarrow \chi_{\mathbf{r}}$, and computes a commitment $\mathbf{w} = \begin{bmatrix} \mathbf{A} & \mathbf{I} \end{bmatrix} \cdot \mathbf{r}$.
- Derive a challenge $c = \mathsf{H}_c(\mathsf{vk}, \mathsf{msg}, \mathbf{w})$.
- Compute the response $\mathbf{z} = c\mathbf{s} + \mathbf{r}$.
- Output $(c, \mathbf{z})$ with probability $\min\left(\chi_{\mathbf{z}}(\mathbf{z}) / (M\chi_{\mathbf{r}}(\mathbf{r}), 1)\right)$ (for some $M > 1$)

The last step of this algorithm corresponds to the rejection sampling recalled in Section 6.2.5, which guarantees that the distribution output by the signer will be close to the ideal distribution $\chi_{\mathbf{z}}$, at the cost of rejecting with probability $1 - 1/M$.

This high-level overview can be seen as a non-optimized version of the ML-DSA signature standard [DKLL+18], without the use of rounding and hints, and where the commitment $\mathbf{w}$ includes an error term used to mask the first coordinate of the response $\mathbf{z}$ (while ML-DSA ensures security with a custom rejection sampling that makes use of the lower bits of $\mathbf{w}$).

In the centralized setting, we can mostly[2] focus on simulating *accepted* transcripts $(c, \mathbf{z})$, since *rejected* transcripts are only computed locally, and can be dropped.

However, in the distributed setting, it is of strong interest to be able to reveal $\mathbf{w}$ even in case the signature is rejected, so as to allow parties to compute the challenge c in clear. We will show that this is indeed possible, and that revealing $\mathbf{w}$ does not endanger the security of the scheme.

We start by bounding the rejection probability in the Fiat-Shamir with Aborts paradigm, and introduce a bound on the statistical distance between the real and ideal target distributions.

Lemma 6.3.1. *Let any $M > 1$, $B > 0, \varepsilon < 1$, $\mathbf{v} \in R^{\ell+k}$, distributions $\chi_{\mathbf{r}}$, $\chi_{\mathbf{z}}$ over $R^{\ell+k}$ such that $R^{\varepsilon}_{\infty}(\chi_{\mathbf{z}} || \chi_{\mathbf{r}} + \mathbf{v}) \leqslant M$. We have:*

$$\frac{1-\varepsilon}{M} \leqslant \Pr\left[\mathsf{Rej}(\mathbf{v}, \chi_{\mathbf{r}}, \chi_{\mathsf{sk}}, M) \neq \perp\right] \leqslant \frac{1}{M}$$
$$\delta \leqslant \frac{2\varepsilon}{1-\varepsilon}$$

where δ is the statistical distance between $(\mathbf{z}|\mathrm{acc})$ and $\chi_{\mathbf{z}}$.

Proof. Let S be the subset of $R^{\ell+k}$ that includes all $\mathbf{x}$ such that $\chi_{\mathbf{z}}(x) > M[\chi_{\mathbf{r}} + \mathbf{v}](x)$ (or equivalently, $\chi_{\mathbf{z}}(x) > M\chi_{\mathbf{r}}(x - \mathbf{v})$). By definition of the smooth Renyi divergence, $0 \leqslant \varepsilon(\mathbf{v}) := \Pr_{x \leftarrow \chi_{\mathbf{z}}}(x \in S) \leqslant \varepsilon$ (c.f. Definition 6.2.4).

We prove the first inequality:

$$\begin{aligned}
\Pr\left[\mathbf{z} \neq \perp\right] &= \sum_{\mathbf{x} \in S} \Pr\left[\mathrm{acc}|\mathbf{z} = \mathbf{x}\right] \Pr\left[\mathbf{z} = \mathbf{x}\right] + \sum_{\mathbf{x} \in R^{\ell+k} \setminus S} \Pr\left[\mathrm{acc}|\mathbf{z} = \mathbf{x}\right] \Pr\left[\mathbf{z} = \mathbf{x}\right] \\
&= \sum_{\mathbf{x} \in S} 1 \cdot \chi_{\mathbf{r}}(\mathbf{x} - \mathbf{v}) + \sum_{\mathbf{x} \in R^{\ell+k} \setminus S} \frac{\chi_{\mathbf{z}}(\mathbf{x})}{M\chi_{\mathbf{r}}(\mathbf{x} - \mathbf{v})} \chi_{\mathbf{r}}(\mathbf{x} - \mathbf{v}) \\
&= \sum_{\mathbf{x} \in S} \left(\chi_{\mathbf{r}}(\mathbf{x} - \mathbf{v}) - \frac{\chi_{\mathbf{z}}(\mathbf{x})}{M}\right) + \sum_{\mathbf{x} \in R^{\ell+k}} \frac{\chi_{\mathbf{z}}(\mathbf{x})}{M} \\
&= \frac{1 - \varepsilon(\mathbf{v})}{M}
\end{aligned}$$

We now derive the explicit distribution of $\mathbf{z}|\mathrm{acc}$:

$$\begin{aligned}
\forall \mathbf{x} \notin S; \quad \Pr\left[\mathbf{z} = \mathbf{x}|\mathrm{acc}\right] &= \tfrac{\Pr[\mathrm{acc}|\mathbf{z}=\mathbf{x}] \cdot \Pr[\mathbf{z}=\mathbf{x}]}{\Pr[\mathrm{acc}]} \\
&= \tfrac{\chi_{\mathbf{z}}(\mathbf{x})}{M\chi_{\mathbf{r}}(\mathbf{x}-\mathbf{v})} \chi_{\mathbf{r}}(\mathbf{x} - \mathbf{v}) \cdot \tfrac{M}{1-\varepsilon(\mathbf{v})} \\
&= \tfrac{\chi_{\mathbf{z}}(\mathbf{x})}{1-\varepsilon(\mathbf{v})} \\
\forall \mathbf{x} \in S; \quad \Pr\left[\mathbf{z} = \mathbf{x}|\mathrm{acc}\right] &= \tfrac{\Pr[\mathrm{acc}|\mathbf{z}=\mathbf{x}] \cdot \Pr[\mathbf{z}=\mathbf{x}]}{\Pr[\mathrm{acc}]} \\
&= 1 \cdot \chi_{\mathbf{r}}(\mathbf{x} - \mathbf{v}) \cdot \tfrac{M}{1-\varepsilon(\mathbf{v})} \\
&= \tfrac{M\chi_{\mathbf{r}}(\mathbf{x}-\mathbf{v})}{1-\varepsilon(\mathbf{v})}
\end{aligned}$$

From this we can get the second inequality:

$$\begin{aligned}
\Delta((\mathbf{z}|\mathrm{acc}), \chi_{\mathbf{z}}) &= \sum_{S} |\Pr\left[\mathbf{z} = \mathbf{x}|\mathrm{acc}\right] - \chi_{\mathbf{z}}(\mathbf{x})| + \sum_{R^{\ell+k} \setminus S} |\Pr\left[\mathbf{z} = \mathbf{x}|\mathrm{acc}\right] - \chi_{\mathbf{z}}(\mathbf{x})| \\
&= \sum_{S} \left|\frac{M\chi_{\mathbf{r}}(\mathbf{x} - \mathbf{v})}{1 - \varepsilon(\mathbf{v})} - \chi_{\mathbf{z}}(\mathbf{x})\right| + \sum_{R^{\ell+k} \setminus S} \left|\frac{\chi_{\mathbf{z}}(\mathbf{x})}{1 - \varepsilon(\mathbf{v})} - \chi_{\mathbf{z}}(\mathbf{x})\right| \\
&\leqslant \frac{1}{1 - \varepsilon(\mathbf{v})} \cdot \left(\sum_{S} |M\chi_{\mathbf{r}}(\mathbf{x} - \mathbf{v}) - \chi_{\mathbf{z}}(\mathbf{x})| + \varepsilon(\mathbf{v}) \sum_{R^{\ell+k}} |\chi_{\mathbf{z}}(\mathbf{x})|\right) \\
&\leqslant \frac{2\varepsilon}{1 - \varepsilon}
\end{aligned}$$

2 The security proof in the centralized setting is actually concretized with subtle arguments, as in the Random Oracle Model (ROM) there is still some outside observation of rejected transcripts. Nonetheless, it can be proven with weaker properties.

□

We now move on to the main result of this section, which states that the commitment $\mathbf{w}$ conditioned on rejection is computationally indistinguishable from uniform, under two MLWE assumptions on $\chi_{\mathbf{r}}$ and $\chi_{\mathbf{z}}$.

Lemma 6.3.2. *Given a secret* $\mathbf{s}$*, and challenge* c*, note* $(\mathbf{z}|\text{rej})_{\mathbf{v}=c\cdot\mathbf{s}}$ *the distribution of rejected vectors* $\mathbf{z}_i$ *conditioned on* $\mathbf{v} = c \cdot \mathbf{s}$*. Let* $\mathcal{A}$ *be an adversary against the* $\mathsf{MLWE}_{R_q,k,\ell,(\mathbf{z}|\text{rej})_{\mathbf{v}=c\cdot\mathbf{s}}}$ *problem. There exist adversaries* $\mathcal{B}_1, \mathcal{B}_2$ *respectively against the* $\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{r}}}$ *and the* $\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{z}}}$ *problems, running in time* $\mathsf{Time}(\mathcal{B}_1) \approx \mathsf{Time}(\mathcal{B}_2) \approx \mathsf{Time}(\mathcal{A})$ *such that:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{MLWE}_{R_q,k,\ell,(\mathbf{z}|\text{rej})_{\mathbf{v}=c\cdot\mathbf{s}}}} \leqslant \frac{1}{p} \cdot \left(\mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{r}}}} + \mathsf{Adv}_{\mathcal{B}_2}^{\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{z}}}} + \delta\right)$$

where $p = \Pr\left[\mathsf{Rej}(c\mathbf{s}, \chi_{\mathbf{r}}, \chi_{\mathsf{sk}}, M; \mathbf{r} \leftarrow \chi_{\mathbf{r}}) = \perp\right]$ *is the probability of rejection of* $\mathbf{z}$ *conditioned on* $c \cdot \mathbf{s}$ *and* δ *is the statistical distance between* $(\mathbf{z}|\text{acc})_{\mathbf{v}=c\cdot\mathbf{s}}$ *and the target distribution* $\chi_{\mathbf{z}}$*.*

Note that the above statement is equivalent to stating that $\mathbf{w}$ *conditioned on rejection is computationally indistinguishable from uniform, as* $\mathbf{w} = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{z} - c \cdot \mathbf{b}$*.*

Applying Lemma 6.3.1, if $R_\infty^\varepsilon(\chi_{\mathbf{z}}||\chi_{\mathbf{r}} + c \cdot \mathbf{s}) \leqslant M$*, then* $p \geqslant 1 - \frac{1}{M}$*.*

Remark 6.3.3. Although we don't use this formalism here, the FSwA paradigm can be generically captured by the notion of Σ-protocol with aborts (Definition 6.2.11), see for instance [DFPS23, Fig. 3]. The above Lemma 6.3.2 allows to tightly prove the stronger notion of HVZK (Definition 6.2.12), instead of naHVZK (Definition 6.2.13), i.e. aborted transcripts can safely be revealed in this Σ-protocol with aborts.

Proof. Let $\mathcal{A}$ be an adversary against the $\mathsf{MLWE}_{R_q,k,\ell,(\mathbf{z}|\text{rej})_{\mathbf{v}=c\cdot\mathbf{s}}}$ problem.

In the rest of this proof, we will note $\delta_{\mathcal{A}}^{\chi} := \Pr\left[1 \leftarrow \mathcal{A}\left(\mathbf{A}, \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{r}\right) \mid \mathbf{r} \leftarrow \chi\right]$ and $\gamma_{\mathcal{A}} := \Pr\left[1 \leftarrow \mathcal{A}\left(\mathbf{A}, \mathbf{u}\right) \mid \mathbf{u} \leftarrow \mathcal{U}(R_q^k)\right]$ for conciseness.

Let $\mathcal{B}_1 := \mathcal{A}$ be an adversary against $\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{r}}+c\cdot\mathbf{s}}$.

Recall the definition of the MLWE advantage of $\mathcal{B}_1$ for the distribution $\chi_{\mathbf{r}} + c \cdot \mathbf{s}$ with the above notations:

$$\mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{r}}+c\cdot\mathbf{s}}} = \left|\delta_{\mathcal{B}_1}^{\chi_{\mathbf{r}}} - \gamma_{\mathcal{B}_1}\right|$$

Interestingly, we can equivalently express $\chi_{\mathbf{r}} + c \cdot \mathbf{s}$ as a process involving the distributions $(\mathbf{z}|\text{rej})_{\mathbf{v}=c\cdot\mathbf{s}}$ and $(\mathbf{z}|\text{acc})_{\mathbf{v}=c\cdot\mathbf{s}}$. Noting p the probability of rejection of $\mathbf{z}$ conditioned on $\mathbf{v} = c \cdot \mathbf{s}$, $\chi_{\mathbf{r}}$ has the same distribution as:

- Sample b: 1 with probability p, 0 otherwise.
- If $b = 1$, sample and return $\mathbf{z} \leftarrow (\mathbf{z}|\text{rej})_{\mathbf{v}=c\cdot\mathbf{s}}$.
- If $v = 0$, sample and return $\mathbf{z} \leftarrow (\mathbf{z}|\text{acc})_{\mathbf{v}=c\cdot\mathbf{s}}$.

We can thus rewrite

$$\delta_{\mathcal{B}_1}^{\chi_{\mathbf{r}}+c\cdot\mathbf{s}} = p \cdot \delta_{\mathcal{B}_1}^{(\mathbf{z}|\text{rej})_{\mathbf{v}=c\cdot\mathbf{s}}} + (1-p) \cdot \delta_{\mathcal{B}_1}^{(\mathbf{z}|\text{acc})_{\mathbf{v}=c\cdot\mathbf{s}}}$$

At this point, we can define an adversary $\mathcal{B}_2$ that on inputs $(\mathbf{A}, \mathbf{u})$ calls $\mathcal{B}_1$ with $(\mathbf{A}, \mathbf{u} - c \cdot \begin{bmatrix}\mathbf{I}_k & \mathbf{A}\end{bmatrix} \cdot \mathbf{s})$, and returns the response. Then,

$$\delta_{\mathcal{B}_1}^{\chi_{\mathbf{r}}-c\cdot\mathbf{s}} = \delta_{\mathcal{B}_2}^{\chi_{\mathbf{r}}}$$

We additionally have $\gamma_{\mathcal{B}_1} = \gamma_{\mathcal{B}_2}$, so we can finally rewrite:

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{B}_2}^{\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{r}}}} &= \left|p \cdot (\delta_{\mathcal{B}_1}^{(\mathbf{z}|\text{rej})_{\mathbf{v}=c\cdot\mathbf{s}}} - \gamma_{\mathcal{B}_1}) + (1-p) \cdot (\delta_{\mathcal{B}_1}^{(\mathbf{z}|\text{acc})_{\mathbf{v}=c\cdot\mathbf{s}}} - \gamma_{\mathcal{B}_1})\right| \\
&\geqslant p \cdot \left|\delta_{\mathcal{B}_1}^{(\mathbf{z}|\text{rej})_{\mathbf{v}=c\cdot\mathbf{s}}} - \gamma_{\mathcal{B}_1}\right| - (1-p) \cdot \left|\delta_{\mathcal{B}_1}^{(\mathbf{z}|\text{acc})_{\mathbf{v}=c\cdot\mathbf{s}}} - \gamma_{\mathcal{B}_1}\right| \\
&\geqslant p \cdot \mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{MLWE}_{R_q,k,\ell,(\mathbf{z}|\text{rej})_{\mathbf{v}=c\cdot\mathbf{s}}}} - \mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{MLWE}_{R_q,k,\ell,(\mathbf{z}|\text{acc})_{\mathbf{v}=c\cdot\mathbf{s}}}}
\end{aligned}$$

By the properties of the statistical distance, we have,

$$\mathsf{Adv}^{\mathsf{MLWE}_{R_q,k,\ell,(\mathbf{z}|\mathrm{acc})_{\mathbf{v}=c\cdot\mathbf{s}}}}_{\mathcal{B}_1} \leqslant \mathsf{Adv}^{\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{z}}}}_{\mathcal{B}_1} + \delta$$

with δ the statistical distance between $(\mathbf{z}|\mathrm{acc})_{\mathbf{v}=c\cdot\mathbf{s}}$ and $\chi_{\mathbf{z}}$.

As $\mathcal{B}_1 = \mathcal{A}$, we conclude the proof by reordering the terms. □

6.4 Threshold ML-DSA

This section presents Threshold ML-DSA, a threshold variant of the ML-DSA digital signature scheme [LDKL+22].

Our construction builds upon the result of Section 6.3, which allows parties to safely reveal commitments in the Fiat-Shamir with Aborts paradigm. At a high level, parties will independently perform the *unoptimized* rejection sampling described in Section 6.3, first revealing and aggregating their commitments to derive an overall challenge, and then producing their responses, which are finally aggregated into a valid signature. To ensure compatibility with ML-DSA, we additionally round the aggregated commitment before deriving the challenge, and proceed to a second round of rejection when the aggregated response does not satisfy the size or hint verification condition of ML-DSA.

Note that for the security proof of the above scheme to go through, we need to ensure that the aggregated commitment is not biased by rushing adversaries. This is achieved as in prior works [CKM23; DKMM+24] by first revealing a hash of their commitment, only revealing the actual commitment after all parties have sent their hash. This gives rise to a simple and efficient 3-round protocol for one ML-DSA signing attempt.

The high-level blueprint of Threshold ML-DSA in the N-out-of-N setting is thus as follows:

- **Key Generation.** We sample one ML-DSA secret $\mathbf{s}_i$ per party, and compute the corresponding public key shares $\mathbf{b}_i = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{s}_i$. The overall public key is obtained by summing all public key shares, and rounding them as in ML-DSA.
- **Signing.**
  - **Round 1: Commitment phase.** Each party samples a random commitment $\mathbf{w}_i$, and broadcasts a hash of it.
  - **Round 2: Reveal phase.** After receiving all hashes, parties reveal their commitment shares, which are aggregated, and rounded as in ML-DSA, to obtain the overall commitment.
  - **Round 3: Response phase.** The challenge is derived from the overall commitment, and each party decides whether to output a response share or $\perp$ based on its local rejection sampling. Finally, when all parties output valid response shares, they are aggregated into the final signature. We further check whether it satisfies the size and hint verification conditions of ML-DSA, otherwise outputting $\perp$.
- **Verification.** The verification procedure is identical to the one of ML-DSA.

Interestingly, the above protocol can be adapted to the T-out-of-N setting leveraging the insights on short secret sharings from Chapter 5. Concretely, we use a replicated secret sharing, sampling one ML-DSA secret $\mathbf{s}_I$ for each possible subset I of $N - T + 1$ parties (e.g., this can be interpreted as guessing the set of honest parties), and distributing these secrets to the parties in I accordingly. The overall public key is again obtained by summing all public key shares $\mathbf{b}_I = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{s}_I$.

This secret sharing structure ensures that at least one secret $\mathbf{s}_I$ is not known by the selected set of up to $T-1$ corrupted parties, and any subset of T parties can jointly recover all secrets $\mathbf{s}_I$. In order to sign, parties proceed as in the N-out-of-N case, but decide on a partition of the secrets $\mathbf{s}_I$ among the T signing parties, such that each party i produces a response for the sum of the secrets $\mathbf{s}_I$ he is responsible for.

While this already provides a theoretical procedure to thresholdize ML-DSA, it needs to be carefully instantiated to ensure practical efficiency. Several challenges limit the scalability of this approach:

- The number of secrets $\mathbf{s}_I$ grows exponentially in N and T, quickly leading to a blow-up in the scheme's parameters as the efficiency of rejection sampling critically depends on the norm of the secret keys. We will show that the efficiency remains acceptable for small thresholds, and can be partly mitigated with targeted optimizations.
- The rejection sampling performed independently by each party leads to a multiplicative decrease in the overall acceptance probability, requiring several parallel repetitions to achieve a reasonable success rate.
- It is crucial that the aggregated signature remains compatible with ML-DSA. To do so, the partial responses produced by each party must be sufficiently small.

We propose several targeted optimizations to minimize the size of partial signatures, while maximizing the overall acceptance probability.

1. **Tight hyperball distributions.** Although the original ML-DSA scheme relies on uniform distributions for its rejection sampling – which have the benefit of simple implementation – we replace them with tighter distributions over hyperballs (as described in Section 6.2.6). This allows us to reduce the size of partial signatures for the same acceptance probability. We refer to [DFPS22] for a detailed discussion on the benefits of hyperball distributions in rejection sampling.

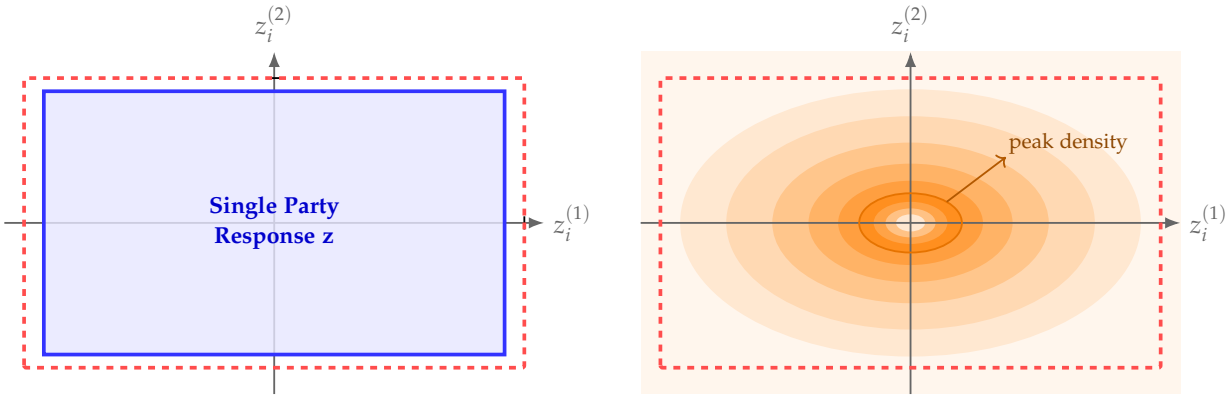
   This technique can be further optimized for ML-DSA by using imbalanced hyperballs, where the first ℓ coordinates of the randomness and response are sampled from a larger hyperball than the last k coordinates. This targets the *hint* technique used in ML-DSA, as it only compresses the second portion $\mathbf{z}^{(2)}$ of the signature, and thus introduces a much stricter verification condition on $\mathbf{z}^{(2)}$ than $\mathbf{z}^{(1)}$. The imbalance is parametrized by a factor ν. See HRej (Fig. 6.2) for a concrete description, and Fig. 6.3 for a visual comparison with the original uniform distribution.

2. **Optimized secret sharing.** To minimize the norm of the partial secrets used by each party to compute their response, we optimize the partition of secrets $\mathbf{s}_I$ among parties. Concretely, we introduce a deterministic function RSSRecover that computes a partition of the secret such that each party is responsible for at most $\left\lceil \binom{N}{T}/T \right\rceil$ secrets $\mathbf{s}_I$ during a session.

3. **Parallel repetitions.** As each party performs rejection sampling independently, the overall acceptance probability is the product of the individual acceptance probabilities. We counterbalance this effect by running K parallel repetitions of the protocol, and accepting the first valid signature produced. Concretely, assuming each party succeeds with probability p, taking $K = \lceil 1/p \rceil$ guarantees a correctness probability at least $1 - e^{-1} \approx 0.632$.

6.4.1 Construction

We now concretize the above ideas into a formal construction of Threshold ML-DSA. As defined in Section 6.2, threshold signature schemes primarily consist of key generation, signing,



(a) Rejection sampling with a hypercube. To achieve a reasonable acceptance probability, the hypercube must saturate the ML-DSA bounds, leaving no room for summing threshold shares.

(b) Projected 2-D density of an imbalanced hyperball ($\|(\mathbf{z}^{(1)}, \nu \cdot \mathbf{z}^{(2)})\|_2 \leqslant r$). The ball extends beyond the verification bounds (pale hue outside the red rectangle), but concentration of measure in high dimension m collapses almost all mass to a small inner ellipse with $|z_i^{(1)}| \approx r/\sqrt{m}$ and $|z_i^{(2)}| \approx r/(\nu\sqrt{m})$, well inside the bounds. ν in particular allows for an imbalance between the two coordinates, which maximizes acceptance probability while keeping the aggregated response $\mathbf{z} = \sum_i \mathbf{z}_i$ within the ML-DSA bound.

Figure 6.3: Comparison of rejection sampling domains for ML-DSA signatures.

combining, and verifying procedures. To formalize these procedures for Threshold ML-DSA, we first outline the parameters used in the single-party ML-DSA scheme, along with the additional hyperball parameters $r_{N,T}$ and $r'_{N,T}$ required for our threshold variant. We collect all of them in Table 6.2.

Key Generation. Our construction relies on the *short secret sharing* notion introduced in Chapter 5 (RSS), which is particularly fit for lattice-based constructions. These secret sharings are special instances of Linear Secret Sharing Schemes (LSSS) that additionally guarantee shares with *small norm and small reconstruction coefficients*. This is particularly suited for rejection-sampling-based schemes, as it enables each party to perform rejection sampling locally on a *short* partially reconstructed secret. This *short replicated secret sharing* scheme consists in sampling $\binom{N}{N-T+1}$ ML-DSA secrets $(\mathsf{sk}_I)_I$ from χ_{sk}, that is, one for each distinct set I of $N-T+1$ users. Since there are at most $T-1$ corrupted parties, at least one of these sets is fully non-corrupted and its secret remains hidden from the corrupted parties.

The final public key is derived by summing all secrets, multiplying by $\begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix}$, and rounding the result as in ML-DSA. Thanks to at least one secret remaining hidden from corrupted parties, the public key maintains pseudorandomness under the same MLWE assumption as ML-DSA. In addition, any set J of T parties can reconstruct the aggregated secret: by construction, there exists a partition $\mathcal{P}_{j\in J}$ so that $\mathsf{sk} = \sum_{j\in J}(\sum_{I\in\mathcal{P}_j} \mathsf{sk}_I)$. As an example, for $N=3$ and $T=2$, we have $\mathsf{sk} = \mathsf{sk}_{01} + \mathsf{sk}_{02} + \mathsf{sk}_{12}$. If users $0, 2$ want to sign, then user 0 will use $\mathsf{sk}_{01} + \mathsf{sk}_{12}$ as partial key, while user 2 will simply use sk_{12}.

The key generation procedure is detailed in Fig. 6.4.

Signing. The distributed signing protocol is detailed in Fig. 6.5. Any set act of T signers collectively know all the secrets $\mathbf{s}_I$. The core idea is to publicly decide on a partition $\bigsqcup_{i\in\mathsf{act}} \mathbf{m}_i = \{I\}_I$ of the secrets (or rather their indices) among the T parties using a deterministic function (RSSRecover), and then run T *unoptimized* Fiat-Shamir protocols in parallel to produce a partial signature for each partial secret $\mathbf{s}_i^{\mathsf{part}} = \sum_{I\in\mathbf{m}_i} \mathbf{s}_I$: partial commitments $\mathbf{w}_i = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{r}_i$ and signatures $\mathbf{z}_i = c \cdot \mathbf{s}_i^{\mathsf{part}} + \mathbf{r}_i$.

Table 6.2: Parameters used in Threshold ML-DSA.

	ML-DSA
R_q	Polynomial ring $R_q = \mathbb{Z}[X]/(q, X^n + 1)$
(k, ℓ)	Dimension of the public matrix $\mathbf{A} \in R_q^{k \times \ell}$
τ	Number of ± 1's in challenge c
d	Amount of bits dropped from the public key $\mathbf{b}$
η	Secret key range: $\chi_{\mathsf{sk}} = \mathcal{U}([-\eta, \eta]^{n(\ell+k)})$
$(\gamma_1, \gamma_2, \beta)$	Ranges for the signature generation and verification ($\beta = \tau \cdot \eta$)
ω	Number of 1's in the hint h
	Threshold ML-DSA
K	Number of parallel protocol repetitions
ν	Expansion factor for the first ℓ coordinates of the randomness
r'	Randomness ball rad. $\chi_{\mathbf{r}} = \left\{ \lfloor (\nu \mathbf{x}_1, \mathbf{x}_2) \rceil \mid (\mathbf{x}_1, \mathbf{x}_2) \xleftarrow{\$} \mathcal{B}_{R,\ell+k}(r') \right\}$
r	Target ball rad. $\chi_{\mathbf{z}} = \left\{ \lfloor (\nu \mathbf{x}_1, \mathbf{x}_2) \rceil \mid (\mathbf{x}_1, \mathbf{x}_2) \xleftarrow{\$} \mathcal{B}_{R,\ell+k}(r) \right\}$
M	Rejection parameter (per party) $M = \left(\frac{r'}{r}\right)^{n(\ell+k)}$

$\mathsf{RSS}(N, T, \chi_{\mathsf{sk}}) \to (\mathbf{s}, \mathsf{sk} := (\mathsf{sk}_1, \ldots, \mathsf{sk}_N))$

1: **for** $I \subset [N]$ such that $|I| = N - T + 1$ **do**
2: $\quad \mathbf{s}_I \leftarrow \chi_{\mathsf{sk}}$
3: $\mathbf{s} := \sum_I \mathbf{s}_I$
4: **for** $i \in [N]$ **do**
5: $\quad \mathsf{sk}_i := \{I : \mathbf{s}_I \mid I \subset [N] \text{ s.t. } i \in I \wedge |I| = N - T + 1\}$
6: **return** $(\mathbf{s}, \mathsf{sk} := (\mathsf{sk}_0, \cdots, \mathsf{sk}_{N-1}))$

$\mathsf{Keygen}(1^\lambda, N, T) \to (\mathsf{vk}, \mathsf{sk})$

1: $\rho \leftarrow \{0,1\}^{256}$
2: $\mathbf{A} := \mathsf{H}_{\mathbf{A}}(\rho)$
3: $(\mathbf{s}, (\mathsf{sk}_i)_{i \in [N]}) \leftarrow \mathsf{RSS}(N, T, \chi_{\mathsf{sk}})$
4: $\mathbf{b} := [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{s}$
5: $(\mathbf{b}_\top, \mathbf{b}_\perp) := \mathsf{Power2Round}(\mathbf{b}, d)$
6: $tr \in \{0,1\}^{256} := H(\rho || \mathbf{b}_\top)$
7: **return** $(\mathsf{vk} := (\rho, \mathbf{b}_\top), \mathsf{sk} := (tr, \mathsf{sk}_i)_{i \in [N]})$

Figure 6.4: Key generation procedure of Threshold ML-DSA.

In particular, each party uses a full MLWE sample as commitment $\mathbf{w}_i$ (i.e. $\mathbf{w}_i = \mathbf{A} \cdot \mathbf{y} + \mathbf{e}$) instead of using $\mathbf{A} \cdot \mathbf{y}$ as in ML-DSA. This change allows leveraging our result from Section 6.3 for direct reveal of $\mathbf{w}_i$ before rounding, even in case of rejection. To ensure correctness, we verify that the sum of the responses $\mathbf{z} = \sum_{i \in \mathsf{act}} \mathbf{z}_i$ satisfies the ML-DSA verification equation $[\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{z} = c \cdot \mathbf{b} + \mathbf{w}$: the challenge c can be computed as $c = \mathsf{SampleInBall}(\tilde{c})$, where $\tilde{c} = \mathsf{H}(\mu || \mathsf{HighBits}(\mathbf{w}, 2\gamma_2))$ and $\mathbf{w} = \sum_{i \in \mathsf{act}} \mathbf{w}_i$. SampleInBall maps $\tilde{c}$ to a polynomial c such that $\|c\|_1 = \tau$ and $\|c\|_\infty = 1$.

Following prior works [DKMM+24; CKM23], we design a three-round scheme secure against rushing adversaries (who target $\mathbf{w}$):

1. **First round:** parties sample randomness $\mathbf{r}_i \leftarrow \chi_{\mathbf{r}}$ and compute the commitment $\mathbf{w}_i = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{r}_i$. They publish a hash of the commitment $\mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, i, \mathbf{w}_i)$.
2. **Second round:** parties reveal the values of $(\mathbf{w}_i)_{i \in \mathsf{act}}$.
3. **Third round:** Each party computes the response $\mathbf{z}_i = c \cdot \mathbf{s}_i^{\mathsf{part}} + \mathbf{r}_i$, where c is derived from the aggregated commitment $\mathbf{w} = \sum_{i \in \mathsf{act}} \mathbf{w}_i$, and apply rejection sampling.

In the complete scheme, we omit the second part $\mathbf{z}_i^{(2)} \in R_q^k$ of the responses since the verifier can reconstruct it from public values. Any public party can then aggregate the partial signatures into $\mathbf{z}^{(1)} = \sum_{i \in \mathsf{act}} \mathbf{z}_i^{(1)} \in R_q^\ell$ to obtain the signature and compute the corresponding hint $\mathbf{h}$ to output the signature $(c, \mathbf{z}^{(1)}, \mathbf{h})$.

Remark 6.4.1 (Balanced partition of the shares for Replicated Secret Sharing). The sizes of partial secrets $\mathbf{s}_i^{\mathsf{part}}$ in Threshold ML-DSA impact greatly the efficiency of parameters as it directly correlates with the norm 2 of the hyperballs that are used, and it is important to use *as few secrets as possible* in a session for each party in RSSRecover. To tackle this, we propose in Section 6.4.2 a graph based algorithm that computes a balanced partition of the shares. In practice, we can hardcode the optimal partition produced by our algorithm and include it as part of the public description of the Threshold ML-DSA scheme.

Combine and Verify. The combine and verify procedures are detailed in Fig. 6.6. Combine involves computing the hint $\mathbf{h}$ that allows verifiers to recover the value $\mathsf{HighBits}(\mathbf{w}, 2\gamma_2)$ from $\mathbf{A} \cdot \mathbf{z}^{(1)} - 2^d \cdot c \cdot \mathbf{b}_\top$ – which is used as an approximation of $\mathbf{w}$. Similarly to ML-DSA, we compute the difference $\boldsymbol{\delta} = \mathbf{w} - \left(\mathbf{A} \cdot \mathbf{z}^{(1)} - 2^d \cdot c \cdot \mathbf{b}_\top\right)$ and use the function MakeHint with $\boldsymbol{\delta}$ to compute the hint. The high bits of $\mathbf{w}$ are correctly recovered as long as $\|\boldsymbol{\delta}\|_\infty \leqslant \gamma_2$; we thus restart the protocol if this is not the case. Verification proceeds just as in ML-DSA: the procedure also restarts if $\|\mathbf{z}^{(1)}\|_\infty \geqslant \gamma_1 - \beta$ or if $\mathbf{h}$ has more than ω 1's. While this check ensures security in ML-DSA, it is only used for correctness here. Indeed, the hiding property of the rejection step in $\mathsf{ShareSign}_3$ already ensures that $\mathbf{z}$ does not leak any secret information. Thanks to parallel repetitions we can minimize the need for restarts as we ensure that any execution of the protocol will output a signature with probability at least $1/2$.

6.4.2 Balanced Partition of the Shares for Replicated Secret Sharing

Recall that we aim at computing a balanced partition of the shares among the parties during the signing session, that is for a partition $(m_i)_{i \in \mathsf{act}}$ of the secrets, which minimizes $\max_{i \in \mathsf{act}} |m_i|$. As a general solution when fixing $U = \max_{i \in \mathsf{act}} |m_i|$, we can efficiently look for a solution using a max-flow modelization of the problem. We consider a bipartite graph consisting of (i) the parties on one side, (ii) the secrets on the other side, and we add an edge between a party and a secret when said party owns that secret. We can solve the above balanced partition problem as follows:

1. We direct the edges from the parties to the secrets.
2. We add a flow U entering the party nodes $i \in \mathsf{act}$.
3. We add an exit flow 1 on the secret nodes.

This is represented in Fig. 6.7. If the max-problem solution covers all the secrets, then we obtain a partition of maximal weight K. We can find the optimal weight K by testing values in increasing order.

For our concrete parameters, we consider $N \leqslant 6$, for which we can easily verify that the optimal assignations verify $K = \left\lceil \binom{N}{T-1} / T \right\rceil$.

$\mathsf{RSSRecover}(\mathsf{act}) \to \mathcal{P}(S)^{\mathsf{act}}$

Compute partition $\mathbf{m} = (m_i)_{i \in \mathsf{act}}$ s.t.
$\bigsqcup m_i = \{I \subset [N] \mid |I| = N - T + 1\}$
and minimizing $\max_{i \in \mathsf{act}} |m_i|$, using a
maximum flow algorithm. See Section 6.4.2.
return $\mathbf{m}$

$\mathsf{ShareSign}_1(\mathsf{vk}, i, \mathsf{sk}_i, \mathsf{st}_i)$

$\mathbf{r}_i \leftarrow \chi_{\mathbf{r}}$
$\mathbf{w}_i := [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}_i$
$\mathsf{cmt}_i = \mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, i, \mathbf{w}_i) \in \{0,1\}^{2\lambda}$
$\mathsf{st}_i := \mathsf{st}_i \cup \{(\mathsf{cmt}_i, \mathbf{w}_i, \mathbf{r}_i)\}$
return $(\mathsf{pm}_1^i := \mathsf{cmt}_i, \mathsf{st}_i)$

$\mathsf{ShareSign}_2(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, i, \mathsf{sk}_i, \mathsf{st}_i, \mathsf{pm}_1)$

require $\mathsf{act} \subseteq [N] \wedge i \in \mathsf{act}$
require $(\mathsf{pm}_1^i, \cdot, \cdot) \in \mathsf{st}_i$
Pick $(\mathsf{cmt}_i, \mathbf{w}_i, \mathbf{r}_i)$ from st_i with $\mathsf{pm}_1^i = \mathsf{cmt}_i$
$\mathsf{st}_i := \mathsf{st}_i \backslash \{(\mathsf{cmt}_i, \mathbf{w}_i, \mathbf{r}_i)\}$
$\mathsf{st}_i := \mathsf{st}_i \cup \{(\mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, \mathbf{w}_i, \mathbf{r}_i)\}$
return $(\mathsf{pm}_2^i := \mathbf{w}_i, \mathsf{st}_i)$

$\mathsf{ShareSign}_3(\mathsf{vk}, i, \mathsf{sk}_i, \mathsf{st}_i, \mathsf{pm}_2)$

require $(\cdot, \mathsf{pm}_2^i, \cdot) \in \mathsf{st}_i$
Pick $(\mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, \mathbf{w}_i, \mathbf{r}_i)$ from st_i with $\mathsf{pm}_2^i = \mathbf{w}_i$
Parse $\mathsf{pm}_1 = (\mathsf{cmt}_j)_j$ and $\mathsf{pm}_2 = (\mathbf{w}_j)_{j \neq i}$
require $\forall j \in \mathsf{act}, \mathsf{cmt}_j = \mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, j, \mathbf{w}_j)$
$\mathsf{st}_i := \mathsf{st}_i \backslash \{(\mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, \mathbf{w}_i, \mathbf{r}_i)\}$
$\mu \in \{0,1\}^{512} := H(tr || \mathsf{msg})$
$\mathbf{w} := \sum_{j \in \mathsf{act}} \mathbf{w}_i$
$\mathbf{w}_\top := \mathsf{HighBits}(\mathbf{w}, 2\gamma_2)$
$\tilde{c} \in \{0,1\}^{2\cdot\lambda} := \mathsf{H}(\mu || \mathbf{w}_\top)$
$c := \mathsf{SampleInBall}(\tilde{c})$
$\mathbf{m} := \mathsf{RSSRecover}(\mathsf{act})$
$\mathbf{s}_i^{\mathsf{part}} := \sum_{I \in m_i} \mathbf{s}_I$
$\mathbf{z}_i \leftarrow \mathsf{HRej}(c \cdot \mathbf{s}_i^{\mathsf{part}}, r, r', \nu, M; \mathbf{r}_i)$
if $\mathbf{z}_i = \perp$ **then** abort
$\mathbf{z}_i^{(1)}, \mathbf{z}_i^{(2)} := \mathbf{z} \in R_q^\ell \times R_q^k$
return $(\mathsf{pm}_3^i := \mathbf{z}_i^{(1)}, \mathsf{st}_i)$

Figure 6.5: Signing procedure in Threshold ML-DSA.

CONCRETE IMPLEMENTATION. For simplicity and efficiency, we do not wish to implement a max-flow solver in the Threshold ML-DSA code. Instead, we first observe that when $T = N$, then the partition is easily obtained as each signing party possesses exactly one secret. For the other cases $2 \leqslant T < N \leqslant 6$, we list an optimal solution for $\mathsf{act} = \{1, ..., T\}$ and each value (T, N). We

$\mathsf{Combine}(\mathsf{vk} = (\mathsf{seed}, \mathbf{b}_\top), \mathsf{act}, \mathsf{msg}, (\mathsf{pm}_k)_{k\in\{1,2,3\}})$

1 : Parse $\mathsf{pm}_2 = (\mathbf{w}_i)_i$ and $\mathsf{pm}_3 = (\mathbf{z}_i^{(1)})_i$
2 : $\mu \in \{0,1\}^{512} := H(tr||\mathsf{msg})$
3 : $\mathbf{w} := \sum_{j\in\mathsf{act}} \mathbf{w}_i$ // Aggregated commitment in R_q^k
4 : $\mathbf{w}_\top := \mathsf{HighBits}(\mathbf{w}, 2\gamma_2)$
5 : $\tilde{c} := \mathsf{H}(\mu||\mathbf{w}_\top)$ // $\tilde{c} \in \{0,1\}^{2\cdot\lambda}$
6 : $c := \mathsf{SampleInBall}(\tilde{c})$ // Global challenge
7 : $\mathbf{z}^{(1)} = \sum_{i\in\mathsf{act}} \mathbf{z}_i^{(1)}$
8 : $\delta := \mathbf{w} - \left(\mathbf{A}\cdot\mathbf{z}^{(1)} - 2^d \cdot c \cdot \mathbf{b}_\top\right)$ // Recover error in R_q^k
9 : $\mathbf{h} := \mathsf{MakeHint}_q(\delta, \mathbf{A}\cdot\mathbf{z}^{(1)} - 2^d\cdot c\cdot\mathbf{b}_\top, 2\gamma_2)$
10 : **if** $\left(\|\mathbf{z}^{(1)}\|_\infty \geqslant \gamma_1 - \beta\right)$ **or** $(\|\delta\|_\infty > \gamma_2)$ **or** $(\|\mathbf{h}\|_1 > \omega)$ **then**
11 : **abort**
12 : **return** $\mathsf{sig} := (\tilde{c}, \mathbf{z}^{(1)}, \mathbf{h})$

$\mathsf{Verify}(\mathsf{vk} := (\rho, \mathbf{b}_\top), \mathsf{msg}, \mathsf{sig})$

1 : Parse $\mathsf{sig} := (\tilde{c}, \mathbf{z}^{(1)}, \mathbf{h})$
2 : $\mathbf{A} := \mathsf{H}_\mathbf{A}(\rho)$
3 : $\mu \in \{0,1\}^{512} := H(tr||\mathsf{msg})$
4 : $c := \mathsf{SampleInBall}(\tilde{c})$
5 : $\mathbf{w}'_\top := \mathsf{UseHint}_q\left(\mathbf{h}, \mathbf{A}\cdot\mathbf{z}^{(1)} - 2^d\cdot c\cdot\mathbf{b}_\top, 2\gamma_2\right)$
6 : **return** $\|\mathbf{z}^{(1)}\|_\infty < \gamma_1 - \beta$ **and** $\tilde{c} = \mathsf{H}(\mu||\mathbf{w}'_\top)$ **and** $\|\mathbf{h}\|_1 \leqslant \omega$

Figure 6.6: Combine and verification of Threshold ML-DSA.

obtain a partition for all other possible signing sets act by symmetry by permuting the index of parties. The idea is formalized in Algorithm 1.

6.4.3 Correctness and Security

We prove that Threshold ML-DSA (i) is unforgeable, and (ii) p-terminates. For (i), Threshold ML-DSA is unforgeable if ML-DSA is unforgeable and the MLWE problem used in ML-DSA is hard. For (ii), we introduce a bound B such that for any $\mathbf{m}_i$ returned by RSSRecover and $(\mathbf{u}_1, \mathbf{u}_2) = \sum_{I\in\mathbf{m}_i} \mathbf{s}_I \in R_q^{\ell+k}$, $\|(\frac{1}{\nu}\cdot c\cdot\mathbf{u}_1, c\cdot\mathbf{u}_2)\|_2 \leqslant B$ with overwhelming probability for a uniformly sampled challenge c and over the randomness of the key generation.

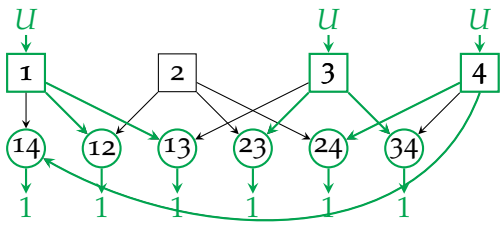


Figure 6.7: Illustration of the replicated secret sharing with $(N, T) = (4, 3)$: users are on top (rectangles), shares at the bottom (circles). Balanced assignment of shares to active users is performed via a max-flow solving algorithm (in green).

Algorithm 1 RSSRecover(act) $\to \mathcal{P}(S)^{\mathsf{act}}$

if $T = N$ **then**
  return $(m_i := (\{i\}))_{i \in \mathsf{act}}$
if $T = 2 \wedge N = 3$ **then**
  shares $:= \{0 : \{01, 02\}, 1 : \{12\}\}$
else if $T = 2 \wedge N = 4$ **then**
  shares $:= \{0 : \{013, 023\}, 1 : \{012, 123\}\}$
else if $T = 3 \wedge N = 4$ **then**
  shares $:= \{0 : \{01, 03\}, 1 : \{12, 13\}, 2 : \{23, 02\}\}$
else if $T = 2 \wedge N = 5$ **then**
  shares $:= \{0 : \{0134, 0234, 0124\}, 1 : \{1234, 0123\}\}$
else if $T = 3 \wedge N = 5$ **then**
  shares $:= \{0 : \{034, 013, 014, 023\}, 1 : \{012, 123, 124, 134\}, 2 : \{234, 024\}\}$
else if $T = 4 \wedge N = 5$ **then**
  shares $:= \{0 : \{01, 03, 04\}, 1 : \{12, 13, 14\}, 2 : \{23, 02, 24\}, 3 : \{34\}\}$
else if $T = 2 \wedge N = 6$ **then**
  shares $:= \{0 : \{02345, 01235, 01245\}, 1 : \{12345, 01234, 01345\}\}$
else if $T = 3 \wedge N = 6$ **then**
  shares $:= \{0 : \{0134, 0124, 0135, 0345, 0125\}, 1 : \{0145, 1345, 1235, 1234, 1245\},$
  $2 : \{0235, 0245, 0234, 0123, 2345\}\}$
else if $T = 4 \wedge N = 6$ **then**
  shares $:= \{0 : \{014, 023, 015, 012, 045\}, 1 : \{135, 134, 125, 145, 124\}, 2 : \{245, 024, 235,$
  $234, 025\}, 3 : \{034, 013, 123, 345, 035\}\}$
else if $T = 5 \wedge N = 6$ **then**
  shares $:= \{0 : \{01, 02, 05\}, 1 : \{12, 13, 15\}, 2 : \{23, 24, 25\}, 3 : \{03, 34, 35\}, 4 : \{45, 04, 14\}\}$
$\phi := \{i : i\}_{i \in [N]}$ ▷ Define a permutation of the party indexes
$i_1 := 0, i_2 := T$
for $j \in [N]$ **do**
  if $j \in \mathsf{act}$ **then**
    $\phi[i_1] = j$
    $i_1 = i_1 + 1$
  else
    $\phi[i_2] = j$
    $i_2 = i_2 + 1$
for $i \in [T]$ **do** ▷ Translate the ideal sharing for act
  $m_{\phi(i)} = \{\phi(u) \mid u \in \mathsf{shares}[i]\}$
return $(m_i)_{i \in \mathsf{act}}$

Let $\varepsilon > 0$. We assume that (r, r', ε) verify the conditions of Lemma 6.2.9 for secret vectors of norm B, i.e. $2\varepsilon = I_{1-1/\phi^2}\left(\frac{n(\ell+k)+1}{2}, \frac{1}{2}\right)$ for some ϕ, and $r'^2 \geqslant r^2 + B^2 + 2rB/\phi$, implying Lemma 6.4.2.

Lemma 6.4.2. *For the above constraints, we have for any $\|\mathbf{v}\|_2 \leqslant B$, $R^{\varepsilon}_{\infty}(\chi_{\mathbf{z}} || \chi_{\mathbf{r}} + \mathbf{v}) = M$*

Proof. The equation checks out by applying Lemma 6.2.9 with the data processing inequality for Rényi divergence. □

Theorem 6.4.3. *Assume $\varepsilon = o(1)$ and $\Pr\left[\mathsf{HighBits}(\mathbf{w}, 2\gamma_2) = \mathsf{HighBits}(\mathbf{w}', 2\gamma_2)\right] = o(1)$, where $\mathbf{w}, \mathbf{w}'$ are sampled from $\begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \sum_{i=1}^{T} \chi_{\mathbf{r}}$, and $\mathbf{A} \xleftarrow{\$} R_q^{k\times\ell}$ – remark that this holds notably under the assumption $\mathsf{MLWE}_{R_q,k,\ell,\sum_{i=1}^{T}\chi_{\mathbf{r}}}$. Threshold ML-DSA p-terminates in the ROM for*

$$p = \mathbb{E}_{k \leftarrow \mathsf{Bin}\left(K, (1-\frac{1}{M})^T\right)}[v_k] + o(1)$$

where v_k, for $k \in [0, K]$, is the probability that in at least one of the k combinations we have $\|\mathbf{z}^{(1)}\|_\infty \leqslant \gamma_1 - \beta$ and $\|\boldsymbol{\delta}\|_\infty \leqslant \gamma_2$ and the number of 1's in $\mathbf{h}$ is less than ω, when $\mathbf{z}$ is sampled from $\sum_{i\in[T]} \chi_{\mathbf{z}}$ and $\boldsymbol{\delta} = c \cdot \mathbf{z}^{(2)} - c \cdot \mathbf{b}_{\perp}$.

Proof. First, observe that by design, any signature produced by the scheme is a valid ML-DSA signature. The signature necessarily (i) verifies $\|\mathbf{z}^{(1)}\| < \gamma_1 - \beta$, and (ii) $\mathbf{h}$ has at most ω 1's. Additionally, the scheme ensures that $\boldsymbol{\delta} := \mathbf{w} - \left(\mathbf{A}\cdot\mathbf{z}^{(1)} - 2^d \cdot c \cdot \mathbf{b}_{\top}\right)$ has an infinity norm at most γ_2. By applying Lemma 6.2.2, we obtain $\mathsf{UseHint}_q\left(\mathbf{h}, \mathbf{A}\cdot\mathbf{z}^{(1)} - 2^d \cdot c \cdot \mathbf{b}_{\top}, 2\gamma_2\right) = \mathbf{w}_{\top}$ during verification. Thus, the verification bounds are satisfied, and the correct challenge c is successfully recovered.

It remains to ensure that the scheme succeeds with probability of at least p. We prove this with a series of games starting from $\mathsf{Game}_0 := \mathsf{Game}_{\mathsf{TS}}^{\mathsf{TS\text{-}CORR\text{-}ABORT}}$. We denote $p_i = \Pr\left[\mathsf{Game}_i() \neq \perp\right]$.

Game_1. In this game, we assert that for any signing party and parallel session during the second round, we have $\left\|\left(\frac{1}{\nu}\cdot c \cdot \mathbf{s}_1^{\mathsf{part}}, c \cdot \mathbf{s}_2^{\mathsf{part}}\right)\right\|_2 \leqslant B$, otherwise the game returns 0. We need to add such assertion up to $T \cdot K$ times. As B is an overwhelming bound, we obtain by union-bound:

$$p_0 \geqslant p_1 - TK \cdot \mathsf{negl}(\lambda) = p_1 - \mathsf{negl}(\lambda)$$

Game_2. In this game, we ensure that the parallel signing sessions use different $\tilde{c}$ and $\mathbf{w}$ with different high bits. Otherwise, the game returns 0. It ensures that different challenges are effectively used in every session. As we have at most K independent sessions, we have from the union bound:

$$p_1 \geqslant p_2 - K^2 \cdot \left(\Pr\left[\mathsf{HighBits}(\mathbf{w}, 2\gamma_2) = \mathsf{HighBits}(\mathbf{w}', 2\gamma_2)\right] + 2^{-2\cdot\lambda}\right)$$

Game_3. In this game, we sample the challenge c of each session in advance, sample $\mathbf{z}_i \leftarrow \mathsf{Rej}(c \cdot \mathbf{s}_i^{\mathsf{part}}, \chi_{\mathbf{z}}, \chi_{\mathbf{r}}, M)$ for each party. If $\mathbf{z}_i = \perp$, we instead sample $\mathbf{z}_i$ from the distribution $(\mathbf{z}|\mathsf{rej})_{\mathbf{v}=c\cdot\mathbf{s}_i^{\mathsf{part}}}$ of rejected $\mathbf{z}$. Then, we program the random oracle after computing $\mathbf{w} = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix}\cdot\mathbf{z} - c\cdot\mathbf{b}$'s, with $\mathbf{z} = \sum_{i\in\mathsf{act}} \mathbf{z}_i$ in each parallel session. The distribution of $\mathbf{w}$ is identical as in Game_2, and Game_2 ensured that we can program c without affecting the probability of winning as no two $\mathbf{w}$ have the same high bits, and then computing $\mathbf{z}$ first in this way is equivalent: $p_2 = p_3$.

Game_4. In this game, we sample the partial responses $\mathbf{z}_i$ using the Ideal rejection sampling from Fig. 6.1. Recall the Game_1 ensured that $c \cdot \mathbf{s}_i^{\mathsf{part}}$ has a twisted norm bounded by B. We can thus apply the result Lemma 6.2.6 on the Rényi divergence of rejection sampling, using the intermediate result on hyperballs of Lemma 6.2.9, since by assumption r, r', B verify its conditions. The Rényi divergence when replacing a single $\mathbf{z}_i$ is $1 + \frac{\varepsilon}{M-1}$.

Let E the event "$\forall(\mathbf{u}_1, \mathbf{u}_2) = \mathbf{s}^{\mathsf{part}}, \|(\frac{1}{\nu} \cdot c \cdot \mathbf{u}_1, c \cdot \mathbf{u}_2)\|_2 \leqslant B$". Formally, we consider the random variable $X = ((\mathsf{sk}_i)_{i\in[N]}, (\mathbf{z}_i^j)_{i\in\mathsf{act}, j\in[K]})$ conditioned on E, where the $\mathbf{z}_i^j$ are the responses output by the parties in each session. After conditioning on $(\mathsf{sk}_i)_{i\in[N]}$ we observe that the $\mathbf{z}_i^j$ become independent. We can thus apply the multiplicativity of the Rényi divergence (Lemma 6.2.5) to bound the Rényi divergence between X in Game_3 and Game_4 by $\left(1 + \frac{\varepsilon}{M-1}\right)^{KT}$.

We then apply the data processing inequality and probability preservation of the Rényi divergence to obtain:

$$\Pr\left[\mathsf{Game}_4() = \perp \mid E\right] \cdot \left(1 + \frac{\varepsilon}{M-1}\right)^{KT} \geqslant \Pr\left[\mathsf{Game}_3() = \perp \mid E\right]$$

As E has the same probability in both games, we have:

$$\Pr\left[\mathsf{Game}_4() = \perp\right] \cdot \left(1 + \frac{\varepsilon}{M-1}\right)^{KT} \geqslant \Pr\left[\mathsf{Game}_3() = \perp\right]$$

Finally, $p_3 \geqslant 1 + \left(1 + \frac{\varepsilon}{M-1}\right)^{KT} \cdot (p_4 - 1)$.

CONCLUSION. At this point, acceptance during the protocol occurs independently of the responses $\mathbf{z}_i$ with probability $(1 - \frac{1}{M})$ for each party. Hence, each session succeeds during the protocol with probability $(1 - \frac{1}{M})^T$.

Additionally, we wish for the combination of responses to succeed. We can combine the above results to evaluate the final success probability of the protocol: $p_4 = \mathbb{E}_{k\leftarrow\mathsf{Bin}\left(K,(1-\frac{1}{M})^T\right)}[v_k]$ □

We now state the unforgeability of Threshold ML-DSA, starting with a simplified theorem statement and a proof sketch. A formal and detailed proof is provided at the end of this section.

Theorem 6.4.4 (Unforgeability). *For any parameter $\phi > 0$, we define a corresponding maximal number of signing queries allowed $Q_s = 2/(K \cdot I_{1-1/\phi^2}\left(\frac{n(\ell+k)+1}{2}, \frac{1}{2}\right))$.*

The TS *from Figs. 6.4 to 6.6 is* ASYNC-TS-UF *secure in the ROM, for up to Q_s calls to the individual signing oracles, under the unforgeability of* ML-DSA *as well as the hardness of the* $\mathsf{MLWE}_{R_q,k,\ell,\chi}$ *assumptions for $\chi \in \{\chi_{\mathsf{sk}}, \chi_{\mathbf{r}}, \chi_{\mathbf{z}}\}$.*

We deduce from Theorem 6.4.4 that breaking the unforgeability of our Threshold ML-DSA scheme is as hard as breaking ML-DSA itself. Indeed, the MLWE assumptions over $\chi_{\mathbf{r}}$ and $\chi_{\mathbf{z}}$ are almost statistically verified due to the large width of the hyperballs we use, and in particular are much harder than the assumptions underlying ML-DSA.

The proof proceeds via a sequence of games, gradually transforming the real attack scenario into one where a forgery against the threshold scheme yields a forgery against single-party ML-DSA, under the hardness of the relevant MLWE problems.

- **Game 2.** We show that honest commitments $\mathbf{w}_i$ have high min-entropy before being revealed, so the adversary cannot guess them in advance. This allows us to program the random oracle $\mathsf{H}_{\mathsf{cmt}}$ lazily, returning random hashes in round 1 and sampling $\mathbf{w}_i$ only in round 2 for honest parties. This step is justified in the random oracle model.

- **Games 3–6.** We sample the challenges c in advance in round 2. The pre-image and collision resistance of $\mathsf{H}_{\mathsf{cmt}}$ ensure that the adversary has already chosen its $\mathbf{w}_i$ before round 2. With the lazy sampling from the previous game, honest commitments are sampled last, so the aggregated commitment $\mathbf{w} = \sum_i \mathbf{w}_i$ has high min-entropy. We can thus program the random oracle to return the desired challenge c without the adversary noticing.

- **Game 7.** We compute $\mathbf{z}_i$ first, even in case of rejection, and then derive $\mathbf{w}_i = \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{z}_i - c \cdot \mathbf{b}_i^{\mathsf{part}}$. If $\mathbf{z}_i$ is rejected, we sample it from the distribution of rejected $\mathbf{z}$. This is statically equivalent to the previous game.
- **Game 8.** We ensure that the norm of the secret vector $c \cdot \mathbf{s}_i^{\mathsf{part}}$ is at most B, which holds with overwhelming probability.
- **Game 9.** We replace the commitments $\mathbf{w}_i$ by uniform values when the corresponding $\mathbf{z}_i$ is rejected and will never be output. This change is indistinguishable to the adversary under the MLWE assumptions for the relevant distributions, following Section 6.3.
- **Games 10–11.** We remove the last dependencies on the secrets by replacing the sampling of $\mathbf{z}_i$ with rejection sampling from the ideal functionality (recalled in Fig. 6.1). The Rényi divergence argument (see Section 6.2.6) ensures that, for up to Q_s queries, the adversary's advantage increases by at most 2 bits.
- **Games 12–13.** We replace the public key of Threshold ML-DSA (composed of $\binom{N}{N-T+1}$ secrets) by an ML-DSA public key (a single secret). Since the signing oracles are now independent of secret values, this is justified by the MLWE assumption.

At the end of this sequence, any forgery output by the adversary yields a valid forgery against an ML-DSA key. Thus, the unforgeability of the threshold scheme reduces to that of ML-DSA, under the stated assumptions and in the ROM.

6.4.4 Full Proof of Unforgeability

We now fully formalize the unforgeability of Threshold ML-DSA. We provide a complete formal statement and its proof. We limit ourselves to the case $K = 1$, as the general case can be easily derived by considering $Q_s' = K \cdot Q_s$, the number of calls to signing queries.

Theorem 6.4.5 (Unforgeability of Threshold ML-DSA). *Formally, let $\mathcal{A}$ be an adversary against the* ASYNC-TS-UF *security of our threshold scheme making Q_s calls to signing oracles, and at most $Q_{\mathsf{H}_{\mathsf{cmt}}}, Q_{\mathsf{H}}$ queries respectively to the random oracles $\mathsf{H}_{\mathsf{cmt}}, \mathsf{H}$. There exist adversaries $\mathcal{B}_{\mathbf{s}}$ against the $\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathsf{sk}}}$, $\mathcal{B}_{\mathbf{r}}$ against the $\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{r}}}$ game, $\mathcal{B}_{\mathbf{z}}$ against the $\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{z}}}$ game, and $\mathcal{B}_{\mathsf{EUF\text{-}CMA}}$ against the unforgeability of* ML-DSA *running in time $\mathsf{Time}(\mathcal{B}_{\mathbf{s}}) \approx \mathsf{Time}(\mathcal{B}_{\mathbf{r}}) \approx \mathsf{Time}(\mathcal{B}_{\mathbf{z}}) \approx \mathsf{Time}(\mathcal{B}_{\mathsf{EUF\text{-}CMA}}) \approx \mathsf{Time}(\mathcal{A})$ such that:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ASYNC\text{-}TS\text{-}UF}}_{\mathcal{A}}(1^\lambda, N, T, T-1) \leqslant\ & \frac{Q_s Q_{\mathsf{H}_{\mathsf{cmt}}}}{q^{nk}} + Q_{\mathsf{H}} Q_s^2 \cdot \left(\frac{2\gamma_2+1}{q}\right)^{kn} + Q_{\mathsf{H}} Q_s \cdot 2^{-2\cdot\lambda} \\ & + \frac{T \cdot Q_s Q_{\mathsf{H}_{\mathsf{cmt}}} + (Q_{\mathsf{H}_{\mathsf{cmt}}} + Q_s)^2}{2^{2\lambda}} + Q_{\mathsf{H}}^2 \cdot 2^{-512} + \mathsf{negl}(\lambda) \\ & + \left(1 + \frac{2\varepsilon}{1-\varepsilon}\right)^{Q_s} \cdot \left(2\mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}_{\mathbf{s}}} + \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{B}_{\mathsf{EUF\text{-}CMA}}}\right) \\ & + Q_s \cdot \frac{3M-2}{M-1} \cdot \left(\mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}_{\mathbf{r}}} + \mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}_{\mathbf{z}}} + \frac{2\varepsilon}{1-\varepsilon}\right)\end{aligned}$$

where $M = \left(\frac{r'}{r}\right)^{n(\ell+k)}$, $2\varepsilon = I_{1-1/\phi^2}\left(\frac{n(\ell+k)+1}{2}, \frac{1}{2}\right)$.

Proof. We proceed with a series of hybrids starting from the ASYNC-TS-UF game. Throughout this proof, we denote the advantage of an adversary $\mathcal{A}$ against a search game G by $\mathsf{Adv}^G_{\mathcal{A}} := \Pr[G(\mathcal{A}) \to 1]$ – in particular, we omit passing $(1^\lambda, N, T, T-1)$ as input.

Game_1. This is the ASYNC-TS-UF game from Fig. 3.5.

Game_2. In this hybrid we defer the computation of the honest $\mathbf{w}_i$ to the second round of the protocol, and program the random oracle to be consistent. We also introduce a flag f_{fail} to explicitly mark additional cases where the adversary loses. Notably, f_{fail} is set to $\top$ when the random oracle is programmed on a previously queried value. This is formalized in Fig. 6.8.

Game_2

$L_{\mathsf{sig}}, \mathsf{UnopenedCmt}[\cdot] := \varnothing$
$f_{\mathsf{fail}} := \bot$
$(\mathsf{CS}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathsf{H}}(1^\lambda, N, T)$
assert $\mathsf{CS} \subseteq [N] \wedge |\mathsf{CS}| \leqslant T-1$
$\mathsf{HS} := [N] \backslash \mathsf{CS}$
$(\mathsf{vk}, (\mathsf{sk}_i)_{i\in[N]}) \leftarrow \mathsf{Keygen}(1^\lambda, N, T)$
for $i \in \mathsf{HS}$ **do**
  $\mathsf{st}_i := \varnothing$
$(\mathsf{msg}, \mathsf{sig}) \leftarrow \mathcal{A}^{\mathsf{H}, (\mathcal{O}_{\mathsf{ShareSign}_r})_{r\in[R_{\mathsf{sig}}]}}(\mathsf{vk}, (\mathsf{sk}_i)_{i\in\mathsf{CS}}, \mathsf{st}_{\mathcal{A}})$
if $f_{\mathsf{fail}} = \top$ **then**
  return 0
if $(\mathsf{msg} \in L_{\mathsf{sig}})$ or $\neg\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$ **then**
  return 0
return 1

$\mathsf{ShareSign}_1(\mathsf{vk}, i, \mathsf{sk}_i, \mathsf{st}_i)$

$\mathsf{cmt}_i \xleftarrow{\$} \{0,1\}^{2\lambda}$
$\mathsf{UnopenedCmt}[(i, \mathsf{cmt}_i)] = \top$
return $(\mathsf{pm}_1^i := \mathsf{cmt}_i, \mathsf{st}_i)$

$\mathsf{ShareSign}_2(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, i, \mathsf{sk}_i, \mathsf{st}_i, \mathsf{pm}_1)$

require $\mathsf{act} \subseteq [N] \wedge i \in \mathsf{act}$
assert $\mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] = \top$
$\mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] := \bot$
$\mathbf{r}_i \leftarrow \chi_{\mathbf{r}}$
$\mathbf{w}_i := [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}_i$
$\mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, \mathbf{w}_i) := \mathsf{pm}_1^i$
// Program random oracle
$\mathsf{st}_i := \mathsf{st}_i \cup \{(\mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, \mathbf{w}_i, \mathbf{r}_i)\}$
return $(\mathsf{pm}_2^i := \mathbf{w}_i, \mathsf{st}_i)$

Figure 6.8: Second hybrid of the unforgeability proof of Threshold ML-DSA. If the random oracle is programmed on a previously queried index, consider that the adversary loses, i.e. f_{fail} is set to $\top$. Difference with the previous hybrid are highlighted.

The view of the adversary $\mathcal{A}$ differs if they called the random oracle of one of the $\mathbf{w}_i$ before round 2 was executed, i.e.

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_1} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_2}\right| \leqslant \Pr[E]$$

where E is the event "$\mathsf{H}_{\mathsf{cmt}}$ was queried on a honest $\mathbf{w}_i$ before round 2 in Game_1".

By union-bound, we can reduce this to a single call to $\mathcal{O}_{\mathsf{ShareSign}_1}$.

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_1} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_2}\right| \leqslant \sum_{s=1}^{Q_s} \Pr[E'_s] \tag{4}$$

where E'_s is the event "The $\mathbf{w}_i$ produced by the s-th call to $\mathcal{O}_{\mathsf{ShareSign}_1}$ is queried on $\mathsf{H}_{\mathsf{cmt}}$ before round 2 in Game_1".

We denote the above individual probabilities p_s for $s \in [Q_s]$. Formally, we introduce in Fig. 6.9 an intermediary game Int_1^s in which $\mathcal{A}$ wins with probability exactly p_s. We also introduce a tweak of Int_1^s, named Int_2^s, where we replace the s-th commitment $\mathbf{w}_i$ by a uniform value in R_q^k.

First, the difference in advantage between Int_1^s and Int_2^s reduces to $\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{r}}}$. Formally, we can define an adversary $\mathcal{E}^s$ against the $\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{r}}}$ problem such that:

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Int}_1^s} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Int}_2^s}\right| = \left|p_s - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Int}_2^s}\right| \leqslant \mathsf{Adv}_{\mathcal{E}^s}^{\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{r}}}} \tag{5}$$

Also, we note that in Int_2^s, the probability that one call of $\mathcal{A}$ to the random oracle queries the s-th $\mathbf{w}_i$ is bounded by the min-entropy of $\mathbf{w}_i \xleftarrow{\$} R_q^k$, i.e. by q^{-nk}. By union bound, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Int}_2^s} \leqslant Q_{\mathsf{H}_{\mathsf{cmt}}} \cdot q^{-nk}$.

Int^s_1

$L_{\mathsf{sig}} := \varnothing$
$\mathsf{Commitments}[\cdot] := \varnothing$
$\mathsf{Cnt} := 0$
$\mathsf{wins} := 0$
$(\mathsf{CS}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathsf{H}}(1^\lambda, N, T, t)$
assert $\mathsf{CS} \subseteq [N] \wedge |\mathsf{CS}| \leqslant T - 1$
$\mathsf{HS} := [N] \backslash \mathsf{CS}$
$(\mathsf{vk}, (\mathsf{sk}_i)_{i \in [N]}) \leftarrow \mathsf{Keygen}(1^\lambda, N, T)$
for $i \in \mathsf{HS}$ **do**
  $\mathsf{st}_i := \varnothing$
$(\mathsf{msg}, \mathsf{sig}) \leftarrow \mathcal{A}^{\mathsf{H}, (\mathcal{O}_{\mathsf{ShareSign}_r})_{r \in [R_{\mathsf{sig}}]}}(\mathsf{vk}, (\mathsf{sk}_i)_{i \in \mathsf{CS}}, \mathsf{st}_{\mathcal{A}})$
return wins

$\mathsf{ShareSign}_1(\mathsf{vk}, i, \mathsf{sk}_i, \mathsf{st}_i)$

$\mathsf{Cnt} := \mathsf{Cnt} + 1$
$\mathbf{r}_i \leftarrow \chi_{\mathbf{r}}$
$\mathbf{w}_i := [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}_i$
$\mathsf{Commitments}[\mathsf{Cnt}] := \mathbf{w}_i$
$\mathsf{st}_i := \mathsf{st}_i \cup \{(\mathsf{cmt}_i, \mathbf{w}_i, \mathbf{r}_i)\}$
$\mathsf{cmt}_i = \mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, \mathbf{w}_i) \in \{0,1\}^{2\lambda}$
return $(\mathsf{pm}^i_1 := \mathsf{cmt}_i, \mathsf{st}_i)$

$\mathsf{ShareSign}_2(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, i, \mathsf{sk}_i, \mathsf{st}_i, \mathsf{pm}_1)$

require $\mathsf{act} \subseteq [N] \wedge i \in \mathsf{act}$
assert $(\mathsf{pm}^i_1, \cdot, \cdot) \in \mathsf{st}_i$
Pick $(\mathsf{cmt}_i, \mathbf{w}_i, \mathbf{r}_i)$ from st_i with $\mathsf{pm}^i_1 = \mathsf{cmt}_i$
if $\mathsf{Commitments}[s] = \mathbf{w}_i$ **then**
  $\mathsf{Commitments}[s] = \bot$
  return $\bot$
$\mathsf{st}_i := \mathsf{st}_i \backslash \{(\mathsf{cmt}_i, \mathbf{w}_i, \mathbf{r}_i)\}$
$\mathsf{st}_i := \mathsf{st}_i \cup \{(\mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, \mathbf{w}_i, \mathbf{r}_i)\}$
return $(\mathsf{pm}^i_2 := \mathbf{w}_i, \mathsf{st}_i)$

$\mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, \mathbf{w})$

if $\mathsf{Commitments}[s] = \mathbf{w}$ **then**
  $\mathsf{wins} = 1$
// Other lines are identical

Int^s_2

// Identical to Int^s_1

$\mathsf{ShareSign}_1(\mathsf{vk}, i, \mathsf{sk}_i, \mathsf{st}_i)$

$\mathsf{Cnt} := \mathsf{Cnt} + 1$
if $\mathsf{Cnt} \neq s$ **then**
  $\mathbf{r}_i \leftarrow \chi_{\mathbf{r}}$
  $\mathbf{w}_i := [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}_i$
  $\mathsf{st}_i := \mathsf{st}_i \cup \{(\mathsf{cmt}_i, \mathbf{w}_i, \mathbf{r}_i)\}$
else
  $\mathbf{w}_i \xleftarrow{\$} R^k_q$
  $\mathsf{st}_i := \mathsf{st}_i \cup \{(\mathsf{cmt}_i, \mathbf{w}_i, \cdot)\}$
$\mathsf{Commitments}[\mathsf{Cnt}] := \mathbf{w}_i$
$\mathsf{cmt}_i = \mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, \mathbf{w}_i) \in \{0,1\}^{2\lambda}$
return $(\mathsf{pm}^i_1 := \mathsf{cmt}_i, \mathsf{st}_i)$

Figure 6.9: Intermediary games for the second hybrid of the unforgeability proof of Threshold ML-DSA. Difference with Game_1 are highlighted.

By combining the above inequality with Eq. (4) and Eq. (5), we conclude:

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_1} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_2}\right| \leqslant Q_s \cdot Q_{\mathsf{H}_{\mathsf{cmt}}} \cdot q^{-nk} + Q_s \cdot \mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{r}}}}$$

where $\mathcal{B}_1$ is the adversary $\mathcal{E}^s$ maximizing $\mathsf{Adv}_{\mathcal{E}^s}^{\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{r}}}}$.

Remark 6.4.6. As a useful remark for the following hybrids, we note that a given $\mathbf{w}_i$ cannot be used twice by the same party after Game_2 as then the random oracle programming would fail, and the adversary loses.

Game_3. In this hybrid, we sample a challenge c and its seed $\tilde{c}$ in advance during round 2. When the last honest hash cmt_i obtains a corresponding $\mathbf{w}_i$ programmed, we program the values $\mathsf{H}(\mu||\mathbf{w}_\top) = \tilde{c}$ and $\mathsf{SampleInBall}(\tilde{c}) = c$. This is formalized in Fig. 6.10.

Game_3

1 : $L_{\mathsf{sig}}, \mathsf{UnopenedCmt}[\cdot] := \varnothing$
2 : $\mathsf{ToProgram}, \mathsf{Programmed}[\cdot] := \varnothing$
3 : // Other lines are identical to Game_2

$\mathsf{ShareSign}_2(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, i, \mathsf{sk}_i, \mathsf{st}_i, \mathsf{pm}_1)$

1 : **require** $\mathsf{act} \subseteq [N] \wedge i \in \mathsf{act}$
2 : **assert** $\mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] = \top$
3 : $\mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] := \bot$
4 : **if** $(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \cdot) \notin \mathsf{ToProgram}$ **then**
5 : $\tilde{c} \xleftarrow{\$} \{0,1\}^{2\cdot\lambda}; c \leftarrow \mathcal{C}$
6 : $\mathsf{ToProgram} := \mathsf{ToProgram} \cup \{(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \tilde{c}, c)\}$
7 : Pick (a_1, a_2, a_3, c)
8 : from $\mathsf{ToProgram}$ s.t. $(a_1, a_2, a_3) = (\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg})$
9 : $\mathbf{r}_i \leftarrow \chi_{\mathbf{r}}$
10 : $\mathbf{w}_i := [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}_i$
11 : $\mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, \mathbf{w}_i) := \mathsf{pm}_1^i$
12 : **for** $(\mathsf{act}', (\mathsf{cmt}'_j)_{j\in\mathsf{act}'}, \mathsf{msg}', c') \in \mathsf{ToProgram}$ **do**
13 : **if** $(i \in \mathsf{act}') \wedge (\mathsf{cmt}'_i = \mathsf{pm}_1^i) \wedge (\forall j \in \mathsf{act}'\backslash\{i\}, \exists \mathbf{w}_j, \mathsf{cmt}_j = \mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, j, \mathbf{w}_j))$ **then**
14 : // Find all the new programmable challenges
15 : $\mathsf{ToProgram} := \mathsf{ToProgram}\backslash\{(\mathsf{act}', (\mathsf{cmt}'_j)_{j\in\mathsf{act}'}, \mathsf{msg}', \tilde{c}', c')\}$
16 : $\mathbf{w} := \sum_{j\in\mathsf{act}'} \mathbf{w}_j$; $(\mathbf{w}_\top, \mathbf{w}_\bot) = \mathsf{HighBits}_q(\mathbf{w}, 2\gamma_2)$
17 : $\mathsf{H}(\mu||\mathbf{w}_\top) := \tilde{c}'$; $\mathsf{SampleInBall}(\tilde{c}') := c'$ // Program random oracle
18 : $\mathsf{st}_i := \mathsf{st}_i \cup \{(\mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, \mathbf{w}_i, \mathbf{r}_i)\}$
19 : **return** $(\mathsf{pm}_2^i := \mathbf{w}_i, \mathsf{st}_i)$

Figure 6.10: Third hybrid of the unforgeability proof of Threshold ML-DSA. If the random oracle is programmed on a previously queried index, consider that the adversary loses, i.e. f_{fail} is set to $\top$. Difference with the previous hybrid are highlighted.

The proof for this hybrid is analogous to the previous one. First, these changes do not bias H and SampleInBall as the sampled challenges c and seeds $\tilde{c}$ are used at most once for programming H and SampleInBall, and are not provided to the adversary before programming. The view of the adversary hence differs only if it queried the random oracle (i) H on some $\mu||\mathbf{w}_\top$, where $\mathbf{w}$ are

the high bits of $\hat{\mathbf{w}} = \sum_{i \in \mathsf{act}} \mathbf{w}_i$ before it is programmed in round 2, or (ii) SampleInBall on $\tilde{c}$ before it is programmed.

First, we can easily bound the probability that SampleInBall is queried on some $\tilde{c}$ before it is programmed by $Q_{\mathsf{H}} Q_s \cdot 2^{-2\cdot\lambda}$.

Then, we are interested in the probability that H is queried on some $\mu||\mathbf{w}_\top$ before it is programmed. For each $s \in Q_s$, we reexpress as an advantage the probability p_s that the s-th $\mathbf{w}_i$ produced during round 2 is used to compute a $\mathbf{w}_\top$ that was previously queried on the random oracle H. Formally, we define the game Int^s_3 in Fig. 6.11, that verifies $p_s = \mathsf{Adv}^{\mathsf{Int}^s_3}_{\mathcal{A}}$. By union bound, we have $\left|\mathsf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Game}_3}_{\mathcal{A}}\right| \leqslant \sum_{s\in[Q_s]} p_s$.

Int^s_3

1 : $L_{\mathsf{sig}}, \mathsf{UnopenedCmt}[\cdot], := \varnothing$
2 : $\mathsf{ToProgram} := \varnothing$
3 : $\mathsf{Cnt} := 0$
4 : $\mathsf{wins} := 0$
5 : $f_{\mathsf{fail}} := \bot$
6 : $(\mathsf{CS}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathsf{H}}(1^\lambda, N, T)$
7 : **assert** $\mathsf{CS} \subseteq [N] \wedge |\mathsf{CS}| \leqslant T-1$
8 : $\mathsf{HS} := [N] \backslash \mathsf{CS}$
9 : $(\mathsf{vk}, (\mathsf{sk}_i)_{i\in[N]}) \leftarrow \mathsf{Keygen}(1^\lambda, N, T)$
10 : **for** $i \in \mathsf{HS}$ **do**
11 : $\quad \mathsf{st}_i := \varnothing$
12 : $(\mathsf{msg}, \mathsf{sig}) \leftarrow \mathcal{A}^{\mathsf{H}, (\mathcal{O}_{\mathsf{ShareSign}_r})_{r\in[R_{\mathsf{sig}}]}}(\mathsf{vk}, (\mathsf{sk}_i)_{i\in\mathsf{CS}}, \mathsf{st}_{\mathcal{A}})$
13 : **return** wins

$\mathsf{ShareSign}_2(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, i, \mathsf{sk}_i, \mathsf{st}_i, \mathsf{pm}_1)$

1 : $\mathsf{Cnt} := \mathsf{Cnt} + 1$
2 : **require** $\mathsf{act} \subseteq [N] \wedge i \in \mathsf{act}$
3 : **assert** $\mathsf{UnopenedCmt}[(i, \mathsf{pm}^i_1)] = \top$
4 : $\mathsf{UnopenedCmt}[(i, \mathsf{pm}^i_1)] := \bot$
5 : **if** $(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \cdot) \notin \mathsf{ToProgram}$ **then**
6 : $\quad \mathsf{ToProgram} := \mathsf{ToProgram} \cup \{(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \cdot)\}$
7 : $\mathbf{r}_i \leftarrow \chi_{\mathbf{r}}; \mathbf{w}_i := [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}_i$
8 : $\mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, \mathbf{w}_i) := \mathsf{pm}^i_1$
9 : **for** $(\mathsf{act}', (\mathsf{cmt}'_j)_{j\in\mathsf{act}'}, \mathsf{msg}', \cdot) \in \mathsf{ToProgram}$ **do**
10 : $\quad$ **if** $(i \in \mathsf{act}') \wedge (\mathsf{cmt}'_i = \mathsf{pm}^i_1) \wedge (\forall j \in \mathsf{act}' \backslash \{i\}, \exists \mathbf{w}_j, \mathsf{cmt}_j = \mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, j, \mathbf{w}_j))$ **then**
11 : $\quad\quad \mathsf{ToProgram} := \mathsf{ToProgram} \backslash \{(\mathsf{act}', (\mathsf{cmt}'_j)_{j\in\mathsf{act}'}, \mathsf{msg}', \tilde{c}', c')\}$
12 : $\quad\quad \mathbf{w} := \sum_{j\in\mathsf{act}'} \mathbf{w}_j; (\mathbf{w}_\top, \mathbf{w}_\bot) = \mathsf{HighBits}_q(\mathbf{w}, 2\gamma_2)$
13 : $\quad\quad$ **if** $\mathsf{Cnt} = s \wedge \exists (\mu||\mathbf{w}_\top) \in \mathsf{H}$ **then** // H already queried on $\mathbf{w}_\top$
14 : $\quad\quad\quad \mathsf{wins} = 1$
15 : $\mathsf{st}_i := \mathsf{st}_i \cup \{(\mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, \mathbf{w}_i, \mathbf{r}_i)\}$
16 : **return** $(\mathsf{pm}^i_2 := \mathbf{w}_i, \mathsf{st}_i)$

Int^s_4

1 : // Identical to Int^s_3

$\mathsf{ShareSign}_2(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, i, \mathsf{sk}_i, \mathsf{st}_i, \mathsf{pm}_1)$

1 : $\mathsf{Cnt} := \mathsf{Cnt} + 1$
2 : **require** $\mathsf{act} \subseteq [N] \wedge i \in \mathsf{act}$
3 : **assert** $\mathsf{UnopenedCmt}[(i, \mathsf{pm}^i_1)] = \top$
4 : $\mathsf{UnopenedCmt}[(i, \mathsf{pm}^i_1)] := \bot$
5 : **if** $(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \cdot) \notin \mathsf{ToProgram}$ **then**
6 : $\quad \mathsf{ToProgram} := \mathsf{ToProgram} \cup \{(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \cdot)\}$
7 : **if** $\mathsf{Cnt} = s$ **then** $\mathbf{w}_i \xleftarrow{\$} R^k_q$
8 : **else** $\mathbf{r}_i \leftarrow \chi_{\mathbf{r}}; \mathbf{w}_i := [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}_i$
9 : // Rest is identical

Figure 6.11: Intermediary games for the third hybrid of the unforgeability proof of Threshold ML-DSA. Difference with Game_2 are highlighted.

We additionally define the game Int^s_4 in Fig. 6.11, where we replace the s-th $\mathbf{w}_i$ by a uniform value in R^k_q. The advantage difference between Int^s_3 and Int^s_4 again reduces to $\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{r}}}$: $\left|\mathsf{Adv}^{\mathsf{Int}^s_3}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Int}^s_4}_{\mathcal{A}}\right| \leqslant \mathsf{Adv}^{\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{r}}}}_{\mathcal{E}^s}$ for an adversary $\mathcal{E}^s$ against the $\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{r}}}$ problem.

Then, we can see that the probability that $\mathcal{A}$ wins in Int^s_4 can be bounded using the min-entropy of $\mathsf{HighBits}_q(\mathcal{U}(R^k_q), 2\gamma_2)$. Indeed, the s-th $\mathbf{w}_i$ is sampled uniformly from R^k_q and ensures that each $\mathbf{w}$ that needs to be programmed follows the distribution $\mathsf{HighBits}_q(\mathcal{U}(R^k_q), 2\gamma_2)$. As there are at most Q_s $\mathbf{w}$ to check, we obtain the bound $\mathsf{Adv}^{\mathsf{Int}^s_4}_{\mathcal{A}} \leqslant Q_{\mathsf{H}} Q_s \cdot \left(\frac{2\gamma_2+1}{q}\right)^{kn}$.

Putting together the different inequalities obtained, we finally deduce the existence of an adversary $\mathcal{B}_2$ against the $\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{r}}}$ such that:

$$\left|\mathsf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}} - \mathsf{Adv}^{\mathsf{Game}_3}_{\mathcal{A}}\right| \leqslant Q_{\mathsf{H}} Q_s^2 \cdot \left(\frac{2\gamma_2+1}{q}\right)^{kn} + Q_{\mathsf{H}} Q_s \cdot 2^{-2\cdot\lambda} + Q_s \cdot \mathsf{Adv}^{\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{r}}}}_{\mathcal{B}_2}$$

Game_4. In this hybrid, we ensure that $\mathsf{H}_{\mathsf{cmt}}$ has no collisions and that the adversary cannot find preimages of hashes cmt_i after passing them to the second signing oracle. We introduce a table Cmt that records every cmt sampled in $\mathsf{H}_{\mathsf{cmt}}$ or in $\mathcal{O}_{\mathsf{ShareSign}_1}$, as well as those passed to the second signing oracle. Whenever a new hash cmt is sampled, we ensure that it was not previously added to Cmt, or the challenger sets the flag f_{fail} to $\top$ in order to abort the game. This is formalized in Fig. 6.12.

Game_4

$L_{\mathsf{sig}}, \mathsf{UnopenedCmt}[\cdot], \mathsf{Cmt} := \varnothing$
// Rest is identical to Game_3

$\mathsf{ShareSign}_1(\mathsf{vk}, i, \mathsf{sk}_i, \mathsf{st}_i)$

$\mathsf{cmt}_i \xleftarrow{\$} \{0,1\}^{2\lambda}$
$\mathsf{UnopenedCmt}[(i, \mathsf{cmt}_i)] = \top$
if $\mathsf{cmt}_i \in \mathsf{Cmt}$ **then**
  $f_{\mathsf{fail}} := \top$
  return $\bot$
$\mathsf{Cmt} := \mathsf{Cmt} \cup \{\mathsf{cmt}_i\}$
return $(\mathsf{pm}_1^i := \mathsf{cmt}_i, \mathsf{st}_i)$

$\mathsf{ShareSign}_2(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, i, \mathsf{sk}_i, \mathsf{st}_i, \mathsf{pm}_1)$

$\mathsf{Cmt} := \mathsf{Cmt} \cup \{\mathsf{cmt}_j := \mathsf{pm}_1^j\}_{j \in \mathsf{act}}$
// The rest is identical

$\mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, \mathbf{w})$

if $Q_{\mathsf{H}_{\mathsf{cmt}}}[(\mathsf{vk}, \mathbf{w})] = \bot$ **then**
  $\mathsf{cmt} \xleftarrow{\$} \{0,1\}^{2\lambda}$
  $Q_{\mathsf{H}_{\mathsf{cmt}}}[(\mathsf{vk}, \mathbf{w})] = \mathsf{cmt}$
  if $\mathsf{cmt}_i \in \mathsf{Cmt}$ **then**
    $f_{\mathsf{fail}} := \top$
    return $\bot$
  $\mathsf{Cmt} := \mathsf{Cmt} \cup \{\mathsf{cmt}_i\}$
return $Q_{\mathsf{H}_{\mathsf{cmt}}}[(\mathsf{vk}, \mathbf{w})] = \bot$

Figure 6.12: Fourth hybrid of the unforgeability proof of Threshold ML-DSA. Difference with the previous hybrid are highlighted.

The view of the adversary differs in Game_4 if it was previously able to (i) find a pre-image of a cmt_i after passing it to $\mathcal{O}_{\mathsf{ShareSign}_2}$, or (ii) if a given cmt was sampled twice.

We can bound the probability of (i) due to the pre-image resistance of the hash function $\mathsf{H}_{\mathsf{cmt}}$, and with an union bound over all the commits passed by the adversary to $\mathcal{O}_{\mathsf{ShareSign}_2}$ which gives a bound $Q_{\mathsf{H}_{\mathsf{cmt}}} \cdot T \cdot Q_s \cdot 2^{-2\lambda}$.

As for (ii), we rely on the collision resistance of $\mathsf{H}_{\mathsf{cmt}}$. Since the commitments are sampled uniformly from $\{0,1\}^{2\lambda}$, we can bound the probability of (ii) by $\frac{(Q_{\mathsf{H}_{\mathsf{cmt}}}+Q_s)^2}{2^{2\lambda}}$.

We conclude that,

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_3} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_4}\right| \leqslant \frac{Q_{\mathsf{H}_{\mathsf{cmt}}} \cdot T \cdot Q_s + (Q_{\mathsf{H}_{\mathsf{cmt}}} + Q_s)^2}{2^{2\lambda}}$$

Game_5. In this hybrid, we ensure that H on $tr||\mathsf{msg}$ does never return twice the same $\tilde{c}$ for two different messages. This is formalized in Fig. 6.13.

Game_5

1 : // Identical to Game_4

$\mathsf{H}(\mathsf{str})$

1 : **if** $\mathsf{str} = tr||\mathsf{msg}$ and $Q_{\mathsf{H}}[tr||\mathsf{msg}] = \bot$ **then**
2 : $\quad\tilde{c} \leftarrow \{0,1\}^{512}$
3 : $\quad Q_{\mathsf{H}}[tr||\mathsf{msg}] = \tilde{c}$
4 : $\quad$ **if** $\exists \mathsf{msg}', Q_{\mathsf{H}}[tr||\mathsf{msg}'] = \tilde{c}$ or $(\cdot, \tilde{c}, \cdot) \in \mathsf{ToProgram}$ **then**
5 : $\quad\quad f_{\mathsf{fail}} := \top$
6 : $\quad\quad$ **return** $\bot$
7 : // Rest is identical

Figure 6.13: Fifth hybrid of the unforgeability proof of Threshold ML-DSA. Difference with the previous hybrid are highlighted.

As for the previous hybrid, this is ensured by the collision resistance of H due to the large space of $\tilde{c}$.

We conclude that,

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_4} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_5}\right| \leqslant Q_{\mathsf{H}}^2 \cdot 2^{-512}$$

Game_6. In this hybrid, we ensure that the challenge used in round 3 is the same as the one sampled in round 2, and we precompute the responses $\mathbf{z}_i$ in advance. This is formalized in Fig. 6.14.

We want to show that responses are identically distributed in Game_6. This is the case if programmed responses use the same c in both round 2 and round 3, and if they are used at most once.

Now, we can first observe that if the hash checks in round 3 pass, then all the $\mathsf{cmt}_i := \mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, \mathbf{w}_i)$ are correctly defined. Recall that Game_4 ensured that $\mathsf{H}_{\mathsf{cmt}}$ has no collisions, and that pre-images cannot be found after round 2 is called. Hence, if these checks pass, then it means that in round 2, all the hashes cmt_i either: (i) already had pre-images, or (ii) were produced by an honest party in $\mathcal{O}_{\mathsf{ShareSign}_1}$ and were waiting for programming. Eventually, all the missing cmt_i from (ii) must thus have been programmed in $\mathcal{O}_{\mathsf{ShareSign}_2}$ and as the challenger programs SampleInBall as soon as all the $\mathbf{w}_i$ are defined, then it ensures that the same c is used in round 2 and in round 3.

Additionally, responses are used at most once by a given party due to the collision resistance of $\mathsf{H}_{\mathsf{cmt}}$ and Remark 6.4.6 which ensures that party i will accept cmt_i at most once.

All in all, we conclude that the adversary's view is identically distributed in Game_5 and Game_6 so,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_6} = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_5}$$

Game_6

$L_{\mathsf{sig}}, \mathsf{UnopenedCmt}[\cdot], \mathsf{Cmt}, \mathsf{Responses}[\cdot] := \varnothing$
// Rest is identical to Game_4

$\mathsf{ShareSign}_2(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, i, \mathsf{sk}_i, \mathsf{st}_i, \mathsf{pm}_1)$

require $\mathsf{act} \subseteq [N] \wedge i \in \mathsf{act}$
assert $\mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] = \top$
$\mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] := \bot$
if $(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \cdot) \notin \mathsf{ToProgram}$ **then**
  $\tilde{c} \xleftarrow{\$} \{0,1\}^{2\cdot\lambda}; c \leftarrow \mathcal{C}$
  $\mathsf{ToProgram} := \mathsf{ToProgram} \cup \{(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \tilde{c}, c)\}$
Pick (a_1, a_2, a_3, c) from ToProgram s.t. $(a_1, a_2, a_3) = (\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg})$
$\mathbf{r}_i \leftarrow \chi_{\mathbf{r}}$
$\mathbf{w}_i := [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}_i$
$\mathbf{m} := \mathsf{RSSRecover}(\mathsf{act})$
$\mathbf{s}_i^{\mathsf{part}} := \sum_{I\in m_i} \mathbf{s}_I$
$\mathsf{Responses}[(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, i)] := \mathbf{z}_i$
// Rest is identical

$\mathsf{ShareSign}_3(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, \mathsf{pm}_2, i, \mathsf{sk}_i, \mathsf{st}_i)$

assert $(\mathsf{act}, \mathsf{msg}, \cdot, \mathsf{pm}_2^i, \cdot) \in \mathsf{st}_i$
Pick $(\mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, \mathbf{w}_i, \cdot)$ from st_i with $\mathsf{pm}_2^i = \mathbf{w}_i$
Parse $\mathsf{pm}_1 = (\mathsf{cmt}_j)_j, \mathsf{pm}_2 = (\mathbf{w}_j)_{j\neq i}$
for $j \in \mathsf{act}$ **do**
  if $\mathsf{cmt}_j \neq \mathsf{H}_{\mathsf{cmt}}(\mathsf{vk}, j, \mathbf{w}_j)$ **then**
    abort
      // Check hash commit.
$\mathsf{st}_i := \mathsf{st}_i \backslash \{(\mathsf{act}, \mathsf{msg}, \mathsf{pm}_1, \mathbf{w}_i, \cdot)\}$
if $\mathbf{z}_i^1, \mathbf{z}_i^2 := \mathsf{Responses}[(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, i)]$ **then**
  return $(\mathbf{z}_i^1, \mathsf{st}_i)$
else abort

Figure 6.14: Sixth hybrid of the unforgeability proof of Threshold ML-DSA. Difference with the previous hybrid are highlighted.

Game_7. In this hybrid, we no longer sample $\mathbf{r}_i$ out of the rejection sampling. In case the rejection sampling aborts, we sample a fresh $\mathbf{z}_i$ from the aborting distribution $(\mathbf{z}|\mathsf{rej})_{\mathbf{v}=c\cdot\mathbf{s}_i^{\mathsf{part}}}$. Then, we compute $\mathbf{w}_i$ as $[\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{z}_i - \mathbf{b}_i^{\mathsf{part}}$, where $\mathbf{b}_i^{\mathsf{part}} = [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{s}_i^{\mathsf{part}}$. This is formalized in Fig. 6.15.

We can see that the view of the adversary is identically distributed. Indeed, in Game_6 we have $[\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{z}_i = c \cdot \underbrace{[\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{s}_i^{\mathsf{part}}}_{\mathbf{b}_i^{\mathsf{part}}} + \underbrace{[\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{r}_i}_{\mathbf{w}_i}$ for $\mathbf{z}_i = \mathbf{r}_i + c \cdot \mathbf{s}_i^{\mathsf{part}}$ (even in case it is rejected). We can thus equivalently write $\mathbf{w}_i = [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{z}_i - c \cdot \mathbf{b}_i^{\mathsf{part}}$ and sample $\mathbf{z}_i$ first. The sampling of rejected $\mathbf{z}_i$ from $(\mathbf{z}|\mathsf{rej})_{\mathbf{v}=c\cdot\mathbf{s}_i^{\mathsf{part}}}$ finally ensures the same distribution of $\mathbf{z}_i$ in the above equality in case of rejections.

So,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_7} = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_6}$$

Game_7

1 : // Identical to Game_6

$\mathsf{ShareSign}_2(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, i, \mathsf{sk}_i, \mathsf{st}_i, \mathsf{pm}_1)$

1 : **require** $\mathsf{act} \subseteq [N] \wedge i \in \mathsf{act}$
2 : **assert** $\mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] = \top$
3 : $\mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] := \bot$
4 : **if** $(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \cdot) \notin \mathsf{ToProgram}$ **then**
5 : $\quad \tilde{c} \xleftarrow{\$} \{0,1\}^{2\cdot\lambda}; c \leftarrow \mathcal{C}$
6 : $\quad \mathsf{ToProgram} := \mathsf{ToProgram} \cup \{(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \tilde{c}, c)\}$
7 : Pick (a_1, a_2, a_3, c) from ToProgram s.t. $(a_1, a_2, a_3) = (\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg})$
8 : $\mathbf{m} := \mathsf{RSSRecover}(\mathsf{act})$
9 : $\mathbf{s}_i^{\mathsf{part}} := \sum_{I\in m_i} \mathbf{s}_I$
10 : $\mathbf{z}_i \leftarrow \mathsf{Rej}(c \cdot \mathbf{s}_i^{\mathsf{part}}, \chi_{\mathbf{z}}, \chi_{\mathbf{r}}, M)$
11 : **if** $\mathbf{z}_i = \bot$ **then** // if reject
12 : $\quad \mathbf{z}_i \leftarrow (\mathbf{z}|\mathrm{rej})_{v=c\cdot\mathbf{s}_i^{\mathsf{part}}}$
13 : $\mathbf{b}_i^{\mathsf{part}} := \sum_{I\in m_i} \mathbf{b}_I$
14 : $\mathbf{w}_i := \begin{bmatrix}\mathbf{A} & \mathbf{I}\end{bmatrix} \cdot \mathbf{z}_i - c \cdot \mathbf{b}_i^{\mathsf{part}}$
15 : **if** $\mathbf{z}_i \neq \bot$ **then**
16 : $\quad \mathsf{Responses}[(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, i)] := \mathbf{z}_i$
17 : // Rest is identical

Figure 6.15: Seventh hybrid of the unforgeability proof of Threshold ML-DSA. Difference with the previous hybrid are highlighted.

Game_8. In this hybrid, we ensure that the values $c \cdot \mathbf{s}_i^{\mathsf{part}}$ have a twisted norm at most B. That is, whenever this is not the case we set the flag f_{fail} to $\top$ to abort the game. This is formalized in Fig. 6.16.

By assumption on B, for any choice of act this is true with overwhelming probability over the randomness of c and of the secrets.

Since the adversary chooses act, it may not be independent of the secrets and we cannot directly conclude. However, we can simply guess act since the number of possible sets $\binom{N}{T}$ is polynomially bounded.

By union-bound over all the calls to a signing oracle, we conclude:

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_7} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_8}\right| \leqslant Q_s \cdot \binom{N}{T} \cdot \mathsf{negl}(\lambda)$$

Game_9. In this hybrid, we replace the aborting $\mathbf{w}_i$ by a uniform value in R_q^k. This is formalized in Fig. 6.17.

There are up to Q_s $\mathbf{w}_i$ to replace. We define a family of games G^p for $p \in [Q_s + 1]$, replacing one by one the $\mathbf{w}_i$ by a uniform vector, which verifies:

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_8} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_9}\right| \leqslant \sum_{s\in[Q_s]} \left|\mathsf{Adv}_{\mathcal{A}}^{G^{p+1}} - \mathsf{Adv}_{\mathcal{A}}^{G^p}\right|$$

Let $p \in [Q_s + 1]$. We want to bound $\left|\mathsf{Adv}_{\mathcal{A}}^{G^{p+1}} - \mathsf{Adv}_{\mathcal{A}}^{G^p}\right|$.

Game_8

// Identical to Game_7

$\mathsf{ShareSign}_2(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, i, \mathsf{sk}_i, \mathsf{st}_i, \mathsf{pm}_1)$

require $\mathsf{act} \subseteq [N] \wedge i \in \mathsf{act}$
assert $\mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] = \top$
$\mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] := \bot$
if $(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \cdot) \notin \mathsf{ToProgram}$ **then**
  $\tilde{c} \xleftarrow{\$} \{0,1\}^{2\cdot\lambda}; c \leftarrow \mathcal{C}$
  $\mathsf{ToProgram} := \mathsf{ToProgram} \cup \{(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \tilde{c}, c)\}$
Pick (a_1, a_2, a_3, c) from $\mathsf{ToProgram}$ s.t. $(a_1, a_2, a_3) = (\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg})$
$\mathbf{m} := \mathsf{RSSRecover}(\mathsf{act})$
$\mathbf{s}_i^{\mathsf{part}} := \sum_{I\in m_i} \mathbf{s}_I; (\mathbf{u}_1, \mathbf{u}_2) = \mathbf{s}_i^{\mathsf{part}}$
if $\|(v \cdot c \cdot \mathbf{u}_1, c \cdot \mathbf{u}_2)\|_2 > B$ **then**
  $f_{\mathsf{fail}} := \top$
  return $\bot$
$\mathbf{z}_i \leftarrow \mathsf{Rej}(c \cdot \mathbf{s}_i^{\mathsf{part}}, \chi_{\mathbf{z}}, \chi_{\mathbf{r}}, M)$
if $\mathbf{z}_i = \bot$ **then** // if reject
  $\mathbf{z}_i \leftarrow (\mathbf{z}|\mathrm{rej})_{v=c\cdot\mathbf{s}_i^{\mathsf{part}}}$
$\mathbf{b}_i^{\mathsf{part}} := \sum_{I\in m_i} \mathbf{b}_I$
$\mathbf{w}_i := [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{z}_i - c \cdot \mathbf{b}_i^{\mathsf{part}}$
if $\mathbf{z}_i \neq \bot$ **then**
  $\mathsf{Responses}[(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, i)] := \mathbf{z}_i$
// Rest is identical

Figure 6.16: Eighth hybrid of the unforgeability proof of Threshold ML-DSA. Difference with the previous hybrid are highlighted.

The core idea is to condition on the value of $v := (\mathsf{act}, i, \mathbf{s}_i^{\mathsf{part}}, c)$ used to define the distribution $(\mathbf{z}|\mathrm{rej})_{\mathbf{v}=c\cdot\mathbf{s}_i^{\mathsf{part}}}$:

$$\left|\mathsf{Adv}_{\mathcal{A}}^{G^{p+1}} - \mathsf{Adv}_{\mathcal{A}}^{G^p}\right| \leqslant \sum_{\mathbf{s},c} \left|\mathsf{Adv}_{\mathcal{A}}^{F'^v} - \mathsf{Adv}_{\mathcal{A}}^{F^v}\right| \cdot \Pr\left[v \text{ used by } G^s\right] \tag{6}$$

where F'^v and F^v are respectively the games G^{p+1} and G^p where we impose the set of signers act, signer i, and challenge c for the p-th $\mathbf{w}_i$ to replace, and sample the secrets $\mathbf{s}_I$ to be consistent with the value $\mathbf{s}_i^{\mathsf{part}}$.

We can then define adversaries $\mathcal{E}^v$ against the $\mathsf{MLWE}_{R_q,k,\ell,(\mathbf{z}|\mathrm{rej})_{\mathbf{v}=c\cdot\mathbf{s}_i^{\mathsf{part}}}}$ such that

$$\left|\mathsf{Adv}_{\mathcal{A}}^{F'^v} - \mathsf{Adv}_{\mathcal{A}}^{F^v}\right| = \mathsf{Adv}_{\mathcal{E}^v}^{\mathsf{MLWE}_{R_q,k,\ell,(\mathbf{z}|\mathrm{rej})_{\mathbf{v}=c\cdot\mathbf{s}_i^{\mathsf{part}}}}} \tag{7}$$

Game_8 ensured that the twisted norm of $\mathbf{v} = c \cdot \mathbf{s}_i^{\mathsf{part}}$ is bounded by B which ensures that $R_\infty^\varepsilon(\chi_{\mathbf{z}}||\chi_{\mathbf{r}} + \mathbf{v}) \leqslant M$ by Lemma 6.2.9. We can thus apply Lemma 6.3.2: there exists adversaries $\mathcal{E}'^v$ and $\mathcal{E}''^v$ such that

$$\mathsf{Adv}_{\mathcal{E}^v}^{\mathsf{MLWE}_{R_q,k,\ell,(\mathbf{z}|\mathrm{rej})\mathbf{v}}} \leqslant \frac{1}{1-\frac{1}{M}} \cdot \left(\mathsf{Adv}_{\mathcal{E}'^v}^{\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{r}}}} + \mathsf{Adv}_{\mathcal{E}''^v}^{\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{z}}}} + \delta^v\right) \tag{8}$$

where $\mathbf{v} = c \cdot \mathbf{s}_i^{\mathsf{part}}$, δ^v is the statistical distance between $\chi_{\mathbf{z}}$ and $(\mathbf{z}|\mathrm{rej})_{\mathbf{v}=c\cdot\mathbf{s}_i^{\mathsf{part}}}$.

Game_9

1 : // Identical to Game_8

$\mathsf{ShareSign}_2(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, i, \mathsf{sk}_i, \mathsf{st}_i, \mathsf{pm}_1)$

1 : **require** $\mathsf{act} \subseteq [N] \wedge i \in \mathsf{act}$
2 : **assert** $\mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] = \top$
3 : $\mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] := \bot$
4 : **if** $(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \cdot) \notin \mathsf{ToProgram}$ **then**
5 : $\quad \tilde{c} \xleftarrow{\$} \{0,1\}^{2\cdot\lambda}; c \leftarrow \mathcal{C}$
6 : $\quad \mathsf{ToProgram} := \mathsf{ToProgram} \cup \{(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \tilde{c}, c)\}$
7 : Pick (a_1, a_2, a_3, c) from ToProgram s.t. $(a_1, a_2, a_3) = (\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg})$
8 : $\mathbf{m} := \mathsf{RSSRecover}(\mathsf{act})$
9 : $\mathbf{s}_i^{\mathsf{part}} := \sum_{I\in m_i} \mathbf{s}_I; (\mathbf{u}_1, \mathbf{u}_2) = \mathbf{s}_i^{\mathsf{part}}$
10 : **if** $\|(v \cdot c \cdot \mathbf{u}_1, c \cdot \mathbf{u}_2)\|_2 > B$ **then**
11 : $\quad f_{\mathsf{fail}} := \top$
12 : $\quad$ **return** $\bot$
13 : $\mathbf{z}_i \leftarrow \mathsf{Rej}(c \cdot \mathbf{s}_i^{\mathsf{part}}, \chi_{\mathbf{z}}, \chi_{\mathbf{r}}, M)$
14 : **if** $\mathbf{z}_i = \bot$ **then**
15 : $\quad \mathbf{w}_i \xleftarrow{\$} R_q^k$
16 : **else**
17 : $\quad \mathbf{b}_i^{\mathsf{part}} := \sum_{I\in m_i} \mathbf{b}_I$
18 : $\quad \mathbf{w}_i := [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{z}_i - c \cdot \mathbf{b}_i^{\mathsf{part}}$
19 : $\quad \mathsf{Responses}[(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, i)] := \mathbf{z}_i$
20 : // Rest is identical

Figure 6.17: Nineth hybrid of the unforgeability proof of Threshold ML-DSA. Difference with the previous hybrid are highlighted.

We can also apply Lemma 6.3.1 to bound $\delta^v \leqslant \frac{2\varepsilon}{1-\varepsilon}$.
By combining Equations 6, 7, and 8, we obtain:

$$\left|\mathsf{Adv}_{\mathcal{A}}^{G^{p+1}} - \mathsf{Adv}_{\mathcal{A}}^{G^{p}}\right| \leqslant \frac{M}{M-1} \max_{\mathsf{act}, i, \|c\mathbf{s}\|\leqslant B} \left(\mathsf{Adv}_{\mathcal{E}'^v}^{\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{r}}}} + \mathsf{Adv}_{\mathcal{E}''^v}^{\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{z}}}}\right) + \frac{M}{M-1} \cdot \frac{2\varepsilon}{1-\varepsilon} + \mathsf{negl}(\lambda)$$

Finally, by summing all the above inequalities, we obtain

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_8} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_9}\right| \leqslant Q_s \frac{M}{M-1} \left(\mathsf{Adv}_{\mathcal{B}_3}^{\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{r}}}} + \mathsf{Adv}_{\mathcal{B}_4}^{\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{z}}}}\right) + Q_s \frac{M}{M-1} \cdot \frac{2\varepsilon}{1-\varepsilon} + \mathsf{negl}(\lambda)$$

where $\mathcal{B}_3$ is an adversary against the $\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{r}}}$ problem, $\mathcal{B}_4$ is an adversary against the $\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathbf{z}}}$ problem.

Game_{10}. In this hybrid, we replace the real rejection sampling algorithm $\mathsf{Rej}(c \cdot \mathbf{s}_i^{\mathsf{part}}, \chi_{\mathbf{z}}, \chi_{\mathbf{r}}, M; \mathbf{r}_i)$ by an ideal functionality $\mathsf{Ideal}(\chi_{\mathbf{z}}, M)$ which is independent of the secret. This is formalized in Fig. 6.18.

Game_{10}

// Identical to Game_6

$\mathsf{ShareSign}_2(\mathsf{vk}, \mathsf{act}, \mathsf{msg}, i, \mathsf{sk}_i, \mathsf{st}_i, \mathsf{pm}_1)$

require $\mathsf{act} \subseteq [N] \wedge i \in \mathsf{act}$
assert $\mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] = \top$
$\mathsf{UnopenedCmt}[(i, \mathsf{pm}_1^i)] := \bot$
if $(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \cdot) \notin \mathsf{ToProgram}$ **then**
  $\tilde{c} \xleftarrow{\$} \{0,1\}^{2\cdot\lambda}; c \leftarrow \mathcal{C}$
  $\mathsf{ToProgram} := \mathsf{ToProgram} \cup \{(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, \tilde{c}, c)\}$
Pick (a_1, a_2, a_3, c) from $\mathsf{ToProgram}$ s.t. $(a_1, a_2, a_3) = (\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg})$
$\mathbf{m} := \mathsf{RSSRecover}(\mathsf{act})$
$\mathbf{s}_i^{\mathsf{part}} := \sum_{I\in m_i} \mathbf{s}_I; (\mathbf{u}_1, \mathbf{u}_2) = \mathbf{s}_i^{\mathsf{part}}$
if $\|(\nu \cdot c \cdot \mathbf{u}_1, c \cdot \mathbf{u}_2)\|_2 > B$ **then**
  $f_{\mathsf{fail}} := \top$
  return $\bot$
$\mathbf{z}_i \leftarrow \mathsf{Ideal}(\chi_{\mathbf{z}}, M)$
if $\mathbf{z}_i = \bot$ **then**
  $\mathbf{w}_i \xleftarrow{\$} R_q^k$
else
  $\mathbf{b}_i^{\mathsf{part}} := \sum_{I\in m_i} \mathbf{b}_I$
  $\mathbf{w}_i := [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{z}_i - c \cdot \mathbf{b}_i^{\mathsf{part}}$
  $\mathsf{Responses}[(\mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}}, \mathsf{msg}, i)] := \mathbf{z}_i$
// Rest is identical

Figure 6.18: Tenth hybrid of the unforgeability proof of Threshold ML-DSA. Difference with the previous hybrid are highlighted.

We rely on the Rényi divergence to prove that this only slightly increase the probability that $\mathcal{A}$ wins the game. For conciseness, we will abuse Rényi divergence notations to take a random variable as input, implicitly corresponding to the divergence between the marginal distributions from Game_9 and Game_{10}.

We wish to apply a Rényi divergence argument, and its multiplicativity, to the responses $(\mathbf{z}_j)_{j\in[Q_s]}$ produced in signing oracles. However, these responses all depend on the secrets and are not independent, which prevent a simple application of multiplicativity.

To solve that, we will instead apply the Rényi properties from Lemma 6.2.5 to the tuple $X = ((\mathsf{sk}_i)_{i\in[N]}, \mathbf{z}_1, ..., \mathbf{z}_{Q_s})$.

By the data processing inequality of the Rényi divergence, since we can see Game_9 and Game_{10} as functions of X, their Rényi divergence is bounded by the Rényi divergence of X.

We then apply the multiplicativity of Rényi divergence Lemma 6.2.5:

$$R_\infty(\mathsf{Game}_9 || \mathsf{Game}_{10}) \leqslant \underbrace{R_\infty((\mathsf{sk}_i)_{i\in[N]})}_{=1} \cdot \prod_j r_j$$

where $r_j = \max_{(\mathsf{sk}_i)_{i\in[N]}} R_\infty(\mathbf{z}_j \mid (\mathsf{sk}_i)_{i\in[N]})$.

We can apply again the multiplicativity of the Rényi divergence:

$$R_\infty(\mathbf{z}_j \mid (\mathsf{sk}_i)_{i\in[N]}) \leqslant \max_{\|\mathbf{v}\|\leqslant B} R_\infty(\mathbf{z}_j \mid c_j \cdot \mathbf{s}_j^{\mathsf{part}} = \mathbf{v})$$

as $\mathbf{z}_j$ only differs between the games when $\|c \cdot \mathbf{s}_j^{\mathsf{part}}\| \leqslant B$.

We finally apply Lemma 6.3.1 to deduce that,

$$R_\infty(\mathsf{Game}_9 || \mathsf{Game}_{10}) \leqslant \left(1 + \frac{\varepsilon}{M-1}\right)^{Q_s}$$

Applying the probability preservation of the Rényi divergence, we get:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_9} \leqslant \left(1 + \frac{\varepsilon}{M-1}\right)^{Q_s} \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_{10}}$$

Game_{11}. In this hybrid, we remove the abort condition on the norm of $c \cdot \mathbf{s}_i^{\mathsf{part}}$.

This can only increase the winning probability of the adversary, hence

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_{10}} \leqslant \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_{11}}$$

Game_{12}. In this hybrid, we replace the public key by the rounding of a uniform value, i.e. $\mathsf{Power2Round}(\mathbf{b}, d)$ where $\mathbf{b} \xleftarrow{\$} R_q^k$. To preserve consistency, we fix a secret index I such that $I_0 \subseteq \mathsf{HS}$, i.e. $\mathbf{s}_{I_0}$ is not given to the adversary, and we replace $\mathbf{b}_{I_0}$ with $\mathbf{b}_{I_0} := \mathbf{b} - \sum_{I \neq I_0} \mathbf{b}_I$. This is formalized in Fig. 6.19.

Game_{12}

$\mathsf{Keygen}(1^\lambda, N, T) \to (\mathsf{vk}, \mathsf{sk})$

$\rho \leftarrow \{0,1\}^{256}$
$\mathbf{A} := \mathsf{H}_{\mathbf{A}}(\rho)$
for $I \subset [N]$ s.t. $|I| = N - T + 1$ and $I \neq I_0$ **do**
  $\mathbf{s}_I \leftarrow \chi_{\mathsf{sk}}$
  $\mathbf{b}_I := [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{s}_I$

  // Compute partial public keys
  for $i \in I$ **do** // Distribute the keys
    $\mathsf{sk}_i = \mathsf{sk}_i \cup \{\mathbf{s}_I\}$
$\mathbf{b} \xleftarrow{\$} R_q^k$
$\mathbf{b}_{I_0} := \mathbf{b} - \sum_{I \neq I_0} \mathbf{b}_I$
$(\mathbf{b}_\top, \mathbf{b}_\bot) := \mathsf{Power2Round}(\mathbf{b}, d)$
$\mathsf{vk} := (\rho, \mathbf{b}_\top)$
  // Verification key
$\mathsf{sk} := (tr, \mathsf{sk}_i)_{i \in [N]}$
return $(\mathsf{vk}, \mathsf{sk})$

Figure 6.19: Twelfth hybrid of the unforgeability proof of Threshold ML-DSA. Difference with the previous hybrid are highlighted.

As secrets are no longer used by honest parties in the scheme at this stage, we can replace $\mathbf{b}_{I_0}$ by a uniform value by the $\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathsf{sk}}}$ assumption. Equivalently, we can then sample $\mathbf{b}$ uniformly at random and define $(\mathbf{b}_\top, \mathbf{b}_\bot) := \mathsf{Power2Round}(\mathbf{b}, d)$ and $\mathbf{b}_{I_0} := \mathbf{b} - \sum_{I \neq I_0} \mathbf{b}_I$.

Formally, there exists an adversary $\mathcal{B}_5$ against the $\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathsf{sk}}}$ problem such that:

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_{11}} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_{12}}\right| \leqslant \mathsf{Adv}_{\mathcal{B}_5}^{\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathsf{sk}}}}$$

Game_{13}. In this hybrid, we first sample the high bits of the public key $\mathbf{b}_\top$ from an ML-DSA public key, then we sample the full key $\mathbf{b}$ from the distribution R_q^k conditioned on the value of $\mathbf{b}_\top$. This is formalized in Fig. 6.20.

Game_{13}

// Identical to Game_8

$\mathsf{Keygen}(1^\lambda, N, T) \to (\mathsf{vk}, \mathsf{sk})$

$\rho \leftarrow \{0,1\}^{256}$
$\mathbf{A} := \mathsf{H}_{\mathbf{A}}(\rho)$
for $I \subset [N]$ s.t. $|I| = N - T + 1$ and $I \neq I_0$ **do**
  $\mathbf{s}_I \leftarrow \chi_{\mathsf{sk}}$
  $\mathbf{b}_I := [\mathbf{A} \quad \mathbf{I}] \cdot \mathbf{s}_I$
    // Compute partial public keys
  for $i \in I$ **do** // Distribute the keys
    $\mathsf{sk}_i = \mathsf{sk}_i \cup \{\mathbf{s}_I\}$
$(\mathbf{s}, \mathbf{e}) \leftarrow \chi_{\mathsf{sk}}$
$\widehat{\mathbf{b}} := \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$
$(\mathbf{b}_\top, \cdot) := \mathsf{Power2Round}(\widehat{\mathbf{b}}, d)$
Sample $\mathbf{b}$ from $\mathcal{U}(R_q^k)$ s.t. $(\mathbf{b}_\top, \cdot) = \mathsf{Power2Round}(\mathbf{b}, d)$
$\mathbf{b}_{I_0} := \mathbf{b} - \sum_{I \neq I_0} \mathbf{b}_I$
$\mathsf{vk} := (\rho, \mathbf{b}_\top)$
  // Verification key
$\mathsf{sk} := (tr, \mathsf{sk}_i)_{i \in [N]}$
return $(\mathsf{vk}, \mathsf{sk})$

Figure 6.20: Thirteenth hybrid of the unforgeability proof of Threshold ML-DSA. Difference with the previous hybrid are highlighted.

We can equivalently sample $(\mathbf{b}, \mathbf{b}_\top)$ in Game_{12} by first sampling the high bits from $(\mathbf{b}_\top, \cdot) = \mathsf{Power2Round}(\mathcal{U}(R_q^k, d))$ and then sampling $\mathbf{b}$ conditioned on $\mathbf{b}_\top$. Then, we can see that Game_{13} just replaces $\mathcal{U}(R_q^k)$ in this distribution by an MLWE sample with secrets sampled in χ_{sk}, which is again indistinguishable under the $\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathsf{sk}}}$ assumption.

Formally, there exists an adversary $\mathcal{B}_6$ against the $\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathsf{sk}}}$ problem such that:

$$\left|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_{12}} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}_{13}}\right| \leqslant \mathsf{Adv}_{\mathcal{B}_6}^{\mathsf{MLWE}_{R_q,k,\ell,\chi_{\mathsf{sk}}}}$$

CONCLUSION. Finally, we reduce Game_{13} to the unforgeability of ML-DSA. We design an adversary $\mathcal{B}_7$ against the unforgeability of ML-DSA, and simulate the view of the adversary in Game_{13}.

$\mathcal{B}_7$ takes as input an ML-DSA public key $\mathsf{pk} = (\rho, \mathbf{b}_\perp)$, and random oracles $\mathsf{H}_{\mathbf{A}}$, H and SampleInBall. It does the following changes:

- It programs $\mathsf{H}_{\mathbf{A}}(\rho) = \mathbf{A}$.
- It chooses $\mathbf{b}_\top$ and ρ provided by the unforgeability challenger in the threshold Keygen.
- Queries to the random oracles H and SampleInBall are lazily passed to the random oracles provided by the unforgeability challenger instead of honestly sampling their values. Note that in case Game_{12} programs them otherwise, this behavior is not affected.

$\mathcal{B}_7$ forwards any forgery it receives from $\mathcal{A}$ to the ML-DSA unforgeability challenger.

The view of the adversary $\mathcal{A}$ is identically distributed since the ML-DSA random oracles sample images from the same distributions as Game_{13}, and the ML-DSA public key is sampled as $(\rho, \mathbf{b}_\top)$ in Game_{13}.

Then, we can see that a valid forgery in Game_{13} will also be a valid forgery for ML-DSA. Indeed, the verification procedure only depends on ρ and $\mathbf{b}_\top$ which are chosen identical for the ML-DSA signature and the threshold signature. Also, the challenge random oracles coincide on messages for which forgeries are valid (i.e. messages for which no signing oracle was called) as a $\tilde{c}$ programmed by Game_{13} will never be used by the ML-DSA oracles thanks to the condition added in Game_5. Hence,

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{B}_7} = \mathsf{Adv}^{\mathsf{Game}_{13}}_{\mathcal{A}}$$

We finally collect all the bounds to obtain the theorem statement, noting that all the intermediary adversaries have an execution time roughly equal to $\mathsf{Time}(\mathcal{A})$. □

6.4.5 Parameter Selection

To maintain backward compatibility with ML-DSA, we keep the same public parameters listed in Table 6.2, introducing only the new parameters K, r', r, and ν (respectively, the number of parallel repetitions, the respective radii of the randomness $\mathbf{r}'_i$ and rejected randomness $\mathbf{z}_i$, and the expansion factor for the first part of the rejected randomness $\mathbf{z}_i^{(1)}$). Following the security analysis in Section 6.4.3, we first choose ϕ such that $\varepsilon = I_{1-1/\phi^2}\left(\frac{n(\ell+k)+1}{2}, \frac{1}{2}\right)/2 \leqslant 1/(KQ_s)$. Then, given an upper bound B on the norm of partial secrets (holding with overwhelming probability), and choosing $r'^2 \geqslant r^2 + B^2 + \frac{2rB}{\phi}$, we ensure that Threshold ML-DSA retains the same level of security as ML-DSA for up to Q_s signing queries. Concretely, we take $Q_s = 2^{50}$. We set $B = 1.3 \cdot \sqrt{\tau} \cdot \sqrt{n \cdot (k + \ell/\nu^2)} \cdot \sqrt{\mathsf{Var}(\mathcal{U}(-\eta, \eta))} \cdot \sqrt{\left\lceil \binom{N}{T-1}/T \right\rceil}$ as a heuristic overwhelming bound on the norm of partial secrets[3]. Moreover, RSSRecover ensures that each party uses at most $\left\lceil \binom{N}{T-1}/T \right\rceil$ secrets in a session for our selected parameters, c.f. Section 6.4.2. Finally, we select K, r, r' for each possible pair (T, N), targeting a total success probability of at least $1/2$ for one protocol execution. We evaluate the success probability following Theorem 6.4.3, evaluating v numerically.

To showcase the efficiency of our scheme, we provide the communication cost for each threshold $2 \leqslant T \leqslant N \leqslant 6$ of Threshold ML-DSA-44 in Table 6.3.

N \ T	2	3	4	5	6
2	10.5 kB				
3	15.8 kB	21.0 kB			
4	15.8 kB	36.8 kB	42.0 kB		
5	15.8 kB	73.5 kB	157.4 kB	84.0 kB	
6	21.0 kB	99.8 kB	388.4 kB	524.8 kB	194.2 kB

Table 6.3: Communication costs of Threshold ML-DSA-44 for $2 \leqslant T \leqslant N \leqslant 6$, aiming for a success probability $1/2$. Full parameters are given in Section 6.4.6.

PERFORMANCE SCALING RATIONALE. Figure 6.21 shows that costs grow sharply once T or N increases, so we focus on $N \leqslant 6$ for practicality. Two effects drive this. (i) Let p be the probability a party accepts one local rejection step (for fixed r, r'). All T parties must accept together, so

3 We verify experimentally that $\|\mathbf{s}c\|$ behaves like a Gaussian distribution and take a bound $1.3 \cdot \|\mathbf{s}\| \cdot \|c\|$ corresponding to 13 standard deviations. If this heuristic were wrong, we would only incur a small loss in the number of queries since the bound on $\|\mathbf{s}c\|$ only affects the Rényi divergence of rejection sampling.

one joint attempt succeeds with probability about p^T; the expected number of sequential retries therefore scales like p^{-T}. Using $K > 1$ parallel repetitions only trades extra bandwidth for lower latency – it cannot remove the inherent p^T factor. (ii) A partial secret may aggregate up to $\lceil \binom{N}{T-1}/T \rceil$ base secrets, inflating its norm (captured in B) and forcing larger radii r, r'. This aggregation overhead vanishes at $T = N$, where each party holds exactly one base secret, explaining the mild improvement in that case.

We provide the full parameters of Threshold ML-DSA in Section 6.4.6, including the parameters (r, r', K, ν) for each security level of ML-DSA as well as the communication costs for each threshold $2 \leqslant T \leqslant N \leqslant 6$. The parameters selected ensure a success probability at least $1/2$ for one protocol execution. For Threshold ML-DSA-44, we include a visual representation of the bandwidth and runtime costs in Fig. 6.21.

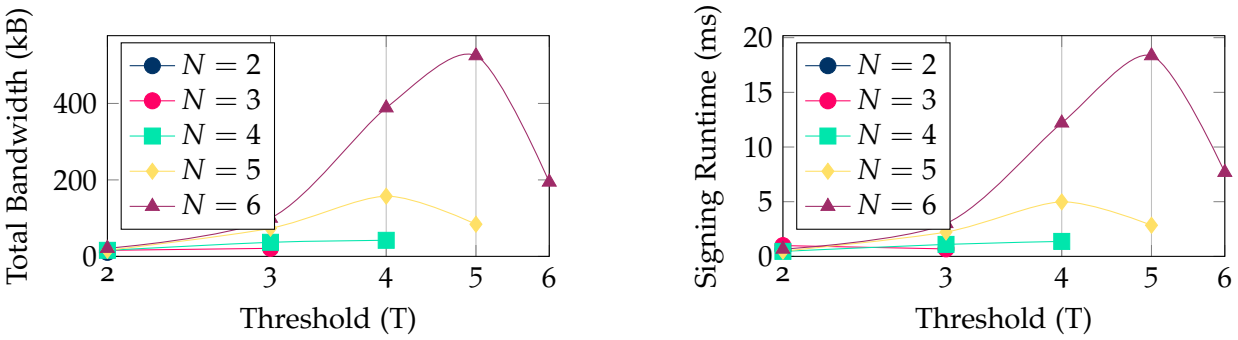


Figure 6.21: Bandwidth costs (top) and runtime signing costs (bottom) for Threshold ML-DSA-44. Note that the values for $N = 2$ and $N = 3$ overlap for all T.

6.4.6 Complete Threshold ML-DSA Parameters

This section provides the complete parameter sets for Threshold ML-DSA, compatible with the different parameter sets of ML-DSA, ML-DSA 44 in Fig. 6.22, ML-DSA 65 and ML-DSA 87 in Fig. 6.23. These parameter sets are designed to support threshold signatures with up to 6 parties, ensuring efficient signing and verification processes.

6.5 PERFORMANCE

We fully implemented our scheme for the parameter set Threshold ML-DSA 44, targeting a success probability of $1/2$ per signing attempt. Our implementation is written in `Golang`, and builds on the code of the CIRCL library [FK19], that we extended to support our threshold variant of ML-DSA.

EXPERIMENTAL ANALYSIS. We evaluated the performance of Threshold ML-DSA locally. All parties ran on a consumer-grade MacBook M3 with 25 GB RAM. We conducted single-threaded local simulations of our scheme to benchmark key generation, signing (one attempt), and verification. Each party's computation was emulated in a single process. The runtime (i.e., computational cost) for various (T, N) values is shown in Table 6.4, for the parameter set 44 of Threshold ML-DSA, as well as visualizations of its bandwidth and runtime in Fig. 6.21.

(T, N)	r	r'	K	**Comm. per party**
(2, 2)	252778	252833	2	10.5 kB
(2, 3)	310060	310138	3	15.8 kB
(3, 3)	246490	246546	4	21.0 kB
(2, 4)	305919	305997	3	15.8 kB
(3, 4)	279235	279314	7	36.8 kB
(4, 4)	243463	243519	8	42.0 kB
(2, 5)	285363	285459	3	15.8 kB
(3, 5)	282800	282912	14	73.5 kB
(4, 5)	259427	259526	30	157.4 kB
(5, 5)	239924	239981	16	84.0 kB
(2, 6)	300265	300362	4	21.0 kB
(3, 6)	277014	277139	19	99.8 kB
(4, 6)	268705	268831	74	388.4 kB
(5, 6)	250590	250686	100	524.8 kB
(6, 6)	219245	219301	37	194.2 kB

Figure 6.22: Parameter sets for Threshold ML-DSA 44, aiming for a success probability 1/2. All sets use the same multiplicative factor $\nu = 3$.

(T, N)	**KeyGen** (ms)	**Sign+Combine** (ms)	**Verify** (ms)
(3,3)	0.3669	0.6810	
(2,4)	0.1709	0.4570	
(3,4)	0.2062	1.0961	
(4,4)	0.1655	1.3672	
(3,5)	0.2870	2.2263	0.0306
(4,5)	0.2940	4.9832	
(5,5)	0.1956	2.8453	
(4,6)	0.5016	12.1949	
(6,6)	0.2181	7.6784	

Table 6.4: Local simulation for the average (mean) costs per party of a single signing attempt of Threshold ML-DSA-44. All costs are computed on a MacOS M3 machine and grouped by N. Green and light green indicate the most and second-most optimal cases per group (N) and per scheme, where applicable.

(T, N)	r	r'	K	**Comm./party**
(2, 2)	501495	501613	3	22.9 kB
(2, 3)	540212	540378	5	38.1 kB
(3, 3)	510387	510504	9	68.5 kB
(2, 4)	540212	540378	6	45.7 kB
(3, 4)	506761	506928	20	152.3 kB
(4, 4)	433594	433711	26	198.0 kB
(2, 5)	552371	552575	8	61.0 kB
(3, 5)	552909	553145	62	472.2 kB
(4, 5)	474331	474535	205	1561.3 kB
(5, 5)	425914	426032	78	594.1 kB
(2, 6)	571208	571412	8	61.0 kB
(3, 6)	536793	537058	95	723.5 kB
(4, 6)	488704	488969	804	6123.3 kB
(5, 6)	461324	461529	1200	9139.2 kB
(6, 6)	414896	415013	250	1904.0 kB

(a) Threshold ML-DSA 65. All thresholds (T, N) use the same multiplicative factor $\nu = 6$.

(T, N)	r	r'	K	**Comm./party**
(2, 2)	503119	503192	3	31.1 kB
(2, 3)	631601	631703	4	41.5 kB
(3, 3)	483107	483180	6	62.2 kB
(2, 4)	632903	633006	4	41.5 kB
(3, 4)	551752	551854	11	114.1 kB
(4, 4)	487958	488031	14	145.2 kB
(2, 5)	607694	607820	5	51.9 kB
(3, 5)	577400	577546	26	269.6 kB
(4, 5)	518384	518510	70	725.8 kB
(5, 5)	468214	468287	35	362.9 kB
(2, 6)	665106	665232	5	51.9 kB
(3, 6)	577541	577704	39	404.4 kB
(4, 6)	517689	517853	208	2156.6 kB
(5, 6)	479692	479819	295	3058.6 kB
(6, 6)	424124	424197	87	902.0 kB

(b) Threshold ML-DSA 87. All thresholds (T, N) use the same multiplicative factor $\nu = 7$.

Figure 6.23: Parameter sets for Threshold ML-DSA 65 and 87, aiming for a success probability per signing attempt of $1/2$.

7 CONCLUSION

The deployment of classical threshold signature schemes is already ongoing. The FROST protocol [KG20] has become the state-of-the-art for threshold Schnorr signatures, having been codified in an IRTF RFC [CKGW24], submitted to the NIST workshop on multi-party threshold cryptography [Kom26], and deployed in various real-world systems [Zca25; Cha25; Sol26; Pen25]. In parallel, post-quantum threshold cryptography has progressed rapidly over the past few years. Several recent proposals now cover many of the desirable properties of threshold signatures in the post-quantum setting. However, combining all these properties—such as robustness, optimal round complexity, and standard compatibility—into a single, highly efficient protocol remains an open challenge.

In particular, this thesis aimed to narrow the gap between classical and post-quantum threshold signatures by developing a toolkit for designing and deploying practical lattice-based threshold signatures. We presented new techniques and practical protocols demonstrating that desirable properties—such as DKG, robustness, efficient abort identification, and compatibility with ML-DSA—can be achieved efficiently. Some techniques developed here have already been integrated into upcoming NIST submissions.

The ongoing NIST Multi-Party Threshold Cryptography (MPTC) effort promises to be a major catalyst for the field. With the recent call for threshold primitives [NIST-IR8214C], concrete proposals will be submitted in the coming months. This process should bring clarity regarding the most promising designs, while providing the community with a structured opportunity to thoroughly analyze, cryptanalyze, and compare these schemes.

In addition, the coming years will likely put post-quantum threshold schemes to the test, and should provide crucial feedback on practical expectations for real-world deployments. For instance, it remains unclear whether the industry will require threshold versions of standardized algorithms like ML-DSA [Nat24], or whether alternative designs—for instance based on the Raccoon signature [PKPR24]—will be acceptable. We also need clarity on which communication models are realistic in practice, whether features like full robustness, identifiable aborts, and distributed key generation (DKG) are strict requirements, and what thresholds (T out of N) and communication overheads are considered acceptable.

7.1 OPEN QUESTIONS AND FUTURE DIRECTIONS

Although we are starting to see a rich and diverse landscape of lattice-based threshold schemes, many questions remain open. These challenges can be broadly categorized by the underlying design paradigms of the signatures.

FIAT–SHAMIR WITH ABORTS. For schemes based on Fiat-Shamir with aborts, and in particular for ML-DSA, the primary hurdles lie in scalability and accountability. Scaling our protocol from Chapter 6 to a larger number of parties (e.g., $N \leqslant 32$ or even 64) while maintaining practical communication costs is a key open question. Our proposal is limited to small thresholds (e.g., $N \leqslant 6$, though we believe it can be extended to slightly larger thresholds). Currently, the only known approach to support large thresholds is with MPC-based techniques [BCEP+25], but it is not clear if this approach can be made sufficiently efficient for real-world applications. Addition-

ally, it is currently unknown how to identify misbehavior during signing for these schemes, as the inherent aborts of the rejection sampling appear hard to distinguish from misbehavior.

FIAT–SHAMIR WITHOUT ABORTS. Conversely, schemes based on Fiat-Shamir with noise flooding avoid the complexities of rejection sampling, and have attracted considerable attention thanks to appealing properties proposed in the literature, including partially non-interactive signing, identifiable aborts, robustness and adaptive security. However, they also face their own set of challenges. For these constructions, next steps include designing efficient proactive refresh mechanisms[1] for long-lived keys and developing more efficient protocols proven secure against adaptive adversaries. Since the underlying centralized signatures for these schemes are not currently standardized by NIST, the community may want to converge on a shared basis for interoperability. A natural candidate is Raccoon, which already underlies the most efficient proposals. Converging on a common set of public parameters, encoding, and verification would help interoperability, while different threshold constructions could remain compatible with this shared base and target different numbers of parties and signing queries.

Besides, while highly efficient two-round schemes exist in this paradigm, the most efficient ones rely on interactive assumptions like Algebraic One-More MSIS (AOM-MSIS) instantiated with parameters that fall outside formal reductions to standard lattice problems. It will be important to further analyze the security of these assumptions and parameters to increase confidence in the security of these schemes.

IMPLEMENTATION, EVALUATION, AND DEPLOYMENT ENVIRONMENTS. Finally, there is a clear need to implement, evaluate, and compare lattice threshold schemes more thoroughly. Currently, schemes can be difficult to compare in practice because they often lack optimized and constant-time implementations, or because they are evaluated under different network assumptions (e.g., synchronous broadcast vs. asynchronous point-to-point). It is also often unclear what kind of environment they will be deployed in—for example, whether partially non-interactive signing (e.g., a 2-round offline/online phase like FROST [KG20]) is a strict requirement, or if highly interactive protocols (e.g., 3 to 5 rounds) are perfectly fine for the target use case. Clarifying these deployment constraints will be essential to guide the design of future schemes and to ensure that they meet real-world needs.

BEYOND THRESHOLD SIGNATURES. While this thesis focused on threshold signatures, the transition to post-quantum cryptography also necessitates distributing other primitives. There is a growing interest in thresholdizing Public Key Encryption (PKE) [DF90; FP01; GLOV25; BD10; BS23; PS24; CLW25; BLPP26], as well as more advanced cryptographic primitives like threshold Identity-Based Encryption (IBE) [GWWW25; BLPP26], threshold Password-based Authenticated Key Exchange (PAKE) [MSJ02; ACFP05], threshold ring signatures [DV09; HS20; LWWS+25], or threshold blind signatures [VZK03; LNÖ25; FNR25]. Although the industry currently appears less urgently focused on deploying these advanced distributed primitives compared to standard threshold signatures, exploring their theoretical feasibility and pushing the boundaries of their concrete efficiency remains an interesting research direction.

1 Hermine [BCPE+26b] proposes such a mechanism when using our short secret sharing techniques. There is no known proactive refresh mechanism when using other secret sharing schemes, and consequently, for higher thresholds than those supported by our techniques.

BIBLIOGRAPHY

[ABCP23] Shahla Atapoor, Karim Baghery, Daniele Cozzo, and Robi Pedersen. "VSS from Distributed ZK Proofs and Applications." In: *ASIACRYPT 2023, Part I*. Ed. by Jian Guo and Ron Steinfeld. Vol. 14438. LNCS. Springer, Singapore, Dec. 2023, pp. 405–440. DOI: 10.1007/978-981-99-8721-4_13.

[ABHR25] Karen Azari, Cecilia Boschini, Kristina Hostáková, and Michael Reichle. *Security Amplification of Threshold Signatures in the Standard Model*. Cryptology ePrint Archive, Report 2025/1705. 2025. URL: https://eprint.iacr.org/2025/1705.

[ACFP05] Michel Abdalla, Olivier Chevassut, Pierre-Alain Fouque, and David Pointcheval. "A Simple Threshold Authenticated Key Exchange from Short Secrets." In: *ASIACRYPT 2005*. Ed. by Bimal K. Roy. Vol. 3788. LNCS. Springer, Berlin, Heidelberg, Dec. 2005, pp. 566–584. DOI: 10.1007/11593447_31.

[ADN06] Jesús F. Almansa, Ivan Damgård, and Jesper Buus Nielsen. "Simplified Threshold RSA with Adaptive and Proactive Security." In: *EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. LNCS. Springer, Berlin, Heidelberg, May 2006, pp. 593–611. DOI: 10.1007/11761679_35.

[ADP24] Nabil Alkeilani Alkadri, Nico Döttling, and Sihang Pu. "Practical Lattice-Based Distributed Signatures for a Small Number of Signers." In: *ACNS 2024, Part I*. Ed. by Christina Pöpper and Lejla Batina. Vol. 14583. LNCS. Springer, Cham, Mar. 2024, pp. 376–402. DOI: 10.1007/978-3-031-54770-6_15.

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. "Post-quantum Key Exchange - A New Hope." In: *USENIX Security 2016*. Ed. by Thorsten Holz and Stefan Savage. USENIX Association, Aug. 2016, pp. 327–343. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/alkim.

[Ajt96] Miklós Ajtai. "Generating Hard Instances of Lattice Problems (Extended Abstract)." In: *28th ACM STOC*. ACM Press, May 1996, pp. 99–108. DOI: 10.1145/237814.237838.

[AL07] Yonatan Aumann and Yehuda Lindell. "Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries." In: *TCC 2007*. Ed. by Salil P. Vadhan. Vol. 4392. LNCS. Springer, Berlin, Heidelberg, Feb. 2007, pp. 137–156. DOI: 10.1007/978-3-540-70936-7_8.

[ANP23] Benny Applebaum, Oded Nir, and Benny Pinkas. "How to Recover a Secret with $O(n)$ Additions." In: *CRYPTO 2023, Part I*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14081. LNCS. Springer, Cham, Aug. 2023, pp. 236–262. DOI: 10.1007/978-3-031-38557-5_8.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. "On the concrete hardness of Learning with Errors." In: *J. Math. Cryptol.* 9.3 (2015), pp. 169–203. URL: http://www.degruyter.com/view/j/jmc.2015.9.issue-3/jmc-2015-0016/jmc-2015-0016.xml.

[ASY22] Shweta Agrawal, Damien Stehlé, and Anshu Yadav. "Round-Optimal Lattice-Based Threshold Signatures, Revisited." In: *ICALP 2022*. Ed. by Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff. Vol. 229. LIPIcs. Schloss Dagstuhl, July 2022, 8:1–8:20. DOI: 10.4230/LIPIcs.ICALP.2022.8.

[BBCM+25] Michele Battagliola, Giacomo Borin, Giovanni Di Crescenzo, Alessio Meneghetti, and Edoardo Persichetti. "Enhancing Threshold Group Action Signature Schemes: Adaptive Security and Scalability Improvements." In: *IACR Cryptol. ePrint Arch.* (2025), p. 85. URL: https://eprint.iacr.org/2025/085.

[BBRS24] Henry Bambury, Hugo Beguinet, Thomas Ricosset, and Éric Sageloli. "Polytopes in the Fiat-Shamir with Aborts Paradigm." In: *CRYPTO 2024, Part I*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14920. LNCS. Springer, Cham, Aug. 2024, pp. 339–372. DOI: 10.1007/978-3-031-68376-3_11.

[BCEP+25] Alexander Bienstock, Leo de Castro, Daniel Escudero, Antigoni Polychroniadou, and Akira Takahashi. *Efficient, Scalable Threshold ML-DSA Signatures: An MPC Approach*. Cryptology ePrint Archive, Report 2025/1163. 2025. URL: https://eprint.iacr.org/2025/1163.

[BCKM+22] Mihir Bellare, Elizabeth C. Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. "Better than Advertised Security for Non-interactive Threshold Signatures." In: *CRYPTO 2022, Part IV*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Vol. 13510. LNCS. Springer, Cham, Aug. 2022, pp. 517–550. DOI: 10.1007/978-3-031-15985-5_18.

[BCPE+26a] Giacomo Borin, Sofía Celi, Rafael del Pino, Thomas Espitau, Shuichi Katsumata, Guilhem Niot, Thomas Prest, and Kaoru Takemure. *Hermine*. Tech. rep. Submission to the NIST Multi-Party Threshold Cryptography (MPTC) Call. National Institute of Standards and Technology (NIST), 2026. URL: https://csrc.nist.gov/csrc/media/Projects/threshold-cryptography/documents/TCall-1/Hermine-PW01.pdf.

[BCPE+26b] Giacomo Borin, Sofía Celi, Rafael del Pino, Thomas Espitau, Shuichi Katsumata, Guilhem Niot, Thomas Prest, and Kaoru Takemure. *Hermine: An Efficient Lattice-based FROST-like Threshold Signature*. Cryptology ePrint Archive, Paper 2026/419. 2026. URL: https://eprint.iacr.org/2026/419.

[BD10] Rikke Bendlin and Ivan Damgård. "Threshold Decryption and Zero-Knowledge Proofs for Lattice-Based Cryptosystems." In: *TCC 2010*. Ed. by Daniele Micciancio. Vol. 5978. LNCS. Springer, Berlin, Heidelberg, Feb. 2010, pp. 201–218. DOI: 10.1007/978-3-642-11799-2_13.

[BDGL16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. "New directions in nearest neighbor searching with applications to lattice sieving." In: *27th SODA*. Ed. by Robert Krauthgamer. ACM-SIAM, Jan. 2016, pp. 10–24. DOI: 10.1137/1.9781611974331.ch2.

[BDLR25a] Renas Bacho, Sourav Das, Julian Loss, and Ling Ren. "Adaptively Secure Three-Round Threshold Schnorr Signatures from DDH." In: *CRYPTO 2025, Part VI*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16005. LNCS. Springer, Cham, Aug. 2025, pp. 390–422. DOI: 10.1007/978-3-032-01887-8_13.

[BDLR25b] Renas Bacho, Sourav Das, Julian Loss, and Ling Ren. "Glacius: Threshold Schnorr Signatures from DDH with Full Adaptive Security." In: *EUROCRYPT 2025, Part II*. Ed. by Serge Fehr and Pierre-Alain Fouque. Vol. 15602. LNCS. Springer, Cham, May 2025, pp. 304–334. DOI: 10.1007/978-3-031-91124-8_11.

[BGGJ+18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. "Threshold Cryptosystems from Threshold Fully Homomorphic Encryption." In: *CRYPTO 2018, Part I*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10991. LNCS. Springer, Cham, Aug. 2018, pp. 565–596. DOI: 10.1007/978-3-319-96884-1_19.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract)." In: *20th ACM STOC*. ACM Press, May 1988, pp. 1–10. DOI: 10.1145/62212.62213.

[BH04] Michael Backes and Dennis Hofheinz. "How to Break and Repair a Universally Composable Signature Functionality." In: *ISC 2004*. Ed. by Kan Zhang and Yuliang Zheng. Vol. 3225. LNCS. Springer, Berlin, Heidelberg, Sept. 2004, pp. 61–72. DOI: 10.1007/978-3-540-30144-8_6.

[BKLM+25] Cecilia Boschini, Darya Kaviani, Russell W. F. Lai, Giulio Malavolta, Akira Takahashi, and Mehdi Tibouchi. "Ringtail: Practical Two-Round Threshold Signatures from Learning with Errors." In: *2025 IEEE Symposium on Security and Privacy*. Ed. by Marina Blanton, William Enck, and Cristina Nita-Rotaru. IEEE Computer Society Press, May 2025, pp. 149–164. DOI: 10.1109/SP61157.2025.00070.

[BKP13] Rikke Bendlin, Sara Krehbiel, and Chris Peikert. "How to Share a Lattice Trapdoor: Threshold Protocols for Signatures and (H)IBE." In: *ACNS 2013*. Ed. by Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini. Vol. 7954. LNCS. Springer, Berlin, Heidelberg, June 2013, pp. 218–236. DOI: 10.1007/978-3-642-38980-1_14.

[BL22] Renas Bacho and Julian Loss. "On the Adaptive Security of the Threshold BLS Signature Scheme." In: *ACM CCS 2022*. Ed. by Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi. ACM Press, Nov. 2022, pp. 193–207. DOI: 10.1145/3548606.3560656.

[BL90] Josh Cohen Benaloh and Jerry Leichter. "Generalized Secret Sharing and Monotone Functions." In: *CRYPTO'88*. Ed. by Shafi Goldwasser. Vol. 403. LNCS. Springer, New York, Aug. 1990, pp. 27–35. DOI: 10.1007/0-387-34799-2_3.

[BLLS+15] Shi Bai, Adeline Langlois, Tancrède Lepoint, Damien Stehlé, and Ron Steinfeld. "Improved Security Proofs in Lattice-Based Cryptography: Using the Rényi Divergence Rather Than the Statistical Distance." In: *ASIACRYPT 2015, Part I*. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9452. LNCS. Springer, Berlin, Heidelberg, Nov. 2015, pp. 3–24. DOI: 10.1007/978-3-662-48797-6_1.

[BLPP26] Katharina Boudgoust, Oleksandra Lapiha, Rafaël del Pino, and Thomas Prest. *IND-CCA Lattice Threshold KEM under 30 KiB*. Cryptology ePrint Archive, Paper 2026/021. 2026. URL: https://eprint.iacr.org/2026/021.

[BLSW24] Renas Bacho, Julian Loss, Gilad Stern, and Benedikt Wagner. "HARTS: High-Threshold, Adaptively Secure, and Robust Threshold Schnorr Signatures." In: *ASIACRYPT 2024, Part III*. Ed. by Kai-Min Chung and Yu Sasaki. Vol. 15486. LNCS. Springer, Singapore, Dec. 2024, pp. 104–140. DOI: 10.1007/978-981-96-0891-1_4.

[BN06] Mihir Bellare and Gregory Neven. "Multi-signatures in the plain public-Key model and a general forking lemma." In: *ACM CCS 2006*. Ed. by Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati. ACM Press, Oct. 2006, pp. 390–399. DOI: 10.1145/1180405.1180453.

[Bol03] Alexandra Boldyreva. "Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme." In: *PKC 2003*. Ed. by Yvo Desmedt. Vol. 2567. LNCS. Springer, Berlin, Heidelberg, Jan. 2003, pp. 31–46. DOI: 10.1007/3-540-36288-6_3.

[Bop85] Ravi B. Boppana. "Amplification of Probabilistic Boolean Formulas." In: *26th FOCS*. IEEE Computer Society Press, Oct. 1985, pp. 20–29. DOI: 10.1109/SFCS.1985.5.

[BPLP26] Katharina Boudgoust, Rafael del Pino, Oleksandra Lapiha, and Thomas Prest. *Amber*. Tech. rep. Submission to the NIST Multi-Party Threshold Cryptography (MPTC) Call. National Institute of Standards and Technology (NIST), 2026. URL: https://csrc.nist.gov/csrc/media/Projects/threshold-cryptography/documents/TCall-1/Amber-PW01.pdf.

[BS23] Katharina Boudgoust and Peter Scholl. "Simple Threshold (Fully Homomorphic) Encryption from LWE with Polynomial Modulus." In: *ASIACRYPT 2023, Part I*. Ed. by Jian Guo and Ron Steinfeld. Vol. 14438. LNCS. Springer, Singapore, Dec. 2023, pp. 371–404. DOI: 10.1007/978-981-99-8721-4_12.

[BTT22] Cecilia Boschini, Akira Takahashi, and Mehdi Tibouchi. "MuSig-L: Lattice-Based Multi-signature with Single-Round Online Phase." In: *CRYPTO 2022, Part II*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Vol. 13508. LNCS. Springer, Cham, Aug. 2022, pp. 276–305. DOI: 10.1007/978-3-031-15979-4_10.

[Can01] Ran Canetti. "Universally Composable Security: A New Paradigm for Cryptographic Protocols." In: *42nd FOCS*. IEEE Computer Society Press, Oct. 2001, pp. 136–145. DOI: 10.1109/SFCS.2001.959888.

[Can03] Ran Canetti. *Universally Composable Signatures, Certification and Authentication*. Cryptology ePrint Archive, Report 2003/239. 2003. URL: https://eprint.iacr.org/2003/239.

[CCDG+24] Jung Hee Cheon, Hyeongmin Choe, Julien Devevey, Tim Güneysu, Dongyeon Hong, Markus Krausz, Georg Land, Marc Möller, Damien Stehlé, and MinJune Yi. "HAETAE: Shorter Lattice-Based Fiat-Shamir Signatures." In: *IACR TCHES* 2024.3 (2024), pp. 25–75. DOI: 10.46586/tches.v2024.i3.25-75.

[CCLS+19] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. "Two-Party ECDSA from Hash Proof Systems and Efficient Instantiations." In: *CRYPTO 2019, Part III*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11694. LNCS. Springer, Cham, Aug. 2019, pp. 191–221. DOI: 10.1007/978-3-030-26954-8_7.

[CCLS+23] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. "Bandwidth-efficient threshold EC-DSA revisited: Online/offline extensions, identifiable aborts proactive and adaptive security." In: *Theor. Comput. Sci.* 939 (2023), pp. 78–104. DOI: 10.1016/J.TCS.2022.10.016. URL: https://doi.org/10.1016/j.tcs.2022.10.016.

[CDPE+26] Sofía Celi, Gustavo Delerue, Rafael del Pino, Thomas Espitau, Guilhem Niot, and Thomas Prest. *Mithril*. Tech. rep. Submission to the NIST Multi-Party Threshold Cryptography (MPTC) Call. National Institute of Standards and Technology (NIST), 2026. URL: https://csrc.nist.gov/csrc/media/Projects/threshold-cryptography/documents/TCall-1/Mithril-PW01.pdf.

[CEN25] Sofía Celi, Daniel Escudero, and Guilhem Niot. "Share the MAYO: Thresholdizing MAYO." In: *Post-Quantum Cryptography - 16th International Workshop, PQCrypto 2025, Part I*. Ed. by Ruben Niederhagen and Markku-Juhani O. Saarinen. Springer, Cham, Apr. 2025, pp. 165–198. DOI: 10.1007/978-3-031-86599-2_6.

[CGGM+20] Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. "UC Non-Interactive, Proactive, Threshold ECDSA with Identifiable Aborts." In: *ACM CCS 2020*. Ed. by Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna. ACM Press, Nov. 2020, pp. 1769–1787. DOI: 10.1145/3372297.3423367.

[CGJK+99] Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. "Adaptive Security for Threshold Cryptosystems." In: *CRYPTO'99*. Ed. by Michael J. Wiener. Vol. 1666. LNCS. Springer, Berlin, Heidelberg, Aug. 1999, pp. 98–115. DOI: 10.1007/3-540-48405-1_7.

[CGMA85] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. "Verifiable secret sharing and achieving simultaneity in the presence of faults." In: *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*. IEEE. 1985, pp. 383–395.

[CGS14] Peter Campbell, Michael Groves, and Dan Shepherd. "Soliloquy: A Cautionary Tale." In: *ETSI 2nd Quantum-Safe Crypto Workshop*. Oct. 2014. URL: https://docbox.etsi.org/Workshop/2014/201410_CRYPTO/S07_Systems_and_Attacks/S07_Groves_Annex.pdf.

[Cha25] Chainflip Labs. *FROST Signature Scheme – Chainflip Protocol Documentation*. https://docs.chainflip.io/protocol/frost-signature-scheme. Accessed: February 6, 2026. 2025.

[Che23] Yanbo Chen. "DualMS: Efficient Lattice-Based Two-Round Multi-signature with Trapdoor-Free Simulation." In: *CRYPTO 2023, Part V*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14085. LNCS. Springer, Cham, Aug. 2023, pp. 716–747. DOI: 10.1007/978-3-031-38554-4_23.

[Che25] Yanbo Chen. *Round-Efficient Adaptively Secure Threshold Signatures with Rewinding*. Cryptology ePrint Archive, Report 2025/638. 2025. URL: https://eprint.iacr.org/2025/638.

[CHNR+24] Daniel Collins, Loïs Huguenin-Dumittan, Ngoc Khanh Nguyen, Nicolas Rolin, and Serge Vaudenay. "K-Waay: Fast and Deniable Post-Quantum X3DH without Ring Signatures." In: *USENIX Security 2024*. Ed. by Davide Balzarotti and Wenyuan Xu. USENIX Association, Aug. 2024. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/collins.

[CKGW24] Deirdre Connolly, Chelsea Komlo, Ian Goldberg, and Christopher A. Wood. *The Flexible Round-Optimized Schnorr Threshold (FROST) Protocol for Two-Round Schnorr Signatures*. RFC 9591. Accessed: February 6, 2026. June 2024. DOI: 10.17487/RFC9591. URL: https://www.rfc-editor.org/info/rfc9591.

[CKKT+25] Elizabeth C. Crites, Jonathan Katz, Chelsea Komlo, Stefano Tessaro, and Chenzhi Zhu. "On the Adaptive Security of FROST." In: *CRYPTO 2025, Part VI*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16005. LNCS. Springer, Cham, Aug. 2025, pp. 480–511. DOI: 10.1007/978-3-032-01887-8_16.

[CKM21] Elizabeth Crites, Chelsea Komlo, and Mary Maller. *How to Prove Schnorr Assuming Schnorr: Security of Multi- and Threshold Signatures*. Cryptology ePrint Archive, Report 2021/1375. 2021. URL: https://eprint.iacr.org/2021/1375.

[CKM23] Elizabeth C. Crites, Chelsea Komlo, and Mary Maller. "Fully Adaptive Schnorr Threshold Signatures." In: *CRYPTO 2023, Part I*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14081. LNCS. Springer, Cham, Aug. 2023, pp. 678–709. DOI: 10.1007/978-3-031-38557-5_22.

[CLW25] Valerio Cini, Russell W. F. Lai, and Ivy K. Y. Woo. *Pilvi: Lattice Threshold PKE with Small Decryption Shares and Improved Security*. Cryptology ePrint Archive, Report 2025/1691. 2025. URL: https://eprint.iacr.org/2025/1691.

[CN11] Yuanmi Chen and Phong Q. Nguyen. "BKZ 2.0: Better Lattice Security Estimates." In: *ASIACRYPT 2011*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Vol. 7073. LNCS. Springer, Berlin, Heidelberg, Dec. 2011, pp. 1–20. DOI: 10.1007/978-3-642-25385-0_1.

[CPEN+26] Sofía Celi, Rafaël del Pino, Thomas Espitau, Guilhem Niot, and Thomas Prest. *Efficient Threshold ML-DSA*. Cryptology ePrint Archive, Paper 2026/013. To Appear in USENIX Security 2026. 2026. URL: https://eprint.iacr.org/2026/013.

[CS19] Daniele Cozzo and Nigel P. Smart. "Sharing the LUOV: Threshold Post-quantum Signatures." In: *17th IMA International Conference on Cryptography and Coding*. Ed. by Martin Albrecht. Vol. 11929. LNCS. Springer, Cham, Dec. 2019, pp. 128–153. DOI: 10.1007/978-3-030-35199-1_7.

[CTZ24] Rutchathon Chairattana-Apirom, Stefano Tessaro, and Chenzhi Zhu. "Partially Non-interactive Two-Round Lattice-Based Threshold Signatures." In: *ASIACRYPT 2024, Part IV*. Ed. by Kai-Min Chung and Yu Sasaki. Vol. 15487. LNCS. Springer, Singapore, Dec. 2024, pp. 268–302. DOI: 10.1007/978-981-96-0894-2_9.

[DDB95] Yvo Desmedt, Giovanni Di Crescenzo, and Mike Burmester. "Multiplicative Non-abelian Sharing Schemes and their Application to Threshold Cryptography." In: *ASIACRYPT'94*. Ed. by Josef Pieprzyk and Reihaneh Safavi-Naini. Vol. 917. LNCS. Springer, Berlin, Heidelberg, Nov. 1995, pp. 21–32. DOI: 10.1007/BFb0000421.

[DDLL13] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. "Lattice Signatures and Bimodal Gaussians." In: *CRYPTO 2013, Part I*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8042. LNCS. Springer, Berlin, Heidelberg, Aug. 2013, pp. 40–56. DOI: 10.1007/978-3-642-40041-4_3.

[dEKM+23] Rafael del Pino, Thomas Espitau, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, Mélissa Rossi, and Markku-Juhani Saarinen. *Raccoon*. Tech. rep. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures. National Institute of Standards and Technology, 2023.

[dENP25] Rafaël del Pino, Thomas Espitau, Guilhem Niot, and Thomas Prest. *Simple and Efficient Lattice Threshold Signatures with Identifiable Aborts*. Cryptology ePrint Archive, Report 2025/871. 2025. URL: https://eprint.iacr.org/2025/871.

[Des88] Yvo Desmedt. "Society and Group Oriented Cryptography: A New Concept." In: *CRYPTO'87*. Ed. by Carl Pomerance. Vol. 293. LNCS. Springer, Berlin, Heidelberg, Aug. 1988, pp. 120–127. DOI: 10.1007/3-540-48184-2_8.

[DF90] Yvo Desmedt and Yair Frankel. "Threshold Cryptosystems." In: *CRYPTO'89*. Ed. by Gilles Brassard. Vol. 435. LNCS. Springer, New York, Aug. 1990, pp. 307–315. DOI: 10.1007/0-387-34805-0_28.

[DF92] Yvo Desmedt and Yair Frankel. "Shared Generation of Authenticators and Signatures (Extended Abstract)." In: *CRYPTO'91*. Ed. by Joan Feigenbaum. Vol. 576. LNCS. Springer, Berlin, Heidelberg, Aug. 1992, pp. 457–469. DOI: 10.1007/3-540-46766-1_37.

[DFJN+25] Kévin Duverger, Pierre-Alain Fouque, Charlie Jacomme, Guilhem Niot, and Cristina Onete. "Subversion-resilient Key-exchange in the Post-quantum World." In: *ACM CCS 2025*. Ed. by Chun-Ying Huang, Jyh-Cheng Chen, Shiuh-Pyng Shieh, David Lie, and Véronique Cortier. ACM Press, Oct. 2025, pp. 1200–1214. DOI: 10.1145/3719027.3765165.

[DFPS22] Julien Devevey, Omar Fawzi, Alain Passelègue, and Damien Stehlé. "On Rejection Sampling in Lyubashevsky's Signature Scheme." In: *ASIACRYPT 2022, Part IV*. Ed. by Shweta Agrawal and Dongdai Lin. Vol. 13794. LNCS. Springer, Cham, Dec. 2022, pp. 34–64. DOI: 10.1007/978-3-031-22972-5_2.

[DFPS23] Julien Devevey, Pouria Fallahpour, Alain Passelègue, and Damien Stehlé. "A Detailed Analysis of Fiat-Shamir with Aborts." In: *CRYPTO 2023, Part V*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14085. LNCS. Springer, Cham, Aug. 2023, pp. 327–357. DOI: 10.1007/978-3-031-38554-4_11.

[DH76] Whitfield Diffie and Martin E. Hellman. "New Directions in Cryptography." In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654. DOI: 10.1109/TIT.1976.1055638.

[DKLL+18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. "CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme." In: *IACR TCHES* 2018.1 (2018), pp. 238–268. ISSN: 2569-2925. DOI: 10.13154/tches.v2018.i1.238-268. URL: https://tches.iacr.org/index.php/TCHES/article/view/839.

[DKLs18] Jack Doerner, Yashvanth Kondi, Eysa Lee, and abhi shelat. "Secure Two-party Threshold ECDSA from ECDSA Assumptions." In: *2018 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2018, pp. 980–997. DOI: 10.1109/SP.2018.00036.

[DKLs24] Jack Doerner, Yashvanth Kondi, Eysa Lee, and abhi shelat. "Threshold ECDSA in Three Rounds." In: *2024 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2024, pp. 3053–3071. DOI: 10.1109/SP54263.2024.00178.

[DKLS25] Antonín Dufka, Semjon Kravtšenko, Peeter Laud, and Nikita Snetkov. *Trilithium: Efficient and Universally Composable Distributed ML-DSA Signing*. Cryptology ePrint Archive, Report 2025/675. 2025. URL: https://eprint.iacr.org/2025/675.

[DKMM+24] Rafaël Del Pino, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, and Markku-Juhani O. Saarinen. "Threshold Raccoon: Practical Threshold Signatures from Standard Lattice Assumptions." In: *EUROCRYPT 2024, Part II*. Ed. by Marc Joye and Gregor Leander. Vol. 14652. LNCS. Springer, Cham, May 2024, pp. 219–248. DOI: 10.1007/978-3-031-58723-8_8.

[DN12] Léo Ducas and Phong Q. Nguyen. "Faster Gaussian Lattice Sampling Using Lazy Floating-Point Arithmetic." In: *ASIACRYPT 2012*. Ed. by Xiaoyun Wang and Kazue Sako. Vol. 7658. LNCS. Springer, Berlin, Heidelberg, Dec. 2012, pp. 415–432. DOI: 10.1007/978-3-642-34961-4_26.

[DOTT21] Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. "Two-Round n-out-of-n and Multi-signatures and Trapdoor Commitment from Lattices." In: *PKC 2021, Part I*. Ed. by Juan Garay. Vol. 12710. LNCS. Springer, Cham, May 2021, pp. 99–130. DOI: 10.1007/978-3-030-75245-3_5.

[DOTT22] Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. "Two-Round n-out-of-n and Multi-Signatures and Trapdoor Commitment from Lattices." In: *Journal of Cryptology* 35.2 (Apr. 2022), p. 14. DOI: 10.1007/s00145-022-09425-3.

[DPS23] Julien Devevey, Alain Passelègue, and Damien Stehlé. "G+G: A Fiat-Shamir Lattice Signature Based on Convolved Gaussians." In: *ASIACRYPT 2023, Part VII*. Ed. by Jian Guo and Ron Steinfeld. Vol. 14444. LNCS. Springer, Singapore, Dec. 2023, pp. 37–64. DOI: 10.1007/978-981-99-8739-9_2.

[DPW19] Léo Ducas, Maxime Plançon, and Benjamin Wesolowski. "On the Shortness of Vectors to Be Found by the Ideal-SVP Quantum Algorithm." In: *CRYPTO 2019, Part I*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11692. LNCS. Springer, Cham, Aug. 2019, pp. 322–351. DOI: 10.1007/978-3-030-26948-7_12.

[DT06] Ivan Damgård and Rune Thorbek. "Linear Integer Secret Sharing and Distributed Exponentiation." In: *PKC 2006*. Ed. by Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin. Vol. 3958. LNCS. Springer, Berlin, Heidelberg, Apr. 2006, pp. 75–90. DOI: 10.1007/11745853_6.

[DV09] Léonard Dallot and Damien Vergnaud. "Provably Secure Code-Based Threshold Ring Signatures." In: *12th IMA International Conference on Cryptography and Coding*. Ed. by Matthew G. Parker. Vol. 5921. LNCS. Springer, Berlin, Heidelberg, Dec. 2009, pp. 222–235. DOI: 10.1007/978-3-642-10868-6_13.

[EENP+24] Muhammed F. Esgin, Thomas Espitau, Guilhem Niot, Thomas Prest, Amin Sakzad, and Ron Steinfeld. "Plover: Masking-Friendly Hash-and-Sign Lattice Signatures." In: *EUROCRYPT 2024, Part VII*. Ed. by Marc Joye and Gregor Leander. Vol. 14657. LNCS. Springer, Cham, May 2024, pp. 316–345. DOI: 10.1007/978-3-031-58754-2_12.

[EKT24] Thomas Espitau, Shuichi Katsumata, and Kaoru Takemure. "Two-Round Threshold Signature from Algebraic One-More Learning with Errors." In: *CRYPTO 2024, Part VII*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14926. LNCS. Springer, Cham, Aug. 2024, pp. 387–424. DOI: 10.1007/978-3-031-68394-7_13.

[EKT25] Thomas Espitau, Shuichi Katsumata, and Kaoru Takemure. "Two-Round Threshold Signature from Algebraic One-More Learning with Errors." In: *Journal of Cryptology* 38.4 (Oct. 2025), p. 31. DOI: 10.1007/s00145-025-09549-2.

[ENP24] Thomas Espitau, Guilhem Niot, and Thomas Prest. "Flood and Submerse: Distributed Key Generation and Robust Threshold Signature from Lattices." In: *CRYPTO 2024, Part VII*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14926. LNCS. Springer, Cham, Aug. 2024, pp. 425–458. DOI: 10.1007/978-3-031-68394-7_14.

[FK19] Armando Faz-Hernandez and Kris Kwiatkowski. *Introducing CIRCL: An Advanced Cryptographic Library*. Available at https://github.com/cloudflare/circl. v1.6.0 Accessed Jan, 2025. Cloudflare. June 2019.

[FNR25] Sebastian Faller, Guilhem Niot, and Michael Reichle. *Lattice-based Threshold Blind Signatures*. Cryptology ePrint Archive, Paper 2025/1566. 2025. URL: https://eprint.iacr.org/2025/1566.

[FP01] Pierre-Alain Fouque and David Pointcheval. "Threshold Cryptosystems Secure against Chosen-Ciphertext Attacks." In: *ASIACRYPT 2001*. Ed. by Colin Boyd. Vol. 2248. LNCS. Springer, Berlin, Heidelberg, Dec. 2001, pp. 351–368. DOI: 10.1007/3-540-45682-1_21.

[FS01] Pierre-Alain Fouque and Jacques Stern. "Fully Distributed Threshold RSA under Standard Assumptions." In: *ASIACRYPT 2001*. Ed. by Colin Boyd. Vol. 2248. LNCS. Springer, Berlin, Heidelberg, Dec. 2001, pp. 310–330. DOI: 10.1007/3-540-45682-1_19.

[FS87] Amos Fiat and Adi Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems." In: *CRYPTO'86*. Ed. by Andrew M. Odlyzko. Vol. 263. LNCS. Springer, Berlin, Heidelberg, Aug. 1987, pp. 186–194. DOI: 10.1007/3-540-47721-7_12.

[GG18] Rosario Gennaro and Steven Goldfeder. "Fast Multiparty Threshold ECDSA with Fast Trustless Setup." In: *ACM CCS 2018*. Ed. by David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang. ACM Press, Oct. 2018, pp. 1179–1194. DOI: 10.1145/3243734.3243859.

[GGN16] Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. "Threshold-Optimal DSA/ECDSA Signatures and an Application to Bitcoin Wallet Security." In: *ACNS 2016*. Ed. by Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider. Vol. 9696. LNCS. Springer, Cham, June 2016, pp. 156–174. DOI: 10.1007/978-3-319-39555-5_9.

[GHKR08] Rosario Gennaro, Shai Halevi, Hugo Krawczyk, and Tal Rabin. "Threshold RSA for Dynamic and Ad-Hoc Groups." In: *EUROCRYPT 2008*. Ed. by Nigel P. Smart. Vol. 4965. LNCS. Springer, Berlin, Heidelberg, Apr. 2008, pp. 88–107. DOI: 10.1007/978-3-540-78967-3_6.

[GHKS+25] Kamil Doruk Gur, Patrick Hough, Jonathan Katz, Caroline Sandsbråten, and Tjerand Silde. *Olingo: Threshold Lattice Signatures with DKG and Identifiable Abort*. Cryptology ePrint Archive, Report 2025/1789. 2025. URL: https://eprint.iacr.org/2025/1789.

[GHL22] Craig Gentry, Shai Halevi, and Vadim Lyubashevsky. "Practical Non-interactive Publicly Verifiable Secret Sharing with Thousands of Parties." In: *EUROCRYPT 2022, Part I*. Ed. by Orr Dunkelman and Stefan Dziembowski. Vol. 13275. LNCS. Springer, Cham, May 2022, pp. 458–487. DOI: 10.1007/978-3-031-06944-4_16.

[GJKR96] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. "Robust Threshold DSS Signatures." In: *EUROCRYPT'96*. Ed. by Ueli M. Maurer. Vol. 1070. LNCS. Springer, Berlin, Heidelberg, May 1996, pp. 354–371. DOI: 10.1007/3-540-68339-9_31.

[GJKR99] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. "Secure Distributed Key Generation for Discrete-Log Based Cryptosystems." In: *EUROCRYPT'99*. Ed. by Jacques Stern. Vol. 1592. LNCS. Springer, Berlin, Heidelberg, May 1999, pp. 295–310. DOI: 10.1007/3-540-48910-X_21.

[GJMS+24] Sanjam Garg, Abhishek Jain, Pratyay Mukherjee, Rohit Sinha, Mingyuan Wang, and Yinuo Zhang. "hinTS: Threshold Signatures with Silent Setup." In: *2024 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2024, pp. 3034–3052. DOI: 10.1109/SP54263.2024.00057.

[GJS15] Qian Guo, Thomas Johansson, and Paul Stankovski. "Coded-BKW: Solving LWE Using Lattice Codes." In: *CRYPTO 2015, Part I*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9215. LNCS. Springer, Berlin, Heidelberg, Aug. 2015, pp. 23–42. DOI: 10.1007/978-3-662-47989-6_2.

[GKS24] Kamil Doruk Gür, Jonathan Katz, and Tjerand Silde. "Two-Round Threshold Lattice-Based Signatures from Threshold Homomorphic Encryption." In: *Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part II*. Ed. by Markku-Juhani Saarinen and Daniel Smith-Tone. Springer, Cham, June 2024, pp. 266–300. DOI: 10.1007/978-3-031-62746-0_12.

[GLOV25] Pascal Giorgi, Fabien Laguillaumie, Lucas Ottow, and Damien Vergnaud. *Threshold Niederreiter: Chosen-Ciphertext Security and Improved Distributed Decoding*. Cryptology ePrint Archive, Report 2025/757. 2025. URL: https://eprint.iacr.org/2025/757.

[GN08] Nicolas Gama and Phong Q. Nguyen. "Predicting Lattice Reduction." In: *EUROCRYPT 2008*. Ed. by Nigel P. Smart. Vol. 4965. LNCS. Springer, Berlin, Heidelberg, Apr. 2008, pp. 31–51. DOI: 10.1007/978-3-540-78967-3_3.

[GNR10] Nicolas Gama, Phong Q. Nguyen, and Oded Regev. "Lattice Enumeration Using Extreme Pruning." In: *EUROCRYPT 2010*. Ed. by Henri Gilbert. Vol. 6110. LNCS. Springer, Berlin, Heidelberg, May 2010, pp. 257–278. DOI: 10.1007/978-3-642-13190-5_13.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. "Trapdoors for hard lattices and new cryptographic constructions." In: *40th ACM STOC*. Ed. by Richard E. Ladner and Cynthia Dwork. ACM Press, May 2008, pp. 197–206. DOI: 10.1145/1374376.1374407.

[GRJK00] Rosario Gennaro, Tal Rabin, Stanislaw Jarecki, and Hugo Krawczyk. "Robust and Efficient Sharing of RSA Functions." In: *Journal of Cryptology* 13.2 (Mar. 2000), pp. 273–300. DOI: 10.1007/s001459910011.

[GRSS+21] Alonso González, Hamy Ratoanina, Robin Salen, Setareh Sharifian, and Vladimir Soukharev. *Identifiable Cheating Entity Flexible Round-Optimized Schnorr Threshold (ICE FROST) Signature Protocol*. Cryptology ePrint Archive, Report 2021/1658. 2021. URL: https://eprint.iacr.org/2021/1658.

[GTJL+26] Liming Gao, Guofeng Tang, Dingding Jia, Yijian Liu, Bingqian Liu, Xianhui Lu, Kunpeng Wang, and Yongjian Yin. TalonG*: Bandwidth-Efficient Two-Round Threshold Signatures from Lattices*. Cryptology ePrint Archive, Paper 2026/303. 2026. URL: https://eprint.iacr.org/2026/303.

[GWWW25] Junqing Gong, Brent Waters, Hoeteck Wee, and David J. Wu. *Threshold Batched Identity-Based Encryption from Pairings in the Plain Model*. Cryptology ePrint Archive, Paper 2025/2103. 2025. URL: https://eprint.iacr.org/2025/2103.

[HKNW25] Keitaro Hashimoto, Shuichi Katsumata, Guilhem Niot, and Thom Wiggers. *Revisiting PQ WireGuard: A Comprehensive Security Analysis With a New Design Using Reinforced KEMs*. Cryptology ePrint Archive, Report 2025/1758. 2025. URL: https://eprint.iacr.org/2025/1758.

[HKW25] Keitaro Hashimoto, Shuichi Katsumata, and Thom Wiggers. "Bundled Authenticated Key Exchange: A Concrete Treatment of Signal's Handshake Protocol and Post-Quantum Security." In: *USENIX Security 2025*. Ed. by Lujo Bauer and Giancarlo Pellegrino. USENIX Association, Aug. 2025, pp. 6777–6796. URL: https://www.usenix.org/conference/usenixsecurity25/presentation/hashimoto-key-exchange.

[HNSW+21] Andreas Hülsing, Kai-Chun Ning, Peter Schwabe, Fiona Johanna Weber, and Philip R. Zimmermann. "Post-quantum WireGuard." In: *2021 IEEE Symposium on Security and Privacy (SP)*. 2021, pp. 304–321. DOI: 10.1109/SP40001.2021.00030.

[HPRR20] James Howe, Thomas Prest, Thomas Ricosset, and Mélissa Rossi. "Isochronous Gaussian Sampling: From Inception to Implementation." In: *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*. Ed. by Jintai Ding and Jean-Pierre Tillich. Springer, Cham, 2020, pp. 53–71. DOI: 10.1007/978-3-030-44223-1_4.

[HS15] Dennis Hofheinz and Victor Shoup. "GNUC: A New Universal Composability Framework." In: *Journal of Cryptology* 28.3 (July 2015), pp. 423–508. DOI: 10.1007/s00145-013-9160-y.

[HS20] Abida Haque and Alessandra Scafuro. "Threshold Ring Signatures: New Definitions and Post-quantum Security." In: *PKC 2020, Part II*. Ed. by Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas. Vol. 12111. LNCS. Springer, Cham, May 2020, pp. 423–452. DOI: 10.1007/978-3-030-45388-6_15.

[ISN87] M. Ito, A. Saito, and Takao Nishizeki. "Secret Sharing Schemes Realizing General Access Structure." In: *Proc. IEEE Global Telecommunication Conf. (Globecom'87)*. 1987, pp. 99–102.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. "Private Circuits: Securing Hardware against Probing Attacks." In: *CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. LNCS. Springer, Berlin, Heidelberg, Aug. 2003, pp. 463–481. DOI: 10.1007/978-3-540-45146-4_27.

[JD11] David Jao and Luca De Feo. "Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies." In: *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011*. Ed. by Bo-Yin Yang. Springer, Berlin, Heidelberg, Nov. 2011, pp. 19–34. DOI: 10.1007/978-3-642-25405-5_2.

[JLS86] William B Johnson, Joram Lindenstrauss, and Gideon Schechtman. "Extensions of Lipschitz maps into Banach spaces." In: *Israel Journal of Mathematics* 54.2 (1986), pp. 129–138.

[JMW24] Kelsey A. Jackson, Carl A. Miller, and Daochen Wang. "Evaluating the Security of CRYSTALS-Dilithium in the Quantum Random Oracle Model." In: *EUROCRYPT 2024, Part VI*. Ed. by Marc Joye and Gregor Leander. Vol. 14656. LNCS. Springer, Cham, May 2024, pp. 418–446. DOI: 10.1007/978-3-031-58751-1_15.

[JRS24] Corentin Jeudy, Adeline Roux-Langlois, and Olivier Sanders. "Phoenix: Hash-and-Sign with Aborts from Lattice Gadgets." In: *Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part I*. Ed. by Markku-Juhani Saarinen and Daniel Smith-Tone. Springer, Cham, June 2024, pp. 265–299. DOI: 10.1007/978-3-031-62743-9_9.

[JTZ23] Yunfeng Ji, Yang Tao, and Rui Zhang. "Robust (T, N)-Threshold Lattice Signature." In: (2023). DOI: 10.2139/ssrn.4588269. URL: http://dx.doi.org/10.2139/ssrn.4588269.

[KG20] Chelsea Komlo and Ian Goldberg. "FROST: Flexible Round-Optimized Schnorr Threshold Signatures." In: *SAC 2020*. Ed. by Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn. Vol. 12804. LNCS. Springer, Cham, Oct. 2020, pp. 34–65. DOI: 10.1007/978-3-030-81652-0_2.

[KG25] Chelsea Komlo and Ian Goldberg. "Arctic: Lightweight and Stateless Threshold Schnorr Signatures." In: *PKC 2025, Part V*. Ed. by Tibor Jager and Jiaxin Pan. Vol. 15678. LNCS. Springer, Cham, May 2025, pp. 234–267. DOI: 10.1007/978-3-031-91832-2_8.

[KGS23] Chelsea Komlo, Ian Goldberg, and Douglas Stebila. *A Formal Treatment of Distributed Key Generation, and New Constructions*. Cryptology ePrint Archive, Report 2023/292. 2023. URL: https://eprint.iacr.org/2023/292.

[KLS18] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. "A Concrete Treatment of Fiat-Shamir Signatures in the Quantum Random-Oracle Model." In: *EUROCRYPT 2018, Part III*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10822. LNCS. Springer, Cham, Apr. 2018, pp. 552–586. DOI: 10.1007/978-3-319-78372-7_18.

[KLSS23] Duhyeong Kim, Dongwon Lee, Jinyeong Seo, and Yongsoo Song. "Toward Practical Lattice-Based Proof of Knowledge from Hint-MLWE." In: *CRYPTO 2023, Part V*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14085. LNCS. Springer, Cham, Aug. 2023, pp. 549–580. DOI: 10.1007/978-3-031-38554-4_18.

[KNTW25] Shuichi Katsumata, Guilhem Niot, Ida Tucker, and Thom Wiggers. "Comprehensive Deniability Analysis of Signal Handshake Protocols: X3DH, PQXDH to Fully Post-Quantum with Deniable Ring Signatures." In: *USENIX Security 2025*. Ed. by Lujo Bauer and Giancarlo Pellegrino. USENIX Association, Aug. 2025, pp. 6797–6816. URL: https://www.usenix.org/conference/usenixsecurity25/presentation/katsumata.

[Kom26] Chelsea Komlo. "Updates on the FROST NIST Submission." In: *MPTS 2026: NIST Workshop on Multi-Party Threshold Schemes*. Joint work with Elizabeth Crites, Conrado Gouvea, Jack Grigg, Ian Goldberg, Jonathan Katz, Mary Maller, Simon Rastikian, Stefano Tessaro, Nikita Sorokovikov, Denis Varlakov, and Chenzhi Zhu. Jan. 2026. URL: https://csrc.nist.gov/presentations/2026/mpts2026-1a1.

[KRT24] Shuichi Katsumata, Michael Reichle, and Kaoru Takemure. "Adaptively Secure 5 Round Threshold Signatures from MLWE/MSIS and DL with Rewinding." In: *CRYPTO 2024, Part VII*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14926. LNCS. Springer, Cham, Aug. 2024, pp. 459–491. DOI: 10.1007/978-3-031-68394-7_15.

[KTR20] Ralf Küsters, Max Tuengerthal, and Daniel Rausch. "The IITM Model: A Simple and Expressive Model for Universal Composability." In: *Journal of Cryptology* 33.4 (Oct. 2020), pp. 1461–1584. DOI: 10.1007/s00145-020-09352-1.

[LDKL+22] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. *CRYSTALS-DILITHIUM*. Tech. rep. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022. National Institute of Standards and Technology, 2022.

[Lin17] Yehuda Lindell. "Fast Secure Two-Party ECDSA Signing." In: *CRYPTO 2017, Part II*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10402. LNCS. Springer, Cham, Aug. 2017, pp. 613–644. DOI: 10.1007/978-3-319-63715-0_21.

[Lin24] Yehuda Lindell. "Simple Three-Round Multiparty Schnorr Signing with Full Simulatability." In: *CiC* 1.1 (2024), p. 25. DOI: 10.62056/a36c0l5vt.

[LN18] Yehuda Lindell and Ariel Nof. "Fast Secure Multiparty ECDSA with Practical Distributed Key Generation and Applications to Cryptocurrency Custody." In: *ACM CCS 2018*. Ed. by David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang. ACM Press, Oct. 2018, pp. 1837–1854. DOI: 10.1145/3243734.3243788.

[LNÖ25] Anja Lehmann, Phillip Nazarian, and Cavit Özbay. "Stronger Security for Threshold Blind Signatures." In: *EUROCRYPT 2025, Part II*. Ed. by Serge Fehr and Pierre-Alain Fouque. Vol. 15602. LNCS. Springer, Cham, May 2025, pp. 335–364. DOI: 10.1007/978-3-031-91124-8_12.

[LNS21] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. "Shorter Lattice-Based Zero-Knowledge Proofs via One-Time Commitments." In: *PKC 2021, Part I*. Ed. by Juan Garay. Vol. 12710. LNCS. Springer, Cham, May 2021, pp. 215–241. DOI: 10.1007/978-3-030-75245-3_9.

[LS15] Adeline Langlois and Damien Stehlé. "Worst-case to average-case reductions for module lattices." In: *DCC* 75.3 (2015), pp. 565–599. DOI: 10.1007/s10623-014-9938-4.

[LSW25] Russell W. F. Lai, Monisha Swarnakar, and Ivy K. Y. Woo. "Leaky LWE: Learning with Errors with Semi-Adaptive Secret- and Error-Leakage." In: *IACR Communications in Cryptology* 2.3 (Oct. 6, 2025). ISSN: 3006-5496. DOI: 10.62056/ah89ksuc2.

[LW15] Vadim Lyubashevsky and Daniel Wichs. "Simple Lattice Trapdoor Sampling from a Broad Class of Distributions." In: *PKC 2015*. Ed. by Jonathan Katz. Vol. 9020. LNCS. Springer, Berlin, Heidelberg, Mar. 2015, pp. 716–730. DOI: 10.1007/978-3-662-46447-2_32.

[LWWS+25] Hao Lin, Mingqiang Wang, Weiqiang Wen, Shifeng Sun, and Kaitai Liang. "Generic construction of threshold ring signatures and lattice-based instantiations." In: *DCC* 93.9 (2025), pp. 3955–4017. DOI: 10.1007/s10623-025-01660-6.

[Lyu09] Vadim Lyubashevsky. "Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures." In: *ASIACRYPT 2009*. Ed. by Mitsuru Matsui. Vol. 5912. LNCS. Springer, Berlin, Heidelberg, Dec. 2009, pp. 598–616. DOI: 10.1007/978-3-642-10366-7_35.

[Lyu12] Vadim Lyubashevsky. "Lattice Signatures without Trapdoors." In: *EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. LNCS. Springer, Berlin, Heidelberg, Apr. 2012, pp. 738–755. DOI: 10.1007/978-3-642-29011-4_43.

[Mau10] Ueli Maurer. "Constructive Cryptography - A Primer (Invited Paper)." In: *FC 2010*. Ed. by Radu Sion. Vol. 6052. LNCS. Springer, Berlin, Heidelberg, Jan. 2010, p. 1. DOI: 10.1007/978-3-642-14577-3_1.

[McE78] Robert J. McEliece. *A public-key cryptosystem based on algebraic coding theory*. The Deep Space Network Progress Report 42-44. https://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF. Jet Propulsion Laboratory, California Institute of Technology, Jan. 1978, pp. 114–116.

[Mer90] Ralph C. Merkle. "A Certified Digital Signature." In: *CRYPTO'89*. Ed. by Gilles Brassard. Vol. 435. LNCS. Springer, New York, Aug. 1990, pp. 218–238. DOI: 10.1007/0-387-34805-0_21.

[MI88] Tsutomu Matsumoto and Hideki Imai. "Public Quadratic Polynominal-Tuples for Efficient Signature-Verification and Message-Encryption." In: *EUROCRYPT'88*. Ed. by C. G. Günther. Vol. 330. LNCS. Springer, Berlin, Heidelberg, May 1988, pp. 419–453. DOI: 10.1007/3-540-45961-8_39.

[MP12] Daniele Micciancio and Chris Peikert. "Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller." In: *EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. LNCS. Springer, Berlin, Heidelberg, Apr. 2012, pp. 700–718. DOI: 10.1007/978-3-642-29011-4_41.

[MR04] Daniele Micciancio and Oded Regev. "Worst-Case to Average-Case Reductions Based on Gaussian Measures." In: *45th FOCS*. IEEE Computer Society Press, Oct. 2004, pp. 372–381. DOI: 10.1109/FOCS.2004.72.

[MSJ02] Philip D. MacKenzie, Thomas Shrimpton, and Markus Jakobsson. "Threshold Password-Authenticated Key Exchange." In: *CRYPTO 2002*. Ed. by Moti Yung. Vol. 2442. LNCS. Springer, Berlin, Heidelberg, Aug. 2002, pp. 385–400. DOI: 10.1007/3-540-45708-9_25.

[MW17] Daniele Micciancio and Michael Walter. "Gaussian Sampling over the Integers: Efficient, Generic, Constant-Time." In: *CRYPTO 2017, Part II*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10402. LNCS. Springer, Cham, Aug. 2017, pp. 455–485. DOI: 10.1007/978-3-319-63715-0_16.

[Nat24] National Institute of Standards and Technology. *FIPS 204: Module-Lattice-Based Digital Signature Standard*. Federal Information Processing Standard (FIPS) Publication NIST FIPS 204. U.S. Department of Commerce, National Institute of Standards and Technology, Aug. 2024. DOI: 10.6028/NIST.FIPS.204. URL: https://csrc.nist.gov/pubs/fips/204/final.

[Ngu22] Ngoc Khanh Nguyen. "Lattice-Based Zero-Knowledge Proofs Under a Few Dozen Kilobytes." PhD thesis. ETH Zurich, Zürich, Switzerland, 2022. DOI: 10.3929/ETHZ-B-000574844. URL: https://hdl.handle.net/20.500.11850/574844.

[Nio25] Guilhem Niot. "Practical Deniable Post-Quantum X3DH: A Lightweight Split-KEM for K-Waay." In: *ASIACCS 25*. ACM Press, Aug. 2025, pp. 298–312. DOI: 10.1145/3708821.3736192.

[NRT26] Guilhem Niot, Michael Reichle, and Kaoru Takemure. "Adaptively-Secure Three-Round Threshold Schnorr from DL." In: *Advances in Cryptology – EUROCRYPT 2026*. Lecture Notes in Computer Science. To appear. Springer, 2026.

[Ped91] Torben P. Pedersen. "A Threshold Cryptosystem without a Trusted Party (Extended Abstract) (Rump Session)." In: *EUROCRYPT'91*. Ed. by Donald W. Davies. Vol. 547. LNCS. Springer, Berlin, Heidelberg, Apr. 1991, pp. 522–526. DOI: 10.1007/3-540-46416-6_47.

[Ped92] Torben P. Pedersen. "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing." In: *CRYPTO'91*. Ed. by Joan Feigenbaum. Vol. 576. LNCS. Springer, Berlin, Heidelberg, Aug. 1992, pp. 129–140. DOI: 10.1007/3-540-46766-1_9.

[Pei10] Chris Peikert. "An Efficient and Parallel Gaussian Sampler for Lattices." In: *CRYPTO 2010*. Ed. by Tal Rabin. Vol. 6223. LNCS. Springer, Berlin, Heidelberg, Aug. 2010, pp. 80–97. DOI: 10.1007/978-3-642-14623-7_5.

[Pen25] Penumbra Labs. *Threshold Custody and Signing in Penumbra*. https://guide.penumbra.zone/usage/pcli/wallet/threshold. Accessed: February 6, 2026. 2025.

[PFHK+22] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. *FALCON*. Tech. rep. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022. National Institute of Standards and Technology, 2022.

[PKNR+25] Rafaël Del Pino, Shuichi Katsumata, Guilhem Niot, Michael Reichle, and Kaoru Takemure. "Unmasking TRaccoon: A Lattice-Based Threshold Signature with An Efficient Identifiable Abort Protocol." In: *CRYPTO 2025, Part VI*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16005. LNCS. Springer, Cham, Aug. 2025, pp. 423–456. DOI: 10.1007/978-3-032-01887-8_14.

[PKPR24] Rafaël del Pino, Shuichi Katsumata, Thomas Prest, and Mélissa Rossi. "Raccoon: A Masking-Friendly Signature Proven in the Probing Model." In: *CRYPTO 2024, Part I*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14920. LNCS. Springer, Cham, Aug. 2024, pp. 409–444. DOI: 10.1007/978-3-031-68376-3_13.

[PN25] Rafaël Del Pino and Guilhem Niot. "Finally! A Compact Lattice-Based Threshold Signature." In: *PKC 2025, Part III*. Ed. by Tibor Jager and Jiaxin Pan. Vol. 15676. LNCS. Springer, Cham, May 2025, pp. 169–199. DOI: 10.1007/978-3-031-91826-1_6.

[PP21] Maxime Plançon and Thomas Prest. "Exact Lattice Sampling from Non-Gaussian Distributions." In: *PKC 2021, Part I*. Ed. by Juan Garay. Vol. 12710. LNCS. Springer, Cham, May 2021, pp. 573–595. DOI: 10.1007/978-3-030-75245-3_21.

[Pre17] Thomas Prest. "Sharper Bounds in Lattice-Based Cryptography Using the Rényi Divergence." In: *ASIACRYPT 2017, Part I*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Vol. 10624. LNCS. Springer, Cham, Dec. 2017, pp. 347–374. DOI: 10.1007/978-3-319-70694-8_13.

[PS24] Alain Passelègue and Damien Stehlé. "Low Communication Threshold Fully Homomorphic Encryption." In: *ASIACRYPT 2024, Part I*. Ed. by Kai-Min Chung and Yu Sasaki. Vol. 15484. LNCS. Springer, Singapore, Dec. 2024, pp. 297–329. DOI: 10.1007/978-981-96-0875-1_10.

[Rab98] Tal Rabin. "A Simplified Approach to Threshold and Proactive RSA." In: *CRYPTO'98*. Ed. by Hugo Krawczyk. Vol. 1462. LNCS. Springer, Berlin, Heidelberg, Aug. 1998, pp. 89–104. DOI: 10.1007/BFb0055722.

[Reg05] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography." In: *37th ACM STOC*. Ed. by Harold N. Gabow and Ronald Fagin. ACM Press, May 2005, pp. 84–93. DOI: 10.1145/1060590.1060603.

[RRJS+22] Tim Ruffing, Viktoria Ronge, Elliott Jin, Jonas Schneider-Bensch, and Dominique Schröder. "ROAST: Robust Asynchronous Schnorr Threshold Signatures." In: *ACM CCS 2022*. Ed. by Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi. ACM Press, Nov. 2022, pp. 2551–2564. DOI: 10.1145/3548606.3560583.

[RS06] Alexander Rostovtsev and Anton Stolbunov. *Public-Key Cryptosystem Based On Isogenies*. Cryptology ePrint Archive, Report 2006/145. 2006. URL: https://eprint.iacr.org/2006/145.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems." In: *Communications of the Association for Computing Machinery* 21.2 (Feb. 1978), pp. 120–126. DOI: 10.1145/359340.359342.

[SABD+22] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehlé, and Jintai Ding. *CRYSTALS-KYBER*. Tech. rep. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022. National Institute of Standards and Technology, 2022.

[Sch03] Claus Peter Schnorr. "Lattice Reduction by Random Sampling and Birthday Methods." In: *STACS 2003*. Ed. by Helmut Alt and Michel Habib. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 145–156. ISBN: 978-3-540-36494-8.

[SE94] C. P. Schnorr and M. Euchner. "Lattice basis reduction: Improved practical algorithms and solving subset sum problems." In: *Mathematical Programming* 66.1–3 (Aug. 1994), pp. 181–199. ISSN: 1436-4646. DOI: 10.1007/bf01581144. URL: http://dx.doi.org/10.1007/BF01581144.

[Sha49] Claude E. Shannon. "Communication theory of secrecy systems." In: *Bell Systems Technical Journal* 28.4 (1949), pp. 656–715.

[Sha79] Adi Shamir. "How to Share a Secret." In: *Communications of the Association for Computing Machinery* 22.11 (Nov. 1979), pp. 612–613. DOI: 10.1145/359168.359176.

[Shoo0] Victor Shoup. "Practical Threshold Signatures." In: *EUROCRYPT 2000*. Ed. by Bart Preneel. Vol. 1807. LNCS. Springer, Berlin, Heidelberg, May 2000, pp. 207–220. DOI: 10.1007/3-540-45539-6_15.

[Sho94] Peter W. Shor. "Algorithms for Quantum Computation: Discrete Logarithms and Factoring." In: *35th FOCS*. IEEE Computer Society Press, Nov. 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700.

[Sol26] Solv Protocol Team. *Solv Drives FROST Threshold Signatures for Bitcoin Mainnet Execution*. https://solv.finance/blog/frost-bitcoin-mainnet. Accessed: February 6, 2026. Jan. 2026.

[SS01] Douglas R. Stinson and Reto Strobl. "Provably Secure Distributed Schnorr Signatures and a (t, n) Threshold Scheme for Implicit Certificates." In: *ACISP 01*. Ed. by Vijay Varadharajan and Yi Mu. Vol. 2119. LNCS. Springer, Berlin, Heidelberg, July 2001, pp. 417–434. DOI: 10.1007/3-540-47719-5_33.

[Sta96] Markus Stadler. "Publicly Verifiable Secret Sharing." In: *EUROCRYPT'96*. Ed. by Ueli M. Maurer. Vol. 1070. LNCS. Springer, Berlin, Heidelberg, May 1996, pp. 190–199. DOI: 10.1007/3-540-68339-9_17.

[TPCZ24] Guofeng Tang, Bo Pang, Long Chen, and Zhenfeng Zhang. *Efficient Lattice-Based Threshold Signatures with Functional Interchangeability*. Cryptology ePrint Archive, Report 2024/1067. 2024. URL: https://eprint.iacr.org/2024/1067.

[TX25] Guofeng Tang and Haiyang Xue. "Robust Threshold ECDSA with Online-Friendly Design in Three Rounds." In: *2025 IEEE Symposium on Security and Privacy*. Ed. by Marina Blanton, William Enck, and Cristina Nita-Rotaru. IEEE Computer Society Press, May 2025, pp. 203–221. DOI: 10.1109/SP61157.2025.00115.

[VZK03] Liem Vo, Fangguo Zhang, and Kwangjo Kim. "A New Threshold Blind Signature Scheme from Pairings." In: (Mar. 2003).

[WR22] Bas Westerbaan and Cefan Daniel Rubin. *Defending Against Future Threats: Cloudflare Goes Post-Quantum*. https://blog.cloudflare.com/post-quantum-for-all/. Accessed: 2025-11-01. Cloudflare Blog, Oct. 2022.

[Yao86] Andrew Chi-Chih Yao. "How to Generate and Exchange Secrets (Extended Abstract)." In: *27th FOCS*. IEEE Computer Society Press, Oct. 1986, pp. 162–167. DOI: 10.1109/SFCS.1986.25.

[Zca25] Zcash Foundation. *The State of FROST for Zcash*. https://zfnd.org/the-state-of-frost-for-zcash/. Accessed: February 5, 2026. May 2025.

[ZT25] Chenzhi Zhu and Stefano Tessaro. "The Algebraic One-More MISIS Problem and Applications to Threshold Signatures." In: *CRYPTO 2025, Part I*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16000. LNCS. Springer, Cham, Aug. 2025, pp. 548–581. DOI: 10.1007/978-3-032-01855-7_18.

[NIST-IR8214C] Luís T. A. N. Brandão and René Peralta. *NIST First Call for Multi-Party Threshold Schemes*. (National Institute of Standards and Technology) NIST Internal Report (NISTIR) 8214C. 2026. DOI: 10.6028/NIST.IR.8214C.

COLLEGE MATHS, TELECOMS
DOCTORAL INFORMATIQUE, SIGNAL
BRETAGNE SYSTEMES, ELECTRONIQUE

Titre : Constructions pratiques de signature à seuil post-quantique à partir de réseaux euclidiens

Mot clés : Signature à seuil, Réseaux euclidiens, Robustesse, ML-DSA

Résumé : La cryptographie post-quantique a gagné en importance ces dernières années, motivée par la menace que les ordinateurs quantiques font peser sur les systèmes cryptographiques classiques. Bien que le NIST ait standardisé plusieurs schémas de chiffrement à clé publique et de signature numérique post-quantiques, il subsiste un fossé important dans notre compréhension des primitives cryptographiques avancées dans ce contexte. Parmi celles-ci, les signatures à seuil se distinguent comme une primitive particulièrement importante ; elles permettent à un groupe de participants de produire conjointement une signature valide, offrant ainsi une résilience face aux défaillances et au compromis partiel.

Alors que les schémas de signature à seuil classiques ont atteint un stade de maturité et sont déployés, le développement de schémas de signature à seuil post-quantiques pratiques demeure un défi ouvert. Les constructions proposées manquent souvent de robustesse face aux participants malveillants, d'une génération de clé distribuée efficace, et de compatibilité avec les standards émergents.

Cette thèse fournit une boîte à outils complète pour la conception et le déploiement de signatures à seuil pratiques basées sur les réseaux euclidiens. Premièrement, nous établissons les fondations de la *robustesse*, en introduisant des techniques de génération de clé distribuée et de signature qui garantissent la création d'une signature valide même en présence de participants malveillants. Ensuite, nous nous tournons vers un protocole radicalement plus efficace, permettant l'identification des abandons (identifiable aborts), une propriété qui permet de détecter les comportements malveillants plutôt que de les prévenir. Finalement, nous présentons de nouvelles idées pour répondre au défi de compatibilité avec les standards, en présentant la première version à seuil pratique de ML-DSA.

Title: Achieving Practical Post-Quantum Threshold Signatures from Lattices

Keywords: Threshold signatures, Lattices, Robustness, ML-DSA

Abstract: Post-quantum cryptography has gained momentum in recent years, motivated by the threat that quantum computers pose to widely deployed classical public-key cryptography. While NIST has standardized several post-quantum public-key encryption and digital signature schemes, many advanced cryptographic primitives remain less well understood in the post-quantum setting. Among these, threshold signatures are particularly important: they enable a group of participants to jointly produce a valid signature, providing resilience against faults and partial compromise of the system.

Classical threshold signature schemes are mature and are getting widely adopted, but the development of practical post-quantum threshold signature schemes remains an open challenge. Proposed constructions often lack robustness against malicious participants, efficient distributed key generation, and compatibility with emerging standards.

This thesis develops a toolkit for designing and deploying practical, lattice-based threshold signatures. First, we establish a foundation for *robustness*, introducing techniques for distributed key generation and signing that guarantee the creation of a valid signature even in the presence of malicious participants. Next, we develop a much more efficient protocol that achieves *identifiable aborts*, a property that detects rather than prevents malicious behavior. Finally, we address standard compatibility by presenting the first practical threshold version of ML-DSA (Dilithium).