Concerning Lattice-Based Cryptography, and other matters

Alain Passelègue

Abstract

This habilitation thesis focuses on recent contributions to lattice-based cryptography, with an emphasis on Lyubashevsky-style signatures and threshold fully homomorphic encryption. After providing a high-level overview of my work over the last decade, the manuscript focuses on three technical chapters. The first chapter discusses recent results on Lyubashevsky's signatures, focusing in particular on the design of rejection sampling and on security analyses in the ROM and QROM; it also presents a rejection-free signature scheme based on convolved Gaussians. The second chapter focuses on my contributions to the construction of threshold fully homomorphic encryption schemes, in both synchronous and asynchronous settings, including results on distributed key generation and low-communication threshold FHE. Finally, the third technical chapter bridges the previous two by presenting a threshold variant of Dilithium/ML-DSA, recently standardized by NIST, based on threshold fully homomorphic encryption.

Foreword

A reader of fantasy literature may recognize in the title of this manuscript an allusion to the prologue of *The Fellowship of the Ring* by J. R. R. Tolkien. I have long appreciated that prologue for its concise, unadorned presentation of concepts that structure the world in which the narrative unfolds. In a similar spirit, this manuscript offers a high-level synthesis of my recent contributions to lattice-based cryptography, while leaving full definitions, complete proofs, and technical details to the corresponding papers.

Lattice-based cryptography is a comparatively recent focus of my research, and it was not my initial destination. The introduction that follows recounts my path through cryptography over the past decade, and, to extend the metaphor, could be entitled *"An Unexpected Journey into Lattice-Based Cryptography"*.

Contents

Chapter 1

Introduction

1.1 Provable Security

Looking back on my first encounter with cryptography about a decade ago, I find it somewhat ironic that my very first course was on lattice-based cryptography, with a particular focus on fully homomorphic encryption, and taught by Vadim Lyubashevsky. Beyond the feasibility of FHE and the interplay between computer science and mathematics at the heart of lattice-based cryptography, what struck me most was the paradigm of provable security: the ability to formalize both desired functionality and adversarial capabilities in a way that reflects real-world settings, and to derive rigorous guarantees from explicit assumptions. I found this paradigm deeply fascinating, and it has shaped how I approach cryptographic design and analysis. In particular, I have been repeatedly drawn to the definition and construction of primitives: what a primitive is meant to achieve, and what security notion best captures that goal. This emphasis on provable security frames my later work on lattice-based signatures and (threshold) fully homomorphic encryption, which form the core of this manuscript. Before turning to these results, I briefly return to my earlier works. The first notion I worked on was pseudorandom functions, which remain one of my favorite concepts in cryptography.

1.2 Early Works on Pseudorandom Functions

During my Ph.D., I studied the algebraic structure underlying number-theoretic pseudorandom functions (PRFs). In particular, my thesis introduced an algebraic framework that reduces the pseudorandomness of a certain class of functions to a simple algebraic property. This generic framework captures most existing constructions and extends naturally to related primitives, including related-key secure PRFs, aggregate PRFs, and multilinear PRFs. The resulting security guarantees hold under a family of Matrix Diffie–Hellman (MDDH) assumptions, which include classical instantiations such as DDH, DLin, and k-Lin. By choosing parameters appropriately, the framework yields PRFs (and related primitives) whose security can be based on any of these assumptions.

I particularly investigated related-key security (RKA security), which aims to model an adversary's ability to tamper with a scheme's secret key. First, I proposed a fix and some extensions to the Bellare–Cash framework [BC10; ABPP14], and constructed PRFs secure against broader classes of related-key attacks [ABP15a; ABP15b] (as well as multilinear and aggregate PRFs). Second, I gave the first PRF provably secure against XOR-related-key

attacks, assuming the existence of a weak form of multilinear maps [ABP19].

This early line of work led to the publications listed below, which I do not discuss further in this manuscript.

- **Related-Key Security for Pseudorandom Functions Beyond the Linear Barrier**
  with Michel Abdalla, Fabrice Benhamouda, and Kenneth G. Paterson
  CRYPTO 2014, Journal of Cryptology 2018
- **An Algebraic Framework for Pseudorandom Functions and Applications to Related-Key Security**
  with Michel Abdalla and Fabrice Benhamouda
  CRYPTO 2015
- **Multilinear and Aggregate Pseudorandom Functions: New Constructions and Improved Security**
  with Michel Abdalla and Fabrice Benhamouda
  ASIACRYPT 2015
- **Algebraic XOR-RKA-Secure Pseudorandom Functions from Post-Zeroizing Multilinear Maps**
  with Michel Abdalla and Fabrice Benhamouda
  ASIACRYPT 2019

1.3 Leakage-Resilient Cryptography

The classes of related-key attacks considered in the above works were, in hindsight, somewhat unrealistic (except, perhaps, for XOR-related-key attacks, but the corresponding construction relies on largely theoretical cryptographic objects). Around the same time, seeking a better understanding of realistic side-channel attacks, I turned to leakage-resilient cryptography. I contributed to two works on private multiplication in the probing model [BBP+16; BBP+17], an idealized framework intended to capture a range of side-channel attacks (e.g., attacks exploiting physical measurements from devices running cryptographic algorithms, such as power consumption or electromagnetic emanations). Although idealized, the probing model was shown equivalent (up to non-tight reductions) to a more realistic leakage model, namely the noisy leakage model [PR13; DDF14]. In my most recent work on leakage-resilient cryptography [PGMP19], I tightened the reductions between these models using different mathematical tools, in particular the Rényi divergence. This line of work resulted in the following publications.

- **Randomness Complexity of Private Circuits for Multiplication**
  with Sonia Belaïd, Fabrice Benhamouda, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud
  EUROCRYPT 2016
- **Private Multiplication over Finite Fields**
  with Sonia Belaïd, Fabrice Benhamouda, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud
  CRYPTO 2017
- **Unifying Leakage Models on a Rényi Day**
  with Dahmun Goudarzi, Ange Martinelli, and Thomas Prest
  CRYPTO 2019

1.4 Cryptographic Protocols with Advanced Functionalities

In a different (and fairly opposite) direction, I became interested in some of the most powerful (and theoretical) primitives in modern cryptography, notably functional encryption and indistinguishability obfuscation, as these notions became central topics in the field. More broadly, I developed an interest in advanced cryptographic primitives and in the relationships between them. I explored these questions for a couple of years, and my main contribution in this area is a result showing that (multi-key or succinct) secret-key functional encryption (combined with plain public-key encryption) implies indistinguishability obfuscation [BNPW16]. I later shifted my focus away from these notions, largely due to their limited practical relevance. Below is the full list of works I contributed to in this area.

- **From Cryptomania to Obfustopia through Secret-Key Functional Encryption**
  with Nir Bitansky, Ryo Nishimaki, and Daniel Wichs
  TCC 2016-B, Journal of Cryptology 2020
- **Non-Trivial Witness Encryption and Null-iO from Standard Assumptions**
  with Zvika Brakerski, Aayush Jain, Ilan Komargodski, and Daniel Wichs
  SCN 2018
- **Function-Revealing Encryption**
  with Marc Joye
  SCN 2018
- **Alternative Constructions of Asymmetric Primitives from Obfuscation: Hierarchical IBE, Predicate Encryption, and More**
  with Pooya Farshim and Georg Fuchsbauer
  INDOCRYPT 2020
- **Cumulatively All-Lossy-But-One Trapdoor Functions from Standard Assumptions**
  with Benoît Libert and Ky Nguyen
  SCN 2022

1.5 Zero-Knowledge Proofs

Later, I also worked on zero-knowledge proofs. Initially, this line of work was primarily driven by conceptual interest rather than by a specific application. This perspective has changed significantly with my recent experience with lattice-based cryptography, especially in the context of fully homomorphic encryption. The latter has highlighted concrete use cases for zero-knowledge proofs that still lack practical solutions, hence I plan to return to this area in the near future.

My contributions to this area are as follows. First, I constructed statistical zero-knowledge designated-verifier NIZK arguments for NP in pairing-free groups (and dual-mode NIZKs in bilinear groups) [LPWW20]. Second, I built unbounded simulation-sound NIZK arguments for equality of plaintexts encrypted with Regev PKE, which enable an instantiation of the Naor-Yung paradigm to obtain a KDM-CCA2-secure lattice-based PKE in the standard model [LNPT20]. Finally, I provided a new security analysis of PointProofs, a pairing-based vector commitment scheme, which removed strong assumptions from the prior analysis without modifying the scheme [LPR22]. The details about my contributions to this field can be found in the following works.

- **New Constructions of Statistical NIZKs: Dual-Mode DV-NIZKs and More**
  with Benoît Libert, David J. Wu, and Hoeteck Wee
  EUROCRYPT 2020
- **Simulation-Sound Proofs for LWE and Applications to KDM-CCA2 Security**
  with Benoît Libert, Khoa Nguyen, and Radu Ţiţiu
  ASIACRYPT 2020
- **PointProofs, Revisited**
  with Benoît Libert and Mahshid Riahinia
  ASIACRYPT 2022

1.6 Public-Key Encryption with Additional Properties

I started supervising my first students in 2021. My first Ph.D. student, Calvin Abou Haidar, worked on updatable public-key encryption, a relaxation of forward-secure public-key encryption designed to reduce overhead relative to forward-security. An updatable PKE is a PKE in which there is a mechanism (possibly triggered by a sender) to enforce an update of the recipient's public/secret key pair, so that messages encrypted under prior public keys remain confidential even if the recipient's (updated) secret key is later compromised. The notion is motivated by key-compromise recovery in end-to-end encrypted messaging. Together, we proposed efficient constructions based on DCR [ALP22] and (M-)LWE [APS23]. The latter construction is analogous to Kyber (a.k.a. ML-KEM) [BDK+18], the post-quantum key encapsulation mechanism recently standardized by NIST. We also strengthened the security models and extended our constructions to satisfy the stronger security notion by adding zero-knowledge proofs. In parallel, I worked on partially equivocable PKE with my second Ph.D. student, Mahshid Riahinia [LPR22].

- **Updatable Public Key Encryption from DCR: Efficient Constructions with Stronger Security**
  with Calvin Abou Haidar and Benoît Libert
  CCS 2022
- **New and Improved Constructions for Partially Equivocable Public Key Encryption**
  with Benoît Libert and Mahshid Riahinia
  SCN 2022
- **Efficient Updatable Public-Key Encryption from Lattices**
  with Calvin Abou Haidar and Damien Stehlé
  ASIACRYPT 2023

1.7 More Recent Works on Pseudorandom Functions

More recently, I have worked on extensions of PRFs, including constrained PRFs and PRFs tailored for advanced cryptography. My first contribution to this area is a family of low-complexity PRFs that are designed in particular for MPC [BIP+18] but proved well-suited for FHE as well (see, e.g., Chapter 4). While I had not been active on this topic for a couple of years, I kept following it closely, and got back to it with my second Ph.D. student, Mahshid Riahinia. Together, we drew connections between constrained PRFs and homomorphic secret sharing [CMPR23], leading to constructions of constrained PRFs for

inner-product predicates and for NC^1 relying on various assumptions. This work also had implications for secure computation. In addition, we worked on Pseudorandom Correlation Functions (PCFs). PCFs allow two parties, given correlated evaluation keys, to locally generate arbitrarily many pseudorandom correlated strings, e.g., Oblivious Transfer (OT) correlations, which can then be used by the two parties to jointly run secure computation protocols. In [BCM+24], we provided a novel and simple approach for constructing PCFs for OT correlations, by relying on constrained pseudorandom functions for a class of constraints containing a weak pseudorandom function (wPRF). We then showed that tweaking the Naor-Reingold pseudorandom function and relying on low-complexity pseudorandom functions, such as that from my prior work [BIP+18], allows one to instantiate our paradigm. We further extended our ideas to obtain efficient public-key PCFs, which allow the distribution of correlated keys between parties to be non-interactive. We obtained public-key PCFs with similar throughput to state-of-the-art interactive PCFs and roughly 4 orders of magnitude faster than prior public-key PCFs.

- **Exploring Crypto Dark Matter: New Simple PRF Candidates and Their Applications**
  with Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu
  TCC 2018
- **Constrained Pseudorandom Functions from Homomorphic Secret Sharing**
  with Geoffroy Couteau, Pierre Meyer, and Mahshid Riahinia
  EUROCRYPT 2023
- **Fast Public-Key Silent OT and More from Constrained Naor-Reingold**
  with Dung Bui, Geoffroy Couteau, Pierre Meyer, and Mahshid Riahinia
  EUROCRYPT 2024

1.8 Lattice-Based Signatures, Fully Homomorphic Encryption, and their Threshold Variants

The last area of research I have worked on, and the focus of this manuscript, is lattice-based cryptography. My transition to this area was motivated by two factors. First, changes in my local research environment led me to initiate new lattice-based projects (e.g., [APS23] mentioned above). Second, the outcome of the NIST standardization process for post-quantum signatures and key-exchange (which I had only followed from a distance) significantly changed my view of the maturity of lattice-based primitives for deployment and prompted me to revisit the topic with a renewed perspective. This, in turn, motivated my shift towards fully homomorphic encryption, which I see as a promising next step for privacy-preserving applications built on lattice assumptions.

In this manuscript, I focus on this most recent part of my work, which corresponds to (part of) my contributions from 2022 to 2026. I only briefly explain my first work on FHE in this introduction. In that work [CCP+24], we explained that IND-CPA-D security, long seen as a security notion that only applies to approximate FHE schemes like CKKS, should be seen as a security notion that applies to all schemes with non-negligible decryption failure (and therefore, correctness plays a role in the security of FHE). Indeed, all schemes (dealing either with approximate or exact data) become susceptible to IND-CPA-D attacks as soon as they have large decryption (or bootstrapping) failure probability. This work led me to work on threshold FHE, which can be seen as a generalization of IND-CPA-D security. In IND-CPA-D

security, an attacker is given a decryption oracle to the FHE scheme (on honestly generated plaintexts), while threshold FHE provides a partial decryption oracle to the attacker. Hence, as we remarked in [CCP+24], IND-CPA-D can be seen as the degenerate single-decryptor case of threshold FHE.

Below is a list of my contributions to lattice-based signatures and fully homomorphic encryption. All works except [CCP+24] are discussed in further detail in what follows.

- **On Rejection Sampling in Lyubashevsky's Signature Scheme**
  with Julien Devevey, Omar Fawzi, and Damien Stehlé
  ASIACRYPT 2022
- **A Detailed Analysis of Fiat-Shamir with Aborts**
  with Julien Devevey, Pouria Fallahpour, Damien Stehlé, and Keita Xagawa
  Manuscript, extended version of [DFPS23] (*CRYPTO 2023*)
- **G+G: A Fiat-Shamir Lattice Signature Based on Convolved Gaussians**
  with Julien Devevey and Damien Stehlé
  ASIACRYPT 2023
- **Attacks Against the IND-CPA-D Security of Exact FHE Schemes**
  with Jung Hee Cheon, Hyeongmin Choe, Damien Stehlé, and Elias Suvanto
  CCS 2024
- **Low Communication Threshold Fully Homomorphic Encryption**
  with Damien Stehlé
  ASIACRYPT 2024
- **Asynchronous Lagrange-Based Threshold FHE with Smaller Modulus Overhead**
  with Won Kim, Changmin Lee, Jeonghwan Lee, and Damien Stehlé
  CRYPTO 2026
- **THED: Threshold Dilithium from FHE**
  with Jai Hyun Park and Damien Stehlé
  Manuscript
- **On Threshold Fully Homomorphic Encryption with Synchronized Decryptors**
  with François Colin de Verdière and Damien Stehlé
  Manuscript
- **Distributed Key Generation for Efficient Threshold-CKKS**
  with Seonhong Min, Guillaume Hanrot, Jai Hyun Park, and Damien Stehlé
  Manuscript

1.9 Organization of this manuscript

The rest of this manuscript is organized into three technical chapters. In Chapter 2, I provide an overview of my contributions to lattice-based signatures, covering works on rejection sampling, security analyses in the ROM and QROM, and rejection-free signatures based on convolved Gaussians. In Chapter 3, I highlight my contributions to threshold fully homomorphic encryption, including constructions in the synchronous setting and in the asynchronous setting, as well as works on low communication threshold FHE and distributed key generation for threshold CKKS. Finally, in Chapter 4, I present a threshold variant of Dilithium based on threshold FHE. Chapter 5 concludes with final remarks.

Chapter 2

Lyubashevsky's Signatures

In this chapter, we[1] discuss some recent contributions to Lyubashevsky's signatures, in particular regarding their security analysis and rejection sampling (and how to remove it). The results presented in this chapter are sampled from the following works, and we refer the reader to their full versions for the technical details.

- **On Rejection Sampling in Lyubashevsky's Signature Scheme**
  with Julien Devevey, Omar Fawzi, and Damien Stehlé
  ASIACRYPT 2022
- **A Detailed Analysis of Fiat-Shamir with Aborts**
  with Julien Devevey, Pouria Fallahpour, Damien Stehlé, and Keita Xagawa
  Manuscript, extended version of [DFPS23] (*CRYPTO 2023*)
- **G+G: A Fiat-Shamir Lattice Signature Based on Convolved Gaussians**
  with Julien Devevey and Damien Stehlé
  ASIACRYPT 2023

Contents

[1] From here on, to reflect that the works I discuss are collaborations with many co-authors, I use the pronoun "we". In these works, I led (or co-led) the design of protocol and security models and conducted the security analyses. I also contributed substantially to cryptanalytic results in [PS24; CPS26] (detailed in the corresponding full versions), though I did not lead those efforts. Software implementations were carried out primarily by co-authors.

2.1 Background

Lyubashevsky's signature scheme [Lyu09; Lyu12] may be viewed as a lattice variant of Schnorr's group-based signature scheme [Sch91]. It is hard to overstate the importance of Lyubashevsky's signature scheme in lattice-based cryptography. Thanks to its elementary and flexible design, numerous variants and optimizations have been proposed (see [AFLT16; GLP15; DDLL13; BG14], or [Lyu16], for instance). Notably, it led to Dilithium (a.k.a. ML-DSA) [DKL+18; BDK+21], which was recently standardized by NIST, and to lattice-based zero-knowledge proofs [LNP22b]. Compared to Schnorr's signature, Lyubashevsky's protocol presents a core conceptual difference in its use of rejection sampling and the associated introduction of aborts and repeats in the Fiat-Shamir heuristic [FS86].

Lyubashevsky's signature scheme involves a publicly shared matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. The signing key is a matrix $\mathbf{S} \in \mathbb{Z}^{m \times k}$. It is small in the sense that all its entries have absolute values significantly smaller than q. The verification key associated with $\mathbf{S}$ is $\mathbf{T} = \mathbf{AS}$. Given a message $\mu \in \{0,1\}^*$, the signer samples a small masking vector $\mathbf{y} \in \mathbb{Z}^m$ and computes a random-looking commitment $\mathbf{w} = \mathbf{Ay}$. By using a hash function H taking small values in $\mathbb{Z}^k$, it computes a challenge $\mathbf{c} = \mathsf{H}(\mathbf{w}, \mu)$. Finally, if some (possibly probabilistic) test passes, it outputs a signature $\sigma = (\mathbf{z}, \mathbf{c})$ with $\mathbf{z} = \mathbf{y} + \mathbf{Sc}$, and else it restarts from scratch. Given a signature $\sigma = (\mathbf{z}, \mathbf{c})$ for a message μ, the verifier accepts if and only if $\mathbf{z}$ is small and $\mathsf{H}(\mathbf{Az} - \mathbf{Tc}, \mu) = \mathbf{c}$. We refer the reader to Figure 2.1 for a formal description. As suggested by the choice of terminology, Lyubashevsky's signature can be viewed as an identification protocol made non-interactive by relying on the Fiat-Shamir heuristic, i.e., by replacing a truly random $\mathbf{c}$ by the output of a hash function. To fix notation, we refer to $\mathbf{w}$ as the commitment, to $\mathbf{c}$ as the challenge, and to $\mathbf{z}$ as the response. A triple $(\mathbf{w}, \mathbf{c}, \mathbf{z})$ forms a transcript.

$\mathsf{KeyGen}(1^\lambda)$:

1 $\mathbf{B} \hookleftarrow \mathbb{Z}_q^{n \times (m-n)}$ and $\mathbf{S} \hookleftarrow P_{\mathbf{S}}$
2 $\mathbf{A} \leftarrow (\mathbf{I}_n | \mathbf{B})$
3 $\mathbf{T} \leftarrow \mathbf{AS}$
4 **return** $\mathsf{vk} = (\mathbf{A}, \mathbf{T}), \mathsf{sk} = (\mathbf{A}, \mathbf{S})$

$\mathsf{Verify}(\mu, (\mathbf{z}, \mathbf{c}), (\mathbf{A}, \mathbf{T}))$:

1 **if** $\|\mathbf{z}\| \leq \gamma$ *and* $\mathbf{c} = \mathsf{H}(\mathbf{Az} - \mathbf{Tc}, \mu)$ **then**
2 **return** 1
3 **else**
4 **return** 0

$\mathsf{Sign}((\mathbf{A}, \mathbf{S}), \mu)$:

1 $\mathbf{y} \hookleftarrow Q$
2 $\mathbf{w} \leftarrow \mathbf{Ay}$
3 $\mathbf{c} \leftarrow \mathsf{H}(\mathbf{w}, \mu)$
4 $\mathbf{z} \leftarrow \mathbf{y} + \mathbf{Sc}$
5 $u \hookleftarrow U([0,1])$
6 **if** $u \leq \min\left(\frac{P(\mathbf{z})}{M \cdot Q(\mathbf{y})}, 1\right)$ **then**
7 **return** $(\mathbf{z}, \mathbf{c})$
8 **else**
9 go to Step 1

Figure 2.1: Lyubashevsky's signature scheme.

Compared to Schnorr's signature scheme, the signing key $\mathbf{S}$ and mask $\mathbf{y}$ do not belong to a finite set, preventing the use of a uniform mask $\mathbf{y}$ to hide the sensitive term $\mathbf{Sc}$.[2] One possibility (see, e.g., [DPSZ12]) is to sample $\mathbf{y}$ exponentially larger than $\mathbf{Sc}$ as a function of

[2] If we view $\mathbf{y}$ and $\mathbf{S}$ over $\mathbb{Z}_q$ rather than $\mathbb{Z}$, then they do belong to a finite set; but for security, the masking should preserve smallness relative to q, which the uniform distribution modulo q does not achieve.

the security parameter, so that the distributions of $\mathbf{y}$ and $\mathbf{y} + \mathbf{Sc}$ have exponentially small statistical distance. As q must be larger than $\mathbf{y}$ and the smallness of $\mathbf{S}$ relative to q impacts security, this flooding approach leads to large parameters. Instead, Lyubashevsky [Lyu09; Lyu12] put forward the notion of Fiat-Shamir with Aborts. This is the reason for the test concerning $\mathbf{z}$ in the signing algorithm: it is so that the output signature $(\mathbf{z}, \mathbf{c})$ follows a distribution that is independent of the sensitive term $\mathbf{Sc}$.

2.2 On the Design of Rejection Sampling

In this section, we focus on contributions from [DFPS22], which studies the design of rejection sampling in Lyubashevsky's signatures.

A classic application of rejection sampling (see, e.g., [Dev86, Chapter 2]) is to use a source distribution Q that is convenient to sample from, to create samples from a target distribution P. In Lyubashevsky's scheme, the purpose differs: we start from a pre-source distribution Q for $\mathbf{y}$; it is shifted by $\mathbf{Sc}$, leading to a distribution $Q_{+\mathbf{Sc}}$ for $\mathbf{y} + \mathbf{Sc}$; the latter is the source distribution; it is rejected to a target distribution P for $\mathbf{z}$ that does not depend on the signing key $\mathbf{S}$. The purpose of rejection sampling here is to hide the sensitive data $\mathbf{Sc}$. Diverse choices of pairs of distributions have been put forward in the literature: uniform in hypercubes [Lyu09], Gaussian with the same standard deviation while allowing for some small statistical inaccuracy in the target distribution [Lyu12], a bimodal Gaussian source distribution with a Gaussian target distribution in association with an accommodating arithmetic modification of the scheme [DDLL13] (the modification consists in replacing q with $2q$ and changing key generation to ensure that $\mathbf{T} = -\mathbf{T} = q\mathbf{I} \bmod 2q$). The pre-source distribution Q is shifted by $(-1)^b\mathbf{Sc}$ for a uniform bit b, leading to a source distribution $Q_{\pm\mathbf{Sc}}$. We refer to this as the bimodal setting. In contrast, we now refer to the former two cases where the source distribution is $Q_{+\mathbf{Sc}}$ as the unimodal setting. The first choice (uniform distributions in hypercubes) leads to a simple design, notably used in Dilithium, whereas the latter two allow for more compact signatures. One may also want to add constraints on the number of loop iterations, notably to guarantee a signing runtime upper bound. In the extreme case of removing rejection altogether, it was recently shown in [ASY22] that a limited flooding suffices, compared to the exponential flooding discussed earlier. The first contributions we discuss in this chapter address the following question:

Given signing runtime requirements, which rejection sampling strategy leads to the most compact signatures?

In a signature $\sigma = (\mathbf{z}, \mathbf{c})$, the second component contributes to a small fraction of the bit-size: the main requirement on $\mathbf{c}$ is that it has sufficiently high min-entropy to make it hard to guess. On the other hand, the contribution of $\mathbf{z}$ towards signature length is mostly driven by $\|\mathbf{z}\|$, as this directly impacts security: for a given security level, the smaller $\|\mathbf{z}\|$, the more compact the signatures. For this reason, we simplify the overall objective to minimizing $\mathbb{E}_{\mathbf{x}\hookleftarrow P}(\|\mathbf{x}\|)$ under signing runtime requirements. We note that for classic rejection sampling with target P and source Q, the expected number of samples needed is $R_\infty(P\|Q)$ where $R_\infty(D_1\|D_2) = \sup_x D_1(x)/D_2(x)$ refers to the Rényi divergence of infinite order. Indeed, for classic rejection sampling, one samples x from Q and accepts with probability $P(x)/(M \cdot Q(x))$, for $M = R_\infty(P\|Q)$. This justifies using $R_\infty(P\|Q)$ to quantify the runtime for rejecting Q to P.

2.2.1 Contributions in a Nutshell

Our contributions concern the optimality of rejection sampling design choices towards optimizing signature sizes and signing runtime. We provide lower bounds, and study ways to reach and circumvent them. We start with our lower bounds.

- Considering Lyubashevsky's scheme with perfect rejection sampling to the target distribution P (as in [Lyu09]), the relevant quantity measuring the signing runtime is given by $M = \max_{\mathbf{S},\mathbf{c}} R_\infty(P\|Q_{+\mathbf{Sc}})$. We show (under a mild assumption) that for all P and Q such that M is finite, the expected norm $\mathbb{E}_{\mathbf{x}\hookleftarrow P}(\|\mathbf{x}\|)$ is $\Omega((m/\log M) \cdot \max_{\mathbf{S},\mathbf{c}} \|\mathbf{Sc}\|)$. Interestingly, this bound is a factor $\sqrt{m}$ lower than what is obtained for the typical choice of P and Q set as uniform distributions in hypercubes.
- In the case of perfect rejection with the accommodating arithmetic modification from [DDLL13], then the relevant quantity for measuring the signing runtime is $M = \max_{\mathbf{S},\mathbf{c}} R_\infty(P\|Q_{\pm\mathbf{Sc}})$, where $Q_{\pm\mathbf{Sc}}$ denotes the balanced mixture of $Q_{+\mathbf{Sc}}$ and $Q_{-\mathbf{Sc}}$. In this case, we show (under the same mild assumption) that for all P and Q such that M is finite, the expected norm $\mathbb{E}_{\mathbf{x}\hookleftarrow P}(\|\mathbf{x}\|)$ is $\Omega(\sqrt{m/\log M} \cdot \max_{\mathbf{S},\mathbf{c}} \|\mathbf{Sc}\|)$. This lower bound is actually reached (up to a constant factor) for P and Q Gaussian as in [DDLL13].
- We show that for any algorithm (terminating with probability 1) that selects one out of many samples from Q to get a sample from P, the expected number of required samples from Q is $\geq R_\infty(P\|Q)$. This lower bound is reached by classic rejection sampling. In the case of Lyubashevsky's signatures with exact rejection sampling, this general result implies that classic rejection sampling is the appropriate strategy when it comes to minimizing the expected runtime.

The lower bounds above seem to give little margin of improvement in the design choices of Lyubashevsky's signatures, except for the unimodal case, for which uniform distributions in hypercubes do not reach the lower bound. Our second set of main results considers ways to reach or circumvent these lower bounds. We provide a few more details on these results after the highlights.

- Concerning the unimodal case, one way to circumvent the results above is to consider imperfect rejection sampling, by allowing for an approximation to P whose accuracy is parameterized by some $\varepsilon > 0$ (as introduced in [Lyu12]). Then the relevant quantity to bound the runtime becomes $\max_{\mathbf{S},\mathbf{c}} R^\varepsilon_\infty(P\|Q_{+\mathbf{Sc}})$, where R^ε_∞ is a smoothed variant of R_∞ that we define. In this case, we improve the signature security analysis from [Lyu12] by using the Rényi divergence instead of the statistical distance to quantify the closeness to P of the output distribution. This allows choosing ε larger than previously, leading to a (limited) signature compactness improvement.
- Gaussian distributions provide better signature compactness in the bimodal and imperfect unimodal regimes, than uniforms in hypercubes in the perfect unimodal regime. However, uniforms in hypercubes are sometimes preferred (see, e.g., Dilithium), because they lead to a simpler implementation, which in turn makes protection against timing attacks easier. We consider uniforms in hyperballs as a new alternative for the choice of source and target distributions. We show that this choice reaches the two lower bounds for $\mathbb{E}_{\mathbf{x}\hookleftarrow P}(\|\mathbf{x}\|)$ for perfect rejection sampling and is as good as Gaussians for imperfect rejection sampling in practice in the imperfect regime. Interestingly, the rejection test for uniforms in hyperballs is very simple in the unimodal case, similarly to uniforms in hypercubes. We not only study the choice of uniforms in hyperballs in the asymptotic

regime, but also compare it to Dilithium.
- Finally, imperfect rejection from Q to P allows us to describe and analyze variants of rejection sampling where the maximum number of loop iterations is bounded. This provides trade-offs between maximum signing runtime and signature sizes. When instantiated to rejection-free sampling based on flooding, we recover the scheme and analysis from [ASY22], whereas it quickly converges to Lyubashevsky's signature scheme when the signing runtime bound grows.

The results concerning signature compactness for unbounded (perfect and imperfect) rejection sampling are summarized in Table 2.1.

	Unimodal ($\varepsilon = 0$)	Unimodal ($\varepsilon \geq 2^{-o(m)}$ and $\varepsilon = o(1/m)$)	Bimodal ($\varepsilon = 0$)
Hypercube	$\frac{tm^{3/2}}{\log M}$	$\frac{tm^{3/2}}{\log M}$	$\frac{tm^{3/2}}{\log M}$
Gaussian	∞	$\frac{t\sqrt{m}\sqrt{\log\frac{1}{\varepsilon}+\log M}}{\log M}$	$\frac{t\sqrt{m}}{\sqrt{\log M}}$
Hyperball	$\frac{tm}{\log M}$	$\frac{t\sqrt{m}\sqrt{\log\frac{1}{\varepsilon}+\log M}}{\log M}$	$\frac{t\sqrt{m}}{\sqrt{\log M}}$
Lower bound	$\frac{tm}{\log M}$	?	$\frac{t\sqrt{m}}{\sqrt{\log M}}$

Table 2.1: Compactness (i.e. $\mathbb{E}_{\mathbf{x}\hookleftarrow P}(\|\mathbf{x}\|)$) given the signing runtime constraint for various choices of distributions P and Q. The signing runtime constraint is modeled by $\max_{\mathbf{v}\in\mathcal{B}_m(t)} R^{\varepsilon}_{\infty}(P\|Q_{+\mathbf{v}}) \leq M$ in the unimodal case, where ε quantifies the accuracy of rejection sampling, and by $\max_{\mathbf{v}\in\mathcal{B}_m(t)} R_{\infty}(P\|Q_{\pm\mathbf{v}}) \leq M$ in the bimodal case. The last row gives a lower bound on the compactness for any choice of P and Q. Multiplicative constants are omitted in this table, and we make the assumption that $\log M \leq m$.

2.2.2 On Circumventing our Lower Bounds

Rényi-Based Analysis for Imperfect Rejection Sampling. All lower bounds are for perfect rejection sampling, in the sense that one obtains a sample from (exactly) P. In [Lyu12], Lyubashevsky shows that one can consider imperfect rejection sampling, and shows that it is particularly beneficial in the case of Gaussians. We propose an analysis that replaces the use of the statistical distance as done in [Lyu12] by that of the smooth Rényi divergence, and allows loosening the constraints on imperfectness. We first recall that in [Lyu12], the statistical distance is used to bound the statistical distance between a (single) execution of the imperfect rejection sampling algorithm and the target distribution. Using imperfect rejection sampling in a signature scheme and given bound ε for the above statistical distance, one can then bound the distinguishing advantage of an adversary between the real security game and the ideal game (where signatures are simulated by sampling them from the target distribution) by $q_{\mathsf{sig}} \cdot \varepsilon$. Here q_{sig} is a bound on the number of signature queries an adversary can make. We prove that for P, Q such that $R^{\varepsilon}_{\infty}(P\|Q)$ is finite, the Rényi divergence of infinite order between a (single) execution of the imperfect rejection sampling algorithm and the target distribution is bounded by $1/(1-\varepsilon)$. Combining this result with the multiplicativity of the Rényi divergence, we can then bound the Rényi divergence of infinite order between the adversary's view in the real game and its view in the ideal game by

$1/(1-\varepsilon)^{q_{\mathsf{sig}}}$ for the resulting signature. The probability preservation property of the Rényi divergence then allows completing the analysis. Our analysis leads to potential improvements as the former statistical bound $q_{\mathsf{sig}} \cdot \varepsilon$ imposes that $\varepsilon = 2^{-\Omega(\lambda)}$, while our bound can be used by setting $\varepsilon = 1/q_{\mathsf{sig}}$. Since q_{sig} is a (possibly large) polynomial of the security parameter λ, this puts less constraint on the condition P and Q must satisfy, which results in compactness improvement.

Hyperball Uniforms. We show that (continuous) uniform distributions over hyperballs reach the signature compactness lower bound (up to constant factors) in both unimodal and bimodal settings. We also show that they are experimentally as good as Gaussians for imperfect rejection sampling. These results reduce to Rényi divergence computations, which involve geometric properties of hyperballs. We emphasize that while Gaussian distributions also achieve similar signature size in both unimodal and bimodal settings (but only in the case of imperfect rejection sampling with polynomial loss for the unimodal case), using uniform distributions over hyperballs makes the rejection test as simple as computing $\|\mathbf{z}\|$ in the unimodal case since it consists only in checking that $\mathbf{z}$ is in the hyperball of the target distribution P. In the bimodal case, the rejection test involves computing two norms and flipping a coin. In order to use this distribution in a signature, we propose a generalization of Lyubashevsky's signature that allows for continuous source and target distributions, by adding a rounding step after accepting a sample. Its security relies on the same mechanisms as the discrete case. This strategy could also benefit Gaussian distributions, by allowing to replace discrete Gaussian sampling with possibly simpler continuous Gaussian sampling. To assess the practicality of this new choice of distributions, we propose parameters for a variant of Dilithium with uniform distributions in hyperballs. If considering the sum of bit-sizes of a verification key and a signature, the gains range from $\sim 25\%$ to $\sim 30\%$, depending on the security level, just like for the Gaussian variant, whose parameters we also update.

Bounded Rejection Sampling with Guaranteed Output. We propose an original strategy to use rejection sampling while guaranteeing a (moderate) worst-case runtime. This could be beneficial in the context of real-time systems. A simple strategy could consist in fixing a (very large) bound i on the number of iterations such that it fails to produce a sample with negligible probability. While this guarantees a worst-case runtime, the change is mainly cosmetic since it has to be large enough for the sampling to succeed. Our alternative solution leaves the choice of i open without ever failing: for a fixed bound i, it performs (up to) $i-1$ iterations of the classic rejection sampling and outputs a sample if it ever succeeds, otherwise, the last (i-th) iteration uses one-shot flooding techniques (as done in [ASY22]) to guarantee an output. The analysis makes heavy use of the smooth Rényi divergence and its properties. Different choices for the bound i offer various trade-offs, ranging from one-shot signatures ($i = 1$) as in [ASY22] to Lyubashevsky's expected polynomial-time signatures (i going to ∞).

2.3 On the Security of Lyubashevsky's Signatures

In this section, we focus on contributions from [DFP+23], which analyses the security of Lyubashevsky's signatures.

The combination of the Fiat-Shamir heuristic and rejection sampling leads to several difficulties when analyzing the resulting signature scheme. Most analyses of Lyubashevsky's signatures consider a variant that we refer to as Fiat-Shamir with Bounded Aborts (FSwBA). In this variant, the number of loop iterations is a priori bounded by a parameter B of the

scheme. If no non-aborting iteration is encountered within this bounded number B of loop iterations, the signing algorithm fails (the failure symbol $\perp$ is output). With FSwBA, the runtime analysis is trivial. In the security proof, the upper bound on the number of iterations is technically convenient as it provides a bound on how many random oracle values are being programmed by the challenger, which eases the analysis of the random oracle programming impact on the adversary's view. The most detailed security analyses are provided in [AFLT16] for the ROM, and in [KLS18] for the QROM. An alternative proof strategy in the QROM is suggested in [GHHM21], but not detailed. In concrete instances of Fiat-Shamir with Aborts, such as the Dilithium signature scheme [BDK+21], the signing algorithm typically does not enforce any upper bound on the number of loop iterations. We call this variant Fiat-Shamir with Unbounded Aborts (FSwUA). It is more difficult to analyze, as arbitrarily many hash values may be programmed by the challenger in the security proof.

2.3.1 Contributions in a Nutshell

Our first set of contributions relates to FSwBA. First, we explain that all existing security analyses of FSwBA contain a subtle common flaw, with additional errors in the QROM analysis of [KLS18]. We then provide two security analyses of FSwBA in the QROM, the first one by correcting the one from [KLS18], and the second by adapting the approach suggested in [GHHM21]. As detailed below, it turns out that these QROM analyses are incomparable. In the process, we prove that the underlying Σ-protocol achieves a stronger notion of zero-knowledge than usually considered for Σ-protocols with aborts, which enables the proofs. Still for FSwBA, as far as we are aware, there is no detailed correctness analysis in the literature: this is actually not trivial, and we provide a detailed correctness analysis.

Our second set of results concerns FSwUA. On the negative side, we exhibit an interactive proof system such that applying FSwUA to it leads to a signature scheme such that:

- for all signing keys, with non-zero probability over the random oracle randomness, signing loops forever for all messages; in particular, the expected signing runtime is infinite;
- with overwhelming probability over the random oracle randomness, for all messages and all signing keys, the expected runtime of signing over its own randomness is below a fixed polynomial.

This suggests modifying the signing efficiency requirement, in which the runtime expectation is not taken over the randomness of the random oracle, but should be bounded by a polynomial with overwhelming probability over the randomness of the random oracle. On the positive side, we give analyses of correctness, signing efficiency (with respect to the modified definition) and security for FSwUA in the QROM (with a tighter reduction in the ROM).

Finally, as a side contribution, we generalize our analysis to rely on a Σ-protocol whose simulator's quality is measured in terms of the Rényi divergence (rather than the statistical distance) for non-aborting transcripts. In the case of Lyubashevsky's signature with Gaussian distributions [Lyu12], when the signature is replaced with the non-aborting simulator in the security proof, the analysis based on the divergence provides security for a larger range of parameters. This allows one to decrease the standard deviation of the distribution in the signature, which in turn reduces the signature size by a small amount.

2.3.2 Security Analyses of Lyubashevsky's Signatures

We focus on analyzing the Fiat-Shamir with Aborts transform in the context of digital signatures. We specifically consider how this technique allows the challenger to simulate replies to sign queries without knowing the signing key (which is made possible by allowing the challenger to program the random oracle). More formally, we are interested in reducing the signature unforgeability under chosen message attacks (CMA) to its unforgeability under no-message attacks (NMA) and rely on the literature to argue about NMA security [Lyu09; Lyu12; AFLT16; DFMS19]. An adversary against the CMA security of a digital signature in the random oracle model is allowed to make two types of queries: sign queries and hash queries (the latter queries being classical in the ROM and in quantum superposition in the QROM). In the security analysis, we eventually want the challenger to be able to reply to the queries without relying on the signing key. For this purpose, we can let the challenger modify the way it replies to the queries, as long as the modifications are not visible to the adversary. The fact that the signing algorithm uses the hash function, which is controlled by the challenger, is helpful for simulating signatures without knowing the signing key, but induces a difficulty: the challenger must reply to hash queries and sign queries consistently. In the case of Schnorr's signature, a sign query uses the hash function exactly once. However, in the Fiat-Shamir with Aborts variant, a sign query uses the hash function several times: the hash function is evaluated once in every loop iteration.

In what follows, we first describe flaws from existing analyses in the bounded abort setting, and then how we fix them. In the process, we introduce a stronger zero-knowledge definition for Σ-protocols with aborts, which allows us to fix the analysis, and prove that existing protocols achieve this definition. We finally explain how our analysis extends to the case of unbounded aborts.

2.3.2.1 Flaws in Existing Analyses of FSwBA

An unsubstantiated intuition. We start by describing a first flaw appearing in all existing analyses. These analyses start as follows: in the genuine security experiment (denoted Game 0), all (successful or not) transcripts generated during a sign query use a challenge that is computed with the hash function. Then, a first hybrid (Game 1) changes the sign algorithm by sampling a uniformly random challenge and programming the hash function consistently with the successful proof transcript *only*. All proofs immediately conclude these two games are identical: the (unsubstantiated) intuition is that the adversary does not have access to the aborted transcripts, and hence programming these transcripts does not impact the adversary's view.

F1. Assume the challenger in the genuine CMA (or even CMA_1) security game answers a sign query μ using a sequence of commitments $\mathbf{w}_1, \mathbf{w}_2, \ldots$. Assume that rejecting is a deterministic function of $\mathbf{w}$ and $\mathbf{c}$ (this is for example the case for Lyubashevsky's signatures with the parameters considered in [AFLT16]). Then, as soon as $\mathbf{w}_1$ fails to produce a valid transcript, the hash value $\mathsf{H}(\mathbf{w}_1 \| \mu)$ is fixed and the sign oracle can no longer return a valid signature which uses commitment $\mathbf{w}_1$. This is not the case in Game 1, since the hash value $\mathsf{H}(\mathbf{w}_1 \| \mu)$ is not programmed by the failed attempt, and the sign query could return a signature $(\mathbf{w}_1, \mathbf{c}', \mathbf{z}')$ for $\mathbf{c}' \neq \mathbf{c}$.

The fact that the adversary can make hash queries on superposition of all inputs in the QROM makes it even more difficult to argue that the adversary cannot detect random oracle programmings, which induces additional errors.

Correlated challenges and implicit quantum sign queries in the QROM. To avoid the latter difficulty in the QROM, the reduction from [KLS18] is made history-free, i.e., the random oracle is never reprogrammed during the execution of the security game (see [BDF+11] for a general treatment of history-free reductions in the QROM). For this purpose, the authors let the hash function call the signing algorithm, to guarantee that the hash and sign queries are handled consistently. On the downside, any subsequent signing algorithm modification in the security proof is of a quantum nature, as the adversary can make quantum queries to the hash function. This leads the analysis to two additional errors.

Consider the CMA_1 security analysis [KLS18, Theorem 3.2]: as above, the reduction starts with Game 0, which is the genuine security experiment. In Game 1, on a hash query $(\mathbf{w}\|\mu)$, the oracle calls a $\mathsf{GetTrans}$ function which runs the signing algorithm on input μ and checks if the commitment $\mathbf{w}$ of the *non-aborting* transcript matches the hash query. If the random oracle is called on that input (possibly as part of a quantum superposition), it is programmed to reply with the challenge programmed by the signing algorithm. This guarantees consistencies of hash values defined by hash queries and by sign queries. In addition, Game 1 replaces the Σ-protocol execution in the $\mathsf{GetTrans}$ function (called in both sign and hash queries) by the simulator. The authors bound the advantage loss of that game hop by $Bq_{\mathsf{sig}}\varepsilon_{zk}$, with B being the maximum number of loop iterations, q_{sig} the number of sign queries, and ε_{zk} the zero-knowledge error of the underlying interactive protocol.

As pinpointed above, a first flaw **F1** comes from the fact *only the non-aborting* challenge is programmed by $\mathsf{GetTrans}$, but two additional flaws are induced by relying on the simulator in $\mathsf{GetTrans}$.

- **F2.** Recall that the zero-knowledge property of the underlying Σ-protocol is for a single execution of the protocol (as opposed to correlated executions). Hence, replacing executions which rely on challenges computed as hash values by simulated transcripts requires challenges to be statistically independent. This is only possible if the hash function is evaluated on distinct inputs $(\mathbf{w}\|\mu)$, which is not guaranteed: there might be collisions on commitments $\mathbf{w}$'s used within a sign query for a message μ.
- **F3.** Since the adversary can only make classical sign queries, it could seem that transitioning from real to simulated transcripts is required only for those that are generated by the sign queries (there are at most Bq_{sig} of them, leading to the $Bq_{\mathsf{sig}}\varepsilon_{zk}$ term). However, the adversary can make quantum hash queries, and for consistency of hash evaluation, these queries make calls to the $\mathsf{GetTrans}$ function. Hence this transition has to be done for all possible sign queries (not only those that are actually made). In particular, even in the ROM, the reduction loss should already be $B(q_{\mathsf{sig}} + q_{\mathsf{H}})\varepsilon_{zk}$ as each sign query and hash query induces up to B simulated transcripts. In the QROM, the loss is even larger as the adversary can make q_{H} quantum hash queries.

Flaws **F2** and **F3** both appear in the QROM analyses of [Kat21, Lemma 4.6] and [KLS18, Theorems 3.2 and 3.3].

2.3.2.2 Corrected Analyses of FSwBA

The security analysis in the ROM, which only suffers from **F1**, can be readily modified to handle this difficulty by bounding the probability the random oracle gets evaluated twice on a previously defined input. If the commitment has high min-entropy, this event happens with negligible probability. **F1** vanishes as the hash function is never evaluated twice on the same input. It would seem that the rest of the analysis goes through (e.g., following [AFLT16]), but this fix induces an additional problem, which also occurs in our QROM fixed analyses.

Insufficiency of the usual simulators for Σ-protocols with aborts. While the above approach seems sound, it induces an additional subtle problem: we now run all aborting and non-aborting executions of the underlying Σ-protocol at every step of the reduction, and in particular in the game hop replacing real transcripts by simulated ones. The no-abort Honest-Verifier Zero-Knowledge (naHVZK) property usually considered for Σ-protocols with aborts is insufficient to analyze this game hop. Rather than trying to rely on the prior naHVZK notion, we choose an alternative route and exploit a stronger Honest-Verifier Zero-Knowledge (HVZK) for Σ-protocols with aborts, which requires the simulator to be able to simulate both aborting and non-aborting transcripts. Equipped with this definition, the above proof goes through immediately.

There is still one major issue to solve: this definition of strong simulation is not known to be achieved by Σ-protocols involved in Lyubashevsky's signatures (which might be the reason for the existence of the naHVZK notion). We construct a simulator for this setting, which works as follows: With probability p, it generates a non-aborting transcript (using the well-known naHVZK simulator), with p being the known probability that a protocol iteration does not trigger an abort. Else, with probability $1 - p$, it returns a uniform commitment $\mathbf{w}$ (and $\perp$ for the z-part of the transcript). The main technicality is to show that uniform commitments are indeed indistinguishable from aborting transcripts. Recall that the commitment $\mathbf{w}$ is of the form $\mathbf{Ay}$ for a public matrix $\mathbf{A}$ and a vector $\mathbf{y}$ sampled from a source distribution Q. If Q has high min-entropy, we use the fact that aborting does not decrease much the min-entropy of Q and use the leftover hash lemma to conclude that the protocol is *statistical* zero-knowledge. While this already handles many settings of Lyubashevsky's signature, we want the source distribution to have lower entropy in some cases. We prove that if the distribution Q is such that LWE is hard for noise distribution set to Q, the protocol is *computational* zero-knowledge, for a variant of computational zero-knowledge that is compatible with the Fiat-Shamir transform.

In addition to this fixed analysis in the ROM, we provide two different analyses in the QROM. The first one follows and fixes the [KLS18] analysis. The second one extends the adaptive reprogramming technique of [GHHM21] to Fiat-Shamir with Aborts and achieves CMA security. When instantiated to the ROM, the latter analysis is arguably simpler than the one from [AFLT16], for which reason we will only describe this one.

A security analysis in the QROM based on adaptive reprogramming. We provide a different security analysis based on the technique developed in [GHHM21]. In the latter, the authors study adaptive reprogramming in the QROM and exploit it to analyze the (no-abort) Fiat-Shamir heuristic.

Adaptive reprogramming considers a setting in which a quantum adversary has access to a random oracle, and in addition can query a reprogramming oracle $\mathcal{O}$ with inputs μ. The oracle answers to such a query by sampling $\mathbf{w}$ from a target distribution (the commitment

space in our case) and returning it to the adversary. In addition, the oracle either leaves the random oracle unchanged, or reprograms it on input $(\mathbf{w}\|\mu)$. In the classical setting, it is clear that an adversary cannot tell whether $\mathcal{O}$ affects the random oracle unless it has already made the hash query $(\mathbf{w}\|\mu)$. In [GHHM21], the authors provide a bound for the distinguishing advantage of a quantum adversary.

Adaptive reprogramming allows one to immediately move from Game 0 to a Game 1 in which the GetTrans function, on input μ, samples fresh uniformly random and independent challenges $\mathbf{c}$ and reprograms the random oracle according to $\mathbf{c}$ on input $\mathbf{w}\|\mu$. This immediately solves **F1** as it programs all intermediate values (even though some values can get programmed multiple times), as well as **F2** since challenges are now set to uniformly random and independent values thanks to reprogramming. Note that hash queries do not need to run GetTrans as adaptive reprogramming guarantees the adversary cannot find inconsistencies (which would allow to distinguish Games 0 and 1). It remains to replace real transcripts by simulated ones, which is easily argued with a security loss of $Bq_{\mathsf{sig}}\varepsilon_{zk}$, since only the (classical) sign queries rely on running the simulator. Doing so, we circumvent **F3**. One then needs to keep consistency in the hash values, which is done by keeping track of the last values reprogrammed by the (polynomial number of classical) sign queries.

Correctness analysis of FSwBA signatures. In addition to these technical issues regarding the security analysis, it turns out that bounding the number of loop iterations and returning $\perp$ when the bound is reached makes the correctness analysis somewhat non-trivial. The goal is to provide a small upper bound on the probability that the signing algorithm outputs $\perp$. For this purpose, it is tempting to argue that at each loop iteration, the abort probability is the failure probability $\beta \in (0, 1)$ of the underlying proof system, and hence that the signing abort probability is β^B where B is the bound on the number of iterations. This is incorrect, as the executions of the underlying proof system are not statistically independent: all challenges are derived from the hash function. It hence seems unavoidable to assume the ROM not only for security but also for correctness, but this is not sufficient, as statistical dependencies between the loop iterations can stem from collisions between inputs of the hash function: if the hash inputs are the same in two iterations, the returned challenges are the same.

We provide a detailed proof of correctness. For this, we observe that the security analyses involves a game in which the signing loop iterations are statistically independent: the β^B bound above holds in these experiments. We then argue that the failure probability in the genuine execution is close to β^B, as otherwise we would be able to distinguish the genuine security experiment from the one in which the signing loop iterations are statistically independent. Our correctness analysis for FSwBA is actually a corollary of our (runtime) analysis of FSwUA.

Wrapping up on FSwBA. We obtain several complete analyses with distinct security claims for signatures based on FSwBA, both in the ROM and the QROM. We provide an overview of our results in Table 2.2, using the same notation as above. The "reduction loss" is a bound on the difference of success probabilities of the adversary in the CMA and NMA security experiments. We assume the circuit model for quantum computations, except when mentioned otherwise. The table assumes that $q_{\mathsf{H}} \geq B \cdot q_{\mathsf{sig}}$ (this assumption is justified by the fact that hash evaluations can be made without restriction whereas sign queries require interaction with the signer). Similarly, the zero-knowledge simulation time is neglected (unless it is very large, its contribution is typically dominated by the terms in the table). We also

omit constant factors. Note that the reduction in the ROM simulates the random oracle using the lazy sampling method. However, the QROM reductions are relative to another random oracle that is accessible to the challenger (this assumption may be removed by relying on a quantum pseudo-random function [Zha12]).

Analysis	Hash function	Reduction loss	Reduction runtime overhead
Adaptive reprogramming	ROM	$2^{-\alpha} B q_{\mathsf{sig}} q_{\mathsf{H}} + \varepsilon_{zk} B q_{\mathsf{sig}}$	$q_{\mathsf{H}} \log(q_{\mathsf{H}})$
Adaptive reprogramming	QROM	$2^{-\alpha/2} B q_{\mathsf{sig}} q_{\mathsf{H}}^{1/2} + \varepsilon_{zk} B q_{\mathsf{sig}}$	$q_{\mathsf{H}} \log(B q_{\mathsf{sig}})$ with QRACM $B q_{\mathsf{sig}} q_{\mathsf{H}}$ without
History-free for CMA_1 security	QROM	$2^{-\alpha/2} B q_{\mathsf{H}} + \varepsilon_{zk}^{1/2} B^{1/2} q_{\mathsf{H}}^{3/2}$	$B q_{\mathsf{H}}$
History-free for CMA security	QROM	$2^{-\alpha/2} B q_{\mathsf{sig}} q_{\mathsf{H}} + \varepsilon_{zk}^{1/2} B^{1/2} q_{\mathsf{H}}^{3/2}$	$B q_{\mathsf{sig}} q_{\mathsf{H}}$

Table 2.2: Comparison of the security analyses of FSwBA, with α denoting the min-entropy of commitments, q_{sig} the number of sign queries, q_{H} the number of hash queries, and B the parameter bounding the number of loop iterations.

We observe that the QROM analyses are incomparable. In particular, the adaptive reprogramming technique from [GHHM21] is tight only when assuming quantum random access classical memory (QRACM), which is a stronger assumption than the quantum circuit model of computation. The history-free technique from [KLS18] is tight only when considering adversaries that may make at most one sign query for any message (CMA_1 security). This covers the deterministic version of the resulting signature, obtained by deriving the randomness from the message via a pseudo-random function evaluation. For CMA security, the reduction is not tight (even assuming QRACM) and the reduction loss is higher than the one obtained with the adaptive reprogramming technique.

2.3.3 Concrete Analysis of FSwUA

On the termination of FSwUA signatures. For FSwUA, we start by exhibiting an underlying identification scheme with the following peculiar property: for any execution of the key generation algorithm of the resulting signature, there exists a hash function such that the resulting signing algorithm loops forever on every input message. Yet, with overwhelming probability over the random choice of the hash function, the expected runtime is polynomially bounded. The scheme is a variant of Lyubashevsky's [Lyu09; Lyu12], with carefully crafted source and target distributions. To make sure that every loop iteration always fails, we use a source and a target distribution that are uniform over some sets X_S and X_T, respectively, with $X_T \subseteq X_S$. The choice of uniform distributions leads to a deterministic rejection test: an iteration takes a uniform $\mathbf{y} \in X_S$ and maps it to a vector $\mathbf{z}$, and an abort occurs if $\mathbf{z} \notin X_T$. Going a little further into the details, the vector $\mathbf{z}$ is of the form $\mathbf{z} = \mathbf{y} + \mathbf{S} \cdot \mathbf{c}$, where the integer matrix $\mathbf{S}$ is the signing key and $\mathbf{c}$ is the output of the hash function H on a function of $\mathbf{y}$ and the message. We want to design X_S and X_T such that: (1) for all $\mathbf{S}$, the probability over $\mathbf{y} \leftarrow U(X_S)$ and H that $\mathbf{z} = \mathbf{y} + \mathbf{S} \cdot \mathbf{c}$ belongs to X_T is at least a positive constant, and

(2) for all $\mathbf{S}$, there exists an H such that for all $\mathbf{y}$ and message, the vector $\mathbf{z} = \mathbf{y} + \mathbf{S} \cdot \mathbf{c}$ does not belong to X_T.

The first condition forces us to set X_S not much larger than X_T. For the second condition, we design H so that any $\mathbf{y}$ is sent outside of X_T. As H depends on a function of $\mathbf{y}$, we first make sure that this function is injective, so that H is a function of $\mathbf{y}$ itself (else we would have to consider the set of predecessors and design H to jointly send them all outside of X_T). This injectivity is obtained by relying on the lossy version of Lyubashevsky's signature scheme [AFLT16]. Then for a vector $\mathbf{y}$, we design $\mathbf{c}$ so that $\mathbf{y} + \mathbf{S} \cdot \mathbf{c}$ is not in X_T and set $\mathbf{c}$ as the output of H on $\mathbf{y}$ and the message. For this purpose, we set X_S as a hyperball and X_T as an inner crust (a corona that almost aligns with the hyperball boundary), as we illustrate in Figure 2.2. As hyperballs are concentrated on their surface, the volume ratio can be bounded by a positive constant even with a thin crust. Now, if $\mathbf{y} \in X_S \setminus X_T$, we set $\mathbf{c} = \mathbf{0}$ (for every message). If $\mathbf{y}$ belongs to X_T, we choose $\mathbf{y}' \in X_S \setminus X_T$ near $\mathbf{y}$ and define $\mathbf{c}$ such that $\mathbf{y}+\mathbf{S}\cdot\mathbf{c}$ is very close to $\mathbf{y}'$: for this purpose, it suffices to round $\mathbf{y}'$ to the lattice spanned by $\mathbf{S}$; by taking $\mathbf{S}$ that is well-conditioned, we can guarantee that the rounded vector is close to $\mathbf{y}'$ and remains outside of X_T. As a result, all loop iterations of the resulting FSwUA signing algorithm fail.

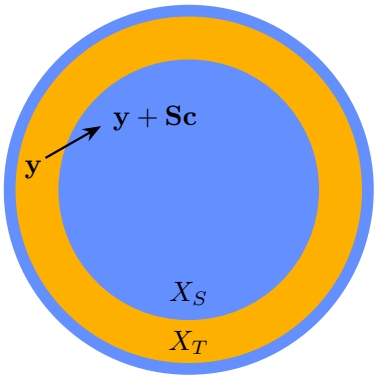

Figure 2.2: Counter-example in dimension 2.

The above counter-example is admittedly contrived, but illustrates the fact that specific difficulties arise when analyzing the unbounded version of Fiat-Shamir with Aborts. In particular, this suggests modifying the requirement of signing runtime, so that it is authorized to take longer than desired, but only with small probability over the randomness of the random oracle. We show that the signature obtained with the FSwUA transform indeed fulfills this requirement (in the random oracle model).

Security and correctness analyses of FSwUA signatures. We reduce the CMA security of FSwUA signatures to their NMA security, both in the ROM and the QROM. For this purpose, it is tempting to add a bound on the number of loop iterations, argue that the adversary cannot notice the difference, and then use the NMA security to CMA security reduction of FSwBA signatures. To prove that the adversary cannot notice the difference between the unbounded and bounded versions of the signing algorithm, one would argue that the probability of reaching that bound in at least one sign query is negligible, as the number of loop iterations follows a geometric law. But as discussed earlier, this is not true since there is a statistical dependency between different iterations of the rejection sampling. However, we show that the probability of the number of iterations until a success outcome is larger than B is small, over the randomness of the random oracle. This allows us to show the expected equivalence between FSwBA and FSwUA when the bound B is large, as an adversary will likely never see $\perp$ with the FSwBA variant. Using the same notations as before, the reduction loss of this step is bounded by $q_{\mathsf{sig}} \cdot \beta^B + 2^{-\alpha/2} \cdot B q_{\mathsf{sig}} \cdot \sqrt{q_{\mathsf{H}}}$ in the QROM and by $q_{\mathsf{sig}} \cdot \beta^B + 2^{-\alpha} \cdot B q_{\mathsf{sig}} q_{\mathsf{H}}$ in the ROM. (As above, these bounds assume that $q_{\mathsf{H}} \geq B q_{\mathsf{sig}}$ and omit constant terms; we additionally assume that $\beta \in (0, 1)$ is a constant.)

We provide a correctness analysis of FSwUA signatures that proceeds in a similar way.

2.4 Rejection-Free Signatures with Convolved Gaussians

In this final section, we focus on contributions from [DPS23], which provides a solution to remove rejection sampling from Lyubashevsky's signature without relying on noise flooding, leading to compact signatures.

We introduce a new paradigm for adapting Schnorr's identification protocol to the lattice setting. It relies on Gaussian convolution, rather than flooding or rejection sampling. Our $\mathsf{G}+\mathsf{G}$ (Gaussian Plus Gaussian) identification protocol can be compiled into a signature using the Fiat-Shamir heuristic (without aborts), in the Quantum Random Oracle Model (QROM). The resulting signature is asymptotically more compact than those based on rejection sampling. Finally, we provide concrete parameters which show that $\mathsf{G}+\mathsf{G}$ is competitive with the state-of-the-art optimizations of Lyubashevsky's signature.

$\mathsf{G}+\mathsf{G}$ involves two Gaussians that are being summed. The first one is $\mathbf{y}$ and the second one corresponds to $\mathbf{Sc}$. The first difficulty that we face is that $\mathbf{S}$ is fixed and $\mathbf{c}$ is publicly known as part of the resulting signature and hence cannot be assumed random for the sake of studying the distribution of $\mathbf{z}$.

To introduce the required new randomness, we start from BLISS [DDLL13], which uses bimodal rejection sampling with Gaussians. The verification key $\mathbf{A} \in \mathbb{Z}_{2q}^{m\times k}$ and the signing key $\mathbf{S} \in \mathbb{Z}^{k\times m}$ satisfy the relation $\mathbf{AS} = q\mathbf{I}_m \bmod 2q$. The commitment of the prover is $\mathbf{w} = \mathbf{Ay} \bmod 2q$, and upon receiving $\mathbf{c} \in \{0,1\}^m$, the prover replies with either $\mathbf{z} = \mathbf{y} + \mathbf{Sc}$ or $\mathbf{z} = \mathbf{y} - \mathbf{Sc}$ with probability $1/2$ each. The verifier checks that $\mathbf{z}$ is short and $\mathbf{Az} = \mathbf{w} + q\mathbf{c} \bmod 2q$. This check works for both values of $\mathbf{z}$ that the prover chose from. This can be explained by observing that the verification views $\mathbf{c}$ modulo 2, i.e., as a coset of $\mathbb{Z}^m/2\mathbb{Z}^m$, and negating it does not change the coset. This observation was used in [Duc14] to take negations of individual coordinates of $\mathbf{c}$ to minimize the Euclidean norm of $\mathbf{Sc}$ and hence decrease the standard deviation of $\mathbf{y}$ necessary to hide $\mathbf{Sc}$ via rejection sampling. We go further and let the prover extend the coset $\mathbf{c}$ sent by the verifier to a Gaussian sample with support $2\mathbb{Z}^m + \mathbf{c}$. The verification equation still holds, and we have our second Gaussian.

At this stage, the prover samples a Gaussian $\mathbf{y}$ over $\mathbb{Z}^k$, receives a uniform coset $\mathbf{c} \in \mathbb{Z}^m/2\mathbb{Z}^m$ from the verifier, produces a Gaussian sample $\mathbf{x}$ with support $2\mathbb{Z}^m+\mathbf{c}$ and computes $\mathbf{z} = \mathbf{y}+\mathbf{Sx}$. Equivalently, it samples $\mathbf{k}$ from a Gaussian with support $2\mathbf{S}\mathbb{Z}^m + \mathbf{Sc}$ and returns $\mathbf{z} = \mathbf{y} + \mathbf{k}$. In order to obtain the zero-knowledge property (i.e., be able to simulate signatures without knowing the signing key), we aim to prove that the distribution of the Gaussian convolution $\mathbf{z}$ can be sampled publicly. If $\mathbf{y}$ and $\mathbf{k}$ were continuous Gaussians, we would set their covariance matrices $\mathbf{\Sigma_y}$ and $\mathbf{\Sigma_k}$ such that $\mathbf{\Sigma_y} + \mathbf{\Sigma_k} = \mathbf{\Sigma_z}$ for a known covariance matrix $\mathbf{\Sigma_z}$ for $\mathbf{z}$. To fix the ideas, we could set $\mathbf{\Sigma_z} = \sigma^2\mathbf{I}$ for some $\sigma > 0$, i.e., the distribution of $\mathbf{z}$ is a spherical Gaussian, and set $\mathbf{\Sigma_y} = \sigma^2\mathbf{I} - \mathbf{\Sigma_k}$. If we sample $\mathbf{x}$ from a spherical Gaussian with standard deviation $s > 0$, then $\mathbf{\Sigma_k} = s^2\mathbf{S}\mathbf{S}^\top$ and $\mathbf{\Sigma_y} = \sigma^2\mathbf{I} - s^2\mathbf{S}\mathbf{S}^\top$ (by taking σ sufficiently large, the latter is indeed positive definite). This is our choice for $\mathsf{G}+\mathsf{G}$ though there is flexibility.

The above over-simplifies the situation as the Gaussians we manipulate are discrete rather than continuous. Further, their supports do not have the same dimensions. Indeed, the support of $\mathbf{y}$ is $\mathbb{Z}^k$ whereas the support of $\mathbf{k}$ is exactly $2\mathbf{S}\mathbb{Z}^m + \mathbf{Sc}$ whose span has dimension $m < k$: the second Gaussian lives in a smaller dimension and its support is sparser. This is illustrated in Figure 2.3. Convolution of discrete Gaussians has been studied in [BF11], but restricted to the case of full-dimensional supports. As a technical contribution, we extend their result to rank-deficient Gaussians with co-diagonalizable covariance matrices.

Thanks to the above, if the covariance matrices are set appropriately, then $\mathsf{G}+\mathsf{G}$ is honest-

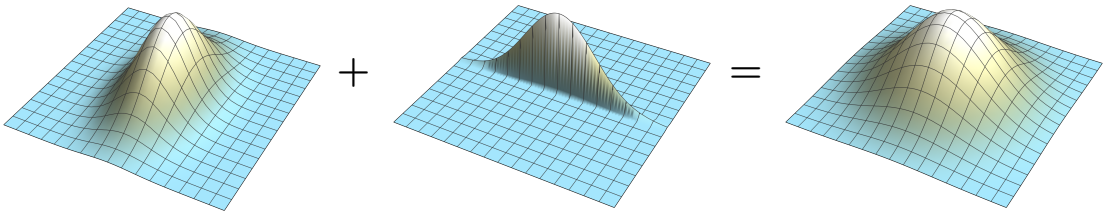

Figure 2.3: The sum of two Gaussians with compensating covariance matrices is a spherical Gaussian, even when the second Gaussian is rank-deficient. In the $\mathsf{G}+\mathsf{G}$ identification protocol and signature, the first Gaussian corresponds to $\mathbf{y}$, the second Gaussian is associated with $\mathbf{Sc}$ and the resulting one corresponds to $\mathbf{z}$.

verifier zero-knowledge (HVZK). The proofs of completeness and soundness are adapted from [DDLL13]. To apply the Fiat-Shamir heuristic, we also need the commitment $\mathbf{Ay}$ to have sufficiently high min-entropy. This is technically more complex than for Lyubashevsky's signatures as $\mathbf{y}$ is distributed from a skewed Gaussian. The properties satisfied by $\mathsf{G}+\mathsf{G}$ allow a conversion to a secure signature scheme, using completeness and commitment-recoverability to obtain the correctness of the signature, HVZK and commitment-min-entropy to reduce security against chosen-message attacks to security against no-message attacks, and computational soundness (resp. lossy-soundness) which implies security against no-message attacks for different parametrizations.

While all key generation techniques presented in [DDLL13] can be used with our $\mathsf{G}+\mathsf{G}$ protocol, we present alternative versions which offer more flexibility. A first improvement is that we can set $\mathbf{AS} = q\mathbf{J} \bmod 2q$, where $\mathbf{J} \in \mathbb{Z}_q^{m\times\ell}$ is only rectangular and full column-rank rather than set to the identity. When instantiating $\mathsf{G}+\mathsf{G}$ with the MLWE and MSIS hardness assumptions [BGV12; LS15] over a ring $R = \mathbb{Z}[X]/(X^N+1)$ with N a power of 2, we take $\mathbf{j} = (x^{N/2}+1, 0, \ldots, 0)$. This allows us to replace the lattice $2\mathbf{s}R$ with $(X^{N/2}-1)\mathbf{s}R$, and to decrease the standard deviation of $\mathbf{z}$ by a factor $\sqrt{2}$. Overall, we obtain signature sizes that are between 12% and 20% smaller than those in [DFPS22], or 30% to 40% smaller than Dilithium [BDK+21].

The resulting signature scheme is described in Figure 2.4, where $\zeta = 1 + X^{N/2}$ and $\zeta^* = 1 - X^{N/2}$ satisfy $\zeta^*\zeta = 2 \bmod X^N + 1$.

$\mathsf{KeyGen}(1^\lambda)$:

$(\mathbf{a}|\mathbf{A}_0) \hookleftarrow U(R_q^{m\times k-m})$
do $(\mathbf{s}_1, \mathbf{s}_2) \hookleftarrow \chi_\eta^{k-m-1} \times \chi_\eta^m$
  $\mathbf{b} \leftarrow \mathbf{a} + \mathbf{A}_0\mathbf{s}_1 + \mathbf{s}_2 \bmod q$
  $(\mathbf{b}_0, \mathbf{b}_1) \leftarrow \mathsf{Decomp}(\mathbf{b}, d)$
  $\mathbf{s} \leftarrow (1|\mathbf{s}_1^\top|\mathbf{s}_2^\top - \mathbf{b}_0^\top)^\top \in R_{2q}^k$
while $\sigma_1(\mathsf{rot}(\zeta\mathbf{s})) \geq S$
$\mathbf{A} \leftarrow (2(\mathbf{a} - \mathbf{b}_1) + q\mathbf{j}|2\mathbf{A}_0|2\mathbf{I}_m)$
return $(\mathsf{vk}, \mathsf{sk}) = (\mathbf{A}, (\mathbf{A}, \mathbf{s}))$

$\mathsf{Sign}((\mathbf{A}, \mathbf{s}), \mu)$:

$\mathbf{y} \hookleftarrow D_{R^k, \mathbf{\Sigma}(\mathbf{s})}$
$\mathbf{w} \leftarrow \mathbf{Ay} \bmod 2q$
$c \leftarrow H(\mathbf{w}, \mu)$
$u \hookleftarrow D_{R, s, -\zeta^*\cdot c/2}$
$\mathbf{z} \leftarrow \mathbf{y} + (\zeta u + c)\mathbf{s}$
return $(\mathbf{z}, c)$

$\mathsf{Verify}(\mathbf{A}, \mu, (\mathbf{z}, \mathbf{c}))$:

$\mathbf{w} \leftarrow \mathbf{Az} - q\mathbf{c}\mathbf{j} \bmod 2q$
if $\|\mathbf{z}\| \leq \gamma$ *and* $\mathbf{c} = \mathsf{H}(\mathbf{w}, \mu)$ **then**
  return 1
else
  return 0

Figure 2.4: The Module $\mathsf{G}+\mathsf{G}$ Signature Scheme.

Chapter 3

Threshold FHE

In this chapter, we discuss some recent contributions to threshold fully homomorphic encryption. The results presented in this chapter are sampled from the following works, and we refer the reader to their full versions for the technical details.

- **On Threshold Fully Homomorphic Encryption with Synchronized Decryptors**
  with François Colin de Verdière and Damien Stehlé
  Manuscript
- **Asynchronous Lagrange-Based Threshold FHE with Smaller Modulus Overhead**
  with Won Kim, Changmin Lee, Jeonghwan Lee, and Damien Stehlé
  CRYPTO 2026
- **Low Communication Threshold Fully Homomorphic Encryption**
  with Damien Stehlé
  ASIACRYPT 2024
- **Distributed Key Generation for Efficient Threshold-CKKS**
  with Seonhong Min, Guillaume Hanrot, Jai Hyun Park, and Damien Stehlé
  Manuscript

Contents

3.1 Background

Fully homomorphic encryption (FHE) has matured significantly over the past decade, with efficient and highly parallelizable schemes such as BGV [BGV12], BFV [Bra12; FV12], and CKKS [CKKS17]. It now enables scalable private computation over real/complex or discrete data. Yet, in many real-world scenarios, the decryption secret key should not reside on a single device. Threshold fully homomorphic encryption (ThFHE) addresses this by distributing the decryption capability across multiple parties. This mitigates single-point compromise and reduces the risk of a single point of failure.

In the t-out-of-n setting, any subset of t parties can jointly decrypt, while any subset of at most $t-1$ of them learns nothing about the encrypted data. In addition to enabling secure key management, threshold FHE also enables secure multiparty computation. Each party can encrypt their private data using ThFHE so that it is possible to compute over their joint encrypted data, while threshold decryption limits access to data by requiring t parties to agree before being able to decrypt a ciphertext.

ThFHE extends the standard FHE syntax by splitting the decryption algorithm into a 2-round distributed decryption protocol. Specifically, a ThFHE scheme is a tuple $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PartDec}, \mathsf{FinDec})$, which satisfies the following: KeyGen outputs a public key pk and n shares $(\mathsf{sk}_i)_{i\in[n]}$. Each party P_i for $i \in [n]$ is given a secret key share sk_i, which allows them to participate in the 2-round distributed decryption of a ciphertext as follows:

1. P_i produces a *partial decryption share* p_i by running $\mathsf{PartDec}$ using its share of the secret key sk_i;
2. Given t partial decryption shares from t distinct parties, one can recover the underlying plaintext by running FinDec.

First, we recall that RLWE-based FHE schemes share the following structure. They are defined over $R_q = R/qR$ where $R = \mathbb{Z}[x]/(x^N+1)$ for N a power-of-two integer and $q \geq 2$. A secret key is an element $\mathsf{sk} \in R_q$. For clarity of exposition, we focus on BFV-like FHE schemes but our results readily extend to other schemes. For BFV, a ciphertext encrypts a plaintext $\mu \in R_p = \mathbb{Z}_p[x]/(x^N+1)$ for some $p < q$, and satisfies $\mathsf{ct} = (a, b) \in R_q^2$ with:

$$b + a \cdot \mathsf{sk} = \left\lceil \frac{q}{p} \right\rfloor \cdot \mu + e \bmod q \ , \tag{3.1}$$

where $e \in R_q$ is a small error with $\|e\|_\infty \leq B_{\mathsf{eval}}$ for some fixed bound $B_{\mathsf{eval}} > 0$. Decrypting a ciphertext $\mathsf{ct} = (a, b)$ using sk is simply done by (i) computing $x \leftarrow a \cdot \mathsf{sk} + b \bmod q$ and (ii) returning $\lceil x \cdot \frac{p}{q} \rfloor \bmod p$. We emphasize that all FHE schemes start decryption by Step (i), which is linear, and only Step (ii) differs depending on the scheme (and it is usually not linear). Constructing threshold FHE is done by distributing Step (i), while Step (ii) is done locally once (a noisy version of) x is recovered. Then, the common approach to build a ThFHE scheme is to exploit the linearity of Step (i) of FHE decryption by relying on a linear secret sharing scheme (LSSS).

We now recall the notion of LSSS. A t-out-of-n LSSS allows one to share a secret, in our case the FHE secret key sk, among n parties by distributing a share sk_i to P_i so that any t distinct shares $(\mathsf{sk}_{i_j})_{j\in[t]}$ can be recombined to obtain sk via a linear equation

$$\mathsf{sk} = \sum_{j\in[t]} L_{i_j} \mathsf{sk}_{i_j} \bmod q, \tag{3.2}$$

where the reconstruction coefficients $(L_{i_j})_{j\in[t]}$ are publicly computable from the set of participants, and therefore depend on $\mathcal{S} = \{i_1, \ldots, i_t\}$ which is only known at combination time in the asynchronous setting.

Given an LSSS, one can construct a ThFHE by implementing $\mathsf{PartDec}$ as follows: P_i runs the (linear) step (i) of decryption using sk_i to produce a partial decryption share. For security, each party further needs to add noise during $\mathsf{PartDec}$ in order to break this linearity and avoid trivial key-recovery attacks. Hence, one defines $\mathsf{p}_i \leftarrow \mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct})$ for ciphertext $\mathsf{ct} = (a, b)$ as:

$$\mathsf{p}_i := a \cdot \mathsf{sk}_i + e_{\mathsf{fl},i} \bmod q \ ,$$

where $e_{\mathsf{fl},i}$ is an error $e_{\mathsf{fl},i} \leftarrow U(R)$ with $\|e_{\mathsf{fl},i}\|_\infty \leq B_{\mathsf{sm}}$ for some fixed bound $B_{\mathsf{sm}} > 0$. Then, FinDec simply relies on the linear reconstruction to combine t distinct partial decryption shares, as:

$$\begin{aligned}\mathsf{FinDec}((\mathsf{p}_{i_j})_{j\in[t]}, \mathsf{ct}) &:= b + \sum_{j\in[t]} L_{i_j} \cdot \mathsf{p}_{i_j} \bmod q \\ &= b + a \cdot \mathsf{sk} + \sum_{j\in[t]} L_{i_j} \cdot e_{\mathsf{fl},i_j} \bmod q \\ &= \left\lceil \frac{q}{p} \right\rfloor \cdot \mu + e + \sum_{j\in[t]} L_{i_j} \cdot e_{\mathsf{fl},i_j} \bmod q\end{aligned}$$

This is the same as Equation 3.1 up to an additional error $e_{\mathsf{fl}} = \sum_{j\in[t]} L_{i_j} \cdot e_{\mathsf{fl},i_j}$. For security, one needs e_{fl} to be large enough to hide e, and for correctness, one needs it to be small enough for Step (ii) to return μ.

The simplest case of LSSS is when $t = n$. In this case, also called additive secret sharing, one can simply split the secret key sk in n uniformly random shares $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$ conditioned on $\sum_{i=1}^n \mathsf{sk}_i = \mathsf{sk} \bmod q$. In this specific case, the reconstruction coefficients are simply 1 and correctness and security are easily achieved.

For the general t-out-of-n case with $t < n$, one needs to rely on more involved LSSS. As of today, there are mainly two categories of t-out-of-n LSSS (for $t < n$).

On the one hand, we have LSSS with binary reconstruction coefficients. These schemes help achieving correctness and security as the analysis is similar to the case $t = n$ mentioned above: the error e_{fl} is a sum of up to t errors $e_{\mathsf{fl},i}$'s, whose magnitude is bounded by B_{sm}. Yet, no scheme achieving this property enables a practical ThFHE, as LSSS with binary reconstruction coefficients produce gigantic shares $(\mathsf{sk}_i)_{i\in[n]}$. For example, counting the FHE secret key as 1 element, the most compact LSSS with binary reconstruction coefficients produces shares containing $O(n^{4.3})$ elements. This makes them impractical already for a moderate number of parties (e.g., $n = 100$).

On the other hand, Shamir secret sharing offers compact shares, whose size is only 1 element (i.e., matches the size of the secret key being shared).[1] Shamir secret sharing relies on Lagrange interpolation, and its reconstruction coefficients are the so-called Lagrange coefficients. Unfortunately, these Lagrange coefficients can be as large as q, which makes correctness and security much harder to guarantee. Threshold FHE based on Shamir secret sharing is the focus of this chapter.

[1] ThFHE based on LSSS with binary reconstruction and Shamir secret sharing actually induce different ciphertext-modulus constraints. Hence, our reference point (secret key size) for comparison is slightly biased. Nevertheless, it is a reasonable baseline for comparing share sizes.

Before diving into more details, let us recall Shamir secret sharing. We aim to share the secret key, which is defined over R_q. Shamir secret sharing over R_q is parameterized by interpolation points $\mathcal{I} = \{\alpha_0, \ldots, \alpha_n\} \subset R_q$ with each $(\alpha_i - \alpha_j)$ invertible in R_q for all $0 \leq i \neq j \leq n$. Sharing a secret $\mathsf{sk} \in R_q$ is done by sampling a uniform polynomial $f \in R_q[Y]$ of degree less than t conditioned on $f(\alpha_0) = \mathsf{sk}$; the share of the i-th party is $\mathsf{sk}_i = f(\alpha_i)$. Given t distinct shares (i.e., evaluations of f at t distinct points), one can recover $f(\beta)$ for any $\beta \in R_q$ via Lagrange interpolation, which is linear. For $i \in \mathcal{S}$ and $\beta \in R_q$, we define the Lagrange coefficients:

$$\lambda_i^{\mathcal{S}\to\beta} \coloneqq \prod_{j\in\mathcal{S}\setminus\{i\}} \frac{\beta - \alpha_j}{\alpha_i - \alpha_j} \in R_q \ .$$

Then, the above coefficients $(L_i)_{i\in\mathcal{S}}$ are the Lagrange coefficients $(\lambda_i^{\mathcal{S}\to\alpha_0})_{i\in\mathcal{S}}$, and we have $\mathsf{sk} = f(\alpha_0) = \sum_{i\in\mathcal{S}} \lambda_i^{\mathcal{S}\to\alpha_0} \cdot f(\alpha_i)$. In what follows, we always set $\alpha_0 = 0$. This is the most common choice and it is actually the best choice for performance as well.

Using Shamir secret sharing, FinDec defined as above then returns

$$\left\lceil \frac{q}{p} \right\rfloor \cdot \mu + e + \sum_{i\in\mathcal{S}} \lambda_i^{\mathcal{S}\to 0} \cdot e_{\mathsf{fl},i} \bmod q \ ,$$

so we have $e_{\mathsf{fl}} = \sum_{i\in\mathcal{S}} \lambda_i^{\mathcal{S}\to 0} \cdot e_{\mathsf{fl},i}$. It is possible to recover μ via rounding if $\|e + e_{\mathsf{fl}}\|_\infty < q/2p$. This raises a challenge: we need $e_{\mathsf{fl}} = \sum_{i\in\mathcal{S}} \lambda_i^{\mathcal{S}\to 0} \cdot e_{\mathsf{fl},i}$ to be small compared to $q/2p$, but Lagrange coefficients can have magnitude as large as q due to their denominators. Two solutions have been proposed to address this difficulty:

1. Assume that decryptors are synchronous, meaning that a party running $\mathsf{PartDec}$ knows which set of t decryptors are participating in the distributed decryption. This is what we refer to as the synchronous setting, which we discuss in Section 3.2.
2. Enforce e_{fl} to be small by tweaking the scheme. Prior works handle this by scaling $e_{\mathsf{fl},i}$ by a denominator-clearing factor C_{sim} in $\mathsf{PartDec}$ so that $C_{\mathsf{sim}} \cdot e_{\mathsf{fl}}$ is small compared to $q/2p$, over the integers. While this addresses the correctness issue, this raises a new challenge regarding security. We discuss the asynchronous setting in Section 3.3.

What is in this chapter. In this chapter, we primarily focus on contributions sampled from [CPS26; KLL+26], which aim to construct efficient threshold FHE based on Shamir secret sharing. We first focus on constructions in the synchronous setting in Section 3.2, and then discuss the more technical asynchronous setting in Section 3.3. In addition, we also briefly explain some contributions from [PS24] on reducing communication in threshold FHE in Section 3.4.1, and about distributed key generation for threshold CKKS [MHP+25] in Section 3.4.2, which are also related to threshold FHE but focus on different challenges.

The objective of this chapter is to provide an overview of how to instantiate threshold FHE in practice, depending on the setting.

What is not in this chapter. We intentionally omit a significant part of the contributions from [PS24; CPS26], and in particular the cryptanalytic contributions therein. In short, besides providing secure constructions of threshold FHE with low communication (in [PS24]) or in the synchronous setting (in [CPS26]), our works also provide attacks against prior schemes that aimed to solve the same issues. These contributions, which are orthogonal to our constructions, can be found in the full version of these works. In [CPS26], we also discuss an extension of our construction to obtain context-dependent threshold FHE, generalizing the notion of context-dependent threshold public key encryption originally defined in [BBN+25].

3.2 ThFHE in the Synchronous Setting

So far, we have implicitly discussed the *asynchronous* setting, in which any set of t partial decryptions should allow decryption without knowing in advance which t decryptors will contribute. This is in contrast with the *synchronous* setting, in which the set of t decryptors is known at the outset of the decryption protocol. We study this setting in [CPS26], and we now explain briefly some of the contributions from this work.

Addressing the error growth in the synchronous setting. A brief remark about the potential advantage of considering synchronous decryption is made in [BGG+18, Footnote 1], though without an explicit scheme description. The synchronous setting *seems* to provide a direct solution to the issue of errors growing due to the reconstruction coefficients. The idea is to change $\mathsf{PartDec}$ such that it returns $\lambda_i^{\mathcal{S}\to 0} \cdot a \cdot \mathsf{sk}_i + e_{\mathsf{fl},i}$ instead of $a \cdot \mathsf{sk}_i + e_{\mathsf{fl},i}$. Note that $\lambda_i^{\mathcal{S}\to 0}$, which depends on $\mathcal{S}$, can be computed at this stage, since decryptors are synchronous. By incorporating the reconstruction coefficient directly in the partial decryption, one prevents the decryption error from mingling with these coefficients, as combining partial decryption shares can now be achieved by adding them, leading to an added error $e_{\mathsf{fl}} = \sum_{1\le j\le t} e_{\mathsf{fl},i_j}$. As we will see in Section 3.3, this provides significant efficiency gains compared to constructions in the asynchronous setting. This led some practical works to focus on this setting, notably [MTBH21; MBH23].

While these works propose concrete constructions and protocols in the synchronous setting, they do not explicitly propose formal (functionality and security) definitions nor provide detailed security analyses in this setting. Yet, the fact that the set of decryptors is known at the time of decryption requires a modification in the definition of ThFHE: the (partial) decryption algorithm should take as an additional input the set of decryptors $\mathcal{S}$ for which it is crafting the partial decryption. Furthermore, this syntactic change to the partial decryption algorithm should be reflected in the security model. For instance, an attacker should be allowed to make decryption queries for different sets of decryptors for the same party. This is essential for practical deployment, as a target set of decryptors might fail to decrypt, e.g., if one of the decryptors is disconnected from the network before revealing its share. This is referred to as *churn* in [MTBH21]. In such a scenario, decryptors will attempt to decrypt with a different set of decryptors by replacing the now offline party and will reveal two partial decryptions of the same ciphertext for different target sets.

3.2.1 Contributions in a Nutshell

We provide a formal treatment of ThFHE in the synchronous setting, showing that prior constructions are insecure and providing a secure and efficient solution. Our contributions are threefold. (1) First, we fill the definitional gap by providing a specific syntax for ThFHE in the synchronous setting, as well as two security models: a weak model in which designating the set of decryptors is only viewed as an enabler of efficient decryption and it is fine for non-designated parties to be able to decrypt as long as there are sufficiently many of them; and a much stronger model where a set of decryptors which does not exactly match the designated set should not learn anything. (2) Next, we describe efficient key-recovery attacks against the schemes proposed in [MBH23; MTBH21] in the weak security model. In particular, the simple scheme described above *is not secure*. It also implies that the scheme proposed in [MTBH21] does not provide security against churn as claimed. (3) Finally, we provide a

simple and efficient construction which we prove secure in the strong security model. This construction achieves performance essentially matching the prior (insecure) constructions and relies on combining an FHE scheme with Shamir secret sharing and pseudorandom functions.

3.2.2 Our Approach to Synchronous ThFHE

We now provide a brief overview of our contributions, starting with syntax and security models for ThFHE in the synchronous setting.

ThFHE in the synchronous setting. We extend the syntax of ThFHE to the synchronous setting by adding the set of target decryptors as an additional input to the partial decryption algorithm. Hence, we define a *target-decryptors ThFHE* as a tuple of PPT algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PartDec}, \mathsf{FinDec})$, with the following syntax. We focus on KeyGen, $\mathsf{PartDec}$, and FinDec, as Enc and Eval have the same syntax as for classical FHE.

- $\mathsf{KeyGen}(1^\lambda, 1^n, 1^t)$ generates the public key pk together with partial decryption keys $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$;
- $\mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}, \mathcal{S})$ takes as input a partial decryption key sk_i, a ciphertext ct, and *a target set of decryptors* $\mathcal{S} \subseteq [n]$ *of size* t, and returns a partial decryption share $\mathsf{p}_i^{\mathcal{S}}$;
- $\mathsf{FinDec}((\mathsf{p}_i^{\mathcal{S}}, \mathsf{ct})_{i \in \mathcal{S}})$ takes as input t partial decryption shares and a ciphertext ct, and returns a plaintext μ.

For correctness, we ask that decryption succeeds when FinDec is fed with the t partial decryption shares $(\mathsf{p}_{i_1}^{\mathcal{S}}, \ldots, \mathsf{p}_{i_t}^{\mathcal{S}})$ corresponding to decryptors $(i_1, \ldots, i_t)$ in the same set $\mathcal{S}$.

We note that a definition for ThFHE in the synchronous setting is proposed in [GKS24], following the same syntax as above, but only providing a very weak security notion. Our main definitional contribution is to propose two models of security for this syntax: a first, weaker notion that existing constructions already fail to satisfy, and a much stronger one, for which we propose a secure construction.

Our weak security notion, termed $\mathsf{wk\text{-}IND\text{-}CPA}$-security, follows an indistinguishability-based game for ThFHE. The adversary can corrupt up to $t-1$ parties (i.e., learn up to $t-1$ partial decryption keys) and request challenge encryptions from an oracle by providing a pair of plaintexts (μ_0, μ_1), to which the challenger replies by encrypting μ_b for a fixed unknown challenge bit $b \in \{0, 1\}$. The adversary can also make evaluation queries with functions of its choice on previously obtained ciphertexts. In addition, it is given access to a partial decryption oracle, which it can query on input of a previously obtained ciphertext ct, an index $i \in [n]$, and a target set of decryptors $\mathcal{S}$. The challenger returns $\mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}, \mathcal{S})$ only if the plaintext underlying ct does not depend on the challenge bit b; otherwise the challenger returns $\perp$. This model is weak for the synchronous setting as it provides no confidentiality guarantee about the plaintext underlying a ciphertext ct as soon as the attacker knows t partial decryptions of ct from distinct parties. In particular, there is no confidentiality guarantee even if the t partial decryptions are crafted for pairwise disjoint sets of decryptors. Yet, in the synchronous setting, it is entirely possible that a partial decryption for a target set of decryptors $\mathcal{S}$ is revealed and that it is still not possible to complete decryption, e.g., if the other decryptors in $\mathcal{S}$ are not corrupted and never reveal their shares. We prove in our work that prior constructions from [MBH23; MTBH21] are not $\mathsf{wk\text{-}IND\text{-}CPA}$-secure, and provide a simple and efficient key-recovery attack (details in the full version).

Let us now focus on our strong security notion. In this model, we require that decryption shares for the same ciphertext but different decryptor sets of size $< t$ should not reveal

information about the underlying plaintext. We provide a simulation-based definition of the strong security model. It is also possible to give an indistinguishability-based definition, but as we focus on proving security, we present the strongest version. Our security notion is analogous to that proposed in [EY24] in the asynchronous setting, with ours being more permissive regarding adaptive queries. Note that the security model from [EY24] is stronger than the original one from [BGG+18].

To start, let us note that according to our definition of target-decryptors ThFHE, a decryption share of a ciphertext ct is associated with two attributes: the index $i \in [n]$ of the decryption key share sk_i used to generate it, as well as a set of decryptors $\mathcal{S} \subseteq [n]$ satisfying $i \in \mathcal{S}$ and $|\mathcal{S}| = t$ for which it is crafted. To simplify the description of our security model, we introduce an auxiliary algorithm $\mathsf{CanCombine}$, described in Figure 3.1, which tells whether a $\mathsf{PartDec}$ query $(i, \mathcal{S})$ on a ciphertext ct and for a target set $\mathcal{S}$ allows the adversary to recover the underlying plaintext based on the information it already has (corrupted keys and prior partial decryption shares). Specifically, $\mathsf{CanCombine}$ takes as inputs the set (of indices) $\mathcal{S}_{\mathsf{cor}} \subseteq [n]$ of corrupted secret keys owned by the adversary, the set $\mathsf{D}[\mathsf{ct}]$ containing all prior partial decryption queries $(i_k, \mathcal{S}_k)$ made on ct by the adversary, and the new query $(i, \mathcal{S})$. Then $\mathsf{CanCombine}$ returns 1 if ct can be decrypted given the answer to query $(i, \mathcal{S})$ by combining it with other decryption shares, i.e., $\mathsf{CanCombine}$ returns 1 if $i \notin \mathcal{S}_{\mathsf{cor}}$ and for all $k \in \mathcal{S} \setminus \{i\}$, at least one of the following conditions holds:

- $k \in \mathcal{S}_{\mathsf{cor}}$, i.e., the adversary knows sk_k;
- $(k, \mathcal{S}) \in \mathsf{D}[\mathsf{ct}]$, i.e., the adversary knows a partial decryption of ct for party P_k and set $\mathcal{S}$.

By convention, we make $\mathsf{CanCombine}$ return 0 when $i \in \mathcal{S}_{\mathsf{cor}}$, as partial decryptions can always be computed by the attacker in this case and therefore the adversary does not learn anything new from this query.

At a high level, our security notion states the following. Consider an adversary that may corrupt up to $t-1$ parties. For a ciphertext ct encrypting a certain plaintext μ, we have two cases:

(1) The adversary has enough information to decrypt ct: in this case, the adversary should learn the underlying plaintext μ but nothing more;
(2) The adversary is not able to decrypt ct: in this case, the adversary should learn nothing.

We formalize this by relying on two simulators that can simulate partial decryption queries based on the view of the attacker and (1) the underlying plaintext, or (2) only the public parameters.

To further illustrate the strength of our definition, note that an adversary corrupting a set $\mathcal{S}_{\mathsf{cor}}$ of $t-1$ parties still learns nothing about the plaintext underlying a ciphertext ct even if it obtains a partial decryption share $\mathsf{p}_i^{\mathcal{S}}$ of ct from a non-corrupted party P_i with $i \notin \mathcal{S}_{\mathsf{cor}}$, as long as $\mathcal{S} \neq \mathcal{S}_{\mathsf{cor}} \cup \{i\}$.

Definition 3.2.1 (Strong td-ThFHE Security)**.** *A target-decryptors ThFHE scheme* $\Pi_{\mathsf{ThFHE}} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PartDec}, \mathsf{FinDec})$ *is said to be* $\mathsf{stg\text{-}SIM}$*-secure if it satisfies the following two properties:*

- *the sub-scheme* $(\mathsf{KeyGen}, \mathsf{Enc})$ *is* $\mathsf{IND\text{-}CPA}$*-secure;*
- *there exists a PPT simulator* $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1, \mathsf{Sim}_2)$ *such that the experiments depicted in Figure 3.1 are computationally indistinguishable; specifically, we require that for any*

PPT adversary $\mathscr{A} = (\mathscr{A}_0, \mathscr{A}_1)$*, we have* $\delta \leq \epsilon(\lambda)$ *where*

$$\delta = \left|\Pr[\mathscr{A}(\mathsf{Exp}^{\mathsf{Sim}}_{\mathsf{Real}}(1^\lambda, 1^n, 1^t)) = 1] - \Pr[\mathscr{A}(\mathsf{Exp}^{\mathsf{Sim}}_{\mathsf{Ideal}}(1^\lambda, 1^n, 1^t)) = 1]\right|.$$

$\mathsf{Exp}^{\mathsf{Sim}}_{\mathsf{Real}}(1^\lambda, 1^n, 1^t)$:

$((\mathsf{sk}_i)_{i\in[n]}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^n, 1^t)$
$\mathsf{ctr} \leftarrow 0,\ \mathsf{L} \leftarrow \varnothing,\ \mathsf{D} \leftarrow \varnothing$
$(\mathcal{S}_{\mathsf{cor}}, \mathsf{st}_{\mathscr{A}}) \leftarrow \mathscr{A}_0(\mathsf{pk})$ with $|\mathcal{S}_{\mathsf{cor}}| < t$
$\mathsf{out} \leftarrow \mathscr{A}_1^{\mathsf{OEnc},\mathsf{OEval},\mathsf{ODec}}((\mathsf{sk}_i)_{i\in\mathcal{S}_{\mathsf{cor}}}, \mathsf{pk}, \mathsf{st}_{\mathscr{A}})$
return $(\mathsf{out} = \mathsf{real})$

$\mathsf{Exp}^{\mathsf{Sim}}_{\mathsf{Ideal}}(1^\lambda, 1^n, 1^t)$:

$((\mathsf{sk}_i)_{i\in[n]}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^n, 1^t)$
$\mathsf{ctr} \leftarrow 0,\ \mathsf{L} \leftarrow \varnothing,\ \mathsf{D} \leftarrow \varnothing$
$(\mathcal{S}_{\mathsf{cor}}, \mathsf{st}_{\mathscr{A}}) \leftarrow \mathscr{A}_0(\mathsf{pk})$ with $|\mathcal{S}_{\mathsf{cor}}| < t$
$(\mathsf{st}, (\mathsf{sk}'_i)_{i\in\mathcal{S}_{\mathsf{cor}}}) \leftarrow \mathsf{Sim}_0(\mathsf{pk}, \mathcal{S}_{\mathsf{cor}})$
$\mathsf{out} \leftarrow \mathscr{A}_1^{\mathsf{OEnc},\mathsf{OEval},\mathsf{OSim}}((\mathsf{sk}'_i)_{i\in\mathcal{S}_{\mathsf{cor}}}, \mathsf{pk}, \mathsf{st}_{\mathscr{A}})$
return $(\mathsf{out} = \mathsf{ideal})$

$\mathsf{OEnc}(\mu)$:

$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mu)$
$\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
if $\mathsf{D}[\mathsf{ct}]$ *does not exist* **then**
  $\mathsf{D}[\mathsf{ct}] \leftarrow \varnothing$
$\mathsf{L}[\mathsf{ctr}] \leftarrow (\mu, \mathsf{ct})$
return ct

$\mathsf{ODec}(j, i, \mathcal{S})$:

if $j > \mathsf{ctr} \vee |\mathcal{S}| \neq t \vee i \notin \mathcal{S}$ **then**
  return $\perp$
$(\mu, \mathsf{ct}) \leftarrow \mathsf{L}[j]$
$\mathsf{p}_i^{\mathcal{S}} \leftarrow \mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}, \mathcal{S})$
return $\mathsf{p}_i^{\mathcal{S}}$

$\mathsf{OEval}(f, (j_1, \ldots, j_\ell))$:

if $\exists k \in [\ell] : j_k > \mathsf{ctr}$ **then**
  return $\perp$
for $k \in [\ell]$ **do**
  $(\mu_k, \mathsf{ct}_k) \leftarrow \mathsf{L}[j_k]$
$\mathsf{ct} \leftarrow \mathsf{Eval}(\mathsf{pk}, f, (\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell))$
$\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
$\mu \leftarrow f(\mu_1, \ldots, \mu_\ell)$
if $\mathsf{D}[\mathsf{ct}]$ *does not exist* **then**
  $\mathsf{D}[\mathsf{ct}] \leftarrow \varnothing$
$\mathsf{L}[\mathsf{ctr}] \leftarrow (\mu, \mathsf{ct})$
return ct

$\mathsf{OSim}(j, i, \mathcal{S})$:

if $j > \mathsf{ctr} \vee |\mathcal{S}| \neq t \vee i \notin \mathcal{S}$ **then**
  return $\perp$
$(\mu, \mathsf{ct}) \leftarrow \mathsf{L}[j]$
if $\mathsf{CanCombine}(\mathcal{S}_{\mathsf{cor}}, \mathsf{D}[\mathsf{ct}], i, \mathcal{S})$ **then**
  $(\mathsf{st}, \mathsf{p}_i^{\mathcal{S}}) \leftarrow \mathsf{Sim}_2(\mathsf{st}, \mathsf{ct}, i, \mathcal{S}, \mu)$
else
  $(\mathsf{st}, \mathsf{p}_i^{\mathcal{S}}) \leftarrow \mathsf{Sim}_1(\mathsf{st}, \mathsf{ct}, i, \mathcal{S})$
$\mathsf{D}[\mathsf{ct}] \leftarrow \mathsf{D}[\mathsf{ct}] \cup \{(i, \mathcal{S})\}$
return $\mathsf{p}_i^{\mathcal{S}}$

$\mathsf{CanCombine}(\mathcal{S}_{\mathsf{cor}}, \mathsf{D}[\mathsf{ct}], i, \mathcal{S})$:

if $i \in \mathcal{S}_{\mathsf{cor}}$ **then**
  return 0
else
  if $\forall k \in \mathcal{S} \setminus \{i\}, k \in \mathcal{S}_{\mathsf{cor}} \vee (k, \mathcal{S}) \in \mathsf{D}[\mathsf{ct}]$ **then**
    return 1
  else
    return 0

Figure 3.1: stg-SIM security games for target-decryptors ThFHE.

A stg-SIM-secure construction. We provide a simple stg-SIM-secure construction. It is obtained by tweaking the above basic construction that combines an FHE scheme $\mathsf{FHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ with Shamir secret sharing (simply denoted Share).

The tweak is inspired by the masking technique used in [PKM+24] in the context of lattice-based threshold signatures and based on pseudorandom functions (PRF). We adapt the PRF masking to our setting as follows: during KeyGen, one further samples n^2 PRF keys $k_{i,j}$ for $i, j \in [n]$, and each party P_i receives $2n$ PRF keys $(k_{i,j}, k_{j,i})_{j\in[n]}$, in addition to its Shamir share sk_i of the FHE secret key sk. When a party P_i computes a partial decryption $\mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}, \mathcal{S})$, it computes a pseudorandom mask $r_i^{\mathcal{S},\mathsf{ct}} := \sum_{j\in\mathcal{S}}(F_{k_{i,j}}(\mathcal{S}, \mathsf{ct}) - F_{k_{j,i}}(\mathcal{S}, \mathsf{ct}))$ and uses it to hide its basic partial decryption share. Concretely, it reveals:

$$\mathsf{p}_i^{\mathcal{S}} = \lambda_i^{\mathcal{S}\to 0} \cdot a \cdot \mathsf{sk}_i + e_{\mathsf{fl},i} + r_i^{\mathcal{S},\mathsf{ct}} \ .$$

Our construction is described in details in Figure 3.2, where we set $B_{\mathsf{sm}} = \Omega(2^\lambda \cdot B_{\mathsf{eval}})$. We assume that FHE is compatible with $(t, U([-B_{\mathsf{sm}}, B_{\mathsf{sm}}]))$, in the sense that for a ciphertext $\mathsf{ct} = (a, b)$ as above, the ciphertext $(a, b + e_{\mathsf{fl}})$ still decrypts to the same plaintext μ with overwhelming probability over $e_{\mathsf{fl}} = \sum_{1\le i\le t} e_{\mathsf{fl},i}$ where $e_{\mathsf{fl},i} \leftarrow U([-B_{\mathsf{sm}}, B_{\mathsf{sm}}])$ for all i.

$\mathsf{KeyGen}(1^\lambda, 1^n, 1^t)$:

$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda)$
for $i, j \in [n]$ **do**
  $k_{i,j} \leftarrow U(\mathcal{K})$
$(\mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{Share}(n, t, \mathsf{sk})$
for $1 \le i \le n$ **do**
  $\mathsf{sk}_i \leftarrow (i, \mathsf{sk}_i, (k_{i,j})_{j\in[n]}, (k_{j,i})_{j\in[n]})$
return $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n, \mathsf{pk})$

$\mathsf{Enc}(\mathsf{pk}, \mu)$:

return $\mathsf{FHE.Enc}(\mathsf{pk}, \mu)$

$\mathsf{Eval}(\mathsf{pk}, f, (\mathsf{ct}_j)_{j\in[\ell]})$:

return $\mathsf{FHE.Eval}(\mathsf{pk}, f, (\mathsf{ct}_j)_{j\in[\ell]})$

$\mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct}, \mathcal{S})$:

Parse ct as (a, b)
Parse sk_i as $(i, \mathsf{sk}_i, (k_{i,j})_{j\in[n]}, (k_{j,i})_{j\in[n]})$
if $i \notin \mathcal{S}$ **then**
  return $\perp$
$e_{\mathsf{fl},i} \leftarrow U([-B_{\mathsf{sm}}, B_{\mathsf{sm}}])$
for $j \in \mathcal{S}$ **do**
  $r_{i,j}^{\mathcal{S},\mathsf{ct}} \leftarrow F_{k_{i,j}}(\mathcal{S}, \mathsf{ct})$
  $r_{j,i}^{\mathcal{S},\mathsf{ct}} \leftarrow F_{k_{j,i}}(\mathcal{S}, \mathsf{ct})$
$r_i^{\mathcal{S},\mathsf{ct}} \leftarrow \sum_{j\in\mathcal{S}}(r_{i,j}^{\mathcal{S},\mathsf{ct}} - r_{j,i}^{\mathcal{S},\mathsf{ct}})$
$\mathsf{p}_i^{\mathcal{S}} \leftarrow \lambda_i^{\mathcal{S}\to 0} \cdot a \cdot \mathsf{sk}_i + e_{\mathsf{fl},i} + r_i^{\mathcal{S},\mathsf{ct}}$
return $\mathsf{p}_i^{\mathcal{S}}$

$\mathsf{FinDec}(\mathsf{ct}, (\mathsf{p}_i^{\mathcal{S}})_{i\in\mathcal{S}})$:

Parse ct as (a, b)
return $b + \sum_{i\in\mathcal{S}} \mathsf{p}_i^{\mathcal{S}}$

Figure 3.2: Our td-ThFHE scheme.

We prove the following theorem:

Theorem 3.2.2. *Assume that F is a secure PRF and that* FHE *is correct, compact and* IND-CPA*-secure. Assume further that $(t, U([-B_{\mathsf{sm}}, B_{\mathsf{sm}}]))$ is compatible with* FHE*. Then* td-ThFHE *is correct, compact and* stg-SIM*-secure.*

The key observations that make our scheme work are the following. First, for any subset $\mathcal{S}$ and any ciphertext ct, we have:

$$\sum_{i\in\mathcal{S}} r_i^{\mathcal{S},\mathsf{ct}} = \sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{S}}(F_{k_{i,j}}(\mathcal{S}, \mathsf{ct}) - F_{k_{j,i}}(\mathcal{S}, \mathsf{ct})) = 0 \ .$$

Then, $(t, U([-B_{\mathsf{sm}}, B_{\mathsf{sm}}]))$-compatibility implies correctness, since pseudorandom masks cancel each other out when one combines t partial decryption shares for the same ciphertext ct and the same decryption set $\mathcal{S}$. Compactness is also directly inherited from that of the underlying FHE scheme, since we do not alter encryption nor evaluation.

Second, consider an attacker that corrupts a set $\mathcal{S}_{\mathsf{cor}}$ of (up to) $t-1$ parties and learns a partial decryption share of a ciphertext ct from an uncorrupted party P_i for a target set of decryptors $\mathcal{S}$. For simplicity, assume the adversary does not yet know any other partial decryption. Then one of the following holds.

- **Case 1:** $\mathcal{S} = \mathcal{S}_{\mathsf{cor}} \cup \{i\}$, i.e., the attacker can now decrypt ct and recover the underlying plaintext μ. Then, by the observation above, we have:

$$r_i^{\mathcal{S},\mathsf{ct}} = -\sum_{j \in \mathcal{S}_{\mathsf{cor}}} \sum_{\ell \in \mathcal{S}} r_j^{\mathcal{S},\mathsf{ct}} \ .$$

  The right-hand side is computable by the attacker. One can therefore simulate the partial decryption share $\mathsf{p}_i^{\mathcal{S}}$ using the standard simulation technique based on noise smudging for ThFHE as if there were no masks (and just add the mask to correct the simulation).
- **Case 2:** $\mathcal{S} \neq \mathcal{S}_{\mathsf{cor}} \cup \{i\}$, i.e., there exists $\ell \in \mathcal{S} \setminus (\mathcal{S}_{\mathsf{cor}} \cup \{i\})$, and the attacker has no information about $k_{i,\ell}$ nor about $k_{\ell,i}$. In this situation, the mask $r_i^{\mathcal{S},\mathsf{ct}}$, which contains a term $F_{k_{i,\ell}}(\mathcal{S}, \mathsf{ct}) - F_{k_{\ell,i}}(\mathcal{S}, \mathsf{ct})$, is pseudorandom to the adversary, and $\mathsf{p}_i^{\mathcal{S}}$ can be simulated as a uniform value.

By extending this analysis, we prove that the above construction is secure.

Overhead. Overall, our construction introduces a small overhead per party compared to the prior (insecure) constructions: each party additionally stores $2(n-1)$ PRF keys and performs $2(n-1)$ PRF evaluations during $\mathsf{PartDec}$. Relying on AES yields a $256 \cdot (n-1)$-bit overhead for the secret key. Note that typical FHE parameters set $N = 2^{16}$ and $\log q \geq 2^7$ (to support noise flooding), so P_i's share sk_i of the FHE secret key sk still dominates sk_i up to $n = 2^{16} = 65536$ parties. This ignores the much larger public and evaluation keys. Moreover, generating the additional masks during $\mathsf{PartDec}$ takes about $2n$ milliseconds on single-thread CPU, based on conservative estimates for AES throughput (1MB/ms).

3.3 ThFHE in the Asynchronous Setting

We study the asynchronous setting in [KLL+26]. This section highlights some contributions from this work.

The main difficulty in the asynchronous setting, as aforementioned, is that Lagrange coefficients amplify the errors $e_{\mathsf{fl},i}$ added during partial decryption. Prior works have handled this issue by scaling $e_{\mathsf{fl},i}$ by a denominator-clearing factor C_{sim} in $\mathsf{PartDec}$ so that $C_{\mathsf{sim}} \cdot \lambda_i^{\mathcal{S}\to 0} \cdot e_{\mathsf{fl},i}$ is small compared to $q/2p$, over the integers, for any set $\mathcal{S}$ of t decryptors. Doing so, FinDec now returns $\left\lceil \frac{q}{p} \right\rfloor \cdot \mu + e + \sum_{i \in \mathcal{S}} \lambda_i^{\mathcal{S}\to 0} \cdot C_{\mathsf{sim}} \cdot e_{\mathsf{fl},i} \bmod q$, where $e_{\mathsf{fl}} = \sum_{i \in \mathcal{S}} \lambda_i^{\mathcal{S}\to 0} \cdot e_{\mathsf{fl},i}$ is now small over the integers. Hence, one can set q such that e_{fl} is small compared to $q/2p$. This requires increasing the modulus q by some overhead, whose sole purpose is to accommodate the error growth resulting from Lagrange reconstruction. As seen in Section 3.2, this overhead is not required in the synchronous setting, leading to more efficient constructions.

While the above technique addresses the correctness challenge, it actually results in a second challenge: for security, we require that $C_{\mathsf{sim}} \cdot e_{\mathsf{fl}}$ hides e, but e lives in R while $C_{\mathsf{sim}} \cdot e_{\mathsf{fl}}$ may range over a proper ideal I of R,[2] and therefore $e \bmod I$ may leak. Prior works solve this problem by assuming the underlying FHE scheme is *special*, i.e. that e lives in an ideal (C_{dec}) for some $C_{\mathsf{dec}} \in R$, with $(C_{\mathsf{dec}}) \subseteq I$. They enforce specialness by tweaking the underlying FHE scheme (using GSW in [BGG+18; CKL24] or BGV in [BFM+25]). Adapting their tweak to CKKS does not seem straightforward due to Rescale operations during CKKS evaluation which introduce rounding errors that break specialness.

Actually, this second difficulty was overlooked in [BGG+18]: the authors set $C_{\mathsf{dec}} = C_{\mathsf{sim}}$, which results in $I \subsetneq (C_{\mathsf{sim}})$ for their choice of Lagrange interpolation points. This leads to an attack, described in [CKL24], but the latter work does not provide a corrected analysis. Note that this is correctly addressed in [BFM+25], but the authors analyze the reconstruction coefficients over the fraction field $K = \mathbb{Q}[x]/(x^N + 1)$ though the construction uses reconstruction coefficients in $R_q = \mathbb{Z}_q[x]/(x^N + 1)$, resulting in a mismatch between the construction and its analysis.[3]

To solve all the aforementioned pitfalls at once, we give a detailed construction which does not require the underlying scheme to be special, and provide a thorough analysis.

3.3.1 Contributions in a Nutshell

We completely revisit the construction of ThFHE based on Shamir secret sharing in the asynchronous setting. First, we revisit the classical construction, remove the need for the underlying FHE scheme to satisfy specialness [BGG+18], and fill minor gaps in prior analyses. Our security and correctness analysis translate into mathematical constraints that relate the scheme parameters (notably, the modulus q which drives the size of keys and ciphertexts) to the L_1-norm of the Lagrange coefficients. The latter depends on the interpolation points picked for Lagrange interpolation.

Then, we compare various choices of interpolation points, over the integers and over monomials in the scheme's underlying cyclotomic ring, in light of these constraints. Our main contributions are tight analytic bounds for meeting these constraints (and therefore correctness and security claims). Our bounds closely match those obtained by heuristics and experimental measures.

This results in t-out-of-n threshold FHE schemes, in the asynchronous setting, with a much smaller modulus. Table 3.1 reports the additional modulus bits needed solely to accommodate the Lagrange reconstruction noise. For $n = 512$, our bounds reduce this modulus overhead (in bits) by about 32% when $t = n/2$ by up to 90% when $t = 500$ (relative to [BFM+25]).

3.3.2 Our Approach to Asynchronous ThFHE

To remove the specialness requirement, we slightly modify distributed decryption as follows. First, we define:

$$\mathsf{PartDec}(\mathsf{sk}_i, \mathsf{ct} = (a, b)) := C_{\mathsf{dec}} \cdot a \cdot \mathsf{sk}_i + C_{\mathsf{sim}} \cdot e_{\mathsf{fl},i} \bmod q \ .$$

[2] I may strictly contain (C_{dec}) as Lagrange coefficients in e_{fl} may cancel part of C_{sim}.
[3] The same mismatch occurs in [CLW25], in the related context of threshold public key encryption.

t	12	13	256	500
[BGG+18] (using our bound)	7918	7932	9016	8404
[BFM+25]	396	397	640	884
[CLW25]	180	185	3840	7500
Ours	81	81	438	86

Table 3.1: Number of modulus bits needed to accommodate the error resulting from Lagrange reconstruction, for $n = 512$ and various threshold values t.

Second, we define FinDec as first computing:

$$x = C_{\mathsf{dec}} \cdot b + \sum_{i \in \mathcal{S}} \lambda_i^{\mathcal{S}\to 0} \cdot \mathsf{p}_i \bmod q = C_{\mathsf{dec}} \cdot \left(\left\lceil \frac{q}{p} \right\rfloor \cdot \mu + e \right) + C_{\mathsf{sim}} \cdot e_{\mathsf{fl}} \bmod q \ ,$$

and then returning $C_{\mathsf{dec}}^{-1} \cdot \lceil x \cdot (p/q) \rfloor \bmod p$. Our analysis derives two conditions for this scheme to be correct, and two other conditions for it to be secure[4]:

(C1) Decryption Invertibility. Parameter $C_{\mathsf{dec}} \in R$ is invertible modulo p.

(C2) Decryption Correctness. For any set $\mathcal{S} \subseteq [n]$ of size t, for any $i \in \mathcal{S}$:

$$\left(B_{\mathsf{eval}} + \frac{p}{4} \right) \cdot (\|C_{\mathsf{dec}} \bmod q\|_1 + 1) + t \cdot B_{\mathsf{sm}} \cdot \|(C_{\mathsf{sim}} \cdot \lambda_i^{\mathcal{S}\to 0}) \bmod q\|_1 < \frac{q}{2p} \ .$$

(C3) Simulation Invertibility. Parameter $C_{\mathsf{sim}} \in R$ is invertible modulo q.

(C4) Simulation Flooding. For any $\mathcal{S}_{\mathsf{cor}} \subseteq [n]$ with $|\mathcal{S}_{\mathsf{cor}}| = t - 1$, denoting $\mathcal{S}^* = \{0\} \cup \mathcal{S}_{\mathsf{cor}}$, for any $i \in [n] \setminus \mathcal{S}^*$:

$$B_{\mathsf{sm}} \geq \left\| \left(\frac{C_{\mathsf{dec}}}{C_{\mathsf{sim}}} \cdot \lambda_0^{\mathcal{S}^*\to \alpha_i} \right) \bmod q \right\|_1 \cdot B_{\mathsf{eval}} \cdot 2^{\lambda} \cdot N \ .$$

Note that **(C2)** depends on B_{sm} and **(C4)** states how small one may set B_{sm}, hence the magnitude of q is driven by Conditions **(C2)** and **(C4)**. Setting B_{sm} as small as possible and ignoring lower-order terms, our analysis directly translates into a lower bound for setting q via the constraint:

$$\frac{q}{2p} > t \cdot B_{\mathsf{eval}} \cdot 2^{\lambda} \cdot N \cdot B_2 \cdot B_3 \ ,$$

where

$$B_2 := \max_{i,\,\mathcal{S}} \left\| \left(C_{\mathsf{sim}} \cdot \lambda_i^{\mathcal{S}\to 0} \right) \bmod q \right\|_1 \ ,$$

$$B_3 := \max_{i,\,\mathcal{S}^*} \left\| \left(\frac{C_{\mathsf{dec}}}{C_{\mathsf{sim}}} \cdot \lambda_0^{\mathcal{S}^*\to \alpha_i} \right) \bmod q \right\|_1 \ .$$

[4]We consider simulation-security with static corruption of $t-1$ parties, which is the standard notion. We do not provide a detailed security definition here (it is provided in [KLL+26]), but it is easily obtained from Definition 3.2.1 by the following tweaks: partial decryptions are no longer associated with a target set $\mathcal{S}$ as we are in the asynchronous setting, and the attacker can always decrypt given one additional decryption share as we assume $t-1$ corruptions. Hence, there is no $\mathsf{CanCombine}$ check, and one always falls into the Sim_2 case in the ideal experiment (therefore there is no Sim_1).

Therefore, we focus on obtaining tight upper bounds on B_2 and B_3, which are driven by the choice and magnitude of $C_{\mathsf{sim}}, C_{\mathsf{dec}}$, and of the Lagrange coefficients (which are directly related to the choice of interpolation points). We study two choices of interpolation points: integers and monomials. Here, we only describe interpolation over the monomials, which is, by far, the most efficient solution.

Interpolation over the monomials. The core contribution of our work is the study of interpolation over the monomials, i.e. $\mathcal{I}_{\text{mon}} := \{0, 1, x, x^2, \ldots, x^{n-1}\}$, assuming $n \leq 2N$. This is the choice made in [CLW25] (and in [BFM+25], up to re-ordering the points). It is shown in [AL21; CLW25] that one can set $C_{\mathsf{dec}} = C_{\mathsf{sim}} = 2^{\lceil \log t \rceil}$, which is also our choice. The denominator of $C_{\mathsf{sim}} \cdot \lambda_i^{\mathcal{S} \to 0}$ then becomes, up to a unit, a product of terms $(x^k - 1)$, for $k \in \{-n, \ldots, n\}$.

To obtain meaningful bounds, we pass through the complex embeddings of the cyclotomic ring, sending x to a primitive $(2N)$-th root of unity $\zeta = \exp(\pi i r/N)$ for an integer r with $\gcd(r, 2N) = 1$. Then, the denominator is a product of $\zeta^k - 1$, i.e., terms whose modulus is $2\big|\sin(\pi r k/(2N))\big|$. We can then leverage the detailed behavior of the sine function to identify worst-case configurations by ordering arguments so as to minimize the modulus of the denominator. Intuitively, this is done by taking terms whose angles are the closest to 0. This leads to a tight bound for B_2. We apply a similar analysis to bound B_3, the main difference being that, here the denominator of $\lambda_0^{\mathcal{S}^* \to \alpha_i}$ is a unit in R, hence we aim to maximize the modulus of the numerator which is again a product of terms $(x^k - 1)$. Intuitively, this is done by taking terms whose angles are the closest to $\pi/2$. We obtain:

$$B_2 \leq 2^{-t+\lceil \log t \rceil + 1} \cdot \sqrt{N} \cdot \prod_{k \in [\lfloor t/2 \rfloor]} \frac{1}{\sin^2\left(\pi \frac{k}{2N}\right)} \ ,$$

$$B_3 \leq 2^{t-1} \cdot \sqrt{N} \cdot \prod_{k \in [\lfloor t/2 \rfloor - 1]} \cos^2\left(\pi \frac{k}{2N}\right) \ .$$

We further optimize the construction in two orthogonal ways. These optimizations were already hinted in [CLW25] but only supported by heuristics and experimental measures. Our rigorous analysis provides solid grounding for them.

Optimization 1: Working over subrings. We observe that placing the n evaluation points evenly among the $2N$ roots of unity (actually among the monomials, which are mapped to roots of unity in our analysis) allows us to optimize bounds. In short, as sketched above, our bounds are obtained by combining terms whose angles are closest to some fixed angle (0 for the bound on B_2, $\pi/2$ for the bound on B_3), and distributing the evaluation points evenly maximizes the angular separation between any two points, therefore obtaining tighter bounds. Equivalently, we work with a generator $y = x^{2N/n'}$ for n' a power-of-two integer satisfying $n \leq n' \leq 2N$ so that the interpolation takes place in a smaller cyclotomic subring. This allows us to replace every N term in our bounds for B_2, B_3 above with n'.

We provide a simple comparison summary for bounds on $B_2 \cdot B_3$ in Table 3.2. In practice, we set $n' = 2^{\lceil \log n \rceil}$, which we simplify as n in Table 3.2.

We also provide more compact bounds for $B_2 \cdot B_3$ by taking logarithms of the product and comparing the resulting sums to integrals. This yields compact analytic bounds for B_2 and B_3 based on Clausen function of order 2 (see the full version for details). Applying this to

Interpolation set	Analysis	$B_2 \cdot B_3$
$\{(-1)^b x^i : b \in \{0,1\},\ i \in [n/2]\}$	[BFM+25]	$2^{\frac{3}{4}n+t}$
Any subset of $\{x^i\}_{0 \le i < 2N}$ of size n	[CLW25]	$\left(\frac{N}{\sqrt{2}}\right)^{2t}$
	Ours	$\prod_{k \in [t/2]} \cot^2\left(\pi \frac{k}{n}\right)$

Table 3.2: Comparison of different analyses of the quality of monomial interpolation points for Shamir secret sharing. For the sake of simplicity: only the dependency in t and n is reported; polynomial factors are omitted; zero, which is common to all interpolation sets, is omitted; t and n are assumed even.

the bound from the last row of Table 3.2, we obtain a compact bound for $B_2 \cdot B_3$:

$$B_2 \cdot B_3 \approx \left(\exp\left(\int_0^{t/(2n)} \ln \cot(\pi u)\, \mathrm{d}u\right)\right)^n ,$$

which is of the form $f(t/n)^n$, where f is defined over $[0,1]$.

The function f is plotted in Figure 3.3. It attains its maximum at $1/2$ (corresponding to $t = n/2$).

Defining $G = \int_0^{\pi/4} \ln \cot t \, \mathrm{d}t$, where G is Catalan's constant, we obtain that the maximum is equal to $\exp(2G/\pi) \approx 1.792$.

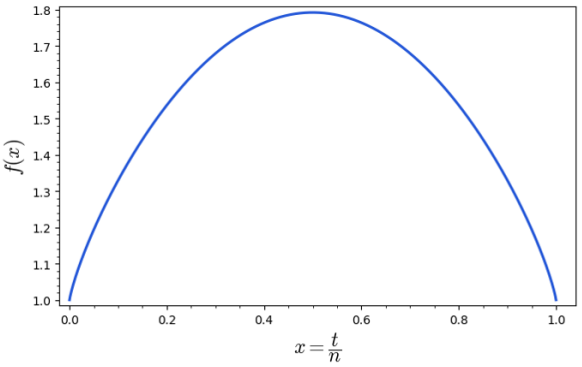

Figure 3.3: The monomial bound.

Optimization 2: Adding virtual parties. As visible in Figure 3.3, for a fixed number of parties n, the modulus q tends to grow as the threshold gets closer to $n/2$. We then observe that adding virtual parties, whose shares are public and therefore known to every real party, can help improve performance. Adding c virtual parties changes the threshold from t-out-of-n to $(t+c)$-out-of-$(n+c)$, but since the c additional shares are public, it does not change the effective t-out-of-n underlying threshold structure among the real parties. On the one hand, this adds a bit of computational and storage overhead to real parties, as they need to store virtual shares and compute the virtual partial decryption shares, but does not introduce any additional communication associated with virtual parties. On the other hand, it allows one to save on the modulus q. We show that the optimal choice to minimize q is $c = 0$ if $\lfloor t/2 \rfloor + \lfloor (t + n' - n)/2 \rfloor \le n'/2 + 1$ and $c = n' - n$ otherwise.

3.4 More Contributions on Threshold FHE

3.4.1 Server-Aided Low Communication Construction in the n-out-of-n Setting

The above two contributions provide constructions of threshold FHE in either the synchronous or the asynchronous model. The first one has a significant performance advantage over the second one, since one does not have to pay for the modulus overhead resulting from Lagrange

reconstruction. As can be seen from Table 3.1, this can save several hundreds of bits for the modulus q. Yet, there is still a significant overhead that both constructions require, which is related to the error $e_{\mathsf{fl},i}$ added during partial decryption. This error typically needs to be λ-bit larger than the computation error e, and therefore also imposes a λ-bit overhead to maintain correctness compared to non-threshold FHE (in some cases, one can reduce this flooding error to $\lambda/2$ bits [MW17], but this is still a large overhead). While it does not seem possible to avoid this flooding overhead in general [LMSS22], we provide a partial solution to this issue in [PS24].

Assuming an external trusted party, which is not colluding with any of the n parties holding key shares, we show that we can reduce the magnitude of $e_{\mathsf{fl},i}$ to only $\mathrm{poly}(\lambda)$ by having this trusted party pre-process ciphertexts before they are decrypted. We think of this trusted party as a server, e.g., the server performing the homomorphic computation. A typical application is delegation of computation on private inputs from multiple clients to a server: the server runs the computation homomorphically on data encrypted with threshold FHE by the clients, and then processes the resulting ciphertext before sending it back to the clients for distributed decryption. If this server is trusted, then applying our processing technique, it can largely reduce the size of the resulting ciphertext before it sends it back to the clients, who can still safely decrypt using only errors of small magnitude. This enables delegation of computation with small communication.

Our result only applies to LWE-based threshold FHE in the n-out-of-n setting, but it was later extended to RLWE-based schemes in the t-out-of-n setting in [OT25]. We only describe our variant.

To account for this additional trusted party, we modify the definition of threshold FHE by adding an algorithm termed ServerDec. We then define a threshold FHE scheme as a tuple (KeyGen, Enc, Eval, ServerDec, PartDec, FinDec) where KeyGen, Enc, Eval, PartDec, and FinDec are the usual threshold FHE algorithms. ServerDec is an additional, public-key, randomized algorithm, which can be performed on fresh or evaluated ciphertexts, and before partial decryption. It returns a ciphertext which fits the input format of PartDec. We suppose that ServerDec is executed by an uncorrupted party (e.g., the server) and uses randomness that remains unknown to all parties (and therefore to the adversary). It may seem that it could then be integrated directly in Eval algorithm, but adding an explicit algorithm increases flexibility: for instance, ServerDec might convert ciphertext into a format which prevents further homomorphic evaluation. Also, adding an explicit ServerDec algorithm allows Eval to possibly remain deterministic. Finally, it allows one to identify precisely the sources of security. Considering an adversary which corrupts all first $n-1$ users, the security is guaranteed thanks to the remaining uncorrupted randomness, i.e., (1) the internal randomness of ServerDec, and (2) the randomness held by P_n (its share $\mathbf{s}_n$ of the key and its internal randomness of PartDec). Speaking of security, we define a simulation-based security notion in a similar fashion as for standard threshold FHE. Note that we are in the n-out-of-n setting, so the secret key is simply shared with additive secret sharing, and we assume static corruptions of $n-1$ parties.

Double-flood-and-round threshold FHE. Our main result is a construction of threshold FHE with low communication. We call our technique *Double-Flood-and-Round*. Consider a generic FHE scheme whose ciphertexts have the above form $(\mathbf{a}, \mathbf{a}^\intercal\mathbf{s} + \lfloor \frac{Q}{p} \rceil \cdot \mu + e)$ over $\mathbb{Z}_Q$ for some (exponentially) large modulus $Q = Q_{\mathsf{fl}} \cdot q_{\mathsf{dec}}$ with $Q_{\mathsf{fl}} = \Omega(2^\lambda)$ and $q_{\mathsf{dec}} = \mathrm{poly}(\lambda)$. The key generation, encryption, and evaluation algorithms are directly inherited from

the underlying FHE scheme, except that the secret key $\mathbf{s}$ is additively secret shared as $\mathbf{s} = \mathbf{s}_1 + \cdots + \mathbf{s}_N \bmod q_{\mathsf{dec}}$. Our technique is to first rely on exponential noise flooding to sanitize evaluated ciphertexts before partial decryption. This is done by the $\mathsf{ServerDec}$ algorithm. It takes a ciphertext $(\mathbf{a}, b) \in \mathbb{Z}_Q^n \times \mathbb{Z}_Q$ and returns a ciphertext $(\mathbf{a}', b') \in \mathbb{Z}_{q_{\mathsf{dec}}}^n \times \mathbb{Z}_{q_{\mathsf{dec}}}$ with:

$$\mathbf{a}' = \left\lfloor \frac{1}{Q_{\mathsf{fl}}} \cdot \mathbf{a} \right\rceil_{\sigma_0} \quad , \quad b' = \left\lfloor \frac{1}{Q_{\mathsf{fl}}} \cdot (b + e_{\mathsf{fl}}) \right\rceil_{\sigma_1} \quad .$$

where $\lfloor \cdot \rceil_\sigma$ denotes a randomized Gaussian rounding, which on input $x \in \mathbb{R}$ returns an element from $\mathcal{D}_{\mathbb{Z},x,\sigma}$, and where e_{fl} is an exponentially large Gaussian noise term. Via standard noise flooding, the error term e_{fl} statistically hides the error term from b. Moreover, thanks to the rescaling to $q_{\mathsf{dec}} = \mathrm{poly}(\lambda)$, the ciphertext $(\mathbf{a}', b')$ sent to parties to be partially decrypted is only $(n+1)\log(q_{\mathsf{dec}})$-bit long. $\mathsf{PartDec}$ and FinDec then follow the same design as before.

We are able to prove that adding a very small amount of noise (whose magnitude is even independent of the number of decryptions Q_D) during partial decryption suffices to guarantee security. Specifically, a ciphertext $(\mathbf{a}', b')$ returned by $\mathsf{ServerDec}$ is of the form $(\mathbf{a}', \mathbf{a}'^{\intercal}\mathbf{s} + \mu + e')$, where e' has the form $\left\lfloor \frac{1}{Q_{\mathsf{fl}}} \cdot (\mathbf{r}_0^{\intercal}\mathbf{s} + e_{\mathsf{fl}}) \right\rceil_{\sigma_1}$ with $\mathbf{r}_0$ denoting the (Gaussian) rounding error of $\mathbf{a}$, i.e., $\mathbf{r}_0 = \mathbf{a} - Q_{\mathsf{fl}} \cdot \mathbf{a}'$. Therefore, following a similar approach as the proof of [BLP+13, Lemma 3.5], one can show that e' is statistically close from a Gaussian distribution whose standard deviation is $\sqrt{(\sigma_0\|\mathbf{s}\|)^2 + (\sigma_{\mathsf{flood}}/Q_{\mathsf{fl}})^2 + \sigma_1^2}$. Assuming $\|\mathbf{s}\|$ is publicly known, this distribution is publicly sampleable. Thanks to this clean distribution of e', the rest of the security analysis is analogous to the recent proof of security of lattice-based threshold public-key encryption from [MS23].

A partial decryption of $(\mathbf{a}', b')$ computed by party P_n owning $\mathbf{s}_n$ is of the form $\mathbf{a}'^{\intercal}\mathbf{s}_n + \delta_n$. Sampling δ_n from $\mathcal{D}_{\mathbb{Z},\eta}$, we then obtain that the view of an adversary corrupting all parties $P_1, \ldots, P_{n-1}$ (and therefore $\mathbf{s}_1, \ldots, \mathbf{s}_{n-1}$) is a triple $(\mathbf{a}', \mathbf{a}'^{\intercal}\mathbf{s} + \lfloor \frac{q_{\mathsf{dec}}}{p} \rceil \cdot \mu + e', \mathbf{a}'^{\intercal}\mathbf{s}_n + \delta_n)$, and adding $\mathbf{a}'^{\intercal} \sum_{i \in [n-1]} \mathbf{s}_n$ to the third term, it is then a triple of the form:

$$(\mathbf{a}', \mathbf{a}'^{\intercal}\mathbf{s} + \mu + e', \mathbf{a}'^{\intercal}\mathbf{s} + \delta_n) \ ,$$

with $e' \sim \mathcal{D}_{\mathbb{Z},\sqrt{(\sigma_0\|\mathbf{s}\|)^2 + (\sigma_{\mathsf{flood}}/Q_{\mathsf{fl}})^2 + \sigma_1^2}}$ and $\delta_N \sim \mathcal{D}_{\mathbb{Z},\eta}$. Assuming LWE, we prove the above distribution is computationally indistinguishable from a triple:

$$(\mathbf{a}', b', b' + \gamma) \ ,$$

where $\gamma \leftarrow \mathcal{D}_{\sigma_\gamma}$ with $\sigma_\gamma := \sqrt{(\sigma_0\|\mathbf{s}\|)^2 + (\sigma_{\mathsf{flood}}/Q_{\mathsf{fl}})^2 + \sigma_1^2 + \eta^2}$. The latter distribution is publicly sampleable if $\|\mathbf{s}\|$ is known, which does not hurt the analysis.

We further note that the partial decryption shares can be rounded, in order to lower communication (we do not use this for security). This explains why the construction is called double-round-and-flood. As a final remark, note that our input FHE ciphertexts are not compact by default, as they are defined over a large modulus Q. This problem is solved by relying on transciphering, e.g., using [BCK+23]. Hence, all communications are small, since all ciphertexts that transit from clients to the server are small thanks to transciphering, and ciphertexts that transit from the server to the clients as well as partial decryption shares that transit between clients are defined over $q_{\mathsf{dec}} = \mathrm{poly}(\lambda)$.

3.4.2 Distributed Key Generation for Efficient Threshold-CKKS

The prior works focus on improving the performance of threshold decryption, but do not provide any solution for key generation. Indeed, the algorithm KeyGen that distributes the secret key shares to parties is assumed to be run by a trusted party. In practice, one would like to distribute the key material without any trusted setup. This is the purpose of distributed key generation, which we discuss in this final section. We study the problem of distributed key generation (for CKKS) in [MHP+25], and now briefly explain some of the contributions from this work.

Motivation. We recall that the bootstrapping multiplicative depth depends on the Hamming weight of the secret key. Then, in order to obtain an efficient CKKS instantiation, the best known approach is to rely on the Sparse Secret Encapsulation (SSE) technique from [BTH22]. SSE generates two (ternary) secret keys: a sparse key and a less-sparse key. The less-sparse key may have Hamming weight ranging from 192 to $2N/3$ where N is the dimension of the key (a power-of-two integer typically between 2^{15} and 2^{17}) and is used most of the time. The sparse key may have Hamming weight as low as 32 and is used inside bootstrapping: the secret for a ciphertext is temporarily switched to the sparse key, and then switched back to the less-sparse key. To enable this, two switching keys are required: an encryption of sk under the sparse key sk' at a low modulus, and an encryption of sk' under sk at a high modulus. SSE is used in all competitive implementations of CKKS (see [Lattigo; OpenFHE; HEaaN]). We note that sparse Ring-LWE secrets have been used in other FHE schemes as well to accelerate their bootstrapping algorithms (see, e.g., [GHS12; LMSS23; BTH22; MHWW24]).

ThFHE puts an additional constraint on bootstrapping: its correctness not only impacts the usability of the scheme, but also its security. Indeed, as we showed in [CCP+24], the bootstrapping algorithms of several FHE schemes including CKKS lead to key-recovery attacks when bootstrapping failures are observed. To obtain sufficiently low failure probability without damaging performance, the only known approach is SSE [BTH22]. Therefore, to enable a high-throughput threshold-CKKS, one needs a key generation protocol that produces properly distributed key material. The prior solutions are far from satisfactory. Key generation algorithms that rely on generic secure multi-party computation or on a trusted dealer fail in terms of efficiency and security, respectively. The state-of-the-art Distributed Key Generation (DKG) protocol for threshold-CKKS [MTBH21] has each party generate its own key pair and, as there is an additive homomorphism between the secret key and the public key in CKKS (up to small noise and assuming that the uniform part of the key is generated using a common reference string), one can then set the joint public key (resp. secret key) as the sum of all public keys (resp. secret keys). Even though the evaluation key involves a quadratic function of the secret key, it is possible to obtain a joint evaluation key for the joint secret key in one more round, or without any extra round but with a produced evaluation key that differs from usual and results in lower throughput [KLSW24; BMM25]. The algorithm from [MTBH21] was extended from the n-out-of-n setting (also referred to as multi-party FHE) to the t-out-of-n setting in [MBH23]. As the joint secret key is the sum of independently sampled keys, with overwhelming probability, it is not sparse, even if the individual keys were sparse. The joint key is unusable for SSE, and bootstrapping then becomes less efficient and consumes more modulus. For a number of users $n = 32$, we estimate that the throughput would be at least 71% lower. We stress that this slowdown stems from the DKG protocol but impacts the throughput of all computations. Further, as the magnitude of the key depends on the number of users n and the bootstrapping circuit depends on the magnitude

of the key, bootstrapping circuits for diverse numbers of users should be considered (or a single one for a large maximum number of users, but that will be unnecessarily inefficient for a smaller number of users). As bootstrapping is a complex algorithm with numerous optimizations, this may lead to significant software development complications. For these reasons, threshold-CKKS implementations typically off-load bootstrapping to an interactive protocol, damaging throughput because of communication delays.

Contributions. To address all the above issues, we describe a DKG protocol between an arbitrary number of parties n that only assumes a common reference string (which can be implemented with a hash function modeled as a random oracle). It outputs a public key and an evaluation key that enable homomorphic evaluations that are near-identical to state-of-the-art non-threshold CKKS based on sparse secret encapsulation. Each party obtains a decryption key share, for n-out-of-n threshold decryption. This extends to t-out-of-n for any $t \leq n$ by relying on [MBH23] and, for this reason, this work focuses on the n-out-of-n case.

We propose several variants of the protocol, with 2 to 4 rounds of communication and various trade-offs between the run-time of the DKG and the throughput performance and precision of the homomorphic computations. We implemented a 4-round variant with an evaluation precision that is slightly degraded. For this purpose, we used a variant of the HEaaN library [HEaaN] based on [CCK+25]. In our experiments, the dominating step of DKG runs in slightly more than 5 minutes on a single-threaded CPU for $n = 32$ users, and slightly more than 2 seconds on a commodity GPU (NVIDIA RTX 4090). This computation can be performed publicly, by any of the parties or by external servers, making it likely that GPUs would be available. Using the resulting key, the computation throughput is identical to that of non-threshold CKKS, and the precision degrades by less than two bits.

The techniques developed can also be used to delegate key generation (in a single-party or multi-party CKKS). Assume a limited client, for example an IoT device, wishes to externalize computations. It may not have the computational resources to generate and transmit an evaluation key, which can be of the order of 5-10GB. Instead, it can ask a number of external parties to call key generators to collectively run a variant of the DKG algorithm so that only the client gets to know the secret key whereas the public key and evaluation key are made public. Security relies on the assumption that the key generators do not collude.

Our DKG in Short. At a conceptual level, the proposed DKG protocol runs the ThFHE-based universal thresholdizer framework from [BGG+18] on the key generation algorithm for a single-key FHE, with a suboptimal but temporary ThFHE scheme. First, the parties collaboratively run a setup protocol of some (n-out-of-n) ThFHE scheme with low communication. Note that the resulting scheme generally has a different key-pair structure from the single-key scheme, but it is only a temporary situation. Then, the parties homomorphically sample from the secret key distribution (optionally, from the noise distribution as well), using encrypted random bits from the parties, using this suboptimal ThFHE scheme. Finally, the parties can jointly decrypt this ciphertext to obtain a share of this secret and run another setup protocol of the ThFHE scheme with these new shares. This new scheme has the same key distribution as the conventional single-key FHE scheme, and ThFHE homomorphic computations can be performed exactly and as efficiently as in the single-key case. We emphasize that this framework can be applied to obtain a DKG protocol for any ThFHE scheme, temporarily relying on an existing ThFHE scheme that is used to homomorphically evaluate the key generation algorithm of the target ThFHE.

In this work, we focus on CKKS, both as target and temporary ThFHE schemes, as it provides high-throughput computations on approximate and exact data. In particular, this allows us to enjoy an efficient evaluation of the key generation circuit. Our main DKG protocol variant first runs the 2-round protocol from [MTBH21] to obtain temporary key material: this step outputs a public key and an evaluation key whose underlying secret key is the sum of n individual secret keys independently sampled by the n users, as well as an n-out-of-n decryption key share for each user. As we have mentioned already, this key material does not allow computations that are as efficient as single-key CKKS, but they suffice for temporary computations internal to DKG. Once we have those keys, we homomorphically run the secret key generation algorithm of the SSE variant of CKKS before the third communication round, to obtain shares of a sparse secret key that can be used for bootstrapping and a less-sparse secret key that can be used during the rest of homomorphic computations, as well as partial evaluation key material. The protocol uses a fourth communication round to create the non-linear component of the evaluation keys (namely, the relinearization key), as in [MTBH21].

We observe that the last round can be removed by homomorphically running the public and evaluation key generation algorithms from [BTH22] at the third round, on top of the secret key generation algorithm. Another round can be saved by handling relinearization keys of the temporary ThFHE scheme as in [KLSW24; BMM25]. In all these variants, the generated global keys have noise terms that grow with the number of users, leading to lower computation precision. Noise terms as in single-key CKKS can be obtained by homomorphically generating as well, at the expense of a slower DKG protocol.

The only computationally intensive component of our main DKG protocol variant is the homomorphic evaluation of the SSE secret key generation algorithm, at the third round. For the sparse key, we have to homomorphically sample a uniform ternary vector of dimension N and Hamming weight exactly h, where N is large (e.g., $N = 2^{16}$) and h is small (e.g., $h = 32$). Note that the value of h is very precise, as it directly impacts the performance of bootstrapping, the security of the evaluation key, and the failure probability of bootstrapping. Comparatively, sampling the less-sparse key is easier, as its precise Hamming weight is not so impactful: only its rough value matters.

We complete this overview by explaining how to generate the sparse key. Our goal is to design an algorithm for sampling a uniform ternary vector of dimension N and Hamming weight $h \ll N$, which can be CKKS-evaluated. For the input randomness, each one of the n parties sends an encryption of a uniform bit-string: these are added modulo 2, by placing the plaintext bits in the most significant bits of the ciphertexts.

Up to multiplying (element-wise) by a vector of uniform signs, we can concentrate on using those bootstrapped uniform bits to generate a uniform binary vector of Hamming weight h. For this purpose, we start with two simpler sub-tasks. First, we consider the task of sampling a uniform one-hot vector, i.e. a vector that is 1 in one of its entries and 0 everywhere else, and second, we propose an algorithm for evaluating an integer indicator function whose distinguished point is at the boundary: for $B \geq 0$, the function δ_B maps $x = 0$ to 1 and $x \in \{1, \ldots, B\}$ to 0. Then, our sparse secret key sampler takes as input $H \geq h$ one-hot vectors, adds the first i of them for all $i \in [h, H]$ and keeps the first sum that has Hamming weight h, and ensures the returned vector binary by relying on our indicator function. The value of H is chosen so that the sum of H one-hot vectors has Hamming weight $\geq h$ with overwhelming probability: by a balls-and-bins argument, it may be shown that H does not need to be much larger than h, allowing the design of an efficient algorithm.

Chapter 4

Threshold Dilithium

In this last technical chapter, we explain how to construct an efficient threshold variant of Dilithium (a.k.a. ML-DSA) using threshold FHE. Indeed, among the numerous applications of ThFHE, a famous one is *Universal Thresholdizers* [BGG+18]: threshold FHE can be used as a tool to thresholdize any other cryptographic primitive, and in particular digital signatures. We apply this paradigm to construct a threshold variant of Dilithium, by combining various contributions from Chapters 2 and 3. The results from this chapter are sampled from the following recent work. Here, we focus on the protocol and its analysis and leave most details regarding how to efficiently perform the homomorphic computation in the full version.

- **THED: Threshold Dilithium from FHE**
  with Jai Hyun Park and Damien Stehlé
  Manuscript

Contents

4.1 Background

Following its standardization by NIST as the primary algorithm for post-quantum digital signatures [AAC+22], Dilithium/ML-DSA [DKL+18; BDK+21] is being rapidly deployed in various environments. We recall that Dilithium is an optimized version of Lyubashevsky's lattice-based signature scheme [Lyu09; Lyu12], which we discussed in Chapter 2. As for all signature schemes, the signing key is particularly sensitive and requires dedicated protection. Similarly to the fact that threshold FHE, which we discussed in Chapter 3, provides secure distributed decryption for FHE, threshold signatures provide a classical approach to strengthen the availability and security of signature issuance, by distributing the signing key among multiple parties [DF89]: if too few parties participate in the execution of the protocol, then they must be unable to produce a signature that passes verification; if sufficiently many parties collaborate, then they should be able to produce valid signatures. For this reason, threshold versions of Dilithium are actively being investigated (see, e.g., [BP25] and the references below). Given that Dilithium is standardized, a particularly desirable property of a threshold Dilithium protocol would be its verification interchangeability with non-threshold Dilithium: a signature produced by executing the threshold Dilithium protocol must be accepted as valid by Dilithium's verification algorithm.

The main difficulties in obtaining a threshold signature scheme that is interchangeable with Dilithium are the signature and verification key compression techniques introduced in [BG14] and [DKL+18]. Many threshold signature schemes have been built by modifying Lyubashevsky's signature, but most of them do not handle Dilithium's optimizations and are not verification interchangeable [CS19; TPCZ23; GKS24; PKM+24; KRT24; EKT24; CTZ24; BKL+25] (see also [DOTT22; Che23; ADP24] for the restricted n-out-of-n case where all users must collaborate to produce valid signatures, and [VSW21; LSV22; SVL24] for the $n = 2$ subcase). Yet, the landscape is rapidly evolving, as a few interchangeable schemes have been proposed very recently. Using secure multi-party computation (MPC), Dufka et al. [DKLS25] obtained a solution for $t = n = 2$. Still using MPC, Bienstock et al. [BCE+26] obtained a solution for arbitrary t and n, but with tens of rounds and MBs of communication per user. A more direct adaptation of ML-DSA was proposed in [CPE+26], with only three rounds of communication, but it is limited to $t, n \leq 6$.

This is where threshold FHE comes into play. From a theoretical point of view, it was shown in [BGG+18] that any (deterministic) signature scheme can be turned into a 1-round threshold version thereof by the use of threshold fully homomorphic encryption: one provides all users with an FHE encryption of the signing key so that, given a message to be signed, all users can locally compute an FHE encryption of a common signature of the message, and parties run distributed decryption to obtain the signature in the clear. This generic construction produces signatures that are interchangeable with the original signature scheme, and could hence be applied to Dilithium. Such an approach has been considered in [ASY22] for a basic version of Lyubashevsky's signature scheme, focusing on removing its use of rejection sampling, but applying it to Dilithium is challenging due to its design.

4.2 Contributions in a Nutshell

We describe the first concrete FHE-based threshold signature scheme that is verification interchangeable with Dilithium. The construction relies on the universal thresholdizer

from [BGG+18], which we modify and instantiate to achieve practical efficiency in 2 rounds of communication.

Using the implementation of the CKKS FHE scheme [CKKS17] available in the HEaaN library [HEaaN], we wrote a proof-of-concept implementation of the FHE components of the signing algorithm for the NIST level-2 instantiation of Dilithium. For a probability of successfully generating a signature that is equal to $p = 0.859$, this costs 1.115s on a commodity GPU (NVIDIA RTX 5090). The communication per party is 22.94KB, assuming up to 128 decryptors in the synchronous setting. In fact, the first round can be preprocessed by making it independent from the message; without taking preprocessing into account, the runtime is decreased to 0.182s, with per-party communication of 3.94KB. We observe that the computation is public (except for threshold decryption) and can be externalized to a server. If the protocol fails to produce a valid signature, then it can be restarted. Alternatively, the protocol can be executed several times in parallel to produce a signature with probability arbitrarily close to 1.

Comparison with Related Works. Our FHE-based solution has the following advantages compared to the other existing approaches for threshold Dilithium [DKLS25; BCE+26; CPE+26].

- It works for every t and n and scales well with them. The main impact is the decryption modulus (appearing at both rounds): it increases pre-decryption computation and communication complexity. Thanks to our prior contributions, the cost of threshold decryption in the asynchronous model (Chapter 3, Sec.3.3) has been significantly reduced, and, if we assume that the decryptors are aware of who they are, this dependency can be almost entirely removed via our synchronous construction (Chapter 3, Sec.3.2). Most of the computation is independent of t and n and so is the number of rounds.
- It requires only 2 rounds: this is to be compared to 3 rounds for [CPE+26] and more than 20 rounds for the MPC-based approaches from [DKLS25; BCE+26]. Note that these can benefit from trade-offs between communication amount and the number of rounds, but communication is already in the MBs without minimizing the number of rounds. In terms of communication volume, our solution involves tens of KB per party, which is similar to [CPE+26] when $t \leq 3$ and significantly less otherwise.
- Almost all the computations can be performed publicly: only the ThFHE decryption keys and decryption steps are secret. The computations can hence be delegated to powerful devices. It also implies that a much smaller amount of software needs to be protected.

The main drawback of our approach is the high computational cost. However, as already observed, most of the computation is public and can be delegated to powerful machines. The main alternative for arbitrary t and n is the scheme from [BCE+26], but the high communication and round complexity quickly translates into high runtimes in environments where participants are distant.

Security Model. We focus on passive security for the sake of simplicity, but note that active security can be achieved by adding zero-knowledge proofs of honest ThFHE partial decryptions, consistent with public commitments of the partial decryption key shares. This provides robustness with identifiable aborts without any honest majority assumption. The statements to be proved are linear with smallness constraints, which is typical of lattice-based cryptography, and we refer to [BS23; LNP22b; NS24; HLSS25] (among others) for potential

instantiations. Independently, note that we can instantiate distributed key generation via our ThFHE distributed key generation [MHP+25] (Chapter 3, Sec. 3.4.2), which enables fast CKKS computations, and by a homomorphic evaluation of Dilithium's key generation algorithm. Hence, our construction does not require a trusted setup.

4.3 Challenges for Thresholdizing Dilithium

Before going further, we first recall, in Figure 4.1, the precise description of (the deterministic version of) Dilithium, as described in [BDK+21]. We use the same notations and elementary functions and refer the reader to the Dilithium documentation [BDK+21] for their definitions. Similarly as in Chapter 2, in the signing algorithm Sign, we refer to $\mathbf{w}_1$ (Step 8) as the commitment, to c (Step 10) as the challenge, and to $\mathbf{z}$ (Step 11) as the response.

Dilithium.KeyGen

1 $\zeta \leftarrow U(\{0,1\}^{256})$
2 $(\rho, \rho', K) \leftarrow \mathsf{H}(\zeta)$
3 $\mathbf{A} \leftarrow \mathsf{ExpandA}(\rho) \in (\mathcal{R}_q^{\mathtt{SIG}})^{k \times \ell}$
4 $(\mathbf{s}_1, \mathbf{s}_2) \leftarrow \mathsf{ExpandS}(\rho') \in S_\eta^\ell \times S_\eta^k$
5 $\mathbf{t} \leftarrow \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 \bmod q$
6 $(\mathbf{t}_0, \mathbf{t}_1) \leftarrow \mathsf{Power2Round}_q(\mathbf{t}, d)$
7 $tr \leftarrow (\rho \parallel \mathbf{t}_1) \in \{0,1\}^{384}$
8 **return** $\mathsf{vk} = (\rho, \mathbf{t}_1)$, $\mathsf{sk} = (\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0)$

Verify(vk, $M, \sigma = (\mathbf{z}, \mathbf{h}, \tilde{c})$)

1 $\mathbf{A} \leftarrow \mathsf{ExpandA}(\rho)$
2 $\mu \leftarrow \mathsf{H}(\mathsf{H}(\rho \parallel \mathbf{t}_1) \parallel M)$
3 $c \leftarrow \mathsf{SampleInBall}(\tilde{c})$
4 $\mathbf{w}_1' \leftarrow \mathsf{UseHint}_q(\mathbf{h},\ \mathbf{A}\mathbf{z} - c\mathbf{t}_1 \cdot 2^d,\ 2\gamma_2)$
5 **if** $\|\mathbf{z}\|_\infty < \gamma_1 - \beta \wedge \tilde{c} = \mathsf{H}(\mu \parallel \mathbf{w}_1') \wedge \mathsf{HW}(\mathbf{h}) \leq \omega$
6 **return** 1
7 **else**
8 **return** 0

Sign(sk, M)

1 $\mathbf{A} \leftarrow \mathsf{ExpandA}(\rho)$
2 $\mu \leftarrow \mathsf{H}(tr \parallel M)$
3 $\kappa \leftarrow 0, \quad (\mathbf{z}, \mathbf{h}) \leftarrow \perp$
4 $\rho' \leftarrow \mathsf{H}(K \parallel \mu)$
5 **while** $(\mathbf{z}, \mathbf{h}) = \perp$ **do**
6 $\mathbf{y} \leftarrow \mathsf{ExpandMask}(\rho', \kappa) \in S_{\gamma_1 - 1}^\ell$
7 $\mathbf{w} \leftarrow \mathbf{A}\mathbf{y} \bmod q$
8 $\mathbf{w}_1 \leftarrow \mathsf{HighBits}(\mathbf{w}, 2\gamma_2)$
9 $\tilde{c} \leftarrow \mathsf{H}(\mu \parallel \mathbf{w}_1) \in \{0,1\}^{256}$
10 $c \leftarrow \mathsf{SampleInBall}(\tilde{c}) \in B_\tau$
11 $\mathbf{z} \leftarrow \mathbf{y} + c\mathbf{s}_1$
12 $\mathbf{r}_0 \leftarrow \mathsf{LowBits}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)$
13 **if** $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta \ \vee \ \|\mathbf{r}_0\|_\infty \geq \gamma_2 - \beta$ **then**
14 $(\mathbf{z}, \mathbf{h}) \leftarrow \perp$
15 **else**
16 $\mathbf{h} \leftarrow \mathsf{MakeHint}_q(-c\mathbf{t}_0,\ \mathbf{w} - c\mathbf{s}_2 + c\mathbf{t}_0,\ 2\gamma_2)$
17 **if** $\|c\mathbf{t}_0\|_\infty \geq \gamma_2 \ \vee \ \mathsf{HW}(\mathbf{h}) > \omega$ **then** $(\mathbf{z}, \mathbf{h}) \leftarrow \perp$
18 $\kappa \leftarrow \kappa + \ell$
19 **return** $\sigma = (\mathbf{z}, \mathbf{h}, \tilde{c})$

Figure 4.1: Deterministic Dilithium.

The main challenge in applying [BGG+18] to thresholdize a signature scheme is to find efficient ways to homomorphically implement the various steps of the signing algorithm. From that perspective, Dilithium comes with a number of challenges. The step numbers below refer to the Sign algorithm in Figure 4.1.

(i) Dilithium relies on SHAKE for expanding part of the verification key (Step 1), hashing the message (Steps 2 and 4), generating the signing mask (Step 6) and generating the signing challenge (Step 9);
(ii) It involves rejection sampling: if any of the four tests (Steps 13 and 17) is invalid, then the signing process has to be restarted;
(iii) All of these rejection tests rely on comparisons, sometimes on integers for which we have extracted the most and least significant bits (with the HighBits, LowBits and MakeHint functions, at Steps 8, 12 and 16);
(iv) The main data type is integers, but some operations involve modular arithmetic (Step 7), others are over $\mathbb{Z}$ (Step 11), and yet others involve comparisons (see above).

In what follows, we focus on our protocol design and analysis, and explain how we tackle difficulties (i) and (ii). We defer low-level implementation/optimization details (notably (iii)

and (iv)) to the full version [PPS26].

Circumventing Challenge (i): Homomorphically evaluating SHAKE has been little explored in the literature. The state-of-the-art approach [HCLK24] relies on [DM15; CGGI16]. It consumes more than 1 hour in single-thread CPU to evaluate one SHA3 round function. Note that we have several kilobytes of data to hash to obtain the signing challenge and the cost of hashing grows linearly with the amount of data to be hashed. We avoid this high cost in several ways. First, we note that some calls to SHAKE can be done publicly, as they do not involve any secret quantity: the expansion of the uniform component of the verification key (Step 1) and the hashing of the message (Steps 2 and 4). Second, we observe that SHAKE is used as a pseudo-random function (PRF) in the generation of the signing mask (Step 6). As we consider a deterministic signature scheme, the PRF key is part of the signing key and is provided in encrypted form. We replace SHAKE by a PRF that is more FHE-friendly, namely, the Learning-With-Rounding Darkmatter PRF from [BIP+18] which we briefly discussed in Chapter 1 (Sec. 1.7). To make it more efficient, we replace the matrix-vector product by a polynomial ring multiplication, where the ring is chosen to be identical to that of CKKS. This leads to signatures that differ from Dilithium's, but does not prevent us from achieving verification interchangeability since we are only changing the internal randomness. Finally, we generate the signing challenge in the clear (Step 9), by decrypting its input with the ThFHE distributed decryption algorithm. This is the reason why our scheme has two communication rounds, instead of a single one as would be obtained by directly applying [BGG+18].

Dealing with Challenge (ii): The use of rejection sampling implies that the signing algorithm is cumbersome to describe as a circuit. If the probability to pass the tests is p, then the distribution of the number of signing attempts follows a geometric law of parameter p. This is difficult to handle with the circuit-based model of homomorphic computations. This difficulty was circumvented in [ASY22; GKS24; PKM+24; KRT24; EKT24; CTZ24; BKL+25] by removing rejection sampling altogether: this is supported by a security proof based on the Rényi divergence rather than the statistical distance, but leads to increased parameters. As this requires parameter changes and handles only one of the four Dilithium rejection tests, this approach seems incompatible with verification interchangeability. We instead take a more direct approach: we perform L signing attempts in parallel, using the high dimension of the FHE plaintexts. In our implementation, one ciphertext contains $L = 8$ parallel attempts. On the one hand, we might have several successful iterations: as we only want to output a single signature, we only keep the first successful signature and (homomorphically) discard the other ones. On the other hand, repeating the protocol boosts the success probability, but it remains non-negligibly away from 1 as increasing the number of parallel attempts too much would harm the performance.

Consequently, our (threshold) signing algorithm has the following structure:

(1) Homomorphically sample L commitments $(\mathbf{w}_1^{(i)})_{i\in[L]}$;
(2) Compute in the clear the L associated challenges $(c^{(i)})_{i\in[L]}$;
(3) Homomorphically compute the L corresponding responses $(\mathbf{z}^{(i)})_{i\in[L]}$.

The signers decrypt the outcome of (1) to run (2) in the clear. Then, (3) is performed homomorphically and only its result is decrypted. Furthermore, our homomorphic evaluation (3) only keeps the first valid $\mathbf{z}_i$'s, such that only one valid signature is revealed. Our two rounds of communication thus correspond to the two times the signers communicate their

partial decryption shares, at the ends of (1) and (3).

Protocol Design to Simplify Tests (iii). The signing algorithm involves several tests, including comparisons which can be expensive to perform homomorphically. While we perform most of them homomorphically (details in the full version), we notice that the tests of Step 17 occur on data that is known in the clear to the signers; this can be seen by rewriting $\mathbf{w} - c\mathbf{s}_2$ as $\mathbf{Az} - c\mathbf{t}$ and noting that $\mathbf{t}$ can be given to the signers without impacting security.

Security Aspects. Creating the signing challenge in the clear decreases the burden on the FHE evaluation of the signing algorithm, but the two-round nature of the threshold signature protocol raises more security questions than if applying the universal thresholdizer [BGG+18] directly. Notably, as several signing attempts are run in parallel (exploiting the SIMD nature of CKKS) when making signing queries, the adversary can see challenges of failed signing attempts, as well as challenges of successful signing attempts that are subsequently discarded as a single signature is issued.

For the security proof, we start from the assumption that deterministic Dilithium is secure when using the Darkmatter PRF for the signing mask (Step 6 of Figure 4.1). We also assume that this Dilithium version remains secure if the adversary has oracle access to aborted transcripts, i.e., it can see the commitment of Step 8 for rejected signatures, and both unselected and selected accepted signatures. Revealing the commitments of rejected signatures is consistent with state-of-the-art secure implementations of Dilithium against side-channel attacks [MGTF19; ABC+23; CGTZ23; CGL+24]. Based on this, the security proof follows the blueprint of [ASY22]. Oracle queries to Round 1 are simulated from aborted transcripts, and oracle queries to Round 2 are simulated using a full signature query (including the full Round-1 transcript). As in [ASY22], we rely on the Rényi divergence to minimize the amount of noise in ThFHE decryption.

4.4 Threshold Dilithium in Two Rounds

Let us now dive into of our protocol.

4.4.1 Definitions

We first recall the definition of threshold signatures. Intuitively, a t-out-of-n threshold signature allows one to distribute a signing key among n parties such that any subset of t parties can collaboratively sign any message, while any smaller subset of parties cannot issue valid signatures. We focus on threshold signatures with two rounds of communication in the semi-honest model.

Definition 4.4.1. *A 2-round threshold signature scheme is a tuple of PPT algorithms* $(\mathsf{KeyGen}, \mathsf{PartSign}_1, \mathsf{Combine}_1, \mathsf{PartSign}_2, \mathsf{Combine}_2, \mathsf{Verify})$ *with the following specifications:*

- $\mathsf{KeyGen}(1^\lambda, 1^n, 1^t)$ *takes as inputs a security parameter* λ*, a number of parties* n *and a threshold* t *and outputs public parameters* pp*, a verification key* vk*, and partial signing keys* $(\mathsf{sk}_i)_{i\in[n]}$*;*
- $\mathsf{PartSign}_1(\mathsf{pp}, \mathsf{sk}_i, \mu)$ *takes as inputs the public parameters* pp*, a partial signing key* sk_i *and a message* μ *and outputs a Round-*1 *contribution* $\mathsf{out}_{1,i}$*;*

- $\mathsf{Combine}_1(\mathsf{pp}, (\mathsf{out}_{1,i_j})_{j\in[t]})$ *is deterministic, takes as inputs the public parameters* pp *and* t *Round-1 contributions* $(\mathsf{out}_{1,i_j})_{j\in[t]}$ *and outputs a Round-2 input message* out_1*;*
- $\mathsf{PartSign}_2(\mathsf{pp}, \mathsf{sk}_i, \mathsf{out}_1, \mu)$ *takes as inputs the public parameters* pp*, a partial signing key* sk_i*, a Round-2 input message* out_1 *and a message* μ *and outputs a Round-2 contribution* $\mathsf{out}_{2,i}$*;*
- $\mathsf{Combine}_2(\mathsf{pp}, (\mathsf{out}_{2,i_j})_{j\in[t]})$ *is deterministic, takes as inputs the public parameters* pp *and* t *Round-2 contributions* $(\mathsf{out}_{2,i_j})_{j\in[t]}$ *and outputs a signature* σ*.*
- $\mathsf{Verify}(\mathsf{vk}, \mu, \sigma)$ *takes as inputs a verification key* vk*, a message* μ*, and a signature* σ *and outputs a bit* $b \in \{0, 1\}$*.*

Correctness. *For* $\varepsilon > 0$*, the scheme is* ε*-correct if for any* μ *and any sets* $\mathcal{S}_1, \mathcal{S}_2 \subseteq [n]$ *with* $|\mathcal{S}_1| = |\mathcal{S}_2| = t$*, we have:*

$$\Pr[\mathsf{Verify}(\mathsf{vk}, \mu, \sigma) = 1] \geq 1 - \varepsilon,$$

where $(\mathsf{pp}, \mathsf{vk}, (\mathsf{sk}_i)_{i\in[n]}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^n, 1^t)$*,* $\mathsf{out}_{1,i} \leftarrow \mathsf{PartSign}_1(\mathsf{pp}, \mathsf{sk}_i, \mu)$ *for all* $i \in \mathcal{S}_1$*,* $\mathsf{out}_1 := \mathsf{Combine}_1(\mathsf{pp}, (\mathsf{out}_{1,i})_{i\in\mathcal{S}_1})$*,* $\mathsf{out}_{2,i} \leftarrow \mathsf{PartSign}_2(\mathsf{pp}, \mathsf{sk}_i, \mathsf{out}_1, \mu)$ *for all* $i \in \mathcal{S}_2$*,* $\sigma := \mathsf{Combine}_2(\mathsf{pp}, (\mathsf{out}_{2,i})_{i\in\mathcal{S}_2})$*, and where the probability is taken over the random coins of* KeyGen*,* $\mathsf{PartSign}_1$*, and* $\mathsf{PartSign}_2$*.*

Compactness. *The scheme achieves compactness if there exists a polynomial* p *such that, for any* μ *and any sets* $\mathcal{S}_1, \mathcal{S}_2 \subseteq [n]$ *with* $|\mathcal{S}_1|, |\mathcal{S}_2| \geq t$*, using the same notation as for correctness, we have that the bit-sizes of* vk *and* σ *are below* $p(\lambda)$*.*

Threshold Unforgeability. *The scheme satisfies threshold unforgeability if for any PPT adversary* $\mathcal{A}$*, the advantage* $\mathsf{Adv}^{\mathsf{thuf}}(\mathcal{A}) := \Pr(\mathsf{Exp}^{\mathsf{thuf}}_{\mathcal{A}}(\lambda, n, t) = 1)$ *is negligible, where experiment* $\mathsf{Exp}^{\mathsf{thuf}}_{\mathcal{A}}$ *is defined in Figure 4.2.*

$\mathsf{Exp}^{\mathsf{thuf}}_{\mathcal{A}}(\lambda, n, t)$**:**

$\mathcal{L} \leftarrow \emptyset$; $\mathcal{W} \leftarrow \emptyset$
$\mathcal{S}_{\mathsf{cor}} \leftarrow \mathscr{A}(1^\lambda)$ with $\mathcal{S}_{\mathsf{cor}} \subseteq [n]$ and $|\mathcal{S}_{\mathsf{cor}}| = t - 1$
$(\mathsf{pp}, \mathsf{vk}, (\mathsf{sk}_i)_{i\in[n]}) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^n, 1^t)$
$(\sigma^*, \mu^*) \leftarrow \mathscr{A}^{\mathsf{OPartSign}_1(\cdot), \mathsf{OPartSign}_2(\cdot)}(\mathsf{pp}, \mathsf{vk}, (\mathsf{sk}_i)_{i\in\mathcal{S}_{\mathsf{cor}}})$
return $\mathsf{Verify}(\mathsf{vk}, \mu^*, \sigma^*) = 1 \wedge (\mu^*, \sigma^*) \notin \mathcal{L}$

$\mathsf{OPartSign}_1(i, \mu)$**:**

if $i \in \mathcal{S}_{\mathsf{cor}}$ **then**
  return $\perp$
$\mathsf{out}_{1,i} \leftarrow \mathsf{PartSign}_1(\mathsf{pp}, \mathsf{sk}_i, \mu)$
for $j \in \mathcal{S}_{\mathsf{cor}}$: $\mathsf{out}_{1,j} \leftarrow \mathsf{PartSign}_1(\mathsf{pp}, \mathsf{sk}_j, \mu)$
$\mathsf{out}_1 \leftarrow \mathsf{Combine}_1(\mathsf{pp}, (\mathsf{out}_{1,j})_{j\in\mathcal{S}_{\mathsf{cor}}\cup\{i\}})$
$\mathcal{W} \leftarrow \mathcal{W} \cup \{(\mu, \mathsf{out}_1)\}$
return $\mathsf{out}_{1,i}$

$\mathsf{OPartSign}_2(i, \mathsf{out}_1, \mu)$**:**

if $i \in \mathcal{S}_{\mathsf{cor}} \vee (\mu, \mathsf{out}_1) \notin \mathcal{W}$ **then return** $\perp$
$\mathsf{out}_{2,i} \leftarrow \mathsf{PartSign}_2(\mathsf{pp}, \mathsf{sk}_i, \mathsf{out}_1, \mu)$
for $j \in \mathcal{S}_{\mathsf{cor}}$:
  $\mathsf{out}_{2,j} \leftarrow \mathsf{PartSign}_2(\mathsf{pp}, \mathsf{sk}_j, \mathsf{out}_1, \mu)$
$\sigma \leftarrow \mathsf{Combine}_2(\mathsf{pp}, (\mathsf{out}_{2,j})_{j\in\mathcal{S}_{\mathsf{cor}}\cup\{i\}})$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mu, \sigma)\}$
return $\mathsf{out}_{2,i}$

Figure 4.2: Security experiment $\mathsf{Exp}^{\mathsf{thuf}}_{\mathcal{A}}$.

We focus on the semi-honest model for our construction. Combining it with non-interactive zero-knowledge proofs (of honest ThFHE partial decryption) allows us to achieve malicious security.

4.4.2 Our protocol

In this section, we describe how we combine Dilithium with ThFHE in order to obtain an efficient 2-round threshold signature scheme. In particular, we explain the tweaks we make to Dilithium to improve the efficiency of the homomorphic computation (without hurting interchangeability). We also analyze it by adapting the analysis from [ASY22] to our 2-round setting. We refer the reader to the full version of this work [PPS26] for details regarding how we efficiently perform the homomorphic computations.

4.4.2.1 Construction

The precise description of our 2-round threshold signature is provided in Figure 4.3. It is mostly the instantiation of the above high-level description and tweaks. We explain them and their impact at the end of this section.

Key Generation. In Figure 4.3, we assume a trusted dealer. The key generation algorithm samples FHE keys and distributes the decryption key to the n parties. It also samples signature keys and encrypts the signing key (which includes the PRF key used to deterministically derive randomness) using the FHE scheme. The verification key is exactly that of the underlying signature.

Distributed Sign. To sign a message μ, the signers homomorphically evaluate the first part of the signature scheme (Figure 4.1, Steps 1–8) using the encrypted signing key, and partially decrypt the resulting ciphertext encrypting $(\mathbf{w}_1^{(i)})_{i\in[L]}$. Assuming that t distinct signers participate, their partial decryption shares can be recombined to recover $(\mathbf{w}_1^{(i)})_{i\in[L]}$, and the challenges $(c^{(i)})_{i\in[L]}$ can be computed in the clear (Figure 4.1, Steps 9 and 10). Finally, at least t users homomorphically evaluate the second part of the signature scheme (Figure 4.1, Steps 11–19) and partially decrypt the resulting ciphertext. Again, assuming that t distinct signers participate, their partial decryption shares can be recombined: this time, the result is a signature. More precisely, we consider the following signing protocol with 2 rounds of communication.

Round 1. For a message μ to be signed, the signers homomorphically evaluate Steps 1–8 of Figure 4.1, L times in parallel, using the encrypted signing key, and partially decrypt the resulting ciphertext. They broadcast their decryption shares.

Round 2. After receiving the decryption shares, each signer runs $\mathsf{Combine}_1$, which consists in evaluating FinDec on t distinct partial decryptions to recover $(\mathbf{w}_1^{(i)})_{i\in[L]}$. This allows signers to compute (in the clear and locally) the challenges $\tilde{c}^{(i)} := \mathsf{H}(\mu \parallel \mathbf{w}_1^{(i)})$, therefore avoiding evaluating the hash function homomorphically. The signers then homomorphically run Steps 11–19 of Figure 4.1 to compute a ciphertext encrypting $(\mathbf{z}^{(i)})_{i\in[L]}$, where only the first valid signature is kept, and partially decrypt it. They broadcast their decryption shares.

Finalize. After receiving the Round-2 decryption shares, the signers recombine t distinct partial shares with FinDec to recover the unique $\mathbf{z}^{(i)} \neq \perp$ (assuming signing is successful), and $(\mathbf{z}^{(i)}, \mathbf{h}^{(i)}, \tilde{c}^{(i)})$ is returned by $\mathsf{Combine}_2$ as the signature of μ.

As can be observed in Figure 4.3, we slightly modify the steps of the original Dilithium signing algorithm (Figure 4.1). This is motivated by performance gain: we modify the sampling of $\mathbf{y}$ (Figure 4.1, Step 6), we modify the rejection sampling based on $\mathbf{r}_0$ (Figure 4.1,

$\mathsf{KeyGen}(1^\lambda, 1^n, 1^t)$

1 $(\mathsf{vk},\ \mathsf{sk}) \leftarrow \mathsf{Dilithium.KeyGen}$
2 parse vk as $(\rho, \mathbf{t}_1)$
3 parse sk as $(\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0)$
4 $k_{\mathsf{PRF}} \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$
5 $(\mathsf{pk}, \mathsf{ek}, \mathsf{fsk}) \leftarrow \mathsf{FhKeyGen}(1^\lambda)$
6 $(\mathsf{sk}_i)_{i\in[n]} \leftarrow \mathsf{Share}(\mathsf{fsk}, n, t)$
7 $[\![\mathbf{s}_1]\!] \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{s}_1)$
8 $[\![\mathbf{s}_2]\!] \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{s}_2)$
9 $[\![k_{\mathsf{PRF}}]\!] \leftarrow \mathsf{Enc}(\mathsf{pk}, k_{\mathsf{PRF}})$
10 $\mathsf{pp} \leftarrow (\mathsf{pk}, \mathsf{ek}, [\![\mathbf{s}_1]\!], [\![\mathbf{s}_2]\!], [\![k_{\mathsf{PRF}}]\!], \rho, tr, \mathbf{t}_0, \mathbf{t}_1)$
11 **return** $(\mathsf{pp}, \mathsf{vk}, (\mathsf{sk}_i)_{i\in[n]})$

$\mathsf{PartSign}_1(\mathsf{pp}, \mathsf{sk}_i, M)$

1 $\mathbf{A} \leftarrow \mathsf{ExpandA}(\rho)$
2 $\mu \leftarrow \mathsf{H}(tr \parallel M)$
3 **for** $k \in [L]$:
4 $\quad [\![\mathbf{y}^{(k)}]\!] \leftarrow \left\lfloor \frac{1}{8}\left(\mathsf{H}(\mu \parallel k) \cdot [\![k_{\mathsf{PRF}}]\!] \bmod 64\right)\right\rceil$
5 $\quad [\![\mathbf{w}^{(k)}]\!] \leftarrow \mathbf{A}[\![\mathbf{y}^{(k)}]\!] \bmod q$
6 $\quad [\![\mathbf{w}_1^{(k)}]\!] \leftarrow \mathsf{HighBits}([\![\mathbf{w}^{(k)}]\!], 2\gamma_2)$
7 $\quad [\![\mathbf{r}_0'^{(k)}]\!] \leftarrow \mathsf{LowBits}_q([\![\mathbf{w}^{(k)}]\!], 2\gamma_2)$
8 $[\![\mathbf{r}_0']\!] \leftarrow [\![(\mathbf{r}_0'^{(1)}, \ldots, \mathbf{r}_0'^{(L)})]\!]$
9 $[\![\mathbf{w}_1]\!] \leftarrow [\![(\mathbf{w}_1^{(1)}, \ldots, \mathbf{w}_1^{(L)})]\!]$
10 $p_{\mathbf{w}_1,i} \leftarrow \mathsf{PartDec}(\mathsf{sk}_i, [\![\mathbf{w}_1]\!])$
11 **return** $\mathsf{out}_{1,i} = ([\![\mathbf{r}_0']\!], [\![\mathbf{y}]\!], [\![\mathbf{w}_1]\!], p_{\mathbf{w}_1,i})$

$\mathsf{Combine}_1(\mathsf{pp}, (\mathsf{out}_{1,j})_{j\in\mathcal{S}})$

1 **for** $j \in \mathcal{S}$:
2 $\quad$ parse $\mathsf{out}_{1,j}$ as $([\![\mathbf{r}_0']\!], [\![\mathbf{y}]\!], [\![\mathbf{w}_1]\!], p_{\mathbf{w}_1,j})$
3 $(\mathbf{w}_1^{(k)})_{k\in[L]} \leftarrow \mathsf{FinDec}([\![\mathbf{w}_1]\!], (p_{\mathbf{w}_1,j})_{j\in\mathcal{S}})$
4 **return** $\mathsf{out}_1 = ([\![\mathbf{r}_0']\!], [\![\mathbf{y}]\!], (\mathbf{w}_1^{(k)})_{k\in[L]})$

$\mathsf{PartSign}_2(\mathsf{pp}, \mathsf{sk}_i, \mathsf{out}_1, \mu)$

1 parse out_1 as $([\![\mathbf{r}_0']\!], [\![\mathbf{y}]\!], (\mathbf{w}_1^{(k)})_{k\in[L]})$
2 **for** $k \in [L]$:
3 $\quad \tilde{c}^{(k)} \leftarrow \mathsf{H}(\mu \parallel \mathbf{w}_1^{(k)})$
4 $\quad c^{(k)} \leftarrow \mathsf{SampleInBall}(\tilde{c}^{(k)})$
5 $\quad [\![\mathbf{z}^{(k)}]\!] \leftarrow [\![\mathbf{y}^{(k)}]\!] + c^{(k)}[\![\mathbf{s}_1]\!]$
6 $\quad$ **if** $\|[\![\mathbf{z}^{(k)}]\!]\|_\infty \geq \gamma_1 - \beta \ \vee\ \|[\![\mathbf{r}_0'^{(k)}]\!] - c^{(k)}[\![\mathbf{s}_2]\!]\|_\infty \geq \gamma_2 - \beta$ **then**
7 $\quad\quad [\![\mathbf{z}^{(k)}]\!] \leftarrow [\![\bot]\!]$
8 $\quad$ **else if** $\|c^{(k)}\mathbf{t}_0\|_\infty \geq \gamma_2$ **then**
9 $\quad\quad [\![\mathbf{z}^{(k)}]\!] \leftarrow [\![\bot]\!]$
10 set $[\![k_0]\!]$ as the first k with $\mathbf{z}_k \neq \bot$; if none, set $[\![k_0]\!] = \varnothing$
11 **for** $k \in [L] \setminus \{k_0\}$:
12 $\quad [\![\mathbf{z}^{(k)}]\!] \leftarrow [\![\bot]\!]$
13 $[\![\mathbf{z}]\!] \leftarrow [\![(\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(L)})]\!]$
14 $p_{\mathbf{z},i} \leftarrow \mathsf{PartDec}(\mathsf{sk}_i, [\![\mathbf{z}]\!])$
15 **return** $\mathsf{out}_{2,i} = ([\![\mathbf{z}]\!], (\tilde{c}^{(k)})_{k\in[L]}, (c^{(k)})_{k\in[L]}, p_{\mathbf{z},i})$

$\mathsf{Combine}_2(\mathsf{pp}, (\mathsf{out}_{2,j})_{j\in\mathcal{S}})$

1 **for** $j \in \mathcal{S}$:
2 $\quad$ parse $\mathsf{out}_{2,j}$ as $([\![\mathbf{z}]\!], (\tilde{c}^{(k)})_{k\in[L]}, (c^{(k)})_{k\in[L]}, p_{\mathbf{z},j})$
3 $(\mathbf{z}_k)_{k\in[L]} \leftarrow \mathsf{FinDec}([\![\mathbf{z}]\!], (p_{\mathbf{z},j})_{j\in\mathcal{S}})$
4 **if** $\exists k \in [L] : \mathbf{z}^{(k)} \neq \bot$ **then**
5 $\quad \sigma = (\mathbf{z}^{(k)}, c^{(k)})$
6 $\quad \mathbf{h} \leftarrow \mathsf{MakeHint}_q(-c^{(k)}\mathbf{t}_0,\ \mathbf{A}\mathbf{z}_k - c^{(k)}\mathbf{t} + c^{(k)}\mathbf{t}_0,\ 2\gamma_2)$
7 $\quad$ **if** $\mathsf{HW}(\mathbf{h}) \leq \omega$ **then**
8 $\quad\quad$ **return** $(\mathbf{z}^{(k)}, \mathbf{h}, \tilde{c}^{(k)})$
9 **else return** $\bot$

$\mathsf{Verify}(\mathsf{vk}, M, \sigma)$

1 **return** $\mathsf{Dilithium.Verify}(\mathsf{vk}, M, \sigma)$

Figure 4.3: The 2-Round Threshold Dilithium signature protocol. The steps in blue involve homomorphic computations.

Steps 12–14), and we postpone the computation of MakeHint (Figure 4.1, Step 16). Let us now explain these changes in more detail.

Sampling y's. Using ExpandMask as a pseudo-random function (PRF) for generating $\mathbf{y}$ (Figure 4.1, Step 6) is not efficient when computed in the encrypted state. We replace this PRF with a ring variant of the Darkmatter PRF [BIP+18]. For a given PRF key k_{PRF} and $m(X) \in \mathcal{R}_{64}^{\mathsf{FHE}} = \mathbb{Z}_{64}[X]/(X^N + 1)$, it outputs

$$\left\lfloor \frac{1}{8}\left(m(X) \cdot k_{\mathsf{PRF}}(X) \bmod 64\right)\right\rceil \ , \tag{4.1}$$

which consists of $3N$ random bits. For $N = 2^{16}$, this instantiation of the learning-with-rounding Darkmatter PRF achieves far more than 128 bits of security against the best known attacks (this may be checked using the lattice estimator [APS15]). This change is reflected in Step 4 of $\mathsf{PartSign}_1$ in Figure 4.3.

Rejecting z's and $\mathbf{r}_0$'s. The original rejection sampling for $\mathbf{z}$ (Figure 4.1, Steps 12–14) requires a high-precision comparison, which is inefficient when computed in the encrypted state. We do not change this step. We are able to perform this comparison by leveraging a small-precision encrypted comparison and using the binary decomposition of $\mathbf{y}$. For the

comparison involving $\mathbf{r}_0$ (Figure 4.1, Step 13), we instead compute $[\![\mathbf{r}'_0]\!] := \mathsf{LowBits}_q([\![\mathbf{w}]\!])$. We emphasize that this does not have any impact, as we have:

$$\|\mathsf{LowBits}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)\|_\infty \geq \gamma_2 - \beta \iff \|\mathsf{LowBits}_q(\mathbf{w}, 2\gamma_2) - c\mathbf{s}_2\|_\infty \geq \gamma_2 - \beta \ .$$

The second test is actually the test run in the publicly available NIST submission implementation of Dilithium, while the first test is the test described in the specification: both tests have identical responses. These changes are reflected in Step 6 of $\mathsf{PartSign}_2$ in Figure 4.3. The computation of $[\![\mathbf{r}'_0]\!]$ is performed directly after the computation of $[\![\mathbf{w}]\!]$ in Step 7 of $\mathsf{PartSign}_1$.

Rejecting Hints. The rejection sampling concerning $\mathsf{MakeHint}$ (Figure 4.1, Step 17) could also generate a computation overhead during homomorphic signing. As observed in [BCE+26], the rejection test based on $\mathbf{h}$ very rarely fails in Dilithium (it rejects $\approx 2\%$ of the signatures that would pass without it). Moreover, this test does not play any role in security, since $\mathbf{h}$ is only there to allow compression of the verification key. We postpone this test to first generate a pre-signature $\mathbf{z}$ that passes all other tests, decrypt $\mathbf{z}$ and then see whether it passes this test (in the clear). Note that the hint vector $\mathbf{h}$ is computable from $\mathbf{z}, \mathbf{w}, \mathbf{t}$ and c via the relation:

$$\mathbf{w} - c\mathbf{s}_2 = \mathbf{A}\mathbf{z} - c\mathbf{t} \ . \tag{4.2}$$

As a result, the signers do not even have to compute $\mathbf{h}$ homomorphically: they can compute it in the clear after decryption of $\mathbf{z}$ and check if it passes the test or not. In 98% of cases, the hint vector $\mathbf{h}$ passes the test and the signers can return $(\mathbf{z}, \mathbf{h}, \tilde{c})$. In the remaining 2% of cases, they have to restart the protocol. As the signing protocol is deterministic, signing a message M is attempted by first signing $M \| \star \| 0$ for a special symbol $\star$. Then, despite determinism, one can attempt to sign M again in case of failure using input $M \| \star \| 1$, etc.

1-Round + Pre-Processing Variant. We note that the start of the signing protocol up to the first round of communication (included) can be performed offline as the dependency on the message can be removed. This may be preprocessed under the assumption that the signers agree on the input to feed to the PRF. By default, for simplicity, we use the message, but it could be any unique identifier that the signers agree on. That identifier can be the message number, the time or other contextual data. This allows us to modify our construction to a 1-round threshold signature with preprocessing. We further remark that the homomorphic computations (in blue in Figure 4.3) are deterministic and only involve public information, so they can be outsourced to a server; only partial decryptions have to be performed by signers.

Distributed Key Generation. Above and in Figure 4.3, we describe our key generation algorithm for the threshold signature assuming a trusted dealer. Note however that one can remove such an assumption as soon as we have a distributed key generation (DKG) protocol for the ThFHE scheme, e.g., the one described in Chapter 3 (Sec. 3.4.2). Equipped with the latter, one can run the DKG protocol to distribute $(\mathsf{pk}, \mathsf{ek}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ and have each party encrypt randomness to homomorphically generate a seed, which then serves to homomorphically run $\mathsf{Dilithium.KeyGen}$ in order to produce $[\![\mathbf{s}_1]\!], [\![\mathbf{s}_2]\!], [\![k_{\mathsf{PRF}}]\!]$, and vk. By default, recovering vk would require an extra round, but it can be performed during our DKG execution without adding a communication round.

4.4.2.2 Analysis

Correctness is inherited from the correctness of Dilithium and of the underlying FHE scheme. Compactness holds as the verification key and the signatures returned by our scheme are precisely the same as those of Dilithium, therefore also guaranteeing verification interchangeability. Further, we prove the following security claim.

Theorem 4.4.2. *Let q_{sig} denote a bound on the number of signing queries made by an attacker, N denote the ring degree, and B_{eval} denote a bound on the homomorphic computation error of the ciphertexts that are decrypted. Then, assuming the underlying FHE scheme is perfectly correct and* IND-CPA*-secure, assuming the threshold flooding parameter η satisfies $\eta \geq B_{\mathsf{eval}} \cdot \sqrt{10 \cdot N \cdot q_{\mathsf{sig}}}$, and assuming security of Dilithium given oracle access to aborted transcripts and with the Darkmatter PRF instantiation of Equation 4.1, the threshold signature scheme described in Figure 4.3 satisfies threshold unforgeability.*

The proof is adapted from [ASY22] and is detailed in the full version [PPS26]. It relies on the security of Dilithium and of the underlying FHE scheme, as well as on a common heuristic: we assume that Dilithium remains secure when the attacker has oracle access to aborted transcripts (i.e., to commitments $\mathbf{w}_1$ output at the end of Round 1). The proof follows a standard hybrid argument, and uses a simulation based on Rényi divergence, which helps reducing the amount of flooding. This is possible as threshold unforgeability is a search-based security game, hence we can benefit from classical Rényi divergence arguments. Thanks to the use of Rényi divergence, one can reduce the magnitude of the error added during partial decryption by a significant margin. In the full version [PPS26], we prove that adding only 49 bits of error (instead of the 128-bit errors $e_{\mathsf{fl},i}$ used in Chapter 3) during partial decryption is sufficient to guarantee 128-bit security of our threshold variant of Dilithium.

Chapter 5

Conclusion and Perspectives

This manuscript describes contributions to Lyubashevsky's signatures and threshold fully homomorphic encryption. These results enable instantiations of these primitives with various trade-offs and lead to a wide variety of schemes. All constructions are supported by rigorous analyses that address issues from prior analyses. The contributions regarding threshold FHE substantially improve the performance of the primitive in the asynchronous setting, and provide a security model and a secure instantiation in the synchronous one. Notably, they yield instantiations of threshold-CKKS with efficient distributed key generation and threshold decryption. The latter can be used to thresholdize a broad range of cryptographic primitives using the universal thresholdizer paradigm, and we illustrate it with a threshold variant of Dilithium/ML-DSA.

Several important challenges remain in these areas.

Threshold Lattice-Based Signatures. As mentioned in Chapter 4, threshold lattice-based signatures, and in particular threshold variants of Dilithium/ML-DSA, are in high demand. While several solutions have been proposed recently [DKLS25; BCE+26; CPE+26; PPS26], the ideal protocol would combine low computational cost, few rounds, low communication, scalability to large numbers of parties n and thresholds t, and malicious (active) security. So far, no existing approach combines all of these properties simultaneously. Our approach addresses most of these challenges, except for computational cost (which can be mitigated via preprocessing or secure outsourcing) and full malicious security (which can be obtained using generic techniques, though at some non-negligible cost). This naturally motivates the main research direction I intend to pursue in the near future.

Zero-Knowledge Proofs for Threshold FHE. FHE, and threshold FHE in particular, is a powerful tool: among other applications, it enables delegation of computation, secure multiparty computation, and privacy-preserving machine learning. However, threshold FHE protocols often consider only semi-honest adversaries. Achieving practical security against malicious adversaries is therefore a critical step towards deployment, and I aim to investigate this topic over the next few years.

This objective involves several intermediate problems, since threshold FHE features multiple roles:

(i) senders (encryptors), who provide encrypted inputs;
(ii) an evaluator, who runs the homomorphic computation;
(iii) receivers (decryptors), who run the distributed decryption.

In adversarial settings, each role can deviate from the protocol in ways that compromise correctness, privacy, or liveness. Depending on the application, one may therefore require:

(i) proofs that ciphertexts are well-formed (honest encryption);
(ii) proofs that the evaluator performs the specified computation correctly (verifiable evaluation);
(iii) proofs that partial decryption shares are honestly generated (verifiable/robust threshold decryption).

Additional efficiency requirements may also be essential. For instance, verifying the evaluator's proof should be significantly cheaper than re-running the computation.

While promising approaches exist, closing the gap to solutions that are efficient and deployable for threshold FHE remains a major challenge, and I plan to pursue this direction in the coming years.

Personal Publications

- **THED: Threshold Dilithium from FHE**
  with Jai Hyun Park and Damien Stehlé
  Manuscript
- **On Threshold Fully Homomorphic Encryption with Synchronized Decryptors**
  with François Colin de Verdière and Damien Stehlé
  Manuscript
- **Distributed Key Generation for Efficient Threshold-CKKS**
  with Seonhong Min, Guillaume Hanrot, Jai Hyun Park, and Damien Stehlé
  Manuscript
- **Asynchronous Lagrange-Based Threshold FHE with Smaller Modulus Overhead**
  with Won Kim, Changmin Lee, Jeonghwan Lee, and Damien Stehlé
  CRYPTO 2026
- **Low Communication Threshold Fully Homomorphic Encryption**
  with Damien Stehlé
  ASIACRYPT 2024
- **Attacks Against the IND-CPA-D Security of Exact FHE Schemes**
  with Jung Hee Cheon, Hyeongmin Choe, Damien Stehlé, and Elias Suvanto
  CCS 2024
- **Fast Public-Key Silent OT and More from Constrained Naor-Reingold**
  with Dung Bui, Geoffroy Couteau, Pierre Meyer, and Mahshid Riahinia
  EUROCRYPT 2024
- **Efficient Updatable Public-Key Encryption from Lattices**
  with Calvin Abou Haidar and Damien Stehlé
  ASIACRYPT 2023
- **G+G: A Fiat-Shamir Lattice Signature Based on Convolved Gaussians**
  with Julien Devevey and Damien Stehlé
  ASIACRYPT 2023
- **A Detailed Analysis of Fiat-Shamir with Aborts**
  with Julien Devevey, Pouria Fallahpour, Damien Stehlé, and Keita Xagawa
  Manuscript, extended version of [DFPS23] (*CRYPTO 2023*)
- **Constrained Pseudorandom Functions from Homomorphic Secret Sharing**
  with Geoffroy Couteau, Pierre Meyer, and Mahshid Riahinia
  EUROCRYPT 2023
- **PointProofs, Revisited**
  with Benoît Libert and Mahshid Riahinia
  ASIACRYPT 2022
- **On Rejection Sampling in Lyubashevsky's Signature Scheme**
  with Julien Devevey, Omar Fawzi, and Damien Stehlé
  ASIACRYPT 2022

- **Cumulatively All-Lossy-But-One Trapdoor Functions from Standard Assumptions**
  with Benoît Libert and Ky Nguyen
  SCN 2022
- **New and Improved Constructions for Partially Equivocable Public Key Encryption**
  with Benoît Libert and Mahshid Riahinia
  SCN 2022
- **Updatable Public Key Encryption from DCR: Efficient Constructions with Stronger Security**
  with Calvin Abou Haidar and Benoît Libert
  CCS 2022
- **Alternative Constructions of Asymmetric Primitives from Obfuscation: Hierarchical IBE, Predicate Encryption, and More**
  with Pooya Farshim and Georg Fuchsbauer
  INDOCRYPT 2020
- **Simulation-Sound Proofs for LWE and Applications to KDM-CCA2 Security**
  with Benoît Libert, Khoa Nguyen, and Radu Ţiţiu
  ASIACRYPT 2020
- **New Constructions of Statistical NIZKs: Dual-Mode DV-NIZKs and More**
  with Benoît Libert, David J. Wu, and Hoeteck Wee
  EUROCRYPT 2020
- **Algebraic XOR-RKA-Secure Pseudorandom Functions from Post-Zeroizing Multilinear Maps**
  with Michel Abdalla and Fabrice Benhamouda
  ASIACRYPT 2019
- **Unifying Leakage Models on a Rényi Day**
  with Dahmun Goudarzi, Ange Martinelli, and Thomas Prest
  CRYPTO 2019
- **Exploring Crypto Dark Matter: New Simple PRF Candidates and Their Applications**
  with Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu
  TCC 2018
- **Non-Trivial Witness Encryption and Null-iO from Standard Assumptions**
  with Zvika Brakerski, Aayush Jain, Ilan Komargodski, and Daniel Wichs
  SCN 2018
- **Function-Revealing Encryption**
  with Marc Joye
  SCN 2018
- **Private Multiplication over Finite Fields**
  with Sonia Belaïd, Fabrice Benhamouda, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud
  CRYPTO 2017
- **From Cryptomania to Obfustopia through Secret-Key Functional Encryption**
  with Nir Bitansky, Ryo Nishimaki, and Daniel Wichs
  TCC 2016-B, Journal of Cryptology 2020

- **Randomness Complexity of Private Circuits for Multiplication**
  with Sonia Belaïd, Fabrice Benhamouda, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud
  EUROCRYPT 2016
- **Multilinear and Aggregate Pseudorandom Functions: New Constructions and Improved Security**
  with Michel Abdalla and Fabrice Benhamouda
  ASIACRYPT 2015
- **An Algebraic Framework for Pseudorandom Functions and Applications to Related-Key Security}** with Michel Abdalla and Fabrice Benhamouda
  CRYPTO 2015
- **Related-Key Security for Pseudorandom Functions Beyond the Linear Barrier**
  with Michel Abdalla, Fabrice Benhamouda, and Kenneth G. Paterson
  CRYPTO 2014, Journal of Cryptology 2018

Bibliography

[AAC+22] Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, and Daniel Smith-Tone. *Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process.* Tech. rep. NIST, 2022 (cit. on p. 44).

[ABC+23] Melissa Azouaoui, Olivier Bronchain, Gaëtan Cassiers, Clément Hoffmann, Yulia Kuzovkova, Joost Renes, Tobias Schneider, Markus Schönauer, François-Xavier Standaert, and Christine van Vredendaal. "Protecting Dilithium against Leakage - Revisited Sensitivity Analysis and Improved Implementations". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* (2023) (cit. on p. 48).

[ABP15a] Michel Abdalla, Fabrice Benhamouda, and Alain Passelègue. "An Algebraic Framework for Pseudorandom Functions and Applications to Related-Key Security". In: *CRYPTO*. 2015 (cit. on p. 1).

[ABP15b] Michel Abdalla, Fabrice Benhamouda, and Alain Passelègue. "Multilinear and Aggregate Pseudorandom Functions: New Constructions and Improved Security". In: *ASIACRYPT*. 2015 (cit. on p. 1).

[ABP19] Michel Abdalla, Fabrice Benhamouda, and Alain Passelègue. "Algebraic XOR-RKA-Secure Pseudorandom Functions from Post-Zeroizing Multilinear Maps". In: *ASIACRYPT*. 2019 (cit. on p. 2).

[ABPP14] Michel Abdalla, Fabrice Benhamouda, Alain Passelègue, and Kenneth G. Paterson. "Related-Key Security for Pseudorandom Functions Beyond the Linear Barrier". In: *CRYPTO*. 2014 (cit. on p. 1).

[ADP24] Nabil Alkeilani Alkadri, Nico Döttling, and Sihang Pu. "Practical Lattice-Based Distributed Signatures for a Small Number of Signers". In: *ACNS*. 2024 (cit. on p. 44).

[AFLT16] Michel Abdalla, Pierre-Alain Fouque, Vadim Lyubashevsky, and Mehdi Tibouchi. "Tightly Secure Signatures From Lossy Identification Schemes". In: *J. Cryptol.* (2016) (cit. on pp. 8, 13, 14, 16, 19).

[AL21] Martin R. Albrecht and Russel W. F. Lai. "Subtractive sets over cyclotomic rings: Limits of Schnorr-like arguments over lattices". In: *CRYPTO*. 2021 (cit. on p. 35).

[ALP22] Calvin Abou Haidar, Benoît Libert, and Alain Passelègue. "Updatable Public Key Encryption from DCR: Efficient Constructions With Stronger Security". In: *ACM CCS*. 2022 (cit. on p. 4).

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. "On the concrete hardness of Learning with Errors". In: *J. Math. Cryptol.* (2015). Software available at `https://github.com/malb/lattice-estimator` (cit. on p. 51).

[APS23] Calvin Abou Haidar, Alain Passelègue, and Damien Stehlé. "Efficient Updatable Public-Key Encryption from Lattices". In: *ASIACRYPT*. 2023 (cit. on pp. 4, 5).

[ASY22] Shweta Agrawal, Damien Stehlé, and Anshu Yadav. "Round-Optimal Lattice-Based Threshold Signatures, Revisited". In: *ICALP*. 2022 (cit. on pp. 9, 11, 12, 44, 47, 48, 50, 53).

[BBN+25] Dan Boneh, Benedikt Bünz, Kartik Nayak, Lior Rotem, and Victor Shoup. "Context-Dependent Threshold Decryption and its Applications". In: *ASIACRYPT*. 2025 (cit. on p. 26).

[BBP+16] Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. "Randomness Complexity of Private Circuits for Multiplication". In: *EUROCRYPT*. 2016 (cit. on p. 2).

[BBP+17] Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. "Private Multiplication over Finite Fields". In: *CRYPTO*. 2017 (cit. on p. 2).

[BC10] Mihir Bellare and David Cash. "Pseudorandom Functions and Permutations Provably Secure against Related-Key Attacks". In: *CRYPTO*. 2010 (cit. on p. 1).

[BCE+26] Alexander Bienstock, Leo de Castro, Daniel Escudero, Antigoni Polychroniadou, and Akira Takahashi. *Quorus: Efficient, Scalable Threshold ML-DSA Signatures from MPC*. USENIX Security. 2026 (cit. on pp. 44, 45, 52, 55).

[BCK+23] Youngjin Bae, Jung Hee Cheon, Jaehyung Kim, Jai Hyun Park, and Damien Stehlé. "HERMES: Efficient Ring Packing Using MLWE Ciphertexts and Application to Transciphering". In: *CRYPTO*. 2023 (cit. on p. 38).

[BCM+24] Dung Bui, Geoffroy Couteau, Pierre Meyer, Alain Passelègue, and Mahshid Riahinia. "Fast Public-Key Silent OT and More from Constrained Naor-Reingold". In: *EUROCRYPT*. 2024 (cit. on p. 5).

[BDF+11] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. "Random Oracles in a Quantum World". In: *ASIACRYPT*. 2011 (cit. on p. 15).

[BDK+18] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. "CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM". In: *EuroS&P*. 2018 (cit. on p. 4).

[BDK+21] Shi Bai, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. *CRYSTALS-Dilithium Algorithm Specifications and Supporting Documentation (Version 3.1)*. 2021 (cit. on pp. 8, 13, 21, 44, 46).

[BF11] Dan Boneh and David Mandell Freeman. "Linearly Homomorphic Signatures over Binary Fields and New Tools for Lattice-Based Signatures". In: *PKC*. 2011 (cit. on p. 20).

[BFM+25] Zvika Brakerski, Offir Friedman, Avichal Marmor, Dolev Mutzari, Yuval Spiizer, and Ni Trieu. *Threshold FHE with Efficient Asynchronous Decryption*. IACR ePrint 2025/712. 2025 (cit. on pp. 33–36).

[BG14] Shi Bai and Steven D. Galbraith. "An Improved Compression Technique for Signatures Based on Learning with Errors". In: *CT-RSA*. 2014 (cit. on pp. 8, 44).

[BGG+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 (cit. on pp. 27, 29, 33, 34, 40, 43–48).

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. "(Leveled) fully homomorphic encryption without bootstrapping". In: *ITCS*. 2012 (cit. on pp. 21, 24).

[BIP+18] Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J. Wu. "Exploring Crypto Dark Matter: New Simple PRF Candidates and Their Applications". In: *TCC*. 2018 (cit. on pp. 4, 5, 47, 51).

[BJK+18] Zvika Brakerski, Aayush Jain, Ilan Komargodski, Alain Passelègue, and Daniel Wichs. "Non-trivial Witness Encryption and Null-iO from Standard Assumptions". In: *SCN*. 2018.

[BKL+25] Cecilia Boschini, Darya Kaviani, Russel W. F. Lai, Giulio Malavolta, Akira Takahashi, and Mehdi Tibouchi. "Ringtail: Practical Two-Round Threshold Signatures from Learning with Errors". In: *S&P*. 2025 (cit. on pp. 44, 47).

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. "Classical hardness of learning with errors". In: *STOC*. 2013 (cit. on p. 38).

[BMM25] Jean-Philippe Bossuat, Christian Mouchet, and Janmajaya Mall. *Compact 1-Round Threshold Multiparty Homomorphic Encryption Setup From Public Aggregatable Transcripts*. Contributed talk in the FHE.org conference, available at https://www.youtube.com/watch?v=9Jsk6b4mFy0. 2025 (cit. on pp. 39, 41).

[BNPW16] Nir Bitansky, Ryo Nishimaki, Alain Passelègue, and Daniel Wichs. "From Cryptomania to Obfustopia Through Secret-Key Functional Encryption". In: *TCC*. 2016 (cit. on p. 3).

[BP25] Luis T. A. N. Brandão and René Peralta. *NIST First Call for Multi-Party Threshold Schemes (second public draft)*. Tech. rep. NIST, 2025 (cit. on p. 44).

[Bra12] Zvika Brakerski. "Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP". In: *CRYPTO*. 2012 (cit. on p. 24).

[BS23] Ward Beullens and Gregor Seiler. "LaBRADOR: Compact Proofs for R1CS from Module-SIS". In: *CRYPTO*. 2023 (cit. on p. 45).

[BTH22] Jean-Philippe Bossuat, Juan Troncoso-Pastoriza, and Jean-Pierre Hubaux. "Bootstrapping for approximate homomorphic encryption with negligible failure-probability by using sparse-secret encapsulation". In: *ACNS*. 2022 (cit. on pp. 39, 41).

[CCK+25] Jung Hee Cheon, Hyeongmin Choe, Minsik Kang, Jaehyung Kim, Seonghak Kim, Johannes Mono, and Taeyeong Noh. "Grafting: Complementing RNS in CKKS". In: *CCS*. 2025 (cit. on p. 40).

[CCP+24] Jung Hee Cheon, Hyeongmin Choe, Alain Passelègue, Damien Stehlé, and Elias Suvanto. "Attacks Against the IND-CPA-D Security of Exact FHE Schemes". In: *CCS*. 2024 (cit. on pp. 5, 6, 39).

[CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. "Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds". In: *ASIACRYPT*. 2016 (cit. on p. 47).

[CGL+24] Jean-Sébastien Coron, François Gérard, Tancrède Lepoint, Matthias Trannoy, and Rina Zeitoun. "Improved High-Order Masked Generation of Masking Vector and Rejection Sampling in Dilithium". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* (2024) (cit. on p. 48).

[CGTZ23] Jean-Sébastien Coron, François Gérard, Matthias Trannoy, and Rina Zeitoun. "Improved Gadgets for the High-Order Masking of Dilithium". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* (2023) (cit. on p. 48).

[Che23] Yilei Chen. "DualMS: Efficient Lattice-Based Two-Round Multi-Signature with Trapdoor-Free Simulation". In: *CRYPTO*. 2023 (cit. on p. 44).

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. "Homomorphic encryption for arithmetic of approximate numbers". In: *ASIACRYPT*. 2017 (cit. on pp. 24, 45).

[CKL24] Wonhee Cho, Jiseung Kim, and Changmin Lee. *(In)Security of Threshold Fully Homomorphic Encryption based on Shamir Secret Sharing*. IACR ePrint 2024/1858. 2024 (cit. on p. 33).

[CLW25] Valerio Cini, Russel W. F. Lai, and Ivy K. Y. Woo. "Pilvi: Lattice Threshold PKE with Small Decryption Shares and Improved Security". In: *ASIACRYPT*. 2025 (cit. on pp. 33–36).

[CMPR23] Geoffroy Couteau, Pierre Meyer, Alain Passelègue, and Mahshid Riahinia. "Constrained Pseudorandom Functions from Homomorphic Secret Sharing". In: *EUROCRYPT*. 2023 (cit. on p. 4).

[CPE+26] Sofia Celi, Rafael del Pino, Thomas Espitau, Guilhem Niot, and Thomas Prest. *Efficient Threshold ML-DSA*. USENIX Security. 2026 (cit. on pp. 44, 45, 55).

[CPS26] François Colin de Verdière, Alain Passelègue, and Damien Stehlé. *On Threshold Fully Homomorphic Encryption with Synchronized Decryptors*. IACR ePrint 2026/031. 2026 (cit. on pp. 7, 26, 27).

[CS19] Daniele Cozzo and Nigel P. Smart. "Sharing the LUOV: Threshold Post-Quantum Signatures". In: *IMACC*. 2019 (cit. on p. 44).

[CTZ24] Rutchathon Chairattana-Apirom, Stefano Tessaro, and Chenzhi Zhu. "Partially Non-interactive Two-Round Lattice-Based Threshold Signatures". In: *ASIACRYPT*. 2024 (cit. on pp. 44, 47).

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. "Unifying Leakage Models: From Probing Attacks to Noisy Leakage". In: *EUROCRYPT*. 2014 (cit. on p. 2).

[DDLL13] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. "Lattice signatures and bimodal Gaussians". In: *CRYPTO*. 2013 (cit. on pp. 8–10, 20, 21).

[Dev86] Luc Devroye. *Non-Uniform random variate generation*. 1986 (cit. on p. 9).

[DF89] Yvo Desmedt and Yair Frankel. "Threshold cryptosystems". In: *CRYPTO*. 1989 (cit. on p. 44).

[DFMS19] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. "Security of the Fiat-Shamir Transformation in the Quantum Random-Oracle Model". In: *CRYPTO*. 2019 (cit. on p. 14).

[DFP+23] Julien Devevey, Pouria Fallahpour, Alain Passelègue, Damien Stehlé, and Keita Xagawa. *A Detailed Analysis of Fiat-Shamir with Aborts*. IACR ePrint 2023/245. 2023 (cit. on p. 12).

[DFPS22] Julien Devevey, Omar Fawzi, Alain Passelègue, and Damien Stehlé. "On Rejection Sampling in Lyubashevsky's Signature Scheme". In: *ASIACRYPT*. 2022 (cit. on pp. 9, 21).

[DFPS23] Julien Devevey, Pouria Fallahpour, Alain Passelègue, and Damien Stehlé. "A Detailed Analysis of Fiat-Shamir with Aborts". In: *CRYPTO*. 2023 (cit. on pp. 6, 7, 57).

[DKL+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: *TCHES* (2018) (cit. on pp. 8, 44).

[DKLS25] Antonin Dufka, Semjon Kravtsenko, Peeter Laud, and Nikita Snetkov. *Trilithium: Efficient and Universally Composable Distributed ML-DSA Signing*. IACR ePrint 2025/675. 2025 (cit. on pp. 44, 45, 55).

[DM15] Léo Ducas and Daniele Micciancio. "FHEW: bootstrapping homomorphic encryption in less than a second". In: *EUROCRYPT*. 2015 (cit. on p. 47).

[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: *J. Cryptol.* (2022) (cit. on p. 44).

[DPS23] Julien Devevey, Alain Passelègue, and Damien Stehlé. "G+G: A Fiat-Shamir Lattice Signature Based on Convolved Gaussians". In: *ASIACRYPT*. 2023 (cit. on p. 20).

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. "Multiparty Computation from Somewhat Homomorphic Encryption". In: *CRYPTO*. 2012 (cit. on p. 8).

[Duc14] Léo Ducas. *Accelerating Bliss: the geometry of ternary polynomials*. IACR ePrint 2014/874. 2014 (cit. on p. 20).

[EKT24] Thomas Espitau, Shuichi Katsumata, and Kaoru Takemure. "Two-Round Threshold Signature from Algebraic One-More Learning with Errors". In: *CRYPTO*. 2024 (cit. on pp. 44, 47).

[EY24] Ehsan Ebrahimi and Anshu Yadav. "Strongly Secure Universal Thresholdizer". In: *ASIACRYPT*. 2024 (cit. on p. 29).

[FFP20] Pooya Farshim, Georg Fuchsbauer, and Alain Passelègue. “Simpler Constructions of Asymmetric Primitives from Obfuscation”. In: *INDOCRYPT*. 2020.

[FS86] Amos Fiat and Adi Shamir. “How to Prove Yourself: Practical Solutions to Identification and Signature Problems”. In: *CRYPTO*. 1986 (cit. on p. 8).

[FV12] Junfeng Fan and Frederik Vercauteren. “Somewhat Practical Fully Homomorphic Encryption”. In: (2012). IACR ePrint 2012/144 (cit. on p. 24).

[GHHM21] Alex B. Grilo, Kathrin Hövelmanns, Andreas Hülsing, and Christian Majenz. “Tight Adaptive Reprogramming in the QROM”. In: *ASIACRYPT*. 2021 (cit. on pp. 13, 16–18).

[GHS12] Craig Gentry, Shai Halevi, and Nigel P. Smart. “Homomorphic Evaluation of the AES Circuit”. In: *CRYPTO*. 2012 (cit. on p. 39).

[GKS24] Kamil Doruk Gur, Jonathan Katz, and Tjerand Silde. “Two-round threshold lattice-based signatures from threshold homomorphic encryption”. In: *PQCrypto*. 2024 (cit. on pp. 28, 44, 47).

[GLP15] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. “Lattice-Based Signatures: Optimization and Implementation on Reconfigurable Hardware”. In: *IEEE T. Comput.* 64.7 (2015), pp. 1954–1967 (cit. on p. 8).

[HCLK24] Deokhwa Hong, Youngjin Choi, Yongwoo Lee, and Young-Sik Kim. *Overlapped Bootstrapping for FHEW/TFHE and Its Application to SHA3*. IACR ePrint 2024/1667. 2024 (cit. on p. 47).

[HEaaN] CryptoLab. *HEaaN Library*. Software available at `https://heaan.it/`. 2022 (cit. on pp. 39, 40, 45).

[HLSS25] Intak Hwang, Hyeonbum Lee, Jinyeong Seo, and Yongsoo Song. “Practical Zero-Knowledge PIOP for Maliciously Secure Multiparty Homomorphic Encryption”. In: *CCS*. 2025 (cit. on p. 45).

[JP18] Marc Joye and Alain Passelègue. “Function-Revealing Encryption - Definitions and Constructions”. In: *SCN*. 2018.

[Kat21] Shuichi Katsumata. “A New Simple Technique to Bootstrap Various Lattice Zero-Knowledge Proofs to QROM Secure NIZKs”. In: *CRYPTO*. 2021 (cit. on p. 15).

[KLL+26] Won Kim, Changmin Lee, Jeonghwan Lee, Alain Passelègue, and Damien Stehlé. “Asynchronous Lagrange-Based Threshold FHE with Smaller Modulus Overhead”. In: *CRYPTO*. 2026 (cit. on pp. 26, 32, 34).

[KLS18] E. Kiltz, Vadim Lyubashevsky, and C. Schaffner. “A Concrete Treatment of Fiat-Shamir Signatures in the Quantum Random-Oracle Model”. In: *EUROCRYPT*. 2018 (cit. on pp. 13, 15, 16, 18).

[KLSW24] Hyesun Kwak, Dongwon Lee, Yongsoo Song, and Sameer Wagh. “A General Framework of Homomorphic Encryption for Multiple Parties with Non-interactive Key-Aggregation”. In: *ACNS*. 2024 (cit. on pp. 39, 41).

[KRT24] S. Katsumata, M. Reichle, and K. Takemure. “Adaptively Secure 5 Round Threshold Signatures from MLWE/MSIS and DL with Rewinding”. In: *CRYPTO*. 2024 (cit. on pp. 44, 47).

[Lattigo] *Lattigo v6*. Software available at https://github.com/tuneinsight/lattigo. EPFL-LDS, Tune Insight SA. 2024 (cit. on p. 39).

[LMSS22] Baiyu Li, Daniele Micciancio, Mark Schultz, and Jessica Sorrell. "Securing Approximate Homomorphic Encryption Using Differential Privacy". In: *CRYPTO*. 2022 (cit. on p. 37).

[LMSS23] Changmin Lee, Seonhong Min, Jinyeong Seo, and Yongsoo Song. "Faster TFHE bootstrapping with block binary keys". In: *AsiaCCS*. 2023 (cit. on p. 39).

[LNP22a] Benoît Libert, Ky Nguyen, and Alain Passelègue. "Cumulatively All-Lossy-But-One Trapdoor Functions from Standard Assumptions". In: *SCN*. 2022.

[LNP22b] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. "Lattice-Based Zero-Knowledge Proofs and Applications: Shorter, Simpler, and More General". In: *CRYPTO*. 2022 (cit. on pp. 8, 45).

[LNPT20] Benoît Libert, Khoa Nguyen, Alain Passelègue, and Radu Titiu. "Simulation-Sound Arguments for LWE and Applications to KDM-CCA2 Security". In: *ASIACRYPT*. 2020 (cit. on p. 3).

[LPR22] Benoît Libert, Alain Passelègue, and Mahshid Riahinia. "New and Improved Constructions for Partially Equivocable Public Key Encryption". In: *SCN*. 2022 (cit. on pp. 3, 4).

[LPWW20] Benoît Libert, Alain Passelègue, Hoeteck Wee, and David J. Wu. "New Constructions of Statistical NIZKs: Dual-Mode DV-NIZKs and More". In: *EUROCRYPT*. 2020 (cit. on p. 3).

[LS15] Adeline Langlois and Damien Stehlé. "Worst-case to average-case reductions for module lattices". In: *Des. Codes Cryptogr.* (2015) (cit. on p. 21).

[LSV22] Peeter Laud, Nikita Snetkov, and Jelizaveta Vakarjuk. *DiLizium-2.0: Revisiting Two-Party Crystals-Dilithium*. IACR ePrint 2022/644. 2022 (cit. on p. 44).

[Lyu09] Vadim Lyubashevsky. "Fiat-Shamir with Aborts: Applications to Lattice and Factoring-based Signatures". In: *ASIACRYPT*. 2009 (cit. on pp. 8–10, 14, 18, 44).

[Lyu12] Vadim Lyubashevsky. "Lattice Signatures Without Trapdoors". In: *EUROCRYPT*. 2012 (cit. on pp. 8–11, 13, 14, 18, 44).

[Lyu16] Vadim Lyubashevsky. "Digital Signatures Based on the Hardness of Ideal Lattice Problems in All Rings". In: *ASIACRYPT*. 2016 (cit. on p. 8).

[MBH23] Christian Mouchet, Elliott Bertrand, and Jean-Pierre Hubaux. "An efficient threshold access-structure for RLWE-based multiparty homomorphic encryption". In: *J. Cryptol.* (2023) (cit. on pp. 27, 28, 39, 40).

[MGTF19] Vincent Migliore, Benoît Gérard, Mehdi Tibouchi, and Pierre-Alain Fouque. "Masking Dilithium - Efficient Implementation and Side-Channel Evaluation". In: *ACNS*. 2019 (cit. on p. 48).

[MHP+25] Seonhong Min, Guillaume Hanrot, Jai Hyun Park, Alain Passelègue, and Damien Stehlé. *Distributed Key Generation for Efficient Threshold-CKKS*. IACR ePrint 2025/2057. 2025 (cit. on pp. 26, 39, 46).

[MHWW24] Shihe Ma, Tairong Huang, Anyu Wang, and Xiaoyun Wang. "Accelerating BGV Bootstrapping for Large p Using Null Polynomials over $\mathbb{Z}_{p^e}$". In: *EUROCRYPT*. 2024 (cit. on p. 39).

[MS23] Daniele Micciancio and Adam Suhl. *Simulation-Secure Threshold PKE from LWE with Polynomial Modulus*. IACR ePrint 2023/1728. 2023 (cit. on p. 38).

[MTBH21] Christian Mouchet, Juan Troncoso-Pastoriza, Jean-Philippe Bossuat, and Jean-Pierre Hubaux. "Multiparty homomorphic encryption from ring-learning-with-errors". In: *PETs* (2021) (cit. on pp. 27, 28, 39, 41).

[MW17] Daniele Micciancio and Michael Walter. "Gaussian Sampling over the Integers: efficient, Generic, Constant-Time". In: *CRYPTO*. 2017 (cit. on p. 37).

[NS24] Ngoc Khanh Nguyen and Gregor Seiler. "Greyhound: Fast Polynomial Commitments from Lattices". In: *CRYPTO*. 2024 (cit. on p. 45).

[OpenFHE] Ahmad Al Badawi, Jack Bates, Flávio Bergamaschi, David Bruce Cousins, Saroja Erabelli, Nicholas Genise, Shai Halevi, Hamish Hunt, Andrey Kim, Yongwoo Lee, Zeyu Liu, Daniele Micciancio, Ian Quah, Yuriy Polyakov, R. V. Saraswathy, Kurt Rohloff, Jonathan Saylor, Dmitriy Suponitsky, Matthew Triplett, Vinod Vaikuntanathan, and Vincent Zucca. *OpenFHE: Open-Source Fully Homomorphic Encryption Library*. WAHC. 2022 (cit. on p. 39).

[OT25] Hiroki Okada and Tsuyoshi Takagi. "Low Communication Threshold FHE from Standard (Module-)LWE". In: *ASIACRYPT*. 2025 (cit. on p. 37).

[PGMP19] Thomas Prest, Dahmun Goudarzi, Ange Martinelli, and Alain Passelègue. "Unifying Leakage Models on a Rényi Day". In: *CRYPTO*. 2019 (cit. on p. 2).

[PKM+24] Rafael del Pino, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, and Markku-Juhani Saarinen. "Threshold Raccoon: Practical Threshold Signatures from Standard Lattice Assumptions". In: *EUROCRYPT*. 2024 (cit. on pp. 31, 44, 47).

[PPS26] Jai Hyun Park, Alain Passelègue, and Damien Stehlé. *THED: Threshold Dilithium from FHE*. Manuscript. 2026 (cit. on pp. 47, 50, 53, 55).

[PR13] Emmanuel Prouff and Matthieu Rivain. "Masking against Side-Channel Attacks: A Formal Security Proof". In: *EUROCRYPT*. 2013 (cit. on p. 2).

[PS24] Alain Passelègue and Damien Stehlé. "Low Communication Threshold Fully Homomorphic Encryption". In: *ASIACRYPT*. 2024 (cit. on pp. 7, 26, 37).

[Sch91] Claus-Peter Schnorr. "Efficient Signature Generation by Smart Cards". In: *J. Cryptol.* (1991) (cit. on p. 8).

[SVL24] Nikita Snetkov, Jelizaveta Vakarjuk, and Peeter Laud. "TOPCOAT: towards practical two-party Crystals-Dilithium". In: *Discov. Comput.* (2024) (cit. on p. 44).

[TPCZ23] Guofeng Tang, Bo Pang, Long Chen, and Zhenfeng Zhang. "Efficient Lattice-Based Threshold Signatures with Functional Interchangeability". In: *IEEE Trans. Inf. Forensics Secur.* (2023) (cit. on p. 44).

[VSW21] Jelizaveta Vakarjuk, Nikita Snetkov, and Jan Willemson. "DiLizium: A Two-Party Lattice-Based Signature Scheme". In: *Entropy* (2021) (cit. on p. 44).

[Zha12] Mark Zhandry. “How to Construct Quantum Random Functions”. In: *FOCS*. 2012 (cit. on p. 18).