ABSTRACT

Title of dissertation: ADVANCES IN CONCRETE CRYPTANALYSIS OF LATTICE PROBLEMS AND INTERACTIVE SIGNATURE SCHEMES

Hunter Kippen, Doctor of Philosophy 2024

Dissertation directed by: Associate Professor Dana Dachman-Soled
Electrical and Computer Engineering

Advanced cryptography that goes beyond what is currently deployed to service our basic internet infrastructure is continuing to see widespread adoption. The enhanced functionality achieved by these schemes frequently yields an increase in complexity. Solely considering the asymptotic security of the underlying computational assumptions is often insufficient to realize practical and secure instantiations.

In these cases, determining the risk of any particular deployment involves analyzing the *concrete* security (the exact length of time it would take to break the encryption) as well as quantifying how concrete security can *degrade* over time due to any exploitable information leakage.

In this dissertation, we examine two such cryptographic primitives where assessing concrete security is paramount. First, we consider the cryptanalysis of lattice problems (used as the basis for current standard quantum resistant cryptosystems). We develop a novel side-channel attack on the FrodoKEM key encapsulation mechanism as submitted to the NIST Post Quantum Cryptography (PQC) standardization

process. Our attack involves *poisoning* the FrodoKEM Key Generation (KeyGen) process using a security exploit in DRAM known as "Rowhammer".

Additionally, we revisit the security of the lattice problem known as Learning with Errors (LWE) in the presence of information leakage. We further enhance the robustness of prior methodology by viewing side information from a geometric perspective. Our approach provides the rigorous promise that, as hints are integrated, the correct solution is a (unique) lattice point contained in an ellipsoidal search space.

Second, we study the concrete security of interactive signature schemes (used as part of many Privacy Enhancing Technologies). To this end, we complete a new analysis of the performance of Wagner's k-list algorithm [CRYPTO '02], which has found significant utility in computing forgeries on several interactive signature schemes that implicitly rely on the hardness of the ROS problem formulated by Schnorr [ICICS '01].

ADVANCES IN CONCRETE CRYPTANALYSIS OF LATTICE PROBLEMS AND INTERACTIVE SIGNATURE SCHEMES

by

Hunter Michael Kippen

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2024

Advisory Committee:
Associate Professor Dana Dachman-Soled: *Advisor, Chair*
Doctor Daniel Apon
Professor Jonathan Katz: *Dean's Representative*
Assistant Professor Yonghwi Kwon
Professor Ankur Srivastava

Dedication

For Mom and Dad, I have no words.

Acknowledgments

I am deeply indebted to my advisor and chair of my committee, Dana Dachman-Soled for her inexhaustible patience and understanding. Without her guidance, I would never have made it this far in cryptologic research. I would also like to extend my deepest appreciation to the rest of my thesis committee, who helped make this dissertation the best it could be. Additionally, this entire endeavor would not have been possible without the support from the Clark School of Engineering in the form of The Clark Doctoral Fellowship as well as Intel Labs for funding my research.

I would like to extend my sincere thanks to all of my colleagues and co-authors, particularly the support of Julian Loss and Antione Joux, who pushed me to finish the final work present in this dissertation. A special thanks goes out to Ian Miers, Arkady Yerukhimovich, and Daniel Genkin for providing ideas, expertise, and the opportunity to learn a great many skills that have helped me along my journey.

Lastly, it would be a grave error to not mention all of my friends and family whose emotional support kept me from giving up. There are far too many of you to list individually, so I apologize in advance if anyone feels left out. Huge thanks to my partner Hearan, my brother Clay, his wife Sisi, and their daughter Romi. Doruk, Julian, Sam, Erica, Noemi, Maurice, Wentao, Noel, Rose, Carolyn; I am so incredibly humbled by your kindness and faith in my ability to complete this degree. You are all amazing, and I am so thankful to have gotten to know each and every one of you. To my oldest friend, I know we joked about this moment since middle school, but it is a bit surreal to finally actually be here writing this!

Table of Contents

Dedication ii

Acknowledgements iii

List of Tables vii

List of Figures viii

List of Abbreviations and Symbols ix

1 Introduction 1

2 Background 5
- 2.1 Notation . . . 5
- 2.2 Statistics . . . 6
- 2.3 Public Key Encryption Preliminaries . . . 7
- 2.4 Lattice Preliminaries . . . 9
- 2.5 FrodoKEM . . . 10
- 2.6 The Rowhammer Bug . . . 14
- 2.7 Linear Algebra . . . 15
- 2.8 Geometry . . . 17
  - 2.8.1 Ellipsoid Halfspace and Hyperplane Intersection . . . 19
  - 2.8.2 Ellipsoid Fusion . . . 21
- 2.9 The DBDD Problem . . . 23
  - 2.9.1 Reduction from DBDD to uSVP . . . 25
  - 2.9.2 Security estimates of uSVP . . . 26

3 When NIST PQC FIPS Flips: End-to-End Key Recovery on FrodoKEM via Rowhammer 30
- 3.1 Introduction . . . 30
  - 3.1.1 Our Contributions . . . 31
  - 3.1.2 Overview of Our New Attack . . . 31
  - 3.1.3 Attacker Model . . . 38
  - 3.1.4 Artifacts and Current Status . . . 39

3.1.5 Chapter Organization . . . 39
3.2 Decryption Failure Attack on Rowhammer-Poisoned FrodoKEM . . . 40
3.2.1 Decryption Failure Rate Analysis . . . 41
3.2.2 FrodoKEM Key recovery . . . 43
3.2.3 Attack Modifications for Rowhammer Assisted Failures . . . 49
3.2.4 Total Attack Cost . . . 50
3.3 Poisoning Hobbits: Rowhammering a FrodoKEM Public Key . . . 51
3.3.1 Experimental Setup . . . 52
3.3.2 Determining Useful Bit Flip Locations . . . 53
3.3.3 Profiling Memory . . . 54
3.3.4 Allocating Pages to Victim Process . . . 56
3.3.5 Performance Degradation . . . 57
3.3.6 Results . . . 59
3.4 Searching for Failing Ciphertexts with a Supercomputer . . . 59
3.5 Completing the Attack: Session-Key Recovery . . . 61
3.6 Attacking Other NIST Lattice KEMs . . . 62
3.7 Possible Countermeasures . . . 65
3.7.1 Rowhammer Defenses . . . 65
3.7.2 Defenses Specific to our Attack . . . 67

4 Revisiting Security Estimation for LWE with Hints from a Geometric Perspective 71
4.1 Introduction . . . 71
4.1.1 Benefits and Drawbacks of Our Geometric Approach . . . 73
4.1.2 Instantiations of our Approach . . . 76
4.1.3 Future Work . . . 82
4.1.4 Related Work . . . 84
4.1.5 Organization . . . 86
4.1.6 Obtaining our initial DBDD embedding . . . 87
4.2 Hints . . . 89
4.2.1 Inequality Hints . . . 89
4.2.2 Combined Hints . . . 90
4.2.3 Perfect Hints, Revisited . . . 93
4.2.4 Short Vector Hints, Revisited . . . 97
4.3 Experimental Validation and Applications . . . 99
4.3.1 Experimental Validation . . . 99
4.3.2 Decryption Failures, Revisited . . . 101
4.3.3 Combining Decryption Failure and SCA . . . 109
4.3.3.1 The Baseline Approach . . . 110
4.3.3.2 Ellipsoid/Ellipsoid Intersection. . . . 111
4.3.3.3 Conditions 1 and 2. . . . 112
4.3.3.4 The known and unknown cases . . . 113

5 A Concrete Analysis of Wagner's k-List Algorithm over $\mathbb{Z}_p$ 116
5.1 Introduction . . . 116
5.1.1 Contributions . . . 117
5.1.2 Related Work . . . 121
5.2 Preliminaries . . . 125
5.3 Analysis of the Degraded k-List Algorithm . . . 126
5.3.1 Analysis of IntervalMerge . . . 129
5.3.2 Output List Size . . . 136
5.3.3 Determining Constants . . . 144

A Appendix for When NISTPQC FIPS flips 151
A.1 Additional Attacks against other NIST Lattice KEMs . . . 151
A.1.1 Attacking Kyber . . . 151
A.1.2 Attacking Saber . . . 152
A.1.3 Attacking NTRU-LPRime . . . 153
A.1.4 Attacking NTRU . . . 154
A.1.5 Attacking Streamlined NTRU Prime . . . 155

B Appendix for Revisiting Security Estimation for LWE with Hints from a Geometric Perspective 158
B.1 Proof of Theorem 4.2.1 . . . 158
B.2 Overview of the Ellipsoid Method . . . 160

Bibliography 162

List of Tables

4.1 Full-size decryption failure experiment. 106
4.2 Comparison of bikz estimates for FrodoKEM with CCS1 parameters. 113

List of Figures

3.1 Probability of recovering a single column of the Frodo-640 secret. . . 50
3.2 Total number of ciphertexts required. 51
3.3 Session key brute-force algorithm. 63

4.1 Experimental verification of the bikz predictions for each type of hint. 100
4.2 Toy decryption failure experiment. 102
4.3 Histograms of α values. 109

5.1 The original version of Wagner's k-list algorithm over $\mathbb{Z}_p$ [143]. 118
5.2 Runtime comparison between the degraded k-list algorithm, and the analysis by Shallue [130] 122
5.3 A degraded version of Wagner's k-list algorithm. 126
5.4 The interval merge procedure in the degraded k-list algorithm. 130
5.5 The rejection sampling procedure used during interval merge. 131
5.6 Example intervals created during interval merge. 132

List of Abbreviations and Symbols

DBDD Distorted Bounded Distance Decoding.

LWE Learning with Errors.

uSVP Unique Shortest Vector Problem.

BSI Federal Office for Information Security.

KEM Key Encapsulation Mechanism.

NIST National Institute for Standards and Technology.

PKE Public-Key Encryption.

Chapter 1: Introduction

In recent years there has been a large uptick in the deployment of cryptographic primitives and protocols that achieve security guarantees beyond classical Public-Key Encryption (PKE) and digital signature schemes. These include quantum-resistant cryptography, end-to-end encrypted messaging, secure multiparty computation, private machine learning protocols, anonymous credentials, cryptocurrencies, and myriad others. Achieving these functionalities (including quantum resistance) often increases algorithmic complexity (as compared to basic PKE) and potentially creates new attacks. Much research is required to fully understand the scope of possible threats to these schemes.

To realize secure and efficient deployments, it is often critical to understand the *concrete security* of the scheme in question. This includes the exact computational cost needed to break the underlying encryption, and the potential leakage of secret information through side-channels. Understanding computational costs involves both (1) designing improved attacks and algorithms for specific deployment settings, and (2) creating a general framework that can track concrete security in an automated fashion.

Studying concrete security is often motivated by practical efficiency concerns.

For example, quantum-resistant cryptography based on the hardness of lattice problems is less storage efficient than classical cryptosystems based on elliptic curves. Lattice cryptosystems can have key and ciphertext sizes orders of magnitude higher when using asymptotic security reductions to set parameters. To reduce this overhead, scheme designers have long since justified lowering parameters below those required by reductions with concrete security claims [1, 8, 17, 29, 133].

Such gains in efficiency have contributed to lattice-based cryptography becoming the current National Institute for Standards and Technology (NIST) standards for post post-quantum PKE / Key Encapsulation Mechanism (KEM) and digital signatures [2, 134]. These standards are expected to be (widely) deployed in the next few years. Large technology firms such as Cloudflare [145] and Apple [49] are already integrating lattice-based cryptography into their products.

Interactive signature schemes such as threshold-, multi-, and blind signature schemes [15, 58, 97, 106, 107, 124] are another example where concrete security is considered. Recently, it has been shown that if an adversary opens multiple concurrent signing sessions, the security of these signature schemes degrades [22, 48]. With enough sessions, an attacker can even perform signature forgeries in polynomial time. However, if the number of parallel sessions is limited, then there exist smooth trade-offs between the number of allowed sessions and attack strength. The runtime of this attack is governed by the conjectured runtime of the k-list algorithm by Wagner [143]. Unfortunately, Wagner's original analysis is heuristic in nature, and the state of the art analysis by Shallue [130] hides large polynomial factors. Thus, it is presently unclear how to size the parameters of these signature schemes to strike

the appropriate balance between performance and security for an application.

This dissertation studies the concrete security of both lattice-based cryptography and interactive signature schemes in detail. The dissertation is organized as follows. In Chapter 2, necessary background material is covered, along with notational conventions and relevant definitions. In Chapters 3 and 4, we cover two works on the concrete security of Learning with Errors (LWE), a common lattice problem used to build a majority of the lattice encryption schemes in use today.

In Chapter 3, we consider deployment risks that arise during the key generation process of NIST PQC KEM constructions. Additionally to security against cryptanalysis, NIST has also made clear throughout the standardization process that PQC candidates also should be resilient against side-channel attacks [28, 71, 109, 115, 116, 127, 137, 141]. While some side-channel attack vectors can be defended against using constant-time coding techniques, other actively-induced effects, such as Rowhammer induced bit flips, cannot be as easily mitigated through careful coding practices. Relatively little is known about the resistance of current PQC constructions to active side-channels like Rowhammer. There are only two prior works investigating such attacks against NIST PQC algorithms, and they examined only the case of digital signatures: specifically, the LUOV and Dilithium signature schemes [74, 102]. In this work, we develop a novel side-channel attack on the FrodoKEM key establishment mechanism. Using Rowhammer, this attack induces bitflips during the key generation process. With this side-channel, we are able to construct an attack strategy that recovers much of the private key material, including user session keys.

In Chapter 4, we examine how the concrete security of LWE is affected by the

leakage of private key material. Currently, one of the commonly used algorithms for solving LWE follows this template: (1) Embed the LWE instance into a Unique Shortest Vector Problem (uSVP) instance, which asks to find the shortest non-zero vector in a *lattice*, and then (2) solve the uSVP instance using a type of algorithm known as *lattice reduction*. In this work, we consider algorithms that follow the above template, and our goal is to develop improved methods for the first step—in the case that side information about the LWE secret or error is available.

This side information is organized into standard forms that we refer to as "hints." We developed an open source toolkit to automate the integration of these hints into the constraint system governing an LWE instance. At each step, the toolkit can provide estimates on the remaining concrete hardness of the instance, and can run a lattice reduction algorithm to attempt to solve for the secret key. See Section 4.1.3 under "Toolkit Extension" for more details about the released framework.

Finally, in Chapter 5, we cover the security of interactive signature schemes. In particular, we present a method to give provable guarantees for the runtime of Wagner's k-list algorithm [143] over the intergers modulo p. We provide a lower bound on the runtime of Wagner's algorithm for practical parameter regimes by analyzing a modified version that provably achieves all heuristic invariants. Notably, this algorithm can be viewed as an inferior version of Wagner's. However, for the parameter regimes in question, our modified algorithm outperforms the provable runtime of Wagner's original algorithm by Shallue [130].

Chapter 2: Background

2.1 Notation

We use bold lower case letters to denote vectors, and bold upper case letters to denote matrices. We use row notation for vectors, and start indexing from 1. We denote by $\boldsymbol{I}_n$ the n-dimensional identity matrix and denote by $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ the inner product of vectors $\boldsymbol{x}, \boldsymbol{y}$ of the same dimension. We denote by $(\boldsymbol{x}||\boldsymbol{y})$ the concatenation of two row vectors $\boldsymbol{x}, \boldsymbol{y}$ whose dimension is the sum of the dimensions of $\boldsymbol{x}$ and $\boldsymbol{y}$. For $\boldsymbol{v} \in \mathcal{R}^d$, $\|\boldsymbol{v}\|$ denotes the ℓ_2 norm of the vector. For a vector $\boldsymbol{v}$, we use both v_i and $\boldsymbol{v}[i]$ to denote the i-th coordinate of the vector. For a matrix $\boldsymbol{M}$ we use both $\boldsymbol{M}_{ij}$ and $\boldsymbol{M}[i][j]$ to denote the (i, j)-th position of the matrix. In addition, we make use of Einstein notation to delineate between rows and columns of a matrix (e.g. For a matrix $\boldsymbol{A}$, $\boldsymbol{a}_i$ is the i^{th} row of the matrix, and $\boldsymbol{a}^j$ is the j^{th} column of the matrix). Random variables—i.e. variables whose values depend on outcomes of a random experiment—are denoted with lowercase calligraphic letters e.g. $\mathcal{a}, \mathcal{b}, \mathcal{e}$, while random vectors are denoted with uppercase calligraphic letters e.g. $\mathcal{C}, \mathcal{X}, \mathcal{Z}$.

2.2 Statistics

Definition 2.2.1 (Univariate normal distribution)**.** We denote by $\mathcal{N}(\mu, \sigma^2)$ the univariate normal (Gaussian) distribution with mean μ, variance σ^2, and probability density function (pdf)

$$x \mapsto \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right).$$

Definition 2.2.2 (Multivariate normal distribution)**.** Let $d \in \mathbb{Z}$, $\boldsymbol{\mu} \in \mathbb{Z}^d$ and let $\boldsymbol{\Sigma}$ be a positive definite matrix of dimension $d \times d$. We denote by $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ the multivariate normal (Gaussian) distribution with mean $\boldsymbol{\mu}$, covariance $\boldsymbol{\Sigma}$, and probability density function (pdf)

$$\boldsymbol{x} \mapsto \frac{1}{\sqrt{(2\pi)^d \cdot \det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})^T\right).$$

Definition 2.2.3. The error function, denoted `erf` is defined as:

$$\texttt{erf}(z) := \frac{2}{\sqrt{\pi}} \int_0^z \exp(-t^2)\, dt.$$

$\texttt{erf}(z)$ is the probability that an x sampled from $\mathcal{N}(0, 1)$ falls in the range $[-z, z]$.

Definition 2.2.4. The discrete Triangle distribution $\mathcal{T}_{z-1}$ is defined by the following

probability mass function:

$$\mathcal{T}_{z-1}(k) = \begin{cases} \frac{k+z}{z^2} & -(z-1) \leq k \leq -1 \\ \frac{1}{z} & k = 0 \\ \frac{z-k}{z^2} & 1 \leq k \leq z-1 \end{cases}$$

In Chapter 5, we make use of a well known application of Azuma's inequality, named McDiarmid's inequality.

Theorem 2.2.5 (McDiarmid's Inequality [99])**.** *A function* $f : \mathcal{X}_1 \times \cdots \times \mathcal{X}_n \mapsto \mathcal{R}$ *satisfies the* bounded differences property *if there are constants* $c_1, c_2, \ldots, c_n$ *such that for all* $i = 1, 2, \ldots, n$ *and every* $x_1 \in \mathcal{X}_1, x_2 \in \mathcal{X}_2, \ldots x_n \in \mathcal{X}_n$*, we have*

$$\sup_{x_i' \in \mathcal{X}_i} \left| \begin{array}{r} f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) - \\ f(x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots x_n) \end{array} \right| \leq c_i.$$

For such an f *satisfying the bounded differences property, consider independent random variables* $\mathcal{x}_1, \mathcal{x}_2, \ldots, \mathcal{x}_n$ *where* $\mathcal{x}_i \in \mathcal{X}_i$ *for all* i*. Then, for any* $\epsilon \geq 0$*,*

$$\mathbf{Pr}\left[f(\mathcal{x}_1, \mathcal{x}_2, \ldots, \mathcal{x}_n) - \mathbf{E}\left[f(\mathcal{x}_1, \mathcal{x}_2, \ldots, \mathcal{x}_n)\right] \leq -\epsilon\right] \leq \exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^n c_i^2}\right)$$

2.3 Public Key Encryption Preliminaries

Definition 2.3.1 (Public-key encryption scheme (PKE))**.** A public-key encryption scheme is a tuple of algorithms (KeyGen, Enc, Dec):

- KeyGen() outputs (pk, sk), where pk is the public key and sk is the secret key.
- Enc(pk, μ) takes as input a public key pk and a message μ and outputs a ciphertext c.
- Dec(sk, c) takes as input a secret key sk and a ciphertext c and outputs a message μ, or fails.

A public-key encryption scheme is correct if an honest in-order execution of KeyGen, Enc, and Dec, results in Dec outputting the same message that is input into Enc (with overwhelming probability). The relevant notion of security for a public-key encryption scheme in our discussion is IND-CPA. A public-key encryption scheme is IND-CPA if any adversary cannot distinguish between ciphertexts of two adversarially-chosen messages with non-negligible advantage.

Definition 2.3.2 (Key Establishment Mechanism (KEM))**.** A key establishment mechanism (KEM) is a tuple of algorithms (KeyGen, Enc, Dec):

- KeyGen() outputs (pk, sk), where pk is the public key and sk is the secret key.
- Enc(pk) takes as input a public key pk and outputs (c, k), where c is the ciphertext that encapsulates the shared session key k.
- Dec(sk, c) takes as input a secret key sk and a ciphertext c, and outputs a shared session key k, or fails.

A key-encapsulation mechanism is correct if an honest in-order execution of KeyGen, Enc, and Dec results in the same shared session key output by Enc and Dec

(with overwhelming probability). Moreover, a KEM is IND-CCA if any adversary with access to a decapsulation oracle, on input of a random challenge session key and a challenge ciphertext, cannot distinguish between the case when the session key is encapsulated in the ciphertext and the case when the session key is independent and uniformly random, with non-negligible advantage.

2.4 Lattice Preliminaries

A *lattice*, denoted by Λ, is a discrete additive subgroup of $\mathcal{R}^d$. It is generated by taking the set of all integer linear combinations of r (where $r \leq d$) linearly independent basis vectors $\{\boldsymbol{b}_j\} \subset \mathcal{R}^d$. Namely,

$$\Lambda := \left\{ \sum\nolimits_j z_j \boldsymbol{b}_j : z_j \in \mathbb{Z} \right\}.$$

We say that d is the *dimension* of Λ and r is its rank. A lattice is *full rank* if $r = d$. A matrix $\boldsymbol{B}$ whose rows are the basis vectors $\{\boldsymbol{b}_j\}$ is called a *basis* of the lattice. The *determinant* or *volume* of a lattice Λ is defined as $\mathsf{Vol}(\Lambda) := \sqrt{\mathsf{det}(\boldsymbol{B}\boldsymbol{B}^T)}$.

Definition 2.4.1 (Unique Shortest Vector Problem)**.** For a lattice Λ and for $i \in [\mathrm{rank}(\Lambda)]$, let $\lambda_i(\Lambda)$ denote the i-th successive minimum (the smallest radius r such that the ball $B(\mathbf{0}, r)$ contains i independent points in the lattice). The unique shortest vector problem (uSVP) is the following:

Given a lattice Λ in which $\lambda_1(\Lambda)$ is significantly shorter than $\lambda_2(\Lambda)$, find a nonzero vector $\boldsymbol{s} \in \Lambda$ where $\|\boldsymbol{s}\| = \lambda_1(\Lambda)$.

Definition 2.4.2 (LWE distribution)**.** The LWE distribution is parametrized by positive integers (m, n, q) and an error distribution χ over $\mathbb{Z}$. The LWE distribution is sampled by sampling a uniformly random $\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}$, $\mathbf{s} \leftarrow \chi^n$, $\mathbf{e} \leftarrow \chi^m$, and outputting $(\mathbf{A}, \mathbf{b} \coloneqq \mathbf{s}\mathbf{A}^T + \mathbf{e} \bmod q)$.

Definition 2.4.3 (The Search-LWE Problem)**.** Given $(\mathbf{A}, \mathbf{b})$ drawn from the LWE distribution, search-LWE problem asks to find $\mathbf{s}$ in the support of χ^n such that $\mathbf{e} \coloneqq \mathbf{b} - \mathbf{A}\mathbf{s}$ is in the support of χ^m modulo q.

Definition 2.4.4 (The Decision-LWE Problem)**.** The decision-LWE problem asks to distinguish between the LWE distribution and the uniform distribution over $(\mathbb{Z}_q^{m \times n}, \mathbb{Z}_q^m)$.

It's been shown (e.g. in [119]) that there is a reduction from search-LWE to decision-LWE.

2.5 FrodoKEM

FrodoKEM is an IND-CCA key-encapusulation mechanism. FrodoKEM is constructed as a Fujisaki-Okamoto transform of FrodoPKE, an IND-CPA public-key encryption scheme whose security is based on the hardness of Learning with Errors. Learning with Errors was first defined and studied as the basis for a public-key cryptosystem by Regev in [119]. Lindner and Peikert later in [92] showed a more efficient public-key encryption scheme based on the hardness of Learning with Errors. FrodoPKE is an instantiation of Lindner and Peikert's construction. We give the definitions of Learning with Errors, both the search and decision variants, below.

We note that the definitions we state here are of so-called normal-form Learning with Errors introduced in [100], which shows that normal-form Learning with Errors is at least as hard as Learning with Errors as originally defined by Regev in [119].

We give a high-level description of FrodoPKE in algorithms 2.5.1, 2.5.2, and 2.5.3, where we omit details about (pseudo)random generation using symmetric primitives, sampling from the error distribution χ, and bit-level representations. We point readers to the Frodo specification [8] for details. In the following, $n, \bar{m}, \bar{n}, B$, and q are integer parameters. Specifically, for Frodo640, $n = 640, \bar{m} = \bar{n} = 8, B = 2$, and $q = 32768$.

Algorithm 2.5.1 FrodoPKE.Keygen()

1: $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times n}$, $\mathbf{S} \leftarrow \chi^{n \times \bar{n}}$, $\mathbf{E} \leftarrow \chi^{n \times \bar{n}}$
2: $\mathbf{B} = \mathbf{AS} + \mathbf{E} \bmod q$
3: **return** $(pk = (\mathbf{A}, \mathbf{B}), sk = \mathbf{S})$

Algorithm 2.5.2 FrodoPKE.Enc(pk, μ)

1: $\mathbf{A}, \mathbf{B} = pk$
2: $\mathbf{S}', \mathbf{E}' \leftarrow \chi^{\bar{m} \times n}$
3: $\mathbf{E}'' \leftarrow \chi^{\bar{m} \times \bar{n}}$
4: $\mathbf{B}' = \mathbf{S}'\mathbf{A} + \mathbf{E}'$
5: $\mathbf{V} = \mathbf{S}'\mathbf{B} + \mathbf{E}''$
6: $(\mathbf{C}_1, \mathbf{C}_2) = (\mathbf{B}', \mathbf{V} + \mathit{Encode}(\mu))$
7: **return** $c = (\mathbf{C}_1, \mathbf{C}_2)$

Algorithm 2.5.3 FrodoPKE.Dec(sk, μ)

1: $\mathbf{S} = sk$
2: $(\mathbf{C}_1, \mathbf{C}_2) = c$
3: $\mathbf{M} = \mathbf{C}_2 - \mathbf{C}_1\mathbf{S}$
4: **return** $\mu = \mathit{Decode}(\mathbf{M})$

We now describe the *Encode* and *Decode* functions to complete our description of FrodoPKE. The *Encode* function encodes an integer k such that $0 \leq k < 2^B \leq q$

as an element in $\mathbb{Z}_q$:

$$Encode(k) \coloneqq k \cdot q/2^B.$$

In Frodo, q is a power of 2, and therefore $q/2^B$ is an integer. The *Decode* function extracts a B-bit integer from an element of $\mathbb{Z}_q$:

$$Decode(k) \coloneqq \lfloor c \cdot 2^B/q \rceil \bmod 2^B.$$

We extend the domain of *Encode* and *Decode* to vectors and matrices by entry-wise application. The *Encode* and *Decode* functions have an error-correcting property stated in [8, Lemma 2.18]. We re-state the result below.

Lemma 2.5.1. *Let $q = 2^D$, $B \leq D$. Then $Decode(Encode(k) + e) = k$ for any $k, e \in \mathbb{Z}$ such that $0 \leq k < 2^B$ and $-q/2^{B+1} \leq e < q/2^{B+1}$.*

The correctness of FrodoPKE follows because the decryption computes

$$\begin{aligned}\mathbf{M} &= \mathbf{C}_2 - \mathbf{C}_1\mathbf{S} \\ &= Encode(\mu) + \mathbf{S}'\mathbf{E} - \mathbf{E}'\mathbf{S} + \mathbf{E}''.\end{aligned}$$

Let $\mathbf{E}''' = \mathbf{S}'\mathbf{E} - \mathbf{E}'\mathbf{S} + \mathbf{E}''$. By Lemma 2.5.1, if the entries of $\mathbf{E}'''$ are sufficiently small, we have that $Decode(\mathbf{M}) = \mu$. A lower bound on the probability of this event can be computed explicitly using Frodo's parameter search script in Frodo submission available at [132]. Conversely, if an entry in the decoding error $\mathbf{E}'''$ exceeds the decoding threshold $[-q/2^{B+1}, q/2^{B+1})$, a decoding failure will occur in

Algorithm 2.5.4 FrodoKEM.KeyGen()

1: $(pk, sk) \leftarrow$ FrodoPKE.KeyGen()
2: **return** $(pk, (pk, sk))$

Algorithm 2.5.5 FrodoKEM.Enc(pk)

1: $\mu \leftarrow \{0,1\}^{\ell}$
2: $r \leftarrow H_1(\mu)$
3: $c \leftarrow$ FrodoPKE.Enc$(pk, \mu; r)$
4: **return** (c, μ)

Algorithm 2.5.6 FrodoKEM.Dec($(pk, sk), c$)

1: $\mu \leftarrow$ FrodoPKE.Dec(sk, c)
2: $r \leftarrow H_1(\mu)$
3: $c' \leftarrow$ FrodoKEM.Enc$(pk, \mu; r)$
4: **if** $c' = c$ **then**
5: **return** $H_2(\mu)$
6: **else**
7: Decapsulation fails.
8: **end if**

the corresponding entry of μ.

Given an IND-CPA public-key encryption scheme such as FrodoPKE above, a generic Fujisaki-Okamoto transform can be applied to obtain an IND-CCA key-encapsulation mechanism. We give a high-level description of the Fujisaki-Okamoto transform used by FrodoKEM in algorithms 2.5.4, 2.5.5, and 2.5.6, where H_1 and H_2 are cryptographic hash functions (e.g. SHAKE) modeled as independent random oracles.

For our attack, we consider Frodo640, the NIST level 1 (as hard as brute-force search on AES-128) parameter set of FrodoKEM. For Frodo640, the parameters are as follows: $n = 640, \bar{n} = \bar{m} = 8, q = 2^{15}, B = 2$. Therefore, the shared session key is decoded from an 8×8 matrix and the decoding threshold is $[-4096, 4096)$.

2.6 The Rowhammer Bug

Rowhammer is a phenomenon wherein repeated accesses to a row in DRAM can induce bit flips in the neighboring rows [81, 103]. This is because activations of the row's wordline cause the capacitors storing bit values in neighboring rows to discharge slightly due to parasitic current. If this occurs a sufficient number of times to drop the voltage below the "charged" threshold before the DRAM refreshes, which typically occurs every 64ms, the logical value of the bit flips. The row that is repeatedly activated, or "hammered" is called the "aggressor row," while the rows containing the induced bit flips are "victim" rows.

Double-Sided Rowhammering. In order to use Rowhammer to intentionally flip inaccessible bits, it is more effective if the attacker repeatedly triggers accesses to memory values both *above and below* the victim rows within the same bank. If the victim row contains bits susceptible to Rowhammer, this memory access pattern is far more likely to induce bit flips than single sided hammering.

Repeatability of Flips. A crucial property of Rowhammer for building attacks is that Rowhammer-induced bit flips are repeatable. This means that a bit that flips at any given point in time is likely to flip again in the future when hammered, and similarly bits that do not flip after being hammered are unlikely to flip upon further hammering. While roughly half of flippable bits can flip in the 1-to-0 direction, and the other half in the opposite direction, any given bit can only possibly flip in one direction per boot. This means that an adversary can "profile" a given machine prior to an attack by hammering memory and building a map of where the flippable

bits are and what directions they flip. This precision allows us to accurately poison the FrodoKEM key in a fairly deterministic manner.

Rowhammer History. A multitude of works from both academia and industry have helped Rowhammer evolve from its conception as a theoretical attack to a realistic attack vector with serious implications. After [81] first uncovered the Rowhammer phenomenon, researchers primarily focused on how to use Rowhammer for privilege escalation attacks and obtaining arbitrary read/writes [31, 54, 62, 89, 120, 126, 138, 140, 146]. Rowhammer attacks from the browser [64, 121], over the network [94, 136], and against ECC memory [39] soon followed, along with new hammering patterns [55, 76, 96] enabling Rowhammer against DDR4 memory.

Cryptographic Applications of Rowhammer. In addition to compromising standard isolation primitives, Rowhammer-induced bit flips were also used to break the security of cryptographic primitives. More specifically, [118] demonstrate how Rowhammer can be used to break RSA signature validation, while [102] demonstrate an attack on the LUOV signature scheme. Finally, [89] show how to use Rowhammer-induced bit flips to directly read bits from memory, allowing for the recovery of RSA keys directly from the target's address space.

2.7 Linear Algebra

Definition 2.7.1 (Positive Semidefinite)**.** A $n \times n$ symmetric real matrix $\boldsymbol{M}$ is positive semidefinite if the scalar quantity $\boldsymbol{x}\boldsymbol{M}\boldsymbol{x}^T \geq 0 \forall \boldsymbol{x} \in \mathcal{R}^n$; if so, we write $\boldsymbol{M} \geq 0$. Given two $n \times n$ real matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, we note that $\boldsymbol{A} \geq \boldsymbol{B}$ if $\boldsymbol{A} - \boldsymbol{B}$ is

positive semidefinite.

Definition 2.7.2. $\boldsymbol{M}$ is a square root of $\boldsymbol{\Sigma}$, denoted $\sqrt{\boldsymbol{\Sigma}}$, if $\boldsymbol{M}^T \cdot \boldsymbol{M} = \boldsymbol{\Sigma}$.

As in the prior work [41], we make use of a generalized notion of the inverse and determinant, where these operations are restricted to operate on the row span of the input matrix. For $\boldsymbol{X} \in \mathcal{R}^{d \times k}$ (with any $d, k \in \mathbb{N}$), we denote by $\boldsymbol{\Pi}_{\boldsymbol{X}}$ the orthogonal projection matrix onto $\mathsf{Span}(\boldsymbol{X})$. More formally, let $\boldsymbol{Y}$ be a maximal set of independent row-vectors of $\boldsymbol{X}$; the orthogonal projection matrix is given by $\boldsymbol{\Pi}_{\boldsymbol{X}} = \boldsymbol{Y}^T \cdot (\boldsymbol{Y} \cdot \boldsymbol{Y}^T)^{-1} \cdot \boldsymbol{Y}$. Its complement (the projection orthogonally to $\mathrm{Span}(\boldsymbol{X})$) is denoted by $\boldsymbol{\Pi}^{\perp}_{\boldsymbol{X}} := \boldsymbol{I}_d - \boldsymbol{\Pi}_{\boldsymbol{X}}$. We naturally extend the notation $\boldsymbol{\Pi}_F$ and $\boldsymbol{\Pi}^{\perp}_F$ to subspaces $F \subset \mathcal{R}^d$. By definition, the projection matrices satisfy $\boldsymbol{\Pi}^2_F = \boldsymbol{\Pi}_F$, $\boldsymbol{\Pi}^T_F = \boldsymbol{\Pi}_F$ and $\boldsymbol{\Pi}_F \cdot \boldsymbol{\Pi}^{\perp}_F = \boldsymbol{\Pi}^{\perp}_F \cdot \boldsymbol{\Pi}_F = \boldsymbol{0}$.

Definition 2.7.3 (Restricted Inverse and Determinant [41])**.** Let $\boldsymbol{\Sigma}$ be a symmetric matrix. We denote a restricted inverse denoted $\boldsymbol{\Sigma}^{\sim}$ as

$$\boldsymbol{\Sigma}^{\sim} := (\boldsymbol{\Sigma} + \boldsymbol{\Pi}^{\perp}_{\boldsymbol{\Sigma}})^{-1} - \boldsymbol{\Pi}^{\perp}_{\boldsymbol{\Sigma}}.$$

It satisfies $\mathsf{Span}(\boldsymbol{\Sigma}^{\sim}) = \mathsf{Span}(\boldsymbol{\Sigma})$ and $\boldsymbol{\Sigma} \cdot \boldsymbol{\Sigma}^{\sim} = \boldsymbol{\Pi}_{\boldsymbol{\Sigma}}$.

We denote by $\mathsf{rdet}(\boldsymbol{\Sigma})$ the restricted determinant: $\mathsf{rdet}(\boldsymbol{\Sigma}) := \mathsf{det}(\boldsymbol{\Sigma} + \boldsymbol{\Pi}^{\perp}_{\boldsymbol{\Sigma}})$.

2.8 Geometry

Definition 2.8.1 (Ellipsoid [61])**.** A set $E \subseteq \mathcal{R}^d$ is a (possibly degenerate) **ellipsoid** if there exist a vector $\boldsymbol{\mu} \in \mathcal{R}^d$ and a positive (semi-)definite $d \times d$-matrix $\boldsymbol{\Sigma}$ such that

$$E = E(\boldsymbol{\mu}, \boldsymbol{\Sigma}) := \{\boldsymbol{x} \in \boldsymbol{\mu} + \mathsf{Span}(\boldsymbol{\Sigma}) \,|\, (\boldsymbol{x} - \boldsymbol{\mu})\boldsymbol{\Sigma}^{\sim}(\boldsymbol{x} - \boldsymbol{\mu})^T \leq 1\}. \tag{2.1}$$

Definition 2.8.1 generalizes the traditional non-degenerate ellipsoid. Note that if $\boldsymbol{\Sigma}$ is full rank, then $\boldsymbol{\mu} + \mathsf{Span}(\boldsymbol{\Sigma}) = \mathcal{R}^d$, and the restricted inverse becomes the regular matrix inverse. Equivalently, a (non-degenerate) ellipsoid can be described by the norm $\|\cdot\|_{\boldsymbol{\Sigma}}$ on $\mathcal{R}^d$

$$E(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \{\boldsymbol{x} \in \mathcal{R}^d \,|\, \|\boldsymbol{x} - \boldsymbol{\mu}\|_{\boldsymbol{\Sigma}} \leq 1\}$$

thus, the ellipsoid $E(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the unit ball around $\boldsymbol{\mu}$ in the vector space $\mathcal{R}^d$ endowed with the norm $\|\cdot\|_{\boldsymbol{\Sigma}}$. In particular, the unit ball around $\mathbf{0}$ in the traditional Euclidean norm is $E(\mathbf{0}, \boldsymbol{I}_d)$. As $\boldsymbol{\Sigma}$ is positive definite, the matrix square root exists. As such, we can express an ellipsoid via the following relation

$$E(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \boldsymbol{\Sigma}^{1/2} E(\mathbf{0}, \boldsymbol{I}_d) + \boldsymbol{\mu}$$

making every ellipsoid the image of the unit ball under a bijective affine transformation. These alternative views of an ellipsoid can also be generalized to work with the degenerate case in a similar fashion to the generalized definition.

Definition 2.8.2 (Volume of a full-rank ellipsoid)**.** A full-rank ellipsoid $E(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ of dimension d has volume $\mathsf{Vol}(E(\boldsymbol{\mu}, \boldsymbol{\Sigma})) = \sqrt{\det(\boldsymbol{\Sigma})} \cdot V_d$, where V_d is the volume of the d-dimensional unit ball.

Definition 2.8.3 (Ellipsoid norm)**.** Let $\boldsymbol{x} \in \boldsymbol{\mu} + \mathsf{Span}\boldsymbol{\Sigma}$. We define the *ellipsoid norm* of $\boldsymbol{x}$ with respect to ellipsoid $E(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ to be the quantity $(\boldsymbol{x} - \boldsymbol{\mu})\boldsymbol{\Sigma}^{\sim}(\boldsymbol{x} - \boldsymbol{\mu})^T$. Note that $\boldsymbol{x}$ is contained in $E(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ if and only if its ellipsoid norm with respect to $E(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is at most 1.

Remark 2.8.4 (Ellipsoid Scaling). Throughout the paper, we make use of two different ellipsoid scalings. For ellipsoid operations defined by the ellipsoid method (Section 2.8.1) or ellipsoid fusion (Section 4.2.2), we make use of the traditional scaling factor of 1 in (2.1). However, the invariant of the DBDD problem (section 2.9) requires that the ellipsoid be scaled such that the right hand side of (2.1) is $\mathsf{Rank}(\boldsymbol{\Sigma})$. To remain consistent with prior work [41], we will treat these ellipsoids as a separate object, the *rank-scaled ellipsoid*.

Definition 2.8.5 (Rank-scaled Ellipsoid)**.** A set $E^{(\mathsf{Rank})} \subseteq \mathcal{R}^d$ is a (possibly degenerate) **rank-scaled ellipsoid** if there exist a vector $\boldsymbol{\mu} \in \mathcal{R}^d$ and a positive semidefinite $d \times d$-matrix $\boldsymbol{\Sigma}$ such that

$$E^{(\mathsf{Rank})} = E^{(\mathsf{Rank})}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) := \{\boldsymbol{x} \in \boldsymbol{\mu} + \mathsf{Span}(\boldsymbol{\Sigma}) \,|\, (\boldsymbol{x} - \boldsymbol{\mu})\boldsymbol{\Sigma}^{\sim}(\boldsymbol{x} - \boldsymbol{\mu})^T \leq \mathsf{Rank}(\boldsymbol{\Sigma})\}. \tag{2.2}$$

Converting a traditional ellipsoid into a rank-scaled ellipsoid follows from the definition. Given a rank-scaled ellipsoid, $E^{(\mathsf{Rank})}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, it is equivalent to the

traditional ellipsoid $E(\boldsymbol{\mu}, \boldsymbol{\Sigma} \cdot \mathsf{Rank}(\boldsymbol{\Sigma}))$. As the mean of the ellipsoid remains the same, let

$$\mathcal{F} : \mathcal{R}^{d \times d} \mapsto \mathcal{R}^{d \times d}, \mathcal{F}(\boldsymbol{\Sigma}) = \boldsymbol{\Sigma} \cdot \mathsf{Rank}(\boldsymbol{\Sigma}) \tag{2.3}$$

denote the transformation between the covariance matrices.

Definition 2.8.6 (Hyperplane)**.** A set $H \subseteq \mathcal{R}^d$ is a **hyperplane** if there exist a vector $\boldsymbol{v} \in \mathcal{R}^d$ and a scalar threshold $\gamma \in \mathcal{R}$ such that

$$H = H(\boldsymbol{v}, \gamma) := \{\boldsymbol{x} \in \mathcal{R}^d \,|\, \langle \boldsymbol{x}, \boldsymbol{v} \rangle = \gamma\}.$$

Definition 2.8.7 (Halfspace)**.** Without loss of generality, A set $H^{\leq} \subseteq \mathcal{R}^d$ is a **halfspace** if there exist a vector $\boldsymbol{v} \in \mathcal{R}^d$ and a scalar threshold $\gamma \in \mathcal{R}$ such that

$$H^{\leq} := \{\boldsymbol{x} \in \mathcal{R}^d \,|\, \langle \boldsymbol{x}, \boldsymbol{v} \rangle \leq \gamma\}.$$

2.8.1 Ellipsoid Halfspace and Hyperplane Intersection

The algorithms in some of our applications are reminiscent of the *ellipsoid method*, the first provably polynomial time algorithm for solving linear programs [80]. While our goal is to solve an *integer program*—a harder problem than linear programming—it is well-known (s.f. [78]) that the ellipsoid method can be combined with lattice reduction to solve integer programs. In practice, however, this method is both inefficient and prone to numerical errors. So we must make crucial changes for the approach to be viable in our setting (see Section 4.3.2). For an overview of

the ellipsoid method, see Appendix B.2.

The main update procedure of the ellipsoid method calculates the Löwner-John ellipsoid corresponding to the intersection of an ellipsoid and halfspace. Given an ellipsoid $E(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and a halfspace $\{\boldsymbol{x} \in \mathcal{R}^d \,|\, \langle \boldsymbol{x}, \boldsymbol{v} \rangle \leq \gamma\}$ (where $\boldsymbol{v} \in \text{Span}(\boldsymbol{\Sigma})$), the Löwner-John ellipsoid of the intersection $E(\boldsymbol{\mu}', \boldsymbol{\Sigma}')$ is:

$$\begin{aligned} \boldsymbol{\mu}' &= \boldsymbol{\mu} - \tau \frac{\boldsymbol{v}\boldsymbol{\Sigma}}{\sqrt{\boldsymbol{v}\boldsymbol{\Sigma}\boldsymbol{v}^T}} \\ \boldsymbol{\Sigma}' &= \delta \left(\boldsymbol{\Sigma} - \sigma \frac{\boldsymbol{\Sigma}\boldsymbol{v}^T\boldsymbol{v}\boldsymbol{\Sigma}}{\boldsymbol{v}\boldsymbol{\Sigma}\boldsymbol{v}^T} \right) \end{aligned} \tag{2.4}$$

This expression generalizes the computation of multiple Löwner-John ellipsoids based on ellipsoid-X intersections. The exact intersection performed depends on the values of the three variables δ, σ, and τ. For a geometric interpretation of the effects of varying these parameters, see the survey on the ellipsoid method by Bland, Goldfarb, and Todd [26].

For an ellipsoid-halfspace intersection,

$$\tau = \frac{1 + r\alpha}{r + 1} \qquad \sigma = \frac{2(1 + r\alpha)}{(r + 1)(1 + \alpha)} \qquad \delta = \frac{r^2}{r^2 - 1}(1 - \alpha^2) \tag{2.5}$$

where r is the rank of $\boldsymbol{\Sigma}$, and α is a distorted measure of the distance between the center of the ellipsoid and the separating hyperplane. In (2.5), α is defined as

$$\alpha = \frac{\boldsymbol{v}\boldsymbol{\mu}^T - \gamma}{\sqrt{\boldsymbol{v}\boldsymbol{\Sigma}\boldsymbol{v}^T}}. \tag{2.6}$$

When $-1 < \alpha \leq 1$, the separating hyperplane intersects the ellipsoid. If $\alpha = 0$, the separating hyperplane bisects the ellipsoid through its center. The optimal circumscription when $-1 < \alpha < -1/r$ is simply the starting ellipsoid.

Ellipsoid-Hyperplane Intersection. The intersection between an ellipsoid $E(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and a hyperplane $H(\boldsymbol{v}, \gamma)$, where $\boldsymbol{v} \in \text{Span}(\boldsymbol{\Sigma})$ can be obtained by plugging appropriate parameters into the formula for parallel cuts given in [26]. Doing so yields τ, δ, and σ:

$$\tau = \alpha \qquad \sigma = 1 \qquad \delta = \frac{r}{r-1}(1-\alpha^2) \tag{2.7}$$

where α remains the same as in (2.6). An ellipsoid-hyperplane intersection is itself an ellipsoid of one fewer dimension. As $\sigma = 1$, the rank one update of $\boldsymbol{\Sigma}$ in (2.4) reduces the rank of the intersection $E(\boldsymbol{\mu}', \boldsymbol{\Sigma}')$ by 1, and ensures it is flat in the direction of $\boldsymbol{v}$. Here $-1 < \alpha \leq 1$, with no additional restrictions, as the hyperplane need simply intersect the starting ellipsoid.

It is possible to prove a tighter bound so that $\delta = (1-\alpha^2)$, which is the setting of δ we will use in our implementation.

2.8.2 Ellipsoid Fusion

The intersection of two ellipsoids is not generally an ellipsoid, so as in the ellipsoid method, some optimal approximation must be used to compute a representation of the intersection efficiently. There are multiple measures of an ellipsoid's size that

could be optimized to produce a good approximation. For our framework, we adopt the Ellipsoid Fusion procedure proposed by Ros et al. [122]. Ros et al. propose a measure based on the volume of a *convex combination* of the two input ellipsoids. This is done through the minimization of the determinant of the combined ellipsoid's covariance matrix.

Theorem 2.8.8 (Theorem 2 in [122])**.** *Given two (possibly degenerate) ellipsoids,* $E(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ *and* $E(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$*, whose intersection is a nonempty bounded region, the region defined by*

$$\{\boldsymbol{x} \,|\, \lambda(\boldsymbol{x}-\boldsymbol{\mu}_1)\boldsymbol{\Sigma}_1{}^{\sim}(\boldsymbol{x}-\boldsymbol{\mu}_1)^T + (1-\lambda)(\boldsymbol{x}-\boldsymbol{\mu}_2)\boldsymbol{\Sigma}_2{}^{\sim}(\boldsymbol{x}-\boldsymbol{\mu}_2)^T \leq 1\},$$

is a real ellipsoid, $E^{\lambda}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma})$*, which coincides with* $E(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ *or* $E(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ *for* $\lambda = 1$ *or* $\lambda = 0$ *respectively; and it is given by*

$$\left.\begin{aligned} \boldsymbol{\Sigma} &= k\boldsymbol{X}^{\sim} \\ \boldsymbol{X} &= \lambda\boldsymbol{\Sigma}_1{}^{\sim} + (1-\lambda)\boldsymbol{\Sigma}_2{}^{\sim} \\ \boldsymbol{\mu}_0\Pi_{\boldsymbol{X}} &= \left(\boldsymbol{\mu}_1\lambda\boldsymbol{\Sigma}_1{}^{\sim} + \boldsymbol{\mu}_2(1-\lambda)\boldsymbol{\Sigma}_2{}^{\sim}\right)\boldsymbol{X}^{\sim} \\ k &= 1 - \lambda(1-\lambda)(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)\boldsymbol{\Sigma}_2{}^{\sim}\boldsymbol{X}^{\sim}\boldsymbol{\Sigma}_1{}^{\sim}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)^T \end{aligned}\right\}$$

for $\lambda \in [0, 1]$.

Note that $\boldsymbol{\mu}_0$ can be set arbitrarily so long as $\boldsymbol{\mu}_0\Pi_{\boldsymbol{X}}$ satisfies the above. While this combination is not necessarily the optimal circumscription, it does not contain points that are in neither $E(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ nor $E(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$.

Definition 2.8.9 (Ellipsoid Fusion (Def. 5 in [122]))**.** The fusion of $E(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $E(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$, whose intersection is a nonempty bounded region is $E^{\tilde{\lambda}}(\boldsymbol{\Sigma}, \boldsymbol{\mu}_0)$ for the value of $\tilde{\lambda} \in [0, 1]$ that minimizes its volume.

Theorem 2.8.10 (Fusion (Theorem 3 in [122]))**.** *The fusion of* $E(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ *and* $E(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ *is:* $E(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$*; or* $E(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$*; or it is* $E^{\tilde{\lambda}}(\boldsymbol{\Sigma}, \boldsymbol{\mu}_0)$ *where* $\tilde{\lambda}$ *is the only root in* $[0, 1]$ *of the following polynomial of degree* $2n - 1$ *:*

$$k(\mathsf{rdet}(\boldsymbol{X}))\mathsf{Trace}(\boldsymbol{X}(\boldsymbol{\Sigma}_1{}^{\sim} - \boldsymbol{\Sigma}_2{}^{\sim})) - n(\mathsf{rdet}(\boldsymbol{X}))^2(2\boldsymbol{\mu}_0\boldsymbol{\Sigma}_1{}^{\sim}\boldsymbol{\mu}_1^T - 2\boldsymbol{\mu}_0\boldsymbol{\Sigma}_2{}^{\sim}\boldsymbol{\mu}_2 + \boldsymbol{\mu}_0(\boldsymbol{\Sigma}_2{}^{\sim} - \boldsymbol{\Sigma}_1{}^{\sim})\boldsymbol{\mu}_0^T - \boldsymbol{\mu}_1\boldsymbol{\Sigma}_1{}^{\sim}\boldsymbol{\mu}_1^T + \boldsymbol{\mu}_2\boldsymbol{\Sigma}_2{}^{\sim}\boldsymbol{\mu}_2^T) \tag{2.8}$$

2.9 The DBDD Problem

Definition 2.9.1 (Distorted Bounded Distance Decoding (DBDD) problem)**.** Let $\Lambda \subset \mathbb{R}^{d+1}$ be a lattice, $\boldsymbol{\Sigma} \in \mathbb{R}^{(d+1)\times(d+1)}$ be a symmetric matrix and $\boldsymbol{\mu} \in \mathrm{Span}(\Lambda) \subset \mathbb{R}^{(d+1)}$ such that

$$\mathrm{Span}(\boldsymbol{\Sigma}) \subsetneq \mathrm{Span}(\boldsymbol{\Sigma} + \boldsymbol{\mu}^T \cdot \boldsymbol{\mu}) = \mathrm{Span}(\Lambda). \tag{2.9}$$

The Distorted Bounded Distance Decoding problem $\mathsf{DBDD}_{\Lambda,\boldsymbol{\mu},\boldsymbol{\Sigma}}$ is:

Given $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ and a basis of Λ.

Find the unique vector $\boldsymbol{x} \in \Lambda \cap E^{(\mathsf{Rank})}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

where $E^{(\mathsf{Rank})}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the (possibly degenerate) rank-scaled ellipsoid

$$E^{(\mathsf{Rank})}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) := \{\boldsymbol{x} \in \boldsymbol{\mu} + \mathsf{Span}(\boldsymbol{\Sigma}) \,|\, (\boldsymbol{x} - \boldsymbol{\mu})\boldsymbol{\Sigma}^{\sim}(\boldsymbol{x} - \boldsymbol{\mu})^T \leq \mathsf{Rank}(\boldsymbol{\Sigma})\}.$$

In [41], $E^{(\mathsf{Rank})}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ corresponds to knowing that the secret vector $\boldsymbol{x}$ to be recovered follows a Gaussian distribution of variance $\boldsymbol{\Sigma}$ and mean $\boldsymbol{\mu}$, and the expected value of $(\boldsymbol{x}-\boldsymbol{\mu})\boldsymbol{\Sigma}^{\sim}(\boldsymbol{x}-\boldsymbol{\mu})^T$ for a Gaussian $\boldsymbol{x}$ of variance $\boldsymbol{\Sigma}$ and mean $\boldsymbol{\mu}$ is $\mathsf{Rank}(\boldsymbol{\Sigma})$. In the current work, we do not view the ellipsoid in the DBDD instance as stemming from the covariance matrix of a multivariate Gaussian distribution. Rather, we view the ellipsoid as defining a region containing a feasible solution to a certain constraint satisfaction problem over the reals. Then we restrict the solutions to those that are also contained in some lattice.

In order to be consistent with the approaches in the literature on Löwner-John ellipsoids, it will be useful for us to consider instances DBDD instances of dimension one lower than those in the prior work. Further, we allow $\mathrm{Span}(\boldsymbol{\Sigma}) = \mathrm{Span}(\boldsymbol{\Sigma} + \boldsymbol{\mu}^T \cdot \boldsymbol{\mu}) = \mathrm{Span}(\Lambda)$, unlike in the prior work. For clarity we formally define below the DBDD variant that we consider in this work.

Definition 2.9.2 (A Variant of the Distorted Bounded Distance Decoding problem)**.** Let $\Lambda \subset \mathbb{R}^d$ be a lattice, $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$ be a symmetric matrix and $\boldsymbol{\mu} \in \mathrm{Span}(\Lambda) \subset \mathbb{R}^d$. such that

$$\mathrm{Span}(\boldsymbol{\Sigma}) = \mathrm{Span}(\boldsymbol{\Sigma} + \boldsymbol{\mu}^T \cdot \boldsymbol{\mu}) = \mathrm{Span}(\Lambda). \tag{2.10}$$

The Distorted Bounded Distance Decoding problem $\mathsf{DBDD}_{\Lambda,\boldsymbol{\mu},\boldsymbol{\Sigma}}$ with respect to $(\Lambda, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ defined as above is:

Given $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ and a basis of Λ.

Find the unique vector $\boldsymbol{x} \in \Lambda \cap E^{(\mathsf{Rank})}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

where $E^{(\mathsf{Rank})}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the (possibly degenerate) rank-scaled ellipsoid

$$E^{(\mathsf{Rank})}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) := \{\boldsymbol{x} \in \mathsf{Span}(\boldsymbol{\Sigma}) \,|\, (\boldsymbol{x} - \boldsymbol{\mu})\boldsymbol{\Sigma}^{\sim}(\boldsymbol{x} - \boldsymbol{\mu})^T \leq \mathsf{Rank}(\boldsymbol{\Sigma})\}.$$

We can convert a DBDD instance $(\Lambda, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ of the type considered above into a DBDD instance $(\Lambda', \boldsymbol{\mu}', \boldsymbol{\Sigma}')$ considered in the prior work as follows:

$$\Lambda' = \{(\boldsymbol{x}||z) \in \mathcal{R}^{d+1} : \boldsymbol{x} \in \Lambda, z \in \mathbb{Z}\}$$

$$\boldsymbol{\mu}' = (\boldsymbol{\mu}||1) \in \mathcal{R}^{d+1}$$

$$\boldsymbol{\Sigma}'[i][j] := \begin{cases} \boldsymbol{\Sigma}[i][j] & \text{if } i, j \leq d \\ 0 & \text{if } i = d+1 \text{ or } j = d+1. \end{cases}$$

2.9.1 Reduction from DBDD to uSVP

Following [41], the conversion of a DBDD instance $(\Lambda, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ into a uSVP instance proceeds in two steps known as *homogenization* and *isotropization*.

Homogenization: The homogenization procedure takes an ellipsoid that is centered at $\boldsymbol{\mu}$ and converts it into an ellipsoid centered at $\mathbf{0}$. The zero-centered ellipsoid *contains* the ellipsoid centered at $\boldsymbol{\mu}$ (see [41] for the proof of this claim). The volume of the ellipsoid remains the same[1], and the rank of its covariance matrix goes up by

[1]This assumes that $\boldsymbol{\mu}'$–corresponding to the first d coorindates of $\boldsymbol{\mu} \in \mathsf{Span}(\boldsymbol{\Sigma})$ and the final coordinate of $\boldsymbol{\mu}$ is equal to 1, which is the case for DBDD instances obtained from DBDD variant instances.

1. Specifically, the conversion is as follows:

$$(\Lambda, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \mapsto (\Lambda, \mathbf{0}, \boldsymbol{\Sigma}' := \boldsymbol{\Sigma} + \boldsymbol{\mu}^T \cdot \boldsymbol{\mu}).$$

Isotropization: The isotropization procedure converts the covariance matrix $\boldsymbol{\Sigma}'$ into an isotropic matrix (i.e. with all its eigenvalues equal to 1), by applying an appropriate linear transformation to the input space. We then perform the same linear transformation on the lattice. Specifically, the conversion is as follows:

$$(\Lambda, \mathbf{0}, \boldsymbol{\Sigma}') \mapsto (\Lambda \cdot M, \mathbf{0}, M \cdot \boldsymbol{\Sigma}' \cdot M^T),$$

where $M = \sqrt{\boldsymbol{\Sigma}'}^{\sim}$. The above can be simplified to

$$(\Lambda \cdot M, \mathbf{0}, M \cdot \boldsymbol{\Sigma}' \cdot M^T) = (\Lambda \cdot M, \mathbf{0}, \boldsymbol{\Pi}_{\boldsymbol{\Sigma}'}) = (\Lambda \cdot M, \mathbf{0}, \boldsymbol{\Pi}_{\Lambda}),$$

see [41] for details on the above simplification. After homogenization and isotropization, we obtain the uSVP instance $\Lambda \cdot \boldsymbol{M}$ (consisting of a lattice only). To complete the reduction, note that from a given solution, $\boldsymbol{x}$, to the $\mathsf{uSVP}_{\Lambda \cdot \boldsymbol{M}}$ problem, one can derive the solution, $\boldsymbol{x}' = \boldsymbol{x} \cdot \boldsymbol{M}^{\sim}$, to the $\mathsf{DBDD}_{\Lambda, \boldsymbol{\mu}, \boldsymbol{\Sigma}}$ problem.

2.9.2 Security estimates of uSVP

We briefly recap the way concrete hardness estimates are computed for a given uSVP instance. Specifically, we consider an attack that consists of applying BKZ-β

to the uSVP lattice Λ for an appropriate block size parameter β. The cost of the attack grows with β, and, as in [41], we will treat β itself as a measurement of the security level in a unit called the *bikz*. Bikz-to-bit conversion can be performed using a conversion factor based on the current best algorithms for SVP in lattices of rank β. Typically, it is assumed that 1 bikz ≈ 0.265 bits. As in [41], the concrete security estimates given in this paper only concern the pure lattice attacks via the uSVP embedding discussed above.

Predicting β for a uSVP instance The state-of-the-art predictions for solving uSVP using BKZ were given in [7, 9]: For a lattice Λ of dimension $\dim(\Lambda)$, it is predicted that BKZ-β can solve a uSVP_Λ instance with secret $(e||s)$ when

$$\sqrt{\beta/\dim(\Lambda)} \cdot \|(e||s)\| \leq \delta_\beta^{2\beta-\dim(\Lambda)-1} \cdot \mathsf{Vol}(\Lambda)^{1/\dim(\Lambda)} \tag{2.11}$$

where δ_β is the so called root-Hermite-Factor of BKZ-β. For $\beta \geq 50$, the Root-Hermite-Factor is predictable using the Gaussian Heuristic [38]:

$$\delta_\beta = \left((\pi\beta)^{\frac{1}{\beta}} \cdot \frac{\beta}{2\pi e}\right)^{1/(2\beta-2)}. \tag{2.12}$$

In [41], the uSVP instances obtained were always isotropic and centered so that the secret has covariance $\boldsymbol{\Sigma} = \boldsymbol{I}$ (or $\boldsymbol{\Sigma} = \boldsymbol{\Pi}_\Lambda$ if Λ is not of full rank) and $\boldsymbol{\mu} = \mathbf{0}$. In this case, $\|(\boldsymbol{e}||\boldsymbol{s})\|^2 = \mathsf{Rank}(\boldsymbol{\Sigma}) = \dim(\Lambda)$, in expectation, and β can be estimated

as the minimum integer that satisfies

$$\sqrt{\beta} \leq \delta_{\beta}^{2\beta-\dim(\Lambda)-1} \cdot \mathsf{Vol}(\Lambda)^{1/\dim(\Lambda)}. \tag{2.13}$$

Importantly, in our case where we do not enforce distributional assumptions, we can no longer assume that after isotropization the secret has covariance $\boldsymbol{\Sigma} = \boldsymbol{I}$ and $\boldsymbol{\mu} = \mathbf{0}$, rather, we just know that the secret is contained in the ellipsoid $E^{(\mathsf{Rank})}(\mathbf{0}, \boldsymbol{I})$, but its norm could be far smaller. Therefore, when performing our final hardness estimates, we sometimes need to take the length of the shortest vector into account (i.e. we will use equation (2.11)) in order to accurately predict β. Throughout the paper, whenever this is the case, we will make note of it. The default is to use the prediction from equation (2.13), which returns β that is at least as large as β from (2.11). As in [41], while β must be an integer as a BKZ parameter, we provide a continuous value.

Remark 2.9.3. To predict security, one does not need the basis of Λ, but only its dimension and its volume. Similarly, it is not necessary to explicitly compute the isotropization matrix $\boldsymbol{M}$ of Section 2.9.1: $\mathsf{Vol}(\Lambda \cdot \boldsymbol{M}) = \mathsf{det}(\boldsymbol{M})\mathsf{Vol}(\Lambda) = \mathsf{det}(\boldsymbol{\Sigma}')^{-1/2}\mathsf{Vol}(\Lambda)$.

Remark 2.9.4. Given a DBDD instance $(\Lambda, \boldsymbol{\mu}, \boldsymbol{\Sigma})$, it is important to note that as the volume of the rank-scaled ellipsoid $E^{(\mathsf{Rank})}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ *decreases*, the volume of the lattice $\Lambda \cdot \boldsymbol{M}$ after homogenization and isotropization *increases*. Applying the hardness estimate from (2.13), this makes the resulting uSVP instance *easier* to solve. Our goal, therefore, when integrating "hints" is to ensure that the volume of the rank-scaled

ellipsoid $E^{(\mathsf{Rank})}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ *decreases* as much as possible.

Chapter 3: When NIST PQC FIPS Flips: End-to-End Key Recovery on FrodoKEM via Rowhammer

3.1 Introduction

Among the lattice-based constructions considered in the 3rd Round of the NIST competition, the FrodoKEM encryption protocol has the most mathematically conservative, security-conscious design. It is the only lattice candidate that has no special algebraic structure but instead bases its security on the *plain* Learning With Errors (LWE) problem [119]. In particular, FrodoKEM was recommended by the German Federal Office for Information Security (BSI) [73] in 2020 as "suitable for long-term confidentiality protection." In this work, we embark on the task of investigating the Rowhammer resilience of NIST's post-quantum KEM candidates, focusing foremost on FrodoKEM as a representative "hard target."

Specifically, we ask the following main questions:

How resilient are PQC KEM constructions to Rowhammer attacks? What would it take for an adversary to mount such attacks, and what information can be extracted using them?

3.1.1 Our Contributions

We demonstrate the first end-to-end implementation of a successful key recovery attack against FrodoKEM using Rowhammer. At a high level, our attack works as follows. First, we use Rowhammer to poison FrodoKEM's KeyGen process. Then, we use a supercomputer to extract private-key material. Next, we synthesize this data using a tailored key-recovery algorithm. Finally, any individual FrodoKEM-encrypted session-key can be recovered in around 2 minutes on a commodity laptop.

Our work demonstrates the near-term and high importance of protecting lattice-based cryptography's key generation processes from active side-channel attacks. Especially, we highlight the scenario of running a lattice KEM's KeyGen in a cloud computing environment, where an adversary will have access to a common, shared-memory architecture on which honest users run KeyGen: In such a setting, honest users are particularly vulnerable to our line of attack and should aim to protect themselves with intention.

3.1.2 Overview of Our New Attack

The main ideas underlying our attack on FrodoKEM are as follows.

Decryption Failure Attacks. We begin with an observation made previously [82] that many lattice-based KEMs, including FrodoKEM, have a non-zero decryption failure rate (DFR),[1] and this may lead to a security vulnerability. That is, validly encrypted ciphertexts may occasionally fail to decrypt properly, and moreover,

[1]This is not an inherent requirement. Our attack may also succeed against rigid lattice-based KEMs with perfect correctness. We defer the details to the body of the paper.

such failing ciphertexts reveal information about the secret key used in decryption. This has led to multiple *decryption failure attacks* and related attack variants in the literature, beginning with Fluhrer's attack on Ring-LWE [52] with many improvements later (e.g. [18, 46, 47, 112]) targeting CPA-secure schemes. These attacks typically make adaptive decryption queries to the receiver of a KEM using carefully crafted ciphertexts. They rely on information of whether a failure occurred or not to gradually recover the secret key.

Protecting FrodoKEM Against Decryption Failures. Decryption failure attacks are mitigated in FrodoKEM and other NIST PQC Round 3 KEMs by use of a Fujisaki-Okamoto (FO) transform [57, 70, 123] that converts a base (IND- or OW-) CPA-secure encryption scheme into a CCA-secure KEM. At a high level, an FO-like transform samples the randomness used to construct a given ciphertext by applying a hash function modeled as a random oracle to its plaintext message. After decryption of a candidate-message, the receiver (deterministically) recomputes a ciphertext encrypting that message and checks if this rederived ciphertext exactly matches the initial ciphertext it received. This makes it impossible for an adversary to craft arbitrary ciphertexts (or maul honestly-constructed ciphertexts), as an overwhelming fraction of these will fail the FO re-encryption check.

Therefore, an adversary against such FO-transformed CCA-secure KEMs is forced to run the honest encryption procedure to produce ciphertexts that will not be rejected outright. Indeed, the parameters of KEM candidates in the 3rd round of the NIST PQC process are chosen to balance the work of an adversary who attempts a decryption failure type of attack and an adversary who attempts a direct

cryptanalytic attack on the mathematics of the system.[2]

Failure-Boosting Attacks. Further works [25, 42, 43] have explored failure-boosting attacks against the CCA-secure forms of NIST PQC candidate-KEMs. The idea here is that the receiver in a real-world KEM will perform only so many decryptions on behalf of various senders attempting to establish a common session key. (The NIST PQC Call For Proposals [105] sets an absolute upper limit of 2^{64} decryptions.) If a random ciphertext fails to decrypt with probability (say) 2^{-128} or 2^{-256}, it's clearly highly unlikely that even 2^{64} decryption-query attempts will yield a single failing ciphertext. But an adversary can do better by querying the random oracle locally, and generating a large list of candidate-ciphertexts to query for decryption. For example, lattice-based ciphertexts with a higher-norm or heavier weight in certain integer-coordinates are more likely to fail to decrypt than a random ciphertext. A failure-boosting adversary chooses a subset of the most-likely-to-fail ciphertexts, and queries only on those. While the above description summarizes the the current state-of-the-art for decryption failure attacks, modern analyses show that the carefully-set parameters of NIST Round 3 KEMs prevent such lines of attack from succeeding in practice.

Intuition for Our Attack. This is where our Rowhammer-assisted attack comes in. The Rowhammer side-channel is a hardware-based security exploit that allows flipping bits in DRAM by "hammering" rows of memory adjacent to a target-victim memory location by repeated memory accesses. Specifically, physically-adjacent

[2]As an interesting historical note, CRYSTALS-Kyber-512 changed a certain binomial-sampling parameter from 2 to 3 between the 2nd and 3rd Rounds of the NIST PQC process explicitly to better balance the cost of these attacks.

memory cells interact electrically between themselves, and repeated activations of rows in memory can cause neighboring capacitors (where the adversary does not have read/write permissions) to discharge before they can be refreshed, causing their logical bits to flip.

By targeting the locations in RAM where the LWE secret key and error are stored during the key generation procedure, we can use Rowhammer to flip some of the higher-order bits of the LWE secret key material. This, in turn, means that the magnitude of certain secret and error coordinates are far higher than would be naturally generated by an honest run of the key generation algorithm. The effect is that decryption failures become somewhat more likely (although not so much that honest users will perceive the difference) – but for an adversary who knows precisely which bits were flipped, obtaining decryption failures via the failure-boosting approach becomes *extremely* more likely (e.g. 2^{-128} becomes $\approx 2^{-12}$).

While this constitutes the core intuition for our new attack, a plethora of additional issues need to be considered to make this approach feasible in the real world.

Rowhammer Timing. A first complication is that Rowhammer bit-flipping requires a minimum window of time at least as long as the electrical refresh rate of the DRAM device being targeted. On typical devices, this is 64 milliseconds, yet FrodoKEM (the slowest of the NIST lattice-based KEM candidates) runs in around 8 milliseconds. We note that FrodoKEM and other NIST PQC KEMs also compute a hash of the generated public key and publish this hash with the associated algebraic material. In turn, this requires that Rowhammer manipulation completes during the

natural KeyGen computation. In other words, the algebraic computation in the key generation process inherently must last for at least 64 milliseconds for Rowhammer to work. To overcome this challenge, we rely on an additional performance degradation side-channel (to sufficiently delay the execution of the FrodoKEM key generation procedure), so that our Rowhammer attack can do its work.

Memory Profiling. A second concern is that the Rowhammer is sensitive to the physical characteristics of individual DRAM modules in the victim device, decided arbitrarily due to process variation during manufacturing at a foundry. For any given page of RAM, some bits will flip frequently when subjected to hammering, and other bits will flip only very slowly. The solution is to profile the target device (before the Lattice KEM algorithm comes online), by hammering every position in the RAM up front, to determine which bits on which pages are more likely to flip.

Memory Massaging. Given such knowledge of particularly "flippy" locations in various pages of the victim DRAM device, we now want to ensure that the FrodoKEM algorithm allocates its memory in a specific manner – exactly aligning with the precise bits of the FrodoKEM algebraic material that we want to manipulate. The mechanism for doing so is "Feng Shui," or memory massaging. Here, we force the victim to allocate its memory in the exact pages of DRAM that we want, by exploiting the low-level details of the data structure underlying the allocation of Linux page frame caches.

Side Stepping Memory Bit Masks. Our Rowhammer procedure physically flips bits in a one-sided manner: from 0 to 1. There are two problems that can arise with this construction. First, logical bits stored in DRAM are distinct from the physical

bits stored in DRAM. In particular, every time the device is booted up, a random bit-mask is applied (akin to a one-time pad). This pad will be consistent, but for N possible pads, the attack succeeds with $1/N$ probability if the machine is reset between profiling and the attack. The solution is to re-profile after any restarts to ensure that the mask does not affect bit locations. Second, 2's complement signed representation dramatically reduces the success probability. For each column, we want to place a single 256-bit flip. The value prior to the flip is equally probable to be negative or positive. The consequence of this representation is that a negative value being targeted to induce a 256-bit flip from 0 to 1 will be unsuccessful because the bit is already a 1. This means that each column can be attacked with probability 2^{-1}, and that each of these locations is an independent probability. So our attack succeeds with additional probability 2^{-N} where N is the number of columns that need to be poisoned, depending on the random sampling of E in the key-generation process.

Generating Failing Ciphertexts. Now we have properly poisoned a FrodoKEM public key. Once the key material has been poisoned, we next need to find sufficiently many failing ciphertexts. For this purpose, we use supercomputing resources (described in Section 3.4) to run an honest encryption procedure on over a trillion distinct messages. Rather than requesting decryptions on this many messages – which would be outside the bounds of an honest receiver's acceptable workload, and thus certainly detected and rejected by a server – we use our knowledge of the hammered positions to filter out ciphertexts that are unlikely to cause a decryption failure. Specifically, we only keep ciphertexts that have either a large positive or a

large negative value in the targeted bit-locations, thus maximizing the probability of failure. By asking that only such ciphertexts be decrypted, we also significantly reduce the probability of detection.

Key Recovery. To perform key recovery given the failing ciphertexts, we present and analyze the following simple algorithm: (1) Construct a set of vectors from the failing ciphertexts generated in the previous stage, (2) Scale these vectors up by a constant factor that depends on the FrodoKEM key distribution and on the rowhammered locations, (3) Take the component-wise average of these scaled-up vectors, (4) Round the resulting vector component-wise to the nearest integer and output this as the candidate key. Comparing our approach with the lattice reduction approach of [41], we find that incorporating the same number of failing ciphertexts using the Toolkit from [41] yields an SVP instance with an estimated hardness of 150 bikz (corresponding to a bit-security of approximately 40), which would not be solvable on a commodity laptop. Further, even obtaining this 150-bikz SVP instance by incorporating decryption failure information using the Toolkit of [41] (as opposed to just computing the hardness *estimates* referenced above), was too computationally intensive for us to run on a commodity laptop due to the large number of failing ciphertexts required in our setting (the required large number of failing ciphertexts is due to the fact that the decryption failure threshold is effectively lowered in our attack, making failures more common, and resulting in less information gained about the secret key from each failure). Thus, for the current setting, our key recovery algorithm outlined above is far more computationally efficient than the approach of [41].

Attacking Session Keys. In our experiment, the attack described thus far allowed us to recover 7/8 columns of the master secret key. However, it is actually computationally hard to recover the remaining bits of the master key. So, instead, we turn to recover the session keys. We show a simple procedure that, when most of the master secret key is known, can recover the full session key by simply trying all possible values for the remaining bits of the session key. We show that this incremental session key recovery step can complete in only a couple minutes on a commodity laptop.

3.1.3 Attacker Model

Our attack relies on inducing bit flips during FrodoKEM's key generation process. We assume the common threat model for Rowhammer attacks. In this model, the attacker and FrodoKEM's key generation process run on the same physical hardware. This can occur with mutually distrusting virtual machines running on the same server, or when the attacker and victim run in two separate processes under the same OS. This threat model is consistent with the current body of literature on Rowhammer [146], as well as the majority of microarchitectural attack papers [147].

Finally, given that FrodoKEM is a CCA-secure scheme, we also assume that the attacker can induce the victim to decrypt honestly generated ciphertexts of the attacker's choice.

3.1.4 Artifacts and Current Status

Artifacts. For completeness, we provide our artifacts at this URL (See `Readme` first): https://github.com/a-as-plus-e/FrodoFLIP

Current Status. After acceptance of this paper, NIST selected Kyber as the initial post-quantum KEM [3]. Nonetheless, Frodo remains a post-quantum recommendation of Germany's BSI [73] and the core, mathematical foundation underlying Kyber [108].

3.1.5 Chapter Organization

The rest of this chapter is organized as follows. First, in Chapter 2, we review some necessary background and prior work as well as notation and definitions. Then, in Sections 3.2 - 3.5, we dive into the details of each component of our attack. Specifically, in Section 3.2, we describe how decryption failures can be used to recover a FrodoKEM secret key. Then, in Section 3.3, we describe how to use a Rowhammer attack to enable creation of decryption failures in FrodoKEM. In Section 3.4 we describe how to use a supercomputer to find sufficiently many failing ciphertexts for our attack. Finally, in Section 3.5, we conclude our attack by showing how we can recover a FrodoKEM session-key. We then discuss how our techniques can be extended to other lattice-based KEMs in Section 3.6 and possible countermeasures in Section 3.7.

3.2 Decryption Failure Attack on Rowhammer-Poisoned FrodoKEM

In this section, we assume the reader has familiarity with decryption failure attacks (see Section 3.1.2) and Rowhammer attacks (see Section 2.6). It will also be helpful for the reader to reference Section 3.1.2 for an overview of our attack and our high-level strategy of combining a decryption failure and Rowhammer attack.

Given the above background, we begin by highlighting an important difference between our attack and a traditional decryption failure attack [18, 25, 42, 43, 46, 47, 52, 112]: In a traditional decryption failure attack, the failing ciphertexts contain a significant amount of information on the secret key, so that only a few thousand failures are required for a full key recovery. Because our effective failure threshold is so much lower, the failing ciphertexts generated using our Rowhammer-altered public key contain less information. The failure event occurs orders of magnitude more often for randomly generated ciphertexts, and in turn, the fraction of secret keys that are consistent with a single failure event is far higher. This induces a trade-off in which we require significantly more failing ciphertexts to gain enough information to recover the key, but require less computation *overall*, since it is much easier to find failing ciphertexts. It is important to quantify this trade-off, as it informs which specific bits of the public key to target with Rowhammer. First, we will analyze the effects of Rowhammer on the decryption failure rate, which gives us an estimate for the total amount of work required to generate a *single* failing ciphertext.

3.2.1 Decryption Failure Rate Analysis

The decoding error in Frodo640 is an 8×8 matrix

$$\mathbf{E}''' = \mathbf{S}'\mathbf{E} - \mathbf{E}'\mathbf{S} + \mathbf{E}''. \tag{3.1}$$

The decoded message, which is also an 8×8 matrix, is correct in each of its entries if the corresponding entry in $\mathbf{E}'''$ mod q is in the interval $[-4096, 4096)$. Otherwise, the decoded entry is incorrect.

Consider the entry $e'''_{i,j}$ at position (i, j) of $\mathbf{E}'''$. Let $\mathbf{s}'_i$ and $\mathbf{e}'_i$ be the i-rows of $\mathbf{S}'$ and $\mathbf{E}'$ respectively, and $\mathbf{e}^j$ and $\mathbf{s}^j$ be the j-columns of $\mathbf{E}$ and $\mathbf{S}$ respectively. Then,

$$e'''_{i,j} = \langle \mathbf{s}'_i, \mathbf{e}^j \rangle - \langle \mathbf{e}'_i, \mathbf{s}^j \rangle + e''_{i,j}.$$

Let χ be the error distribution in Frodo. If key generation is executed correctly, $\mathbf{e}^j$ is sampled from χ^n. However, due to row-hammering, the true distribution of $\mathbf{e}^j$ is $\chi' \times \chi^{n-1}$, where χ' is defined as follows. Let $p_\chi : \mathbb{Z} \to \mathbb{R}$ be the density function of χ. Then, $p_{\chi'}$ is defined by

$$p_{\chi'}(x) = \begin{cases} p_\chi(x) & \text{if } -12 \leq x < 0 \\ p_\chi(x - 256) & \text{if } 256 \leq x < 269 \\ 0 & \text{otherwise} \end{cases} .$$

For Frodo640, the support of p_χ is $\{-12, \ldots 12\}$ whereas the support of $p_{\chi'}$ is

$\{-12, -11, \ldots, -1, 256, 257, \ldots, 268\}$. The reference implementation of FrodoKEM uses 16-bit integers. Here, we are targeting an attack that only requires that one bit flip per column of $\mathbf{E}$, since requiring more bit flips can reduce the success of the Rowhammer attack. By forcing the eighth-order bit of an entry to 1, the Rowhammer attack effectively adds 256 to any positive value and leaves negative values unchanged. Thus explaining the distribution of χ' shown above. Indeed, this analysis can be repeated for other orders of bits. We also consider the failure probabilities for a sixth-order (64) bit Rowhammer and a seventh-order (128) bit Rowhammer.

With $\mathbf{e}_j$ sampled from $\chi' \times \chi^{n-1}$, an honest encapsulation has decoding error distributed as

$$e'''_{i,j} \sim \chi \cdot \chi' + (\chi \cdot \chi)^{\circledast(2n-1)} + \chi, \tag{3.2}$$

where the notations are as follows. If χ_1 and χ_2 are two distributions, then $\chi_1 \cdot \chi_2$ is the distribution of the product of 2 independent random variables having distributions χ_1 and χ_2 respectively. Moreover, $\chi_1 + \chi_2$ is the convolution of χ_1 and χ_2, which is the distribution of the sum of 2 independent random variables having distributions χ_1 and χ_2 respectively. Finally, $\chi_1^{\circledast k} \coloneqq \underbrace{\chi_1 + \chi_1 + \cdots + \chi_1}_{k \text{ times}}$. Concretely, by explicitly computing the distribution in equation (3.2), we have that such $e'''_{i,j}$ causes decryption failure with probability approximately $2^{-27.8}$ ($2^{-113.2}$ for a 64-bit Rowhammer, and 2^{-76} for a 128-bit Rowhammer).

For our adversarial attack, $\mathbf{s}'_i$ is not honestly sampled from χ^n. Instead, we filter the output of the random oracle and select only the values $\mathbf{s}'_i$ that have ± 12 in the same coordinate where χ' is in $\mathbf{e}^j$. Let τ be the distribution with probability 1/2

at 12 and -12. Then, our adversarial $e'''_{i,j}$ has the distribution

$$e'''_{i,j} \sim \tau \cdot \chi' + (\chi \cdot \chi)^{\circledast(2n-1)} + \chi.$$

Concretely, such $e'''_{i,j}$ exceeds the decoding threshold of Frodo640 with probability approximately 2^{-13} ($2^{-98.7}$ for a 64-bit Rowhammer, and 2^{-61} for a 128-bit Rowhammer). To obtain one such $\mathbf{s}'_i$, the expected number of calls to the random oracle is 2^{15}. If there are c target columns of $\mathbf{E}$, since there are 8 rows of $\mathbf{S}'$, the expected number of calls per target column is $2^{15}/(8c)$. In our experiment, $c = 7$.

We note that there is a large variation in decryption failure rates conditioned on whether χ' is negative. The decryption failure rates conditioned on χ' being negative are negligible. Therefore, the decryption failure rates calculated above are negligibly different from decryption failure rates conditioned on χ' being positive. So we can mount our attack to recover $\mathbf{e}^j$ and $\mathbf{s}^j$ with probability $\Pr[\chi' > 0] = \Pr[\chi \geq 0]$.

Next we will describe our key recovery procedure, and analyze the number of failing ciphertexts necessary for recovering (each column of) the secret.

3.2.2 FrodoKEM Key recovery

Each failing entry of the error matrix in (3.1) gives rise to a linear inequality involving a column from each of $\mathbf{S}$, $\mathbf{E}$, a row from each of $\mathbf{S}'$, $\mathbf{E}'$, and the matching coordinate from $\mathbf{E}''$. Let $e'''_{i,j}$ be a failing entry of the error matrix. Assuming we exceed the failure threshold t in the positive direction, we obtain the following linear

inequality

$$e'''_{i,j} = \langle \mathbf{s}'_i, \mathbf{e}^j \rangle + e''_{i,j} - \langle \mathbf{e}'_i, \mathbf{s}^j \rangle \geq t$$

$$\langle (\mathbf{s}'_i \| - \mathbf{e}'_i), ((\mathbf{e}^j)^T \| (\mathbf{s}^j)^T) \rangle \geq t - e''_{i,j}$$

Let us first consider a standard decryption failure attack with adjusted failure threshold $(t - e''_{i,j})$. In this attack, the adversary generates honestly distributed ciphertexts by running the encryption algorithm (this is enforced in practice by the Fujisaki-Okamoto transform) and queries them to the decryption oracle. The attacker then collects all the ciphertexts that led to decryption failure. We will first show how to perform key recovery in the above setting. We will then explain why our Rowhammer attack essentially reduces to a standard decryption failure attack with a further adjusted failure threshold $(t' - e''_{i,j})$, where $t' = t - 12 \cdot e_{i,j}$, and one fewer dimension.

In the above attack, $\mathbf{S}'$ and $\mathbf{E}'$ that produce failing ciphertexts are correlated with $\mathbf{E}$ and $\mathbf{S}$ respectively. Given a set of failing ciphertexts, there are various ways to use the correlation between the failing ciphertexts and the LWE secrets to learn information about the LWE secret. For example, D'Anvers et al. [42] use the correlation to calculate explicit posterior probabilities for the values of the secrets. These then allow them to calculate revised distributions of the secret with a smaller variance.

The posterior probabilities calculated as in D'Anvers et al. [42] are difficult to analyze. We therefore instead consider the distribution of failing ciphertexts as

in [41]. This distribution is parametrized by the LWE secret itself, which is, of course, unknown. We then use our supply of failing ciphertexts–which constitute random samples from this distribution–to learn the hidden parameters. Specifically, as observed in [41], failing ciphertexts can be decomposed into a single component, $\boldsymbol{a}$, distributed as a truncated, univariate Gaussian in the direction of the secret, and components distributed as independent Gaussians in the directions orthogonal to the secret.

Note that for FrodoKEM, the secret and error distributions are discrete approximations of spherical Gaussian distributions. However as we will elaborate on below, the distribution of the failing ciphertexts can be very well approximated as a (continuous) spherical multivariate Gaussian distribution with known variance and unknown mean (where the mean depends on the LWE secret). We can thus reframe the key recovery problem as the problem of estimating the mean of a multivariate Gaussian distribution with a known covariance matrix, given samples from that distribution.

Modeling Failing Ciphertexts More formally, let us assume that $\mathbf{s}, \mathbf{e}, \mathbf{s}', \mathbf{e}'$, and e'' are as above, are all drawn coordinate-wise from a Gaussian distribution with zero mean and covariance σ_{se}^2. We omit the explicit row and column coordinates to reduce notational clutter, as the following holds regardless of the location of the failing coordinate in $\mathbf{E}'''$. Then, let us assume the norm of $(\mathbf{e}^T \| \mathbf{s}^T)$ is exactly $\ell = \sqrt{n}\sigma_{se}$, where n is the dimension of $(\mathbf{e}^T \| \mathbf{s}^T)$. This assumption is rather innocuous due to the high concentration of the norm of a Gaussian vector. As $(\mathbf{s}' \| -\mathbf{e}')$ is the part of the failing ciphertext that induces the failure condition, we can condition its distribution

on the learned linear inequality

$$(\mathbf{s}'\| - \mathbf{e}') \sim \mathcal{N}(\mathbf{0}, \sigma_{se}^2 \mathbf{I}_n) \mid \langle (\mathbf{e}^T \| \mathbf{s}^T), (\mathbf{s}'\| - \mathbf{e}') \rangle \geq t - e''$$

After conditioning, $(\mathbf{s}'\| - \mathbf{e}')$ decomposes into $(\mathbf{s}'\| - \mathbf{e}') = \frac{a}{\ell}(\mathbf{e}^T \| \mathbf{s}^T) + \mathcal{W}'$ where $\mathcal{W}'$ is a random vector distributed as a Gaussian of covariance $\sigma_{se}^2 \mathbf{\Pi}^{\perp}_{(\mathbf{e}^T \| \mathbf{s}^T)}$ (i.e. independent noise in all directions orthogonal to the secret) and a is a random variable that is independent of $\mathcal{W}'$ and follows a distribution that we denote $\mathcal{N}^{\geq t/\ell}_{\sigma_{se}}$—the univariate Gaussian of variance σ_{se}^2 conditioned on $a \geq (t - e'')/\ell$.

In essence, each failing ciphertext gives a draw from the decomposed distribution. If we scale each sample by $\frac{\ell}{a}$, we can simulate draws from the distribution

$$\mathcal{D} := (\mathbf{e}^T \| \mathbf{s}^T) + \mathcal{W}''$$

where $\mathcal{W}'' = \frac{\ell}{a}\mathcal{W}'$. However, we do not know the value of a for a given failing ciphertext. Instead, we will instead make use of the following heuristic. We know that $a \geq (t - e'')/\ell$. Thus, we fix $\alpha = (t - e'')/\ell$ for a, which introduces additional noise into the distribution of $\mathcal{W}''$. If we substitute α for the true mean of a in the calculation of the variance, we obtain

$$\mathbb{E}_{\chi \leftarrow \mathcal{N}^{\geq t/\ell}_{\sigma_{se}}}[((t - e'')/\ell - \chi)^2]$$

This quantity can be shown to be $\leq \sigma_{se}^2$ for any $(t - e'')/\ell \geq 0$. As such, we assume the additional noise is also a Gaussian with mean $\mathbf{0}$ and variance σ_{se}^2.

Thus, we approximate the total noise $\mathcal{W}''$ as a zero-centered multivariate Gaussian with a coordinate-wise variance of at most

$$(\ell/a)^2 = (\ell^2/(t-e''))^2\sigma_{se}^2 = n^2\sigma_{se}^6/(t-e'')^2.$$

Recovering the secret can be performed by obtaining enough failing ciphertexts to estimate the mean of $\mathcal{D}$.

To estimate the mean, we can simply draw sufficient samples from $\mathcal{D}$, take their average, and round coordinate-wise to the nearest integer. After averaging m failing ciphertexts, we obtain a sample from the distribution

$$\mathcal{D}' := (\mathbf{e}^T \| \mathbf{s}^T) + \mathcal{W}''_{(m)},$$

where the error $\mathcal{W}''_{(m)}$ is again a Gaussian and the variance of each coordinate is at most $n^2\sigma_{se}^6/(t^2 m)$. Note here that after averaging, the effects of individual e'' on $\mathcal{D}'$ can be ignored as e'' is zero-centered. For the key recovery to succeed, we require the magnitude of each error coordinate to be less than 0.5.

Thus, for a given value of m, the success probability of the attack is

$$\begin{aligned} P(\mathsf{Success}) &= P(|w''_{(m)i}| < 0.5 \ \forall \ i) \\ &= P(|w''_{(m)i}| < 0.5)^n \\ &= (1 - 2 \cdot P(w''_{(m)i} < -0.5))^n \\ &= \left(-\mathtt{erf}\left(\frac{-0.5 \cdot t\sqrt{m}}{\sqrt{2} \cdot n\sigma_{se}^3}\right)\right)^n \end{aligned} \tag{3.3}$$

Given a target success probability, we then solve for m numerically. Note that it is certainly possible to compute a more accurate value for a numerically, and use it in the calculation of the mean of $\mathcal{D}'$. In a practical attack scenario, both $\alpha = (t - e'')/\ell$ and $\mathbb{E}[a]$ should be used to calculate the candidate secret, as the specific values of the true secret (e.g. if it has a higher than expected norm) can affect the number of failing ciphertexts required when using the heuristic value for a. Using the heuristic for estimation of success probability is preferred, as it is more conservative (i.e. a smaller a results in a larger variance for $\mathcal{W}''$).

Comparison with the 'Hint' framework of [41] We note that decryption failure attacks can also be handled by the DBDD and 'hint' framework of [41], where a single failing ciphertext is viewed as a full dimensional approximate hint on the LWE secret/error. In general, integrating full dimensional approximate hints in the framework of [41] requires explicit inversion of matrices defined over the rationals, which is computationally prohibitive. Due to this, Dachman-Soled et al. provided a lightweight version of their framework, which allows one to estimate the concrete hardness of performing a lattice reduction-based attack in the decryption failure setting. However, the lightweight version is not suitable for our purposes here since we are interested in a full key recovery, not just hardness estimation. We attempted to use the full hints framework of [41] since the matrices that needed to be inverted were all diagonal. Therefore, integrating a single full dimensional approximate hint could be performed reasonably quickly. However, as discussed previously, our attack requires significantly more failing ciphertexts than a traditional decryption failure attack (on the order of 2^{17} failing ciphertexts see Figure 3.1). This additional factor

resulted in the computational cost of using the full framework being untenable. An improved implementation, using parallelism or accelerators would be required to be competitive with our simple average-then-round approach.

3.2.3 Attack Modifications for Rowhammer Assisted Failures

We next explain why our Rowhammer-assisted attack can be viewed as a standard decryption failure attack with a lower threshold and one fewer dimension. More precisely, the exact Rowhammer pattern alters the effective decryption failure threshold. In our attack, one coordinate—$e_{i,j}$—in each column of $\mathbf{E}$ is increased by 256. To increase the chances of failure, we also search for 12s in the corresponding coordinates of $\mathbf{S}'$. This lowers the failure threshold for the remaining coordinates by $12 \cdot (256 + \bar{e}_{i,j})$, where $\bar{e}_{i,j}$ is the original, unhammered value of $e_{i,j}$. We denote this new, smaller threshold as $t' = t - 12 \cdot e_{i,j} = t - 12 \cdot (256 + \bar{e}_{i,j})$.

Note that since we fix one coordinate of $\mathbf{s}'$, the distribution of that coordinate does not depend on the LWE secret as described in the decomposition given in 3.2.2. As such, we simply ignore that coordinate when performing the averaging, thereby reducing the dimension by one. To calculate the proper value for $t' - e''_{i,j}$, we must guess this value in advance for each column. As the error distribution for Frodo-640 has 25 possible values, we simply try all values for $\bar{e}_{i,j}$, and check our recovered key using the LWE equations. The same failing ciphertexts can be used for each candidate value of t', so only the averaging step (which is computationally trivial) needs to be run 25 times. See Figure 3.1 for the relationship between the failure

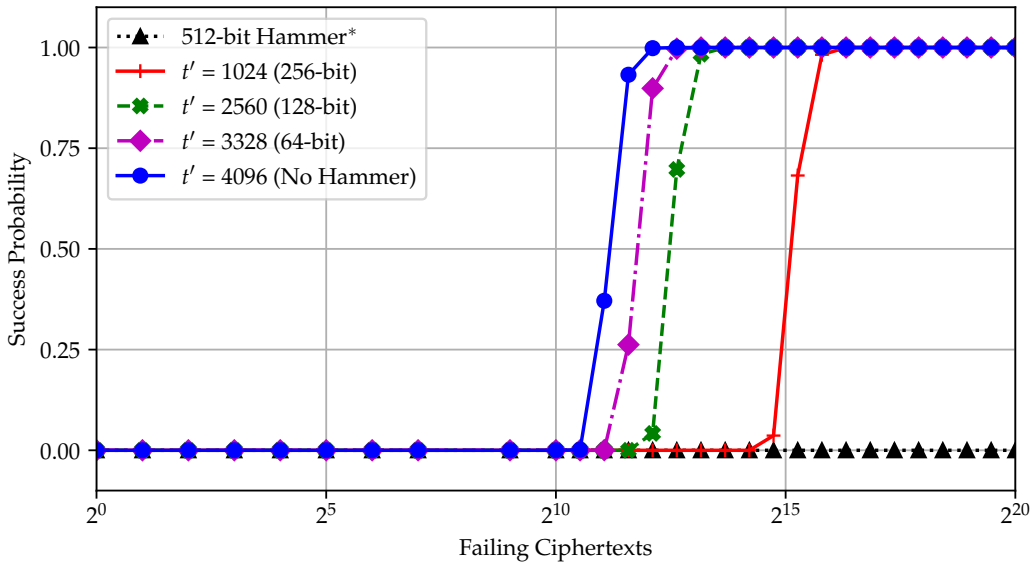


Figure 3.1: Probability of recovering a single column of the Frodo-640 secret (Equation 3.3), for given numbers of failing ciphertexts. Each line denotes a different (adjusted) threshold t' corresponding to rowhammer bit position.

threshold $t' - e''_{i,j}$, and the number of failing ciphertexts m required to mount our attack with various success probabilities. Approximately $100k$ failing ciphertexts are required to succeed with probability close to 1 for a 256-bit rowhammered column. Note that flipping the 512-bit of an entry in $\mathbf{E}$ results in a completely unrecoverable secret, as decryption failure is virtually guaranteed assuming the same filtering behavior.

3.2.4 Total Attack Cost

We present the cost of the attack in the total number of ciphertexts (both succeeding and failing) needed to recover a single column of the secret key. We combine the prior analysis in 3.2.1 and 3.2.2 for 64, 128, and 256-bit rowhammers to produce the graph seen in Figure 3.2. From this, we can conclude that it is indeed best

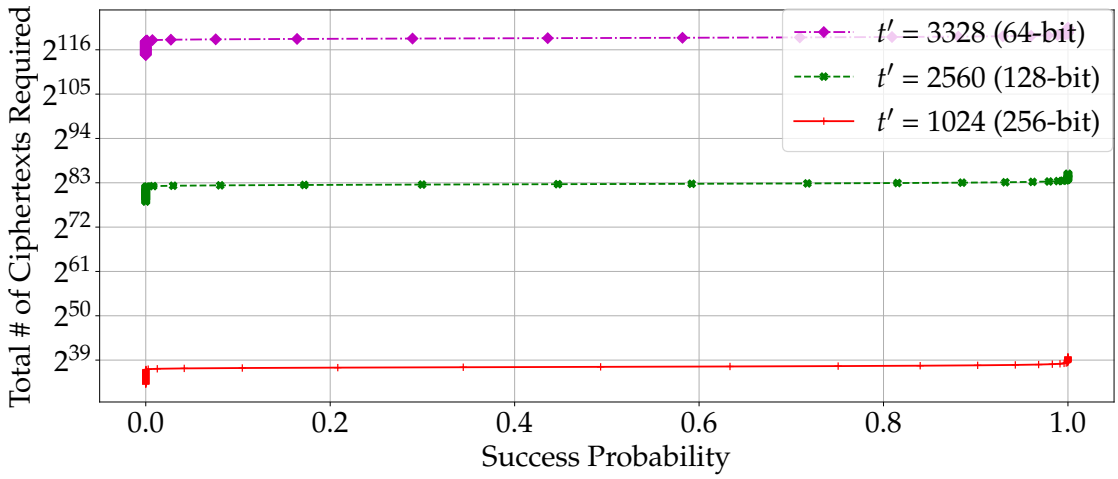


Figure 3.2: Total number of ciphertexts during failure-boosting (not decryption queries) required to recover a single column of the Frodo-640 secret, for (adjusted) thresholds t' corresponding to rowhammer bit position.

to target the 256-bit position with the rowhammer. The total number of ciphertexts we require to generate under failure-boosting to succeed with probability close to 1 is 2^{39} for the 256-bit rowhammer, compared to 2^{86} for the 128-bit rowhammer.

3.3 Poisoning Hobbits: Rowhammering a FrodoKEM Public Key

As outlined above, the main idea of our attack is to "poison" FrodoKEM's public key generation using Rowhammer. More specifically, we aim to flip specific bits inside the error term $\mathbf{E}$ computed during FrodoKEM's public key generation, obtaining a public key with a higher error. For FrodoKEM-640, the error term is an array of 640 x 8 values where each value is 2 bytes. Each of the values in the error matrix $\mathbf{E}$ has about an equal likely chance to be either positive or negative. By adding values to known locations in each of the matrix's columns, an attacker can increase the key's decryption failure rate. Finally, using the knowledge of flipped bit

positions, the attacker can feasibly find a large number of failing ciphertexts, thereby enabling cryptanalytic attacks on FrodoKEM and ultimately recovering the secret key.

A Balancing Act. While the above description is conceptually simple, we note that to extract the secret key in practice, a delicate balance of Rowhammer-induced bit flips is required to execute our attack. More specifically, too many high-order bit flips in the error term boosts the decryption failure rate to an excessive rate, causing failures to be prevalent for both the attacker and honest users. This will alert the target to the possibility that they are under attack, causing them to retire the poisoned key. On the other hand, too few bit flips will cause the decryption failure to be too low, giving the attacker only a slight advantage over honest decryption failures. This will result in an infeasible amount of computation to determine the key exchanged by the KEM.

3.3.1 Experimental Setup

Unless noted otherwise, we performed all of our experiments using a desktop containing a 3.4 GHz i7-4770 CPU and two Samsung DDR3 4 GiB 1333 MHz non-ECC DIMMs (part number M378B5273CH0-CH9). Out machine was running a fully updated Ubuntu 20.04. To simplify the attack, we disabled the machine's address-space layout randomization (ASLR). We note that ASLR was breached on numerous prior occasions [36, 59, 63, 131] making this assumption not overly-restrictive.

The victim process is the FrodoKEM reference code, compiled using the reference optimization level and Shake-128 flags. Additional code was added to

FrodoKEM to output files for the public key, secret key, and error matrix. These files were used for confirming the results of the attack. We took care to add these additional lines of code only after the attack was complete, to not artificially increase the window of opportunity for the Rowhammer attack.

3.3.2 Determining Useful Bit Flip Locations

Prior to mounting a Rowhammer attack, we must first determine which bits inside FrodoKEM's error matrix **E** are useful for the attacker to flip. To that aim, we used code provided in the FrodoKEM submission package in order to examine how specific combinations of bit flips affect the decryption failure rate. As shown in Figure 3.1, we found that flipping bit 8, henceforth called a 256-bit flip because it adds $2^8 = 256$ to the value, in one value per column increased the decryption failure rate to about 2^{-27}, compared to a failure rate of $2^{-138.7}$ mentioned in Frodo-640's specification [8]. This achieves a nice balance of minimizing the number of required bit flips, while still raising the decryption failure rate enough to ensure the feasibility of our attack.

Given that there are 8 columns, the attack then requires a total of 8 256-bit flips in order to fully recover the secret key. Without all 8 flips, parts of the secret key must be recovered through a form of brute force search.

3.3.3 Profiling Memory

Prior to starting the Rowhammer attack itself, we must profile the machine's memory in order to locate the physical locations of pages containing bits susceptible to Rowhammer. We accomplished this using the memory massaging techniques described by [89]. More specifically, the method of [89] involves exploiting Linux's buddy allocator to obtain 2 MB of contiguous memory regions required for precise double-sided Rowhammering. Unfortunately, following the disclosure of [89], modern Linux versions restrict access to the `/proc/pagetypeinfo` file used by [89] to root users. While this does required us to use elevated privileges to run our attack, a concurrent work by [138] demonstrated how to perform the same attack on the allocator by using the world readable `/proc/buddyinfo` file, avoiding the use of root privileges.

After attacking the allocator to obtain tuples of memory locations for double-sided rowhammering, we iteratively hammer each pair of aggressor rows and then check the victim row for bit flips. Since bit flips are repeatable, we store the map of which bit flips were within each page so that we can select which pages are best to use for the attack.

Filtering Candidate Pages. For FrodoKEM-640, the error matrix has dimensions 640 x 8 of 2 bytes values, resulting in the matrix spanning 2.5 pages of size 4 KiB. Therefore, the error matrix can either be spread across 3 or 4 pages, depending on whether the array's page offset is past 0x800. Because of this, there must be multiple columns containing a 256-bit flip per page to satisfy all 8 required flips. Additionally,

the first and last page storing the error matrix also store other nearby variables in the program; on our victim, this ended up being the S matrix and the array for storing randomness, respectively. While it may be possible to tolerate bit flips in S, we only considered pages containing no flips in these locations.

When choosing which 3 pages to use for our attack, we also must filter out both pages that have insufficient bit flips in the desired locations, and pages that have too many bit flips in undesired locations. Firstly, we filter out all sets of pages that don't provide exactly one 256-bit flip in at least 7 of the 8 columns. Otherwise, the Rowhammer attack will not increase the decryption failure rate enough for our attack to succeed. Equally important is that the pages do not exhibit bit flips during the attack in locations that will increase the decryption failure rate to be *too high*. For our attack, this means that any bit flips in bit positions 9 through 15, henceforth referred to as *high-order* bit flips, are not allowed. This is because increase in the error term will be very large in magnitude, and have an overly strong affect on the decryption failure rate. Additionally, only a small number of bits in positions 5 through 7 can be tolerated, since they will also have a substantial effect on the decryption failure rate, albeit less than the higher order bit flips. All other lower bits have an insignificant effect on the decryption failure rate.

Bit Suppression. With regards to requiring a bit to not flip, this can be accomplished by either choosing a page that doesn't contain a bit flip in that location, or by *suppressing* the undesired bit flip. As noted by [39], *stripe patterns*, where the bits above and below a given bit are the opposite value (i.e. 0-1-0 and 1-0-1 configurations), are the most likely to yield bit flips, while *uniform patterns* (

i.e. 0-0-0 or 1-1-1) are unlikely to result in bit flips. Since bits can only ever flip in one direction, this means that we can use 1-1-1 patterns to suppress any 1-to-0 bit flip, and 0-0-0 patterns to suppress 0-to-1 bit flips.

By using the appropriate uniform patterns in the positions where the undesired bit flips are located, we significantly reduce the chance that those bit flips occur during the Rowhammer attack. We used this technique to suppress 117 bit flips in the chosen pages, leaving only 4 bits that could not be suppressed.

We note that not all undesired bit flips need to be suppressed, as there is a 50% chance that the randomly chosen value in **E** of the bit will already be the value that the bit can flip to. This means that for each unsuppressed bit flip, the probability of the attack succeeding decreases by a factor of 2. For example, if 3 bits cannot be suppressed, the attack succeeds 1 out of every 8 attempts.

3.3.4 Allocating Pages to Victim Process

After profiling the memory, the attacker must force the FrodoKEM victim to allocate its memory in a way that stores the secret term E in the attacker's chosen pages. This is accomplished through exploiting the Linux page frame cache in a technique termed "Frame Feng Shui" [89]. The Linux page frame cache stores frames in a first-in-last-out data structure, and returns the most recently deallocated page when receiving a request for a page frame. Therefore, if FrodoKEM allocates a predictable number of frames before the target **E**, the target can be forced into a page of our choice.

First, multiple 4 KiB junk pages are allocated using mmap and the MAP-POPULATE flag to fill the bottom of the page frame cache. The number of junk pages is determined by how many pages the victim process allocates before allocating the target value. Next, the attacker chooses pages from the profiling phase that contain vulnerable bits in the correct positions. The attacker then deallocates the selected page frames using `munmap`, putting them at the top of the page frame stack. Immediately after this, the attacker unmaps all of the previously mapped junk pages, putting these pages on top of the selected pages in the stack. Finally, the the attacker induces the victim process to run its key generation process, during which the FrodoKEM process requests memory to store E, and the buddy allocator returns the pages chosen by the attacker. We used Frame Feng Shui against the FrodoKEM-640 scheme with a total of 297 junk pages to land the E error matrix into the selected pages.

3.3.5 Performance Degradation

Now that the attacker has forced the victim FrodoKEM to allocate the selected pages for the error matrix, and the spatial precision of the bits is sufficient, we must now ensure that the temporal precision of the bit flips is sufficient. That is, the attacker must flip the bits within a precise window during the execution of the FrodoKEM key generation. By examining the source of the reference code, we identified that the bit flips must occur after the generation of E by sampling the Gaussian in the `crypto_kem_keypair` function, and before E is added to AS in the

`frodo_mul_add_as_plus_e` function. In our setup, this window takes roughly 8ms, whereas we empirically found that we required 1300 ms to reliably succeed with the Rowhammer attack. To bridge this gap, we conducted a *performance degradation* attack against FrodoKEM.

A performance degradation attack [10] functions by rapidly forcing the most frequently executed FrodoKEM instructions to be evicted from the cache, which causes the CPU's pipeline to be flushed in a machine clear upon detecting the invalid tag in the L1i cache. This can result in dramatic decreases in performance if it occurs frequently enough. To evict the cacheline, we used the `CLFLUSH` instruction, available on x86 machines to all unprivileged users, to completely evict cachelines from the cache hierarchy.

Choosing a Cacheline. We statically analyzed the FrodoKEM victim's binary to find the most frequently executed cache line of code, and found that the usage of Shake-128 involves a tight loop that executes the `store64` instruction, resulting in 880992 calls to `store64` within the execution window.

This makes `store64` the optimal choice for performance degradation, so we implemented our performance degradation attack by running FrodoKEM-640 on a single core, with performance degraders running on both the sibling core and 2 other virtual cores, including the core running FrodoKEM-640. Without any performance degradation, the time hammering window is 8.2 ms. With performance degradation on only the sibling core, this can be increased to around 663 ms. With additional cores running the degrade code, this reached about 1300 ms. Using the performance degradation code with multiple cores, this allowed for sufficient time to complete the

Rowhammer attack within the alotted window.

3.3.6 Results

For the attack against FrodoKEM-640, the page offset of the error matrix **E** was 0x9b0, meaning that it spanned a total of 4 pages. To prevent hammering any of the shared values on the first page, only the last 3 pages were targeted for the rowhammer attack. Using the criteria listed above, we found 3 pages that satisfy 7 of the 8 256-bit flips. Therefore, additional steps are required at the end for full key recovery. Also, bits in 4 matrix locations were found that could not be suppressed. Thus, accounting for the 7 256-bit locations that must be positive for the flip and the 4 locations that could not be suppressed, the attack succeeded 1 out of every 2^{11} attempts.

3.4 Searching for Failing Ciphertexts with a Supercomputer

In the construction of the poisoned public key, we intentionally keep the failure rate for honestly generated ciphertexts low so that the attack may persist for the duration of a generated key without the victim identifying the issue. This has the effect that finding a failing honestly generated ciphertext is still exceedingly rare. In the key established in Section 3.3, the failure rate of honestly generated ciphertexts is approximately 1:2,700,000. In order to generate approximately 100k failing ciphertexts per column, that would necessitate the creation of over 1 trillion connection requests. Alternatively, the adversary could generate ciphertexts that are

likely to fail, but that would fail the Fujisaki-Okamoto transform.

To meet these constraints, our attack is predicated on a setup phase to generate honest ciphertexts that are *likely to fail* based on the profiled machine and the targeted columns. We use the $\mathbf{A}$ matrix from the public key, and then iterate on the random seed that initializes μ as input to the Frodo.Encode function. Ordinarily, μ is initialized by a randomly sampled 16-byte seed, and thus there are 2^{128} potential seeds. The encode function continues as specified until the $\mathbf{S}'$ matrix is generated. The superset of the rows and columns targeted by Rowhammer are checked to determine if they have high values (i.e. -12 or +12) that would make them candidates to potentially fail to decode. These candidate ciphertexts are saved to attempt decoding. For the poisoned key established in section 4, otherwise honestly generated ciphertexts (i.e. those that pass the FO transform) pass this filter at a ratio of approximately 1:1175.

For our proof-of-concept attack, all candidate ciphertexts are passed as input to the Frodo.Decode function. Ciphertexts that fail are saved to a file including the μ seed to generate the ciphertext, the $\mathbf{S}'$, $\mathbf{E}'$ and $\mathbf{E}''$ matrices, and the ciphertext itself. The attacker would have access to all of these aspects of the ciphertext in a distributed attack. The components are concatenated to files named for the row/column pair where the high value in the $\mathbf{S}'$ matrix is located (e.g. 'mu-4-8.csv'). This helps to identify the likely column that created the failing condition, which aids the key-recovery process. So, we only considered matrices that had a single ± 12 in the checked locations.

To expedite the profiling stage we leveraged resources from two supercomputer clusters — namely Pittsburgh Supercomputing Center (PSC) [34] and Open Science

Grid (OSG) [111, 129]. Total ciphertext generation took 237,806 hours, distributed as follows: 104,856 core-hours on OSG through the Open Science Pool, and 132,950 core-hours on the Bridges-2 Regular Memory nodes from PSC. For PSC, the modified Frodo-KEM encode and decode functions were compiled directly on each compute node. For OSG, the modified code was compiled in a container for each compute node. Jobs were launched with an incrementing seed to generate 150 million ciphertexts per seed. The start seed was shifted 28 bits to the left and stored in μ as the start of the search. Each job produced approximately 128 thousand ciphertexts that passed the filter, from which approximately 20 ciphertexts failed to decrypt. In total 665,343 failing ciphertexts were used to recover 7 of the 8 columns, necessitating 1.53 billion decryption requests—less than 10% of the absolute maximum 2^{64} decryption queries identified by NIST to prevent failure attacks.

3.5 Completing the Attack: Session-Key Recovery

If the full master secret key is unable to be recovered, it is still possible to recover encapsulated session keys from honestly generated ciphertexts. The recovery procedure is enabled by the limited number of bits present in the message μ. Due to the Fujisaki-Okamoto transform, encapsulation is deterministic given the value of μ. If we can determine the value of μ, we can easily recover the session key.

In FrodoKEM.Decaps [8], μ is decoded from an 8x8 message matrix $\mathbf{M}$, which is computed from an honestly generated ciphertext and the secret key. As we are unable to compute $\mathbf{M}$ due to our lack of the full secret key, instead we compute $\mathbf{M}'$

by filling in the missing columns of the secret key with 0's in the computation of $\mathbf{M}$. Note that $\mathbf{M}' = \mathbf{M}$ except in the missing columns. For Frodo-640, the most significant 2 bits of each entry of $\mathbf{M}$ are extracted to produce μ in FrodoKEM.Decode (resulting in a message size of 128 bits). Thus, for every missing column of the secret key, we must brute-force 16 bits of μ. To check correctness, we simply pass each μ' to a modified version of FrodoKEM.Encaps that accepts μ as a parameter rather than sampling it at random. The deterministic nature of the encryption ensures that the output ciphertext, will be equal to the intercepted ciphertext from the victim when we pass in the correct value for μ.

The total cost of the session key recovery is $2^{16 \cdot \mathsf{missingcols}}$ times the cost of FrodoKEM.Encaps. In our attack, we managed to recover 7/8 of the columns of the secret key, and only needed to brute-force 16 bits. This procedure takes at most a couple of minutes on a commodity laptop. In a scenario where super computers are enlisted for this brute-force search in the online phase of the attack, more missing columns could be tolerated at the discretion of the attacker. Sessions could be captured in the meantime for later decryption once the brute-force completes.

3.6 Attacking Other NIST Lattice KEMs

While we only experimentally demonstrated an attack against FrodoKEM, we believe that similar attacks are theoretically possible on other latice-based KEMs; however, there are various reasons these attacks may be more difficult in practice.

Regarding the other lattice KEMs in the NIST PQC Round 3 candidate pool:

Algorithm 3.5.1 SessionKeyBF($\mathbf{SExp}, pk, col, ct = (ct_1, ct_2)$)

```
1: C_1 := FrodoKEM.unpack(ct_1)
2: C_2 := FrodoKEM.unpack(ct_2)
3: Compute M' = C_2 − C_1 · SExp
4: μ' := FrodoKEM.Decode(M')
5: for i := 0 → 2^16 do
6:     Insert(μ', col, i)                ▷ Insert i into col's bit positions in μ'.
7:     ct', ss := FrodoKEM.Encaps(pk, μ')
8:     if ct' = ct then return ss
9:     end if
10: end for
11: return FAIL
```

Figure 3.3: Session key brute-force algorithm given the experimentally derived secret and missing column index.

Kyber, Saber, NTRU, sNTRUprime, and NTRU-LPRime, we note that the strategy will be similar. This is true because decapsulation failures occur with high probability when the product between a noise vector chosen during key generation and a second noise vector chosen during encapsulation is large. Therefore, the strategy is then to:

1. Introduce known additional noise during key generation using Rowhammer techniques, producing a "poisoned key"

2. Use knowledge of the additional noise to select valid ciphertexts that are likely (but not too likely) to produce a decapsulation failure when the honest party attempts to decrypt them using the "poisoned" key.

3. Accumulate information about the unknown parts of the private key by observing which of the ciphertexts are and are not successfully decapsulated.

In order to accomplish step 1, it is likely that one would need to do performance

degradation similar to what was done to FrodoKEM in our experiments. For Kyber and Saber, we identify places in the code where the SHAKE function is used between the time when noise is sampled and when it is used. Cryptographic random number generation is also done in sNTRUprime and NTRU-LPRime in a location that appears similarly useful. However, the reference implementation of each uses AES instead of SHAKE as the core primitive. Since AES has native hardware support, it is likely harder to do performance degradation against the reference implementation of NTRUprime. If the cryptographic random number generation could be performance degraded, sNTRUprime seems like a better candidate for attack than NTRU-LPRime, because in the latter case, the random number generator is called only once, while in the former case it is called repeatedly.

An additional difficulty arises when trying to adapt the attack to NTRU. In that case, the fact that algebraic operations switch between the rings $\mathbb{Z}_q[x]/(x^n - 1)$ and $\mathbb{Z}_q[x]/((x^n - 1)/(x - 1))$ means that any rowhammering which fails to preserve the sum mod q of the coefficients of a polynomial will cause decryption to fail with overwhelming probability for all ciphertexts. This makes it very likely that any attack will be detected, and means that decryption failures will give no information about the private key. It is unclear how to design an attack which avoids this issue.

With regard to step 2, the filtering step, there is an additional quantitatve challenge compared to our attack on FrodoKEM. In the case of FrodoKEM, we were able to Rowhammer a single bit per column of $\boldsymbol{E}$. This was possible because the parameters of the attack could be set so that a decryption failure only occurred with significant probability when one of 56 locations in the ciphertext noise attained

a maximal value. Since this only happened with low probability for an honestly generated ciphertext, filtering was possible. In all the other KEMs, each position in the key generation noise multiplies more positions in the ciphertext noise, and the ciphertext noise distribution has the maximum value occur much more frequently. As a result we expect that attacking any of the other schemes will require a somewhat higher density of rowhammerable bits. The filtering criterion in this case would require the products of several coefficients of the ciphertext noise with the rowhammered coefficients to be large and to have the same sign. We give more details on the methods that would need to be employed to attack the other schemes in Appendix A.1.

3.7 Possible Countermeasures

As described, our attack shows that Frodo is vulnerable against a real-world Rowhammer attack. This justifies further study of the Rowhammer security of any post-quantum protocols that will come out of the NIST standardization process. As such countermeasures are critical, we wish to highlight a few possible ones up front.

3.7.1 Rowhammer Defenses

There is a rich literature surrounding Rowhammer defenses, both proposed and deployed, that attempt to mitigate the Rowhammer bug. Despite these efforts, researchers have managed to circumvent all practically deployed mitigations and empirically demonstrate bit flips against them. Even so, a combination of such countermeasures to provide "defense in depth" may dramatically increase the effort

required to mount a Rowhammer attack.

Hardware Defenses. The two hardware defenses deployed in practice are Error Correcting Code (ECC) memory and the Targeted Row Refresh (TRR) mitigation. ECC memory, most commonly found on server machines, aims to correct bit flips when they are detected upon reading from memory. The memory controller stores additional "control bits" that act as a checksum, enabling bit correction and detection up to a certain threshold of errors. While the ECC mechanism was designed to mitigate errors due to cosmic rays, it also inadvertently assists in mitigating Rowhammer bit flips. [39] defeated ECC with Rowhammer, however, by causing a sufficient number of bit flips within an ECC word such that the mechanism can no longer even detect that a bit flip has occurred.

TRR is a hardware defense implemented in DDR4 that was long touted to be a panacea for Rowhammer. It uses counters to keep track of how many times each row in memory is accessed, and once the counter reaches a specified threshold, TRR automatically refreshes the nearby rows. If this threshold is below the minimum number of hammering attempts required for a Rowhammer attack, then this mitigation refreshes victim bits before they are hammered enough to flip. While this seems to be a perfect defense in theory, [55] were able to bypass TRR and flip bits on DDR4 due to limitations imposed by constraints in the hardware. In particular, TRR can only track accesses to a small number of rows at a time, and attackers can bypass it by hammering many rows at once.

Most recently, [96] addressed the shortcomings of TRR by proposing a scheme that tracks rows optimally, given a limited number of counters and additional refresh

commands. While they demonstrate that their algorithm achieves the best trade-off between these parameters, it still suffers from the inherent limit to how many row counters can be supported by the hardware.

Software Defenses. While no software level defenses against Rowhammer have seen widespread adoption, researchers have proposed mitigations with claims of low overhead and complete Rowhammer protection. [14] aim to backport the same principles behind TRR to DDR3. They use performance counters to determine which rows are accessed most frequently, and then refresh those rows before a bit flip occurs.

[33] aim to protect the kernel from user level Rowhammer attacks by physically separating the kernel memory on the DIMM from user memory. [87] go a step further and physically isolate all data rows in memory from each other.

3.7.2 Defenses Specific to our Attack

We also present practical countermeasures derived from unique insights gleaned from our end-to-end attack against FrodoKEM.

Hardware Accelerated Cryptography. As demonstrated in Section 3.3, having a sufficiently long time window to rowhammer is critical to the success of the attack. To extend the length of this window all the way from 8ms to 1300ms, we relied on a performance degradation attack against the SHAKE hash function, wherein we rapidly flushed a cacheline containing frequently executed FrodoKEM code. The effectiveness of the performance degradation is highly dependent upon how many calls FrodoKEM makes to the flushed cacheline – the one containing `store64` in this

case.

If, however, FrodoKEM were to use hardware accelerated cryptographic instructions in place of its in-software hash function implementation, the opportunity for the attacker to conduct a performance degradation attack would be greatly diminished. This is because the tight inner loop that calls `store64` within Shake would instead be replaced, for example, by just a few instructions from Intel's AES-New Instructions (AES-NI) instruction set. We thus observe that hardware accelerated cryptographic functions seem to be more resilient against performance degradation attacks.

Algorithmic Defenses. In addition, there are two potential algorithmic defenses to our attack. The first is to limit the surface of the performance degradation through proper sequencing of operations, even when not using AES-NI. The main idea is to not perform any expensive operations between the time when **S** and **E** are sampled and when **B** is generated. For example, one could use SHAKE to expand out **A** before **S** and **E** are generated (like in Kyber and Saber) and move the SHAKE calls used to generate the randomness for **S** and **E** before the actual sampling (like in FrodoKem and Saber). Ordering the operations in this manner significantly reduces the window for performance degradation, and therefore our attack.

The second defense is to guard the key generation process. One way to do this is to regenerate **A**, **S**, and **E** from randomness and compute **B** again. If the two **B** are not equivalent, then abort key generation. For the Rowhammer attack to pass this check, the entire attack must succeed twice in a row, for potentially different pages of memory. In addition, to ensure the equality of the **B** matrices, the Rowhammer attack must flip *exactly* the same bits each time. It is highly unlikely, however, that

the attacker would be able to find another set of pages that also exhibits the exact same bit flips under hammering as the original set.

Storing the error matrix E (Frodo overwrites E with B for efficiency) for the duration of the computation of the public key is another possible way to enforce the integrity of key generation. Then, if any value in E (or perhaps its distribution) is abnormal, recompute the public key from scratch. This has the downside of potentially alerting the attacker to the re-keying process, and enabling them to try to rowhammer the new key.

Active Defenses. Another line of defense would be to maintain an active state during the online phase to detect an attack in progress. An adversary needs to send a vast number of filtered ciphertexts to mount a successful key recovery attempt. A potential victim could check the distribution of received ciphertexts during the decapsulation process. Indeed, during the re-encryption check, the victim would be able to see the values of the randomness used during encapsulation (at least where decapsulation would ordinarily be successful). If the distribution of incoming ciphertexts is skewed towards large values (e.g. ± 12), then it is reasonable to assume that the victim is under attack. One must be careful in how to respond to this attack, however. If the decision is to simply re-key when an attack is detected, this leads to an easy Denial of Service attack (an attacker can perform the same filtering procedure to intentionally trigger the detector). On the other hand, if the response is to discard the received ciphertexts that contain such large values then this can lead to discarding communication from honest users.

Also, while this countermeasure would make our attack harder, it would not

make it very much harder – if the intensity of the rowhammering is modestly increased, the attacker can make the decryption failure on random ciphertexts sufficiently high as to not require ciphertexts to come from an unusual distribution. This would, however, result in a significantly higher decryption failure rate for honest parties, making the attack easier to detect.

Chapter 4: Revisiting Security Estimation for LWE with Hints from a Geometric Perspective

4.1 Introduction

Dachman-Soled et al. [41] created a toolkit for integrating so-called "hints" into uSVP instances that can then be solved via lattice-reduction algorithms. To achieve this, they introduced an intermediate lattice problem known as DBDD (Distorted Bounded Distance Decoding). A DBDD instance consists of three parts: A lattice Λ, a mean vector $\boldsymbol{\mu}$, and a covariance matrix $\boldsymbol{\Sigma}$. The original lattice Λ represents the lattice obtained through Kannan's embedding–which is a way to construct a lattice in which the LWE secret/error is the shortest non-zero vector. Subsequently, side information can sometimes be used to sparsify or reduce the dimension of the original lattice. The remaining parts of the instance $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, correspond to a mean vector and covariance matrix, and these represent distributional information known about the LWE secret/error. $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ originally represents the fact that the secret/error is drawn from a distribution with known mean/covariance determined by the specifications of the cryptosystem. Subsequently, it captures the conditional distribution on the secret/error, given the side information, in cases where this conditional distribution

remains well approximated by a Gaussian. Thus, certain types of information on the structure or on the distribution of the secret can be integrated into a DBDD instance, starting with the original instance, and then modifying Λ, and/or $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ appropriately. A DBDD instance can then be converted into a uSVP instance using *homogenization*—centering the ellipsoid at the origin—and *isotropization*—applying a linear transformation that simultaneously transforms the ellipsoid into a ball and transforms the lattice into a different lattice with higher volume. Finally, the resulting uSVP instance is fed into the BKZ [125] lattice reduction algorithm to obtain the shortest non-zero vector in the transformed lattice. This short non-zero vector allows direct recovery of the LWE secret/error. [41] demonstrated their methodology with numerous examples, and provided an open-source implementation to predict the security decay (i.e. reduction in the BKZ blocksize, β, required for key recovery) of an LWE instance given a set of hints.

The approach of the current work is to provide an alternate geometric interpretation to the distributional approach considered in [41]. We alter the reduction from LWE to DBDD, by viewing the solution of an LWE instance (with secret of dimension n and error of dimension m, for a total dimension $d = n + m$) as the (unique) integer point contained in an ellipsoid that is constructed from the given LWE instance. Thus, the problem of finding the unique integer point is captured by the DBDD instance $(\mathbb{Z}^d, \boldsymbol{\mu}, \boldsymbol{\Sigma})$. Here, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the center and the positive semidefinite "shape" matrix defining the ellipsoid.

We now view "hints" as geometric operations on the DBDD ellipsoid, as opposed to viewing a hint as inducing a conditional probability distribution (represented

by a mean and covariance) on the secret/error. In our framework, the information obtained from a hint corresponds to the intersection of the DBDD ellipsoid with another convex body such as a hyperplane, halfspace, or another ellipsoid. This region of intersection is itself a convex body, but may not be an ellipsoid. Thus, to obtain the updated DBDD instance, we compute (an approximation of) the minimal volume ellipsoid that circumscribes the region of intersection. Such ellipsoids are well-studied in the literature on convex geometry and are known as Löwner-John ellipsoids. Since the ellipsoid circumscribes the region of intersection, replacing the original DBDD ellipsoid with the new ellipsoid provably maintains the DBDD invariant that the LWE secret corresponds to a lattice point contained in the ellipsoid.

4.1.1 Benefits and Drawbacks of Our Geometric Approach

Our approach establishes a connection between ellipsoidal approximations in convex geometry and analysis of the impact of side information on the concrete hardness of an LWE instance. This opens up a body of literature which we only begin to tap into in this work.

One benefit of our geometric approach is that in some cases it can more naturally handle the type of information obtained from side channels. For example, in decryption failure attacks, the type of information obtained is exactly an inequality hint on the LWE secret and error. The prior work of [41] proposed an ingenious way of capturing the information obtained from a decryption failure, without the direct use of inequality hints. As we describe in more detail in Section 4.1.2, our

framework allows for direct incorporation of inequality hints as they correspond to the region of intersection of an ellipsoid and a halfspace. Our approach is also inherently compatible with continuous variants of LWE (see Remark 4.1.1 for further details).

Another benefit is that our approach removes the need for the Gaussian assumption. Not all natural distributions on the LWE secret and error are Gaussian. For example, the data obtained from the side-channel attack (SCA) of Bos et al. [30] (which we use for experimentation in Section 4.3.3) gives rise to a probability distribution on each secret key coordinate that is far from a Gaussian distribution. What does one do now if one would like to initialize a DBDD instance with this SCA distribution and then integrate additional hints? One approach is to simply treat the SCA distribution as a Gaussian and apply the hint formulas based on conditional Gaussians from the prior work. In this case, however, there are no guarantees that the DBDD invariant—that the LWE secret corresponds to a lattice point contained in the ellipsoid—holds after hint integration, and so the obtained BKZ-β estimates may be inaccurate (see Figure 4.2 for a case where underestimation of the ellipsoid norm impacts the accuracy of BKZ-β estimates). Our method provides rigorous guarantees even when the data distribution is non-Gaussian. Specifically, our approach can be viewed as a "worst case" approach that guarantees that the secret is contained in the evolving DBDD ellipsoid, even for the "worst case" distribution over the secret.

We show that in the perfect hint setting, even though the data *is* (approximately) Gaussian, our modeling leads to slightly more accurate estimates of β in

certain regimes. [1] Further, despite being a worst-case approach, we are able to show a setting in which our approach yields a decreased predicted BKZ-β, as compared to the prior work of [41]. This is possible since our modeling in that setting is fundamentally different from the prior work (we model decryption failures as inequality hints versus full dimensional approximate hints) so that the two approaches no longer correspond to worst-case/average-case estimates for the same process. Finally, we analyze a setting in which the distribution on the LWE secret, due to incorporation of side-channel data, is far from a Gaussian distribution. We show that our approach allows combining this SCA data with side information from an independent source, without resorting to Gaussian models. We elaborate on these examples in Section 4.1.2 as well as in Sections 4.3.1, 4.3.2, and 4.3.3, respectively. Direct comparisons with the prior work of [41] can be found in Figures 4.1 (in the perfect hints plot), 4.2, 4.1, and 4.2, respectively.

A drawback of our approach is that due to our worst-case modeling, we can get "stuck" and not make progress when integrating new hints. This occurs when the minimal circumscribing ellipsoid works out to be *equivalent* to the original ellipsoid. In Section 4.3.2 such a situation occurred in our experiments and we provide some techniques based on the unique geometry of the problem to allow for continued progress. We discuss constraints on hints for which this situation can occur at the end of Sections 2.8.1 and in Section 4.2.2 (see Theorem 4.2.1). In Section 4.1.3 we discuss additional approaches for circumventing this issue.

[1] We believe our improved accuracy is due to the fact that our modeling incorporates the true distances (w.r.t. the ellipsoid norm) of the intersecting hyperplanes from the center of the ellipsoid with each successive hint, whereas the average-case approach can be viewed as incorporating the expected distance each time.

Further, since our ellipsoids no longer represent Gaussian distributions, we cannot necessarily predict the expected length of the secret (with respect to the "ellipsoid norm") in the evolving DBDD instances. In some cases, we therefore need to scale the DBDD instance (in a way that depends on the actual LWE secret) in order to make accurate predictions on the hardness. We note that this additional scaling is needed only for hardness *estimates* (in which case the LWE secret is, in fact, known) but is not needed to launch a full attack, since the BKZ-β is agnostic to scaling of the instance. See Section 4.3.1 for a more detailed discussion of this phenomenon for the case of inequality hints.

4.1.2 Instantiations of our Approach

We introduce two new types of hints:

Inequality Hints. Here we consider the leakage of the information that $\langle \boldsymbol{v}, \boldsymbol{s} \rangle \geq \gamma$, where $\boldsymbol{v}$ is known and $\boldsymbol{s}$ is the LWE secret/error vector. Given our geometric perspective, this hint now exactly corresponds to the information that the LWE secret is contained in the intersection of the initial ellipsoid and the halfspace $\{\boldsymbol{x} \in \mathcal{R}^d \mid \langle \boldsymbol{x}, \boldsymbol{v} \rangle \geq \gamma\}$. Unfortunately, the geometric perspective does not seem helpful, as this region of intersection is no longer an ellipsoid! Instead, we *approximate* the region of intersection with an ellipsoid. We use the fact that one can efficiently compute the minimal volume ellipsoid (called the Löwner-John ellipsoid) that circumscribes the intersection of an ellipsoid and a halfspace [26]. Using this circumscribed ellipsoid

in our DBDD instance maintains our required invariant, yet the new ellipsoid has *smaller volume* (under the constraints given in Section 2.8.1), making the resulting uSVP problem easier. See Section 4.2.1 for details on integration of inequality hints and Section 4.3.1 for validation of our β estimates for these hints.

Inequality hints are useful in the decryption failure setting since the information that is learned from a decryption failure is exactly of the form of an inequality hint. We show that this yields improved estimation accuracy (see Figure 4.2) compared to modeling decryption failures as full dimensional approximate hints as in [41]. We then show our approach reduces the predicted β value required for key recovery, given a fixed number of decryption failures. We further describe a new, geometric-based failure boosting technique[2] obtained from our approach. See Section 4.3.2 for details and experimental results.

Combined Hints. Combined hints provide a way to "fuse" information from two DBDD instances into a single instance. To motivate this type of hint, consider a situation where we have two sources of side information for a single LWE secret/error, such as data from decryption failures, and data from a side-channel attack. The information from these sources is captured by the two DBDD instances $(\Lambda, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $(\Lambda, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ (for purposes of this example we assume the two lattices are equal in the two instances, but our techniques extend to the case in which the lattice differ).

One might consider using the conditional approximate hints of [41] to in-

[2]The term "failure boosting" (see [44]) refers to techniques that use information from previous decryption failures to increase the failure rate for subsequent queries.

tegrate the information from the second instance $(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ into the first instance $(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$. However, the formulas for conditional approximate hints given by [41] require both distributions to be Gaussian. If those formulas are applied when one or both sources are not well-approximated by a Gaussian, then the DBDD invariant may no longer hold for the evolved instance. In cases where the distribution over individual secret/error coordinates are independent, it is possible to use the *a posteriori* approximate hints of [41], even when the distributions are non-Gaussian. This approach essentially erases certain information from $(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$, and replaces it with corresponding information from $(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$. Our approach, which we discuss next, combines information from the two instances more effectively, rather than simply replacing one with the other.

We first observe that even when no distributional information is available, given the promise of the two DBDD instances, we can conclude that the LWE secret/error vector $\boldsymbol{s}$ lies in the intersection of the two ellipsoids corresponding to $(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$. This region is not necessarily an ellipsoid, so we cannot simply obtain a new DBDD instance by intersecting the ellipsoids. Instead, we adopt the "fusion" approach [122, 144] which is to find the convex combination of the two ellipsoids that optimizes the volume of the resulting ellipsoid. The optimal convex combination has the following properties: (1) It is an ellipsoid, (2) It is guaranteed to contain the intersection of the two ellipsoids, (3) It does not contain points that are outside both ellipsoids, and (4) Points on the surface of both ellipsoids are on the surface of the resulting ellipsoid. This approach is attractive since the fused ellipsoid can be obtained by solving a one-dimensional convex optimization problem, which is

computationally feasible. Further, the approach was shown to be equivalent to several other proposed relaxation methods for finding the optimal circumscribing ellipsoid [144]. See Section 4.2.2 for details on integration of combined hints and discussion of when the resulting ellipsoid achieves smaller volume than both input ellipsoids. Validation of our β estimates for these hints can be found in Section 4.3.1.

We illustrate our approach by using it to fuse information from decryption failures and side-channel leakage, reducing the predicted β value required to recover the secret as compared to the naive approach of combining the information. See Section 4.3.3 for details and experimental results.

Revisiting perfect hints. Once a DBDD instance has evolved via the integration of inequality or combined hints, we can no longer make the Gaussian assumption from the prior work. This means that if there are additional perfect hints, they can no longer be integrated using the prior method. We present a new algorithm for integrating "perfect hints" into an LWE instance that does not require any distributional assumptions. A perfect hint is the leakage of the information that $\langle \boldsymbol{v}, \boldsymbol{s} \rangle = \gamma$, where $\boldsymbol{v}$ is known and $\boldsymbol{s}$ is the LWE secret/error vector. We now view this hint as consisting of the information that the LWE secret lies in the the intersection of the current DBDD ellipsoid and the hyperplane $H := \{\boldsymbol{s} : \langle \boldsymbol{v}, \boldsymbol{s} \rangle = \gamma\}$. We note that the resulting intersection is itself an ellipsoid, thus maintaining our DBDD invariant. We also propose a different way to deal with non-homogenized perfect hints. All perfect hints from [41] were transformed so that the incorporated hint was $\langle \boldsymbol{v}', \boldsymbol{s}' \rangle = 0$ with $\gamma = 0$. This was needed in order to maintain the invariant that the

lattice part of the DBDD instance remains a lattice, and not a lattice coset. However, it also had the by-product that the hint vector $\boldsymbol{v}'$ is not in the span of $\boldsymbol{\Sigma}$, the shape matrix of the ellipsoid corresponding to the DBDD instance. For consistency with our geometric approach we require that our hint vectors $\boldsymbol{v} \in \text{Span}(\boldsymbol{\Sigma})$, and so we suggest an alternative technique for dealing with non-homogenized perfect hints. See Section 4.2.3 for more details.

Experimental results show that our β estimates improve accuracy when more hints are integrated, compared to the estimates of [41]. The actual β needed to recover the secret are the same across the two techniques, since the generated instances differ only by a scaling factor. See Section 4.3.1 for validation of our β estimates for these hints and comparison with the β estimates from [41].

Compatibility with approximate hints. Given an evolved DBDD instance $(\Lambda, \boldsymbol{\mu}, \boldsymbol{\Sigma})$, and $\ell \leq d$ number of approximate hints[3] each having independent error of standard deviation σ_e, we can write the hints as $\boldsymbol{s}\boldsymbol{V} \approx \boldsymbol{\gamma}$, where $\boldsymbol{V}$ is a $d \times \ell$ matrix with each column corresponding to a hint vector. The hints can be integrated by considering the set $\{\boldsymbol{x} : \|\boldsymbol{s}\boldsymbol{V} - \boldsymbol{\gamma}\|^2 \leq \ell \cdot \sigma_e^2\}$, which defines a (possibly degenerate) ellipsoid with mean and shape matrix $(\boldsymbol{\mu}', \boldsymbol{\Sigma}')$. We then apply a *combined hint* on $(\Lambda, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $(\Lambda, \boldsymbol{\mu}', \boldsymbol{\Sigma}')$, to obtain the new instance $(\Lambda, \boldsymbol{\mu}'', \boldsymbol{\Sigma}'')$.

Compatibility with modular hints. We sketch how modular hints can be

[3] ℓ should be large enough that concentration bounds hold for the noise on the aggregate set of hints.

incorporated into an evolved DBDD instance, where the secret distribution is no longer Gaussian. Assume we are given a DBDD instance $(\Lambda, \boldsymbol{\mu}, \boldsymbol{\Sigma})$, and a hint $\langle \boldsymbol{v}, \boldsymbol{s} \rangle \equiv \gamma \mod k$, where $\boldsymbol{s}$ denotes the LWE secret/error. We add a variable c' such that $\langle \boldsymbol{v}, \boldsymbol{s} \rangle - c' \cdot k = \gamma$, where the equation is over the reals. Since $\boldsymbol{s}$ is contained in the ellipsoid defined by $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, we have that $c' \cdot k + \gamma$ is bounded by $\langle \boldsymbol{v}, \boldsymbol{\mu} \rangle \pm \sqrt{r \boldsymbol{v} \boldsymbol{\Sigma} \boldsymbol{v}^T}$ (where r is the rank of $\boldsymbol{\Sigma}$). Therefore, we can first consider the DBDD instance $(\Lambda', \boldsymbol{\mu}', \boldsymbol{\Sigma}')$, where $\Lambda' = \Lambda \times \mathbb{Z}$, $\boldsymbol{\mu}' = (\boldsymbol{\mu} || \frac{1}{k}(\langle \boldsymbol{v}, \boldsymbol{\mu} \rangle - \gamma))$ and $\boldsymbol{\Sigma}'$ has dimension one larger than $\boldsymbol{\Sigma}$, with the final row and column all 0 except the bottom right corner set to $\frac{r \boldsymbol{v} \boldsymbol{\Sigma} \boldsymbol{v}^T}{k^2}$. Note that $(\boldsymbol{s} || c')$ is guaranteed to be contained in the corresponding rank-scaled ellipsoid. Finally, we apply our new perfect hint algorithm for hint $\langle \boldsymbol{v}, \boldsymbol{s} \rangle - c' \cdot k = \gamma$. This results in a new DBDD instance $(\Lambda'', \boldsymbol{\mu}'', \boldsymbol{\Sigma}'')$ with dimension equal to the original DBDD instance.

If some distributional information is known about the instance $(\Lambda, \boldsymbol{\mu}, \boldsymbol{\Sigma})$, one can potentially find a $(\boldsymbol{\mu}', \boldsymbol{\Sigma}')$ where $\boldsymbol{\Sigma}'$ has a smaller bottom right corner coordinate, and for which $(\boldsymbol{s} || c')$ is still guaranteed to be contained in the corresponding rank-scaled ellipsoid. For example, if $\boldsymbol{\Sigma}$ is the original LWE distribution then $\langle \boldsymbol{v}, \boldsymbol{s} \rangle$ is a Gaussian with variance $\boldsymbol{v} \boldsymbol{\Sigma} \boldsymbol{v}^T$. One can choose a constant $h \ll \sqrt{r}$ such that $\langle \boldsymbol{v}, \boldsymbol{s} \rangle \leq h \sqrt{\boldsymbol{v} \boldsymbol{\Sigma} \boldsymbol{v}^T}$ with probability $1 - \epsilon$. The bottom right corner of Σ' can then be set to $\frac{h^2 \boldsymbol{v} \boldsymbol{\Sigma} \boldsymbol{v}^T}{k^2}$, with the guarantee that the secret is contained in the rank-scaled ellipsoid with probability $1 - \epsilon$. We defer implementation of compatible modular hints to future work.

Compatibility with short vector hints. Our new approach remains compatible

with the approach of [41] for short vector hints. See Section 4.2.4 for more details on the adjustment that must be made.

4.1.3 Future Work

In this initial work, our main goal is to establish a connection between techniques in convex geometry and analysis of the security loss of LWE with side information. We believe that this connection can be further explored in several ways.

The maximal inscribed ellipsoid. The Löwner-John ellipsoids we have dealt thus far correspond to the minimal volume ellipsoid circumscribing a convex body. The literature also explores the *maximal* volume ellipsoid that can be *inscribed* in a convex body. For such ellipsoids, closed-form formulas for the case of inequality hints can be obtained from the more general formulas for ellipsoidal slabs [65]. As suggested to us by an anonymous reviewer, security estimations based on the maximal volume inscribed ellipsoid can be viewed as upper bounds on the strength of the optimal algorithm for recovering the LWE secret via lattice-reduction. Combining such estimates with our prior techniques, we would obtain both upper and lower bounds on the optimal algorithm that follows the attack template under consideration.

We note that both in the case of inequality hints and combined hints, it is possible that, while the volume of the intersected region is smaller, the minimal circumscribing ellipsoid is nevertheless equal to the original ellipsoid. This means that the hint yields no progress in our current framework. This, however, cannot

occur with the maximal volume inscribed ellipsoid. Thus, we plan to explore using the maximal inscribed ellipsoid in cases where no progress can be made with the minimal circumscribing ellipsoid. One such example is inequality hints that carry little information about the secret.

Incorporating techniques from Control Theory. There is a rich body of literature in control theory dealing with the "state estimation" problem in which the goal is to integrate new information (obtained from noisy measurements) into a current model of a system. Indeed, the conditional approximate hints from the prior work [41] can be viewed as a special case of the celebrated Kalman filter [79] from control theory (which assumes that all measurements are linear and all noise is Gaussian). A more recent line of work [88, 122, 144] studies the case in which the noise is not guaranteed to be Gaussian (and may even be deterministic) and uses a set of fundamental ellipsoidal operations (known as an "Ellipsoidal Calculus") to combine the information. Hybrid models (where some of the noise is assumed to be Gaussian, whereas worst-case assumptions are made for the rest) have also been studied [68]. We plan to explore further connections with the control theory literature, to understand better the tradeoffs of using worst-case/average-case assumptions to analyze a system.

Toolkit Extension. Alongside this paper, we release an extension to the original python/sage 9.0 toolkit from [41][4] We provide an updated API (which simplifies further extensions to the toolkit), and several new class files. The new EBDD.sage

[4]The updated toolkit can be found at https://github.com/hunterkipt/Geometric-LWE-Estimator.

class is *fully-featured* implementation of our geometric approach. It maintains all information about the instance: the lattice Λ, and the (rank-scaled) ellipsoid $E^{(\mathsf{Rank})}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ as hints are integrated. We leave the lightweight implementations of this extension to future work, as the full implementation is presently required to perform accurate estimation of the hardness loss resulting from our (more-general) geometry-based hints.

4.1.4 Related Work

Concrete security of Lattice Based Cryptosystems. Two LWE attack templates considered in the literature are known as the *primal*, and *dual* attacks. Both of these attack templates reduce the task of breaking LWE to solving an SVP instance. The SVP problem is a long-standing problem that has attracted much attention from the cryptographic as well as quantum communities. The current asymptotically best SVP algorithms (for classical and quantum computers) include [5, 20, 69, 90]. In practice, the BKZ algorithm [125] was found to perform well on parameter regimes of interest, though it is not amenable to provable guarantees on its asymptotic performance. The BKZ algorithm on dimension d includes as a core subroutine an SVP solving step on a smaller block-size $\beta \ll d$. It is compatible with both classical and quantum algorithms for solving the smaller blocksize SVP-β instances. NIST post quantum (PQC) candidates have used the runtime estimates for the BKZ algorithm to inform the setting of their concrete parameters [2]. Several works have sought to create models to accurately predict the behavior of the BKZ algorithm in parameter regimes

of interest [7, 16, 38]. Finally, there has been some work on comparing the lattice reduction-based algorithms described up to now with combinatorial algorithms [4, 6].

Sice-Channel Attacks (SCA). There are various ways in which an attacker can obtain "side-channel" information about the secret key of a cryptographic scheme, greatly reducing the security of the scheme, or even allowing for a full key recovery. These methods include timing attacks [84], power analysis attacks [85], cache side-channel attacks [139], and microarchitectural attacks [83, 93]. Side-channel attacks on NIST PQC candidates include attacks on (earlier versions of) Dilithium, which was recently announced as a selected digital signature algorithm [114, 117], qTESLA [117], NTRUEncrypt [128], as well as Rainbow, NTRU, and McEliece [142]. Template attacks were introduced by [37], who used a device identical to the target to generate a precise "template" of the noise. When noisy side-channel data is obtained, the template can be used to learn information about the secret. Bos et al. [30] applied this approach to FrodoKEM, a Round 3 PQC alternate candidate, simulating a single trace power attack using ELMO [98]. We use their side channel attack as the starting point of our experiments in Section 4.3.3. Other research has focused on active side-channel attacks, where faults are injected during computation involving the secret key, such as RowHammer. Such attacks were performed on the LUOV signature scheme, a Round 2 PQC candidate [102], as well as Dilithium [75].

Decryption Failures. A decryption failure is when the decryption process returns an incorrect message on a validly encrypted ciphertext. Since most lattice-based cryptographic KEM schemes have a non-zero decryption failure rate, several prior works have investigated the possibility of decryption failure attacks. Specifically,

decryption failures leak information about the secret key, and in some cases can be used to fully recover the secret. These attacks were first applied on CPA-secure schemes [18,45,47,53,82,113]. However, CCA-secure schemes use a Fujisaki-Okamoto transform which protects against such attacks and ensures that even a malicious attacker can only cause decryption failures with extremely low probability. Several methods have been suggested to boost the rate at which decryption failures occur, thereby lowering the complexity of the attack [25, 42–44, 66]. Recently, Fahr et al. [51] combined SCA and Decryption Failure attacks by using a Rowhammer attack—which induces bit flips in memory—to artificially boost the failure rate of NIST PQC candidate FrodoKEM. This allowed an end-to-end key recovery attack on Frodo-640.

4.1.5 Organization

In Chapter 2, we present notation and provide necessary background in linear algebra (Section 2.7), geometry (Section 2.8), and lattices (Section 2.4). Background on the ellipsoid method can be found in Appendix B.2.

Section 2.9 defines the DBDD problem, as well as a new variant of the DBDD problem. The reduction from DBDD to uSVP is given in Section 2.9.1, and Section 2.9.2 presents security estimates for the uSVP problem. Section 4.1.6 presents the initial embedding we use in this work from LWE to DBDD.

Section 4.2 introduces inequality hints (Section 4.2.1), combined hints (Section 4.2.2), and revisits perfect (Section 4.2.3) and short vector (Section 4.2.4) hints.

Missing proofs can be found in Appendix B.1.

Section 4.3 presents experimental validation of our β estimates (Section 4.3.1), applications of our new types of hints to the decryption failure setting (Section 4.3.2), and to combining decryption failure and side-channel information (Section 4.3.3).

4.1.6 Obtaining our initial DBDD embedding

Recall that, in the prior work, Kannan's embedding was used to reduce LWE to DBDD. We next present a somewhat different embedding of LWE in DBDD.

The geometric DBDD embedding. Consider an LWE instance $\boldsymbol{s}\boldsymbol{A}^T + \boldsymbol{e} = \boldsymbol{b}$ mod q. We can remove the mod q and transform the above to a system of equations over the integers by adding the vector of variables $\boldsymbol{c}$:

$$\boldsymbol{s}\boldsymbol{A}^T + \boldsymbol{e} - q\boldsymbol{c} = \boldsymbol{b}.$$

Note that given an LWE instance $\boldsymbol{A}, \boldsymbol{b}$ and a solution $(\boldsymbol{c}||\boldsymbol{s})$, there is an *affine* transformation to obtain a solution modulo q of the form $(\boldsymbol{e}||\boldsymbol{s})$. Specifically, $\boldsymbol{e} = q\boldsymbol{c} - \boldsymbol{s}\boldsymbol{A}^T + \boldsymbol{b}$. Further, we assume that we can (w.h.p.) upper bound the squared norm of $(\boldsymbol{e}||\boldsymbol{s})$ by $\sigma^2(n+m) = \sigma^2 \cdot d$ (e.g. in standard LWE σ^2 is the variance of $\boldsymbol{s}, \boldsymbol{e}$). In matrix notation, we define $\boldsymbol{B}$ as:

$$\boldsymbol{B} := \begin{bmatrix} q\boldsymbol{I}_m & 0 \\ -\boldsymbol{A}^T & \boldsymbol{I}_n \end{bmatrix}. \tag{4.1}$$

We obtain the following constraint on the solution $(\boldsymbol{c}||\boldsymbol{s})$ of the transformed system: $\left\| \left((\boldsymbol{c}||\boldsymbol{s})\, \boldsymbol{B} + (\boldsymbol{b}||\boldsymbol{0}) \right) \right\|^2 \leq \sigma^2 \cdot d$. The above defines a rank-scaled ellipsoid $\boldsymbol{E}$ with center $(-\boldsymbol{b}||\boldsymbol{0})\, \boldsymbol{B}^{-1}$:

$$E^{(\mathsf{Rank})}((-\boldsymbol{b}||\boldsymbol{0}, \sigma^2(\boldsymbol{B}\boldsymbol{B}^T)^{-1}) :=$$
$$\left\{ (\boldsymbol{c}||\boldsymbol{s}) \in \mathcal{R}^{n+m} : \left((\boldsymbol{c}||\boldsymbol{s}) - (-\boldsymbol{b})||\boldsymbol{0}) \, \boldsymbol{B}^{-1} \right) \frac{1}{\sigma^2} \boldsymbol{B}\boldsymbol{B}^T \left((\boldsymbol{c}||\boldsymbol{s}) - (\boldsymbol{b}||\boldsymbol{0})\, \boldsymbol{B}^{-1} \right)^T \leq d \right\}.$$

Our DBDD instance is therefore: $\left(\mathbb{Z}^d, (-\boldsymbol{b}||\boldsymbol{0})\, \boldsymbol{B}^{-1}, \sigma^2(\boldsymbol{B}\boldsymbol{B}^T)^{-1} \right)$.

Incorporating a center and shape matrix for $(\boldsymbol{e}||\boldsymbol{s})$. We consider here the case that we are given a center vector $(\boldsymbol{\mu}_e||\boldsymbol{\mu}_s) \in \mathrm{Span}(\boldsymbol{\Sigma})$, and a shape matrix $\boldsymbol{\Sigma}$, along with the guarantee that w.h.p. $((\boldsymbol{e}||\boldsymbol{s}) - (\boldsymbol{\mu}_e||\boldsymbol{\mu}_s))\, \boldsymbol{\Sigma}^{\sim} \left((\boldsymbol{e}||\boldsymbol{s}) - (\boldsymbol{\mu}_e||\boldsymbol{\mu}_s) \right)^T \leq \mathrm{rank}(\boldsymbol{\Sigma})$. As a special case, the above guarantee holds when $(\boldsymbol{e}||\boldsymbol{s}) \sim \mathcal{N}((\boldsymbol{\mu}_e||\boldsymbol{\mu}_s), \boldsymbol{\Sigma})$ follow a multivariate Gaussian distribution. Using the same $\boldsymbol{B}$ as in (4.1), we obtain the constraint:

$$\left\| \left(\left((\boldsymbol{c}||\boldsymbol{s})\, \boldsymbol{B} + (\boldsymbol{b}||\boldsymbol{0}) \right) - (\boldsymbol{\mu}_e||\boldsymbol{\mu}_s) \right) \sqrt{\boldsymbol{\Sigma}^{\sim}} \right\|^2 \leq \mathrm{rank}(\boldsymbol{\Sigma}).$$

This gives the rank-scaled ellipsoid:

$$E^{(\mathsf{Rank})}(((\boldsymbol{\mu}_e - \boldsymbol{b})||\boldsymbol{\mu}_s)\boldsymbol{B}^{-1}, (\boldsymbol{B}^T)^{-1}\boldsymbol{\Sigma}(\boldsymbol{B})^{-1}) := \left\{ (\boldsymbol{c}||\boldsymbol{s}) \in \mathrm{Span}((\boldsymbol{B}^T)^{-1}\boldsymbol{\Sigma}(\boldsymbol{B})^{-1}) : \right.$$
$$\left. \left((\boldsymbol{c}||\boldsymbol{s}) - ((\boldsymbol{\mu}_e - \boldsymbol{b})||\boldsymbol{\mu}_s)\, \boldsymbol{B}^{-1} \right) \boldsymbol{B}\boldsymbol{\Sigma}^{\sim}\boldsymbol{B}^T \left((\boldsymbol{c}||\boldsymbol{s}) - ((\boldsymbol{\mu}_e - \boldsymbol{b})||\boldsymbol{\mu}_s)\, \boldsymbol{B}^{-1} \right)^T \leq \mathrm{rank}(\boldsymbol{\Sigma}) \right\}.$$

Our DBDD instance is now: $\left(\mathbb{Z}^d, ((\boldsymbol{\mu}_e - \boldsymbol{b})||\boldsymbol{\mu}_s)\,\boldsymbol{B}^{-1}, (\boldsymbol{B}^T)^{-1}\boldsymbol{\Sigma}(\boldsymbol{B})^{-1}\right)$. We can now apply hints to our initial DBDD instance.

Remark 4.1.1. Our DBDD embedding extends to $\boldsymbol{s}$ sampled from any distribution $\mathcal{S}$ whose support is contained in a lattice, and to $\boldsymbol{A}^T \in \mathcal{R}^{n\times m}, \boldsymbol{e} \in \mathcal{R}^m$ which are real-valued. Thus, our embedding captures the *Continuous LWE Problem* for secret distributions $\mathcal{S}$ as above [35, 67].

4.2 Hints

4.2.1 Inequality Hints

An inequality hint on the secret $(\boldsymbol{c}||\boldsymbol{s})$ is the knowledge of $\boldsymbol{v} \in \mathcal{R}^d$ and $l \in \mathcal{R}$, such that $\langle(\boldsymbol{c}||\boldsymbol{s}), \boldsymbol{v}\rangle \leq \gamma$. In other words, inequality hints correspond to the knowledge that the secret lies on one side of a halfspace.

The process for integrating inequality hints relies on the ellipsoid-halfspace intersection procedure of the ellipsoid method (2.4,2.5). Given a DBDD instance $\mathsf{DBDD}_{\Lambda,\boldsymbol{\mu},\boldsymbol{\Sigma}}$, an inequality hint with $\boldsymbol{v} \in \mathrm{Span}(\boldsymbol{\Sigma})$ produces a new instance $\mathsf{DBDD}_{\Lambda',\boldsymbol{\mu}',\boldsymbol{\Sigma}'}$,

$$\Lambda' = \Lambda \tag{4.2}$$

$$\boldsymbol{\mu}' = \boldsymbol{\mu} - \left(\frac{1+r\alpha}{r+1}\right)\frac{\boldsymbol{v}\mathcal{F}(\boldsymbol{\Sigma})}{\sqrt{\boldsymbol{v}\mathcal{F}(\boldsymbol{\Sigma})\boldsymbol{v}^T}} \tag{4.3}$$

$$\boldsymbol{\Sigma}' = \mathcal{F}^{-1}\left(\left(\frac{r^2}{r^2-1}(1-\alpha^2)\right)\left(\mathcal{F}(\boldsymbol{\Sigma}) - \left(\frac{2(1+r\alpha)}{(r+1)(1+\alpha)}\right)\frac{\mathcal{F}(\boldsymbol{\Sigma})\boldsymbol{v}^T\boldsymbol{v}\mathcal{F}(\boldsymbol{\Sigma})}{\boldsymbol{v}\mathcal{F}(\boldsymbol{\Sigma})\boldsymbol{v}^T}\right)\right) \tag{4.4}$$

for $-1/r < \alpha \leq 1$, where α is defined as in (2.6). and r is the rank of $\boldsymbol{\Sigma}$. If $-1 < \alpha \leq -1/r$, then $\Lambda' = \Lambda$, $\boldsymbol{\mu}' = \boldsymbol{\mu}$, and $\boldsymbol{\Sigma}' = \boldsymbol{\Sigma}$, meaning that for inequality hints with α in this range, we do not make progress under the approximation stemming from the ellipsoid method.

Quantitative volume reduction. Using the matrix determinant lemma and properties of rdet and $\boldsymbol{\Sigma}^{\sim}$, we have that

$$\mathsf{rdet}(\boldsymbol{\Sigma}') = \left(\frac{r^2}{r^2-1}(1-\alpha^2)\right)^r \cdot \left(1 - \left(\frac{2(1+r\alpha)}{(r+1)(1+\alpha)}\right)\right) \cdot \mathsf{rdet}(\boldsymbol{\Sigma}).$$

Here we can clearly see the power of α on the volume. The closer α is to 1, the smaller the resulting volume of $\boldsymbol{\Sigma}'$ (yielding a larger decrease in security).

4.2.2 Combined Hints

We are given two DBDD instances, $(\Lambda_1, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1), (\Lambda_2, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$, with respect to the *same* secret $(c||s)$ (resp. $(e||s)$). Recall that DBDD instances $(\Lambda, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ provide the promise that the secret $(\boldsymbol{c}||\boldsymbol{s}) \in \Lambda$ and $(\boldsymbol{c}||\boldsymbol{s}) \in E^{(\mathsf{Rank})}(\boldsymbol{\mu}, \boldsymbol{\Sigma},)$ (resp. $(\boldsymbol{e}||\boldsymbol{s}) \in \Lambda$

and $(\boldsymbol{e}||\boldsymbol{s}) \in E^{(\mathsf{Rank})}(\boldsymbol{\mu}, \boldsymbol{\Sigma},))$.

Combined hints take the two DBDD instances and combine them into a single instance $(\Lambda', \boldsymbol{\mu}', \boldsymbol{\Sigma}')$ that captures the information from both. Specifically, Λ' will be equal to the intersection of the two lattices Λ_1, Λ_2. Since the intersection of two ellipsoids $E(\boldsymbol{\mu}_1, \mathcal{F}(\boldsymbol{\Sigma}_1)), E(\boldsymbol{\mu}_2, \mathcal{F}(\boldsymbol{\Sigma}_2))$ is not necessarily an ellipsoid, we define $E(\boldsymbol{\mu}', \mathcal{F}(\boldsymbol{\Sigma}'))$ to be an ellipsoid circumscribing their intersection. Exactly computing the minimal volume ellipsoid that circumscribes the intersection of two ellipsoids is computationally difficult. We instead use Theorem 2.8.8 to find $E(\boldsymbol{\mu}', \mathcal{F}(\boldsymbol{\Sigma}'))$.

$$\Lambda' = \Lambda_1 \cap \Lambda_2 \tag{4.5}$$

$$\boldsymbol{\mu}'\Pi_{\boldsymbol{X}} = \left(\boldsymbol{\mu}_1\tilde{\lambda}\mathcal{F}(\boldsymbol{\Sigma}_1)^{\sim} + \boldsymbol{\mu}_2(1-\tilde{\lambda})\mathcal{F}(\boldsymbol{\Sigma}_2)^{\sim}\right)\boldsymbol{X}^{\sim} \tag{4.6}$$

$$\boldsymbol{\Sigma}' = \mathcal{F}^{-1}(k\boldsymbol{X}^{\sim}), \tag{4.7}$$

where

$$\boldsymbol{X} = \tilde{\lambda}\mathcal{F}(\boldsymbol{\Sigma}_1)^{\sim} + (1-\tilde{\lambda})\mathcal{F}(\boldsymbol{\Sigma}_2)^{\sim},$$

$$k = 1 - \tilde{\lambda}(1-\tilde{\lambda})(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)\mathcal{F}(\boldsymbol{\Sigma}_2)^{\sim}\boldsymbol{X}^{\sim}\mathcal{F}(\boldsymbol{\Sigma}_1)^{\sim}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T$$

and $\tilde{\lambda}$ is the unique value between $[0, 1]$ that minimizes the volume of $E(\boldsymbol{\mu}', \boldsymbol{\Sigma}')$. Theorem 2.8.10 provides a computationally efficient way to find $\tilde{\lambda}$. Given $\boldsymbol{\mu}'\Pi_{\boldsymbol{X}}$, the mean $\boldsymbol{\mu}'$ can be recovered from the known linear constraints on the system.

When does fusion yield a volume reduction? If $\tilde{\lambda} = 0$, then $\boldsymbol{\Sigma}' = \boldsymbol{\Sigma}_2$ and if

$\tilde{\lambda} = 1$, then $\boldsymbol{\Sigma}' = \boldsymbol{\Sigma}_1$. Therefore, ellipsoid fusion does not always yield a reduction in volume. It is not hard to see that if $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2$, and if $\boldsymbol{\mu}_1 \neq \boldsymbol{\mu_2}$ are in the span of both $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$, then the volume of $E^{(\mathsf{Rank})}(\boldsymbol{\mu}', \boldsymbol{\Sigma}')$ is strictly smaller than both the volume of $E^{(\mathsf{Rank})}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and of $E^{(\mathsf{Rank})}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$. In the following, we show that fusion can still sometimes lead to a volume reduction, even in case that the volume of $E^{(\mathsf{Rank})}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ is strictly smaller than the volume of $E^{(\mathsf{Rank})}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$.

Theorem 4.2.1. *Let $\boldsymbol{c} \in \mathcal{R}^d$ denote the d-dimensional vector that has $c \in \mathcal{R}$ in each position. Let $\sigma_1^2, \sigma_2^2 \in \mathcal{R}$ be such that $\sigma_2^2 < \sigma_1^2$. Consider the rank-scaled ellipsoids $E^{(\mathsf{Rank})}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) = E^{(\mathsf{Rank})}(\mathbf{0}, \sigma_1^2 \boldsymbol{I}_d)$ and $E^{(\mathsf{Rank})}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) = E^{(\mathsf{Rank})}(\boldsymbol{c}, \sigma_2^2 \boldsymbol{I}_d)$. Then the volume of $E^{(\mathsf{Rank})}(\boldsymbol{\mu}', \boldsymbol{\Sigma}')$ is lower than both the volume of $E^{(\mathsf{Rank})}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $E^{(\mathsf{Rank})}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ if and only if $c^2 > \sigma_1^2 - \sigma_2^2$.*

We defer the proof of Theorem 4.2.1 to Appendix B.1.

Remark 4.2.2. Consider the setting of Theorem 4.2.1 and let c be such that $(\sigma_1 - \sigma_2)^2 < c^2 < \sigma_1^2 - \sigma_2^2$. Note that $E^{(\mathsf{Rank})}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \not\subseteq E^{(\mathsf{Rank})}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$. This can be seen using the alternate definition of a rank-scaled ellipsoid as a linear transformation and shift of the ball of radius $\sqrt{r}$, where r is the rank. Specificially, since $\|\mathbf{1}\| = \sqrt{d}$ and since

$$\|\mathbf{1} \cdot \sqrt{\boldsymbol{\Sigma}_2} + \boldsymbol{\mu}_2\|^2 = d \cdot (\sigma_2 + c)^2 > d\sigma_1^2,$$

we have that the point $\mathbf{1} \cdot \sqrt{\boldsymbol{\Sigma}_2} + \boldsymbol{\mu}_2$ is contained in $E^{(\mathsf{Rank})}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ but not in $E^{(\mathsf{Rank})}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$. On the other hand, the intersection of the two ellipsoids is not empty, since $\boldsymbol{\mu}_2$ is contained in both ellipsoids. Clearly $\boldsymbol{\mu}_2$ is contained in $E^{(\mathsf{Rank})}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$.

We can see that it is contained in $E^{(\mathsf{Rank})}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ since

$$\boldsymbol{\mu}_2 \boldsymbol{\Sigma}_1^{-1} \boldsymbol{\mu}_2^T = d \cdot c^2 \cdot \frac{1}{\sigma_1^2} < d \cdot \frac{\sigma_1^2 - \sigma_2^2}{\sigma_1^2} < d.$$

However, since $c^2 < \sigma_1^2 - \sigma_2^2$, we have by Theorem 4.2.1 that the volume of $E^{(\mathsf{Rank})}(\boldsymbol{\mu}', \boldsymbol{\Sigma}')$ does not decrease.

Importantly, this means that the ellipsoid fusion technique does not guarantee that we obtain a lower volume ellipsoid, even in the case that $E(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1), E(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ are such that $E(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \cap E(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \neq \emptyset$, $E(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \not\subseteq E(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$, and $E(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \not\subseteq E(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$. This contradicts Theorem 3 of [122].

Remark 4.2.3. If $\boldsymbol{x}$ has ellipsoid norm $0 \leq a \leq 1$ with respect to $E(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and ellipsoid norm $0 \leq b \leq 1$ with respect to $E(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$, then its ellipsoid norm with respect to the fused ellipsoid is

$$0 \leq \frac{\tilde{\lambda} a + (1 - \tilde{\lambda}) b + k - 1}{k} \leq 1.$$

For diagonal ellipsoids for which $0 \leq k \leq 1$, the above implies that the ellipsoid norm of $\boldsymbol{x}$ with respect to the fused ellipsoid is at most $\tilde{\lambda} a + (1 - \tilde{\lambda}) b \leq \max(a, b)$.

4.2.3 Perfect Hints, Revisited

A perfect hint on the secret $(\boldsymbol{c}||\boldsymbol{s})$ is the knowledge of $\boldsymbol{v} \in \mathbb{Z}^d$ and $\gamma \in \mathbb{Z}$, such that $\langle (\boldsymbol{c}||\boldsymbol{s}), \boldsymbol{v} \rangle = \gamma$. We assume that $\boldsymbol{v} \in \mathrm{Span}(\boldsymbol{\Sigma})$.

In our previous work, the resulting instance after incorporating a perfect hint

was based on the conditional distribution of a multi-variate gaussian. Instead, we make use of ellipsoid-hyperplane intersection.

Recall the definition of a hyperplane in 2.8.6. Here, a perfect hint can represent the knowledge that the secret $(\boldsymbol{c}||\boldsymbol{s})$ is located on a hyperplane defined by $(\boldsymbol{v}, \gamma)$. Thus, we can intersect both the ellipsoid and the lattice with $H(\boldsymbol{v}, \gamma)$. An advantage of this approach presents itself when dealing with non-homogeneous instances. If the hyperplane does not cut through the center of the ellipsoid, the volume of the intersection can be lower than before.

Homogeneous instances. For homogeneous instances, the process for integrating perfect hints is fairly straightforward. It relies on the ellipsoid-hyperplane intersection procedure of the ellipsoid method (2.4,2.7). Given a DBDD instance $\mathsf{DBDD}_{\Lambda,\boldsymbol{\mu},\boldsymbol{\Sigma}}$, a homogeneous perfect hint produces a new instance $\mathsf{DBDD}_{\Lambda',\boldsymbol{\mu}',\boldsymbol{\Sigma}'}$,

$$\Lambda' = \Lambda \cap H(\boldsymbol{v}, 0) \tag{4.8}$$

$$\boldsymbol{\mu}' = \boldsymbol{\mu} - \alpha \frac{\boldsymbol{v}\mathcal{F}(\boldsymbol{\Sigma})}{\sqrt{\boldsymbol{v}\mathcal{F}(\boldsymbol{\Sigma})\boldsymbol{v}^T}} \tag{4.9}$$

$$\boldsymbol{\Sigma}' = \mathcal{F}^{-1}\left((1-\alpha^2)\left(\mathcal{F}(\boldsymbol{\Sigma}) - \frac{\mathcal{F}(\boldsymbol{\Sigma})\boldsymbol{v}^T\boldsymbol{v}\mathcal{F}(\boldsymbol{\Sigma})}{\boldsymbol{v}\mathcal{F}(\boldsymbol{\Sigma})\boldsymbol{v}^T}\right)\right) \tag{4.10}$$

for $-1 < \alpha \leq 1$, where α is defined as in (2.6).

Quantitative volume and rank reduction. Note that the rank of $\boldsymbol{\Sigma}'$ is $r-1$, where r is the rank of $\boldsymbol{\Sigma}$. Using (a generalization of) the matrix determinant lemma

and properties of rdet and $\boldsymbol{\Sigma}^{\sim}$, we have that

$$\mathsf{rdet}(\boldsymbol{\Sigma}') = \left(\frac{r(1-\alpha^2)}{(r-1)}\right)^{r-1} \cdot \frac{\boldsymbol{v}\boldsymbol{v}^T}{\boldsymbol{v}\boldsymbol{\Sigma}\boldsymbol{v}^T} \cdot \mathsf{rdet}(\boldsymbol{\Sigma}).$$

Non-homogeneous instances. Note that the above formulation is not complete for non-homogeneous instances. Intersecting the lattice with a hyperplane where $\gamma \neq 0$ results in a shifted lattice *coset*. This severs the connection between the lattice points and the ellipsoid if the exact shift applied is unknown. There are simple geometric relationships that we can use to solve this problem easily, but they quickly become infeasible when applying multiple perfect hints.

Our solution is to find a point $\boldsymbol{y}$ in the lattice coset and shift the entire instance by $\boldsymbol{y}$. This means that the lattice coset is now the zero coset (i.e. it is again a lattice), and the hyperplane contains the origin, while the center of the ellipsoid is now shifted by $\boldsymbol{y}$. This allows us to achieve the smaller intersected volume due to non-homogenized instances, mentioned above. Finally, note that the solution that is obtained from this DBDD instance will now also be shifted by $\boldsymbol{y}$, and so to recover the original solution we must shift back by $\boldsymbol{y}$. Thus, we propose the following procedure.

First, observe that each perfect hint imposes a linear constraint on the space of feasible solutions in Λ. Therefore, we can combine all perfect hints into a linear system. Let $\boldsymbol{V}, \boldsymbol{\gamma}$ denote such a system, where $\boldsymbol{V} \in \mathbb{Z}^{d \times t}, \boldsymbol{\gamma} \in \mathbb{Z}^t$, and t is the number of perfect hints to be integrated. Further, let $\boldsymbol{y} \in \Lambda$ be a solution to the above system (i.e. $\boldsymbol{y}\boldsymbol{V}^T = \boldsymbol{\gamma}$). For non-homogeneous instances, $\boldsymbol{y}$ will not correspond to the origin.

We then shift the ellipsoid (and thus the secret) by $\boldsymbol{y}$ so that $((\boldsymbol{c}||\boldsymbol{s}) - \boldsymbol{y})\boldsymbol{V}^T = 0$. For non-homogeneous instances, given a DBDD instance $\mathsf{DBDD}_{\Lambda,\boldsymbol{\mu},\boldsymbol{\Sigma}}$, we obtain a new instance

$\mathsf{DBDD}_{\Lambda'',\boldsymbol{\mu}'',\boldsymbol{\Sigma}''}$, where

$$\Lambda' = [\Lambda \cap H(\boldsymbol{V}, \boldsymbol{\gamma})] + \boldsymbol{y} \tag{4.11}$$

$$E^{(\mathsf{Rank})}(\boldsymbol{\mu}'', \boldsymbol{\Sigma}'') = E^{(\mathsf{Rank})}(\boldsymbol{\mu}' - \boldsymbol{y}, \boldsymbol{\Sigma}') \tag{4.12}$$

and $E^{(\mathsf{Rank})}(\boldsymbol{\mu}', \boldsymbol{\Sigma}')$ is obtained by applying (4.9) and (4.10) for all perfect hints described by $\boldsymbol{V}, \boldsymbol{\gamma}$. Note also that (with a slight abuse of notation $H(\boldsymbol{V}, \boldsymbol{\gamma})$ refers to the intersection of hyperplanes that meet the constraints of the $\boldsymbol{V}$, $\boldsymbol{\gamma}$ linear system. We note that for our proposed Kannan Ellipsoid embedding, we can solve a system of linear diophantine equations to obtain $\boldsymbol{y}$, as the lattice is $\mathbb{Z}^d$ when perfect hints are integrated first.

Such a system can be solved efficiently using the LLL algorithm or through computing the Hermite Normal Form. We would like to make sure our offset $\boldsymbol{y}$ does not become too large, as this induces numerical errors. Solving the combined system $(\boldsymbol{c}||\boldsymbol{s})\boldsymbol{V} = \boldsymbol{\gamma}$ AND $[(\boldsymbol{c}||\boldsymbol{s})\boldsymbol{B} + (\boldsymbol{b}||\boldsymbol{0})](\boldsymbol{I}_m||\boldsymbol{A})^T - q\boldsymbol{c} = \boldsymbol{b}$ can be done over the integers, (the above uses $\boldsymbol{B}$ defined in (4.1)) but the solutions become extremely large. Instead, as long as there are at most n perfect hints to integrate, we are left with enough free parameters that we can jointly solve the diophantine system and the LWE equations modulo q. More specifically, we first solve the diophantine system $[(\boldsymbol{e}||\boldsymbol{s}) - (\boldsymbol{b}||\boldsymbol{0})]\boldsymbol{B}^{-1})\boldsymbol{V} = \boldsymbol{\gamma}$ to get a small integer solution $\bar{\boldsymbol{y}}$. In fact, we obtain a

solution family $\{\bar{\boldsymbol{y}} + \boldsymbol{x}\boldsymbol{U} \mid \boldsymbol{x} \in \mathbb{Z}^{d-t}\}$, where $\boldsymbol{U}$ is the unimodular transformation matrix of the Hermite Normal Form of $\boldsymbol{V}$ of dimension $(d-t) \times d$. Then, we solve the system, $(\bar{\boldsymbol{y}} + \boldsymbol{x}\boldsymbol{U})(\boldsymbol{I}_m || \boldsymbol{A})^T) = \boldsymbol{b} \mod q$, with m equations and $d - t \geq m$ variables. Thus, our final $\boldsymbol{y} = \bar{\boldsymbol{y}} + \boldsymbol{x}\boldsymbol{U}$ is a solution to both systems. This then bounds the value of each coordinate of $\boldsymbol{y}$ by at most q (given the solutions of the first system are sufficiently small). Finally, $\boldsymbol{y}$ must be converted into the $(\boldsymbol{c}||\boldsymbol{s})$ solution space via the relation in Section 4.1.6.

4.2.4 Short Vector Hints, Revisited

A short vector hint on the lattice Λ is the knowledge of a short vector $\boldsymbol{v}$ such that $\boldsymbol{v} \in \Lambda$.

In [41], these short vector hints were features of the DBDD lattice specific to the LWE embedding derived from Kannan's embedding. For example, LWE instances give rise to q-ary lattices under Kannan's embedding. Therefore so-called "q-vectors" with value q in a single coordinate and 0's in all other coordinates are short vectors (magnitude q) that are always contained in this lattice. The main intuition behind explicitly integrating this information into the DBDD instance is that if one knows a "good enough" lattice vector $\boldsymbol{v}$ that is not the secret, then that vector can be treated as a fixed basis vector. Projecting orthogonally to $\boldsymbol{v}$, then results in a tradeoff between dimension and lattice volume that can result in an easier instance to solve. The lost dimensions can be recovered through solving a system of linear equations over the rationals.

Note that in [41], the short vector hints were integrated into the lattice *before* isotropization. For our embedding of LWE into DBDD, the lattice is simply the integer lattice $\mathbb{Z}^d$ before isotropization (unless perfect hints have been integrated, but even in that case the lattice includes only information about the perfect hints, and not the LWE instance itself). Integrating short vector hints for the lattice $\mathbb{Z}^d$ makes little sense, since providing a good basis vector for the lattice $\mathbb{Z}^d$ with basis $\boldsymbol{I}_d$ is clearly unnecessary. However, we still want to somehow integrate the information contained in the q-vectors discussed above into our DBDD instance. More generally, for full compatibility with the prior work, we would like to be able to integrate any of the short vector hints discussed in [41] into our DBDD instance (e.g. the short vectors can have a different form after perfect hints are integrated). To do this, we can simply (partially) revert back to the Kannan embedding lattice to integrate short vector hints. To accomplish this, we perform the following coordinate space transformation:

$$\Lambda' = \Lambda \boldsymbol{B}$$

$$E(\boldsymbol{\mu}', \boldsymbol{\Sigma}') = E(\boldsymbol{\mu}\boldsymbol{B}, \boldsymbol{B}^T\boldsymbol{\Sigma}\boldsymbol{B})$$

We refer to this transformation as "partial isotropization," since Λ' would be the result of full isotropization only in an instance where no hints were integrated. Note that if the instance was not homogeneous, then the resulting secret is $(\boldsymbol{e} + \boldsymbol{b}||\boldsymbol{s})$. From here, short vector hints can applied as in the prior work [41]. After they are integrated, we perform homogenization and isotropization as normal to complete the

reduction to uSVP.

We can also integrate short vector hints with respect to the fully isotropized lattice $\Lambda \cdot \boldsymbol{M}$, where $\boldsymbol{M} := (\sqrt{\boldsymbol{\Sigma}})^{\sim}$ (assuming such short vectors in $\Lambda \cdot \boldsymbol{M}$ are known).

As pointed out by an anonymous reviewer, the procedure described above is equivalent to integrating knowledge of the short vectors of the DBDD lattice Λ *before* isotropization, but where (1) length is measured with respect to the *ellipsoid norm* defined by the DBDD shape matrix $\boldsymbol{\Sigma}$, and (2) projections are performed using the inner product induced by $\boldsymbol{\Sigma}$.

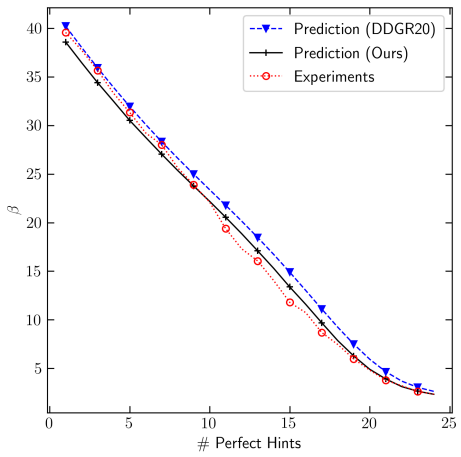
As with the prior work, it is crucial to integrate these hints last. Thus, this coordinate space transformation will not need to be applied to other hints.

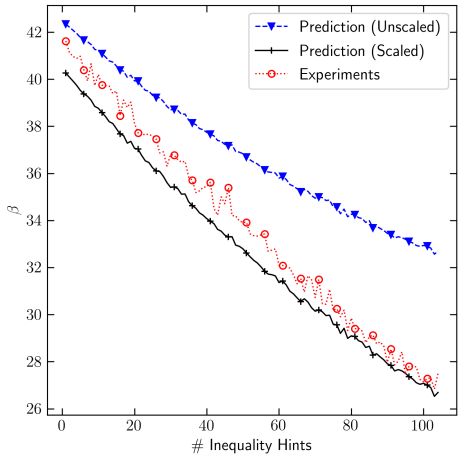
4.3 Experimental Validation and Applications

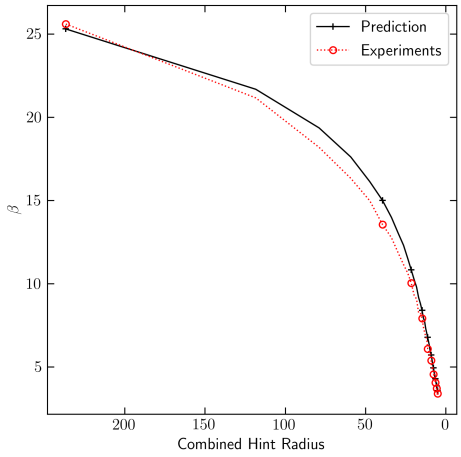
4.3.1 Experimental Validation

For (1) perfect hints, (2) inequality hints, (3) combined hints, we compare the bikz predicted by our tool with the bikz actually needed to launch the attack and recover the LWE secret/error. For (1) and (2), we choose the same set of LWE parameters for the initial instances as in [41], and integrate an increasing number of hints of each type.

For (1), the curve labeled "Prediction (DDGR20)" uses DBDD instances obtained by integrating perfect hints via the approach of [41], while "Prediction (Ours)" uses our new approach. We display a single "Experiments" curve since the EBDD and DBDD instances differ only by a scaling factor, which does not impact the bikz







LWE Parameters $n = m = 70,\ q = 3301,\ \sigma^2 = 20$	
Hint Type	Parameter
Perfect hints	–
Inequality hints	± 2
Combined hints	$[20, (20/49)]$

The hint vectors $\boldsymbol{v}$ for inequality and perfect hints were chosen as random ternary vectors of weight 5.

Figure 4.1: Experimental verification of the bikz predictions for each type of hint. Each data point was averaged over 256 samples. Inequality and perfect hint validation were conducted by integrating successively larger numbers of hints. Combined hint validation was conducted by integrating instances with decreasing ellipsoid volume (see (4.13)).

(as verified experimentally).

For (2), we create inequality hints by simulating a known (small) absolute error. Given a hint vector $\boldsymbol{v}$, we create the hint $\langle \boldsymbol{v}, (\boldsymbol{e}||\boldsymbol{s}) \rangle \geq \gamma - 2$, where γ is the inner product of $\boldsymbol{v}$ with the correct secret. Our predicted bikz–the "scaled" estimate–take into account the length of the shortest vector in our final lattice as in (2.11) as it deviates from the expected value assumed in (2.13). When integrating large numbers

of inequality hints the ellipsoid norm of the secret w.r.t. the DBDD instance is significantly lower than the rank, while (2.13) holds under the assumption that the ellipsoid norm is approximately equal to the rank. This leads to overestimation of the hardness when applying (2.13)–the "unscaled" estimate. As such, we also use this calibration when examining the hardness loss resulting from decryption failures in Section 4.3.2.

For (3) we use the same set of LWE parameters to construct an initial DBDD (see Section 4.1.6) instance. We then perform combined hints with the initial DBDD instance and each of the DBDD instances corresponding to the ellipsoids

$$E^{(\mathsf{Rank})}((\boldsymbol{c}||\boldsymbol{s}) + \mathcal{E}, (20/i) \cdot \boldsymbol{I}_{m+n}), \tag{4.13}$$

where $\mathcal{E} \sim \mathcal{N}(\mathbf{0}, (20/i) \cdot \boldsymbol{I}_{m+n})$ for $i \in [1, 49]$. See Figure 4.1 for details.

4.3.2 Decryption Failures, Revisited

Decryption failures exactly correspond to inequality hints from Section 4.2.1. Thus, the naive approach to running a decryption failure attack is to iteratively integrate each decryption failure as an inequality hint, obtaining a series of ellipsoids with volumes that are strictly decreasing. To test the efficacy of this approach, we mounted a decryption failure attack on a toy FrodoKEM [8] parameter set. We set $n = m = 80$ and $q = 2^{11}$, while we kept the secret/error distribution identical to that of the frodo-640 parameter set. This had the benefit of reducing the initial hardness of each instance to $\beta \approx 45$, while raising the empirical decryption failure rate to 0.44.

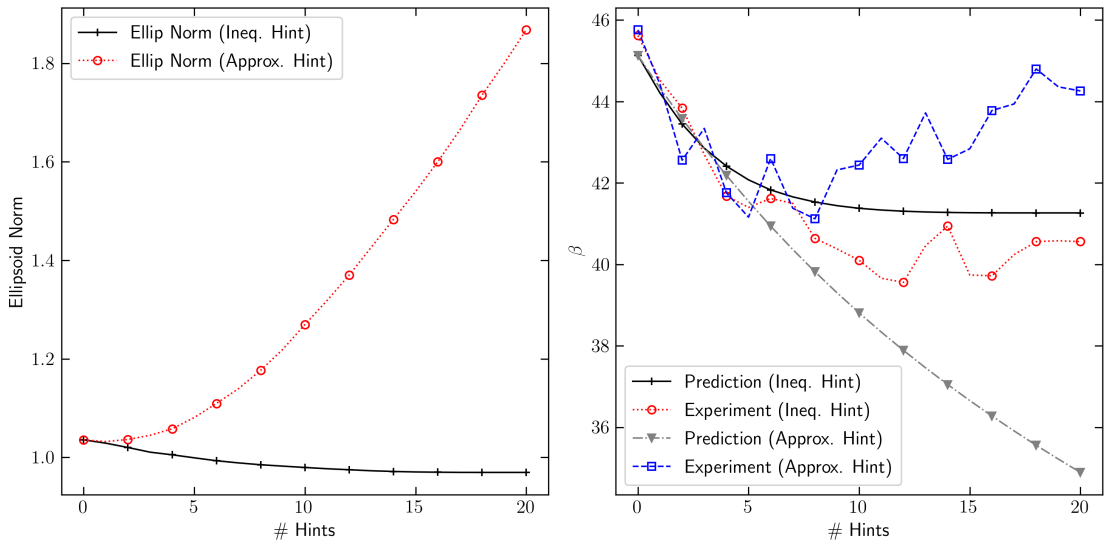


Figure 4.2: 50 key pair average ellipsoid norm (left), predicted β and experimental β (right) for integrating up to 20 decryption failures from a toy frodo-80 parameter set with $n = m = 80$, $q = 2^{11}$. Inequality hints were integrated in order of decreasing α (as defined in (2.6)).

We then generated a small database of 20 decryption failures for each of 50 different key pairs. For each key pair we integrated the decryption failures as both inequality hints and full dimensional approximate hints using the approach from [41]. After integrating each hint, we recorded both the predicted and experimental β as well as the ellipsoid norm for both approaches. A plot of the averages from all 50 key pairs can be seen in Figure 4.2.

In the left figure, the ellipsoid norm of the LWE secret increases with integration of the approximate hints. An ellipsoid norm greater than 1 indicates that the secret is not contained in the DBDD ellipsoid. Since the hardness estimates assume that the secret *is* contained in the DBDD ellipsoid, they are far lower than the experimental BKZ-β. With decryption failures modeled as inequality hints, the predicted loss in β is more modest, but the experimental β effectively matches the predictions. The

decrease in β levels off after around 10 inequality hints are integrated. For full-sized decryption failures (discussed next) we introduce a "regeneration" technique to allow for continued progress.

Full-sized Decryption Failures. We experimented with applying our inequality hint approach to the recent decryption failure attack of Fahr et al. [51]. In their work, the public key of FrodoKEM [8] (NIST level 1 frodo-640) was altered by injecting faults via the Rowhammer exploit to significantly increase the decryption failure rate (by effectively lowering the decryption failure threshold). This enables an attacker to search for failing (honestly generated) ciphertexts in a reasonable amount of time and thus this scenario is more amenable for the experiments in this section.

When instantiating our naive approach in the Fahr et al. [51] setting, we find that while the first batch of hints reduce the volume as expected (e.g. for the first hint if a vector $\boldsymbol{w}$ causes decryption to fail with probability p over choice of secret key, then we see a reduction of volume by nearly a factor of p), hints quickly lose their efficacy, until almost no progress is made in terms of volume reduction as new hints are integrated. In fact, we found that the center of the successive ellipsoids obtained by integrating a sequence of inequality hints converges very quickly to a feasible solution (i.e. a solution that satisfies all the linear constraints). This is due to the one-sided nature of the linear inequalities corresponding to decryption failures. The intersection of a finite number of halfspaces pertaining to these inequalities is an unbounded region of space. Thus, a feasible solution sufficiently inside this region will not be affected by any new constraints of the same form.

After ≈ 200 hints for simulated failures on Frodo-640 [8] the center, $\boldsymbol{\mu}$ itself satisfied *all prior and future* inequality hints, which corresponds to a terminating condition in the ellipsoid method. The full key recovery attack of Fahr et al. [51] required $\approx 100,000$ hints, so reaching a feasible solution after 200 hints is quite surprising. Unfortunately, the Euclidean distance between $\boldsymbol{\mu}$ and the true LWE secret/error remained quite large, so $\boldsymbol{\mu}$ itself was not a good candidate solution. Nevertheless, we found that $\boldsymbol{\mu}$ contains a lot of information about $\boldsymbol{s}$: We argue next that if $\boldsymbol{\mu}$ satisfies all hints, then $\langle \boldsymbol{\mu}, \boldsymbol{s} \rangle \geq \langle \boldsymbol{s}, \boldsymbol{s} \rangle \approx \sigma_s^2 \cdot n$.

As observed in [41], the distribution of hint vectors $\boldsymbol{w}$ decomposes as $\boldsymbol{w} = \alpha \cdot \boldsymbol{s}/||\boldsymbol{s}|| + \boldsymbol{w}'$, where α is a random variable with expectation $\approx t/||\boldsymbol{s}||$ (where t is the decryption failure threshold) and $\boldsymbol{w}'$ is a zero-centered random variable orthogonal to $\boldsymbol{s}$. So for a fixed center $\boldsymbol{\mu}$,

$$\mathbb{E}[\langle \boldsymbol{\mu}, \boldsymbol{w} \rangle] = \mathbb{E}[\alpha] \cdot \langle \boldsymbol{\mu}, \boldsymbol{s} \rangle / ||\boldsymbol{s}|| \approx t \cdot \langle \boldsymbol{\mu}, \boldsymbol{s} \rangle / ||\boldsymbol{s}||^2 = t \cdot \langle \boldsymbol{\mu}, \boldsymbol{s} \rangle / \langle \boldsymbol{s}, \boldsymbol{s} \rangle.$$

If we find empirically, for a sufficiently large hint database, that $\mathbb{E}[\langle \boldsymbol{\mu}, \boldsymbol{w} \rangle] \geq t$– which occurs if $\boldsymbol{\mu}$ satisfies all previous and future hint inequalities–it implies that $\langle \boldsymbol{\mu}, \boldsymbol{s} \rangle \geq \langle \boldsymbol{s}, \boldsymbol{s} \rangle$.

Inequality Hints with Regeneration. To solve the issue of stalled progress as well as issues of numerical precision, we developed the regeneration approach in Algorithm 4.3.1.

When the center $\boldsymbol{\mu}$ of the successive ellipsoids becomes such that $\langle \boldsymbol{\mu}, \boldsymbol{s} \rangle \geq \sigma_s^2 \cdot d$, we simply use $\boldsymbol{\mu}$ itself to perform an inequality hint on a fresh DBDD instance.

Algorithm 4.3.1 Integrating Decryption Failures Using Regeneration

Input: System of decryption failure hints: $(\boldsymbol{W}, \boldsymbol{\gamma})$, LWE instance, Maximum allowed regenerations: MaxRegen

Output: DBDD instance with integrated decryption failures

```
W_regen := [ ]
for i = 0 to MaxRegen − 1 do
    DBDD ← LWE.embed()
    for j = 0 to i do
        DBDD.IntegrateIneqHint(−W_regen[j], −LWE.σ_s^2 · LWE.d)
                                          ▷ Inequality hints formulated for ≤
    end for
    while Mean(DBDD.μ · W^T) < LWE.t do                ▷ Failure threshold
        IntegrateNextHint(DBDD, W, γ)     ▷ Use some hint integration strategy
    end while
    append DBDD.μ to W_regen
end for
return DBDD
```

Specifically, we regenerate the initial ellipsoid according to our embedding and integrate the hint $\langle \boldsymbol{\mu}, \boldsymbol{s} \rangle \geq \sigma_s^2 \cdot d$. Once this is done, we find that we can again make progress for some time by integrating more decryption failures. When progress stalls again, we simply regenerate again.

An attacker cannot directly check the condition for regeneration ($\langle \boldsymbol{\mu}, \boldsymbol{s} \rangle \geq \sigma_s^2 \cdot d$). Instead, the attacker can use the empirical value of $\mathbb{E}[\langle \boldsymbol{w}, \boldsymbol{\mu} \rangle]$, calculated using all $\boldsymbol{w}$ corresponding to failing ciphertexts in the attacker's database. In the case that $\langle \boldsymbol{\mu}, \boldsymbol{s} \rangle \geq \sigma_s^2 \cdot d$, we expect $\mathbb{E}[\langle \boldsymbol{w}, \boldsymbol{\mu} \rangle] \geq (1 + \epsilon) \cdot t$ where ϵ is a safety margin due to the uncertainty in the empirical expected value.

To evaluate the effectiveness of regeneration compared to the full dimensional approximate hint-based hardness estimates in [41], we continued to use the scenario of Fahr et al. [51]. For our experiment, we generated several simulated public keys (i.e. we directly modified honestly generated public keys to reproduce the result of

the fault injection). Then, we searched for 4000 failing ciphertexts for each key. We integrated these 4000 hints as full-dimensional approximate hints and inequality hints with regeneration on two separate DBDD instances. We set the decryption failure threshold $t = 1024$ corresponding to the effect of the fault injection attack. The results can be seen in Table 4.1.

	Full-Dimen. Approx. Hints	Inequality Hints				
		Key 1	Key 2	Key 3	Key 4	Key 5
Initial BKZ–β	**487.08**	487.08	487.08	487.08	487.08	487.08
Ciphertexts	**4000**	1224	1106	959	986	965
Ellipsoid Norm	–	322.61	213.55	590.57	546.58	485.14
Final BKZ–β	**307.85**	295.68	279.78	284.47	283.67	284.70

Table 4.1: Comparison of BKZ blocksize β estimates for a fault injection assisted decryption failure attack using 4000 failing ciphertexts for 5 different (simulated) poisoned Frodo-640 public keys. The final scaled estimates result from shrinking the final ellipsoid by a factor of $\mathsf{Rank}(\boldsymbol{\Sigma})/\|\boldsymbol{s}\|_{\boldsymbol{\Sigma}}$.

Since the obtained $E^{(\mathsf{Rank})}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ no longer represents a multivariate Gaussian distribution (unlike in [41]), the expected value of the rank-scaled ellipsoid norm of the LWE secret may be far less than the rank. Observe that in Table 4.1, the ellipsoid norm of the secret is less than half of the rank, which is 1279. To account for this, we compute the estimated BKZ–β using equation (2.11). This is equivalent to scaling the $E^{(\mathsf{Rank})}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ until the ellipsoid norm of the secret is equal to the rank, and then applying equation (2.13).

As highlighted in Section 4.2.1, the volume reduction incurred when integrating an inequality hint is almost entirely determined by the geometric parameter α defined in (2.6), since the determinant of $\boldsymbol{\Sigma}$ scales by $(1 - \alpha^2)^d$. Using our regeneration approach, we were able to achieve improved β levels compared to the full dimensional

approximate hints approach of [41] by integrating only 959-1224 hints in total, as opposed to 4000 hints. To achieve this, we used a greedy algorithm that at each stage chose the hint with the largest α value to integrate.

We further note that while it is possible to obtain a β estimate using the full-dimensional approximate hint approach from the prior work by utilizing the ultra-lightweight version of the framework [41], it is not possible to compute the uSVP lattice basis itself required to run BKZ due to high computational overhead. In contrast, our new inequality hint-based method is more efficient and so allows us to compute the final ellipsoid covariance matrix necessary for the reduction from DBDD to uSVP, and consequently would allow one to run a full key recovery attack given a sufficient number of hints.

A geometric approach to failure boosting. The α value for a candidate hint, given the current information encapsulated by the ellipsoid, can be used as a proxy for the probability that the query will lead to decryption failure: The smaller α is, the higher the probability of decryption failure. Thus, computing this α value *before* submitting a decryption query provides a geometric analogue to the failure boosting approach of D'anvers et al. [44].

We tested this on a small scale by again using the scenario presented by Fahr et al. [51]. Here, instead of generating a database of 4000 failing ciphertexts, we generated a database of $100k$ candidate ciphertexts (note that in the Fahr et al. [51] there is a way to filter candidate ciphertexts that have a relatively high chance of causing a decryption failure). Of these, only 34 actually caused decryption failures (this is consistent with the decryption failure rate (DFR) for filtered ciphertexts

reported by Fahr et al. [51]). We integrated these 34 failing ciphertexts as inequality hints in order of decreasing α, each time calculating the histogram of α values for the remaining ciphertext database. Figure 4.3 shows the evolution of the histogram as more hints are integrated.

Next, we looked to quantify the number of decryption queries required to find all 34 failures, compared to naively querying the database by a linear scan. All 34 failures had α values in the $[0.07, 0.12]$ range, so we only submitted decryption queries for ciphertexts with corresponding α values at each step sorted in ascending order. To obtain all 34 decryption failures in the database, we found that it took 39785 queries versus 94894 for a linear scan.

Essentially, we can profile the range of α values for the i-th hint and obtain a range $[\alpha^i_{\mathsf{low}}, \alpha^i_{\mathsf{high}}]$, for which a decryption failure is most likely. We can then run the following online algorithm when making decryption queries to find the i-th hint to integrate into the ellipsoid: Let S be the set of all failing decryption queries made up to this moment. First, search S to try to find a query with α value in the range $[\alpha^i_{\mathsf{low}}, \alpha^i_{\mathsf{high}}]$ with respect to the current ellipsoid. If such a query is found, integrate it into the ellipsoid. Otherwise, generate a set S' of candidate hints of some calibrated size s'. For each $\boldsymbol{w} \in S'$, compute its α value. If $\alpha \notin [\alpha^i_{\mathsf{low}}, \alpha^i_{\mathsf{high}}]$ then remove $\boldsymbol{w}$ from S'. Sort the entire set S' from smallest to largest α value. Make decryption queries in this order until a failing ciphertext is found. Once found, add $\boldsymbol{w}$ to S and integrate $\boldsymbol{w}$ into the current ellipsoid.

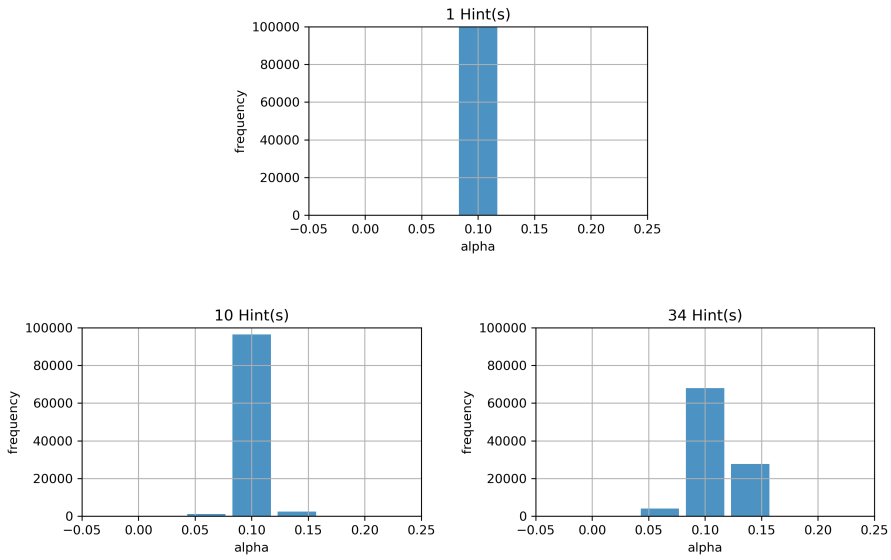


Figure 4.3: Histograms of α values for a database of $100k$ ciphertexts after integrating 1 (top), 10 (bottom-left), and 34 (bottom-right) inequality hints based on the failing ciphertexts in the database.

4.3.3 Combining Decryption Failure and SCA

We illustrate our "Combined Hints" approach from Section 4.2.2 by combining information on a single $(\boldsymbol{e}||\boldsymbol{s})$ pair from a decryption failure and a side-channel attack. In a recent work, Fahr et al. [51] showed that, for FrodoKEM, obtaining m' number of vectors corresponding to random decryption failures, scaling them by a constant that depends on the parameters of the cryptosystem, and taking their coordinate-wise mean, approximates a draw from the distribution $\mathcal{D}' := (\boldsymbol{e}||\boldsymbol{s}) + \mathcal{W}''_{(m)}$, where the error $\mathcal{W}''_{(m)}$ is a d-dimensional Gaussian with mean $\mathbf{0}$ and covariance matrix $\sigma^2_{df} \cdot \boldsymbol{I}_d$, where $\sigma^2_{df} \leq d^2\sigma_1^6/(t^2m')$, σ_1^2 is the error of the original distribution, d is the dimension of the LWE secret/error, and t is the decryption failure threshold. Rearranging terms, given a draw $\boldsymbol{\mu}_{df} \sim \mathcal{D}'$, the secret is equal to $\boldsymbol{s} = \boldsymbol{\mu}_{df} + \mathcal{W}''_{(m)}$. This means that the secret is contained in the rank-scaled ellipsoid $E^{(\mathsf{Rank})}(\boldsymbol{\mu}_{df}, \boldsymbol{\Sigma}_{df})$, where $\boldsymbol{\Sigma}_{df} = \sigma^2_{df} \cdot \boldsymbol{I}_d$.

Note that we could have used the results of Section 4.3.2 to obtain a DBDD instance with better parameters than the one corresponding to $E^{(\mathsf{Rank})}(\boldsymbol{\mu}_{df}, \boldsymbol{\Sigma}_{df})$. Further, this instance would no longer correspond to a non-centered Gaussian distribution over the secret, and in fact the PDF of the secret would be unknown. Obtaining such a DBDD instance with the target σ_{df}^2 value needed for our experiment would be very computationally intensive. Therefore, for our illustration of the combined hints technique, we use the rank-scaled ellipsoid $E^{(\mathsf{Rank})}(\boldsymbol{\mu}_{df}, \boldsymbol{\Sigma}_{df})$ described above to capture the DBDD instance obtained from the decryption failure information.

Bos et al. [30] studied the feasibility of single-trace power analysis of the Frodo Key Encapsulation Mechanism (FrodoKEM). Subsequently, Dachman-Soled et al. [41] used this information to conduct a side-channel attack on FrodoKEM on various parameter sets (CCS1, CCS2, CCS3, CCS4, NIST1, NIST2). Dachman-Soled et al. [41] used the score tables constructed from Bos et al. [30] to form an a posteriori distribution incorporating the side-channel information and used the information from the distribution tables to "guess" a large subset of coordinates when the confidence in the guess (where the confidence was calculated using the aforementioned score tables) was sufficiently high.

4.3.3.1 The Baseline Approach

The prior work [41] incorporated the side-channel information, represented by DBDD instance with $E^{(\mathsf{Rank})}(\boldsymbol{\mu}_{sc}, \boldsymbol{\Sigma}_{sc})$, using approximate *a posteriori hints*. In this method, the mean and covariance matrix of the a posteriori distribution (say on the $\boldsymbol{s}$ variables only) is calculated and then fully replaces the part of the covariance

matrix in the DBDD instance that corresponds to the $\boldsymbol{s}$ variables. This method was suggested by [41] as an alternative to "conditioning" approximate hints. In our case, both $\boldsymbol{\Sigma}_{df}$ and $\boldsymbol{\Sigma}_{sc}$ are diagonal matrices. Therefore, the *a posteriori hints* approach, which we refer to as the *Baseline approach*, yields the following rank-scaled ellipsoid, $E^{(\mathsf{Rank})}(\boldsymbol{\mu}_{ba}, \boldsymbol{\Sigma}_{ba})$: For each $i \in [d]$, if $\boldsymbol{\Sigma}_{sc}[i][i] \leq \boldsymbol{\Sigma}_{df}[i][i]$, set $\boldsymbol{\Sigma}_{ba}[i][i] = \boldsymbol{\Sigma}_{sc}[i][i]$ and $\boldsymbol{\mu}_{ba}[i] = \boldsymbol{\mu}_{sc}[i]$. Otherwise, set $\boldsymbol{\Sigma}_{ba}[i][i] = \boldsymbol{\Sigma}_{df}[i][i]$ and $\boldsymbol{\mu}_{ba}[i] = \boldsymbol{\mu}_{df}[i]$.

4.3.3.2 Ellipsoid/Ellipsoid Intersection.

For an ellipsoid $E = (\mu, \Sigma)$ (resp. rank-scaled ellipsoid $E^{\mathsf{rank}} = (\mu, \Sigma)$), we denote by $E_S = (\mu_S, \Sigma_S)$ (resp. $E_S^{\mathsf{rank}} = (\mu_S, \Sigma_S)$) the ellipsoid (resp. rank-scaled ellipsoid) resulting from the restriction of the center and shape matrix of E (resp. E^{rank}) to a set of coordinates S. For an ellipsoid E, we denote by n_E the ellipsoid norm of the correct solution with respect to E. Let $E_{DF}^{\mathsf{rank}} = E^{(\mathsf{Rank})}(\boldsymbol{\mu}_{df}, \boldsymbol{\Sigma}_{df})$ and $E_B^{\mathsf{rank}} = E^{(\mathsf{Rank})}(\boldsymbol{\mu}_{ba}, \boldsymbol{\Sigma}_{ba})$. Restricting to the set S of secret (no error coordinates), let $E_{int,S}^{\mathsf{rank}} = E^{(\mathsf{Rank})}(\boldsymbol{\mu}_{int,S}, \boldsymbol{\Sigma}_{int,S})$ be the ellipsoid circumscribing the intersection of $E_{DF,S}^{\mathsf{rank}} = E^{(\mathsf{Rank})}(\boldsymbol{\mu}_{df,S}, \boldsymbol{\Sigma}_{df,S})$ and $E_{B,S}^{\mathsf{rank}} = E^{(\mathsf{Rank})}(\boldsymbol{\mu}_{ba,S}, \boldsymbol{\Sigma}_{ba,S})$. Let the diagonal of $\boldsymbol{\Sigma}_{ba,S}$ be denoted by $(\sigma_{2,1}^2, \ldots, \sigma_{2,n}^2)$. Let $\boldsymbol{c} = \boldsymbol{\mu}_{ba,S} - \boldsymbol{\mu}_{df,S}$. We simplify (4.6) and (4.7) as follows:

$$\mathcal{F}(\boldsymbol{\Sigma}_{int,S}) = k\boldsymbol{X}^{-1}; \qquad \boldsymbol{X} = \tilde{\lambda}\mathcal{F}(\boldsymbol{\Sigma}_{df,S})^{-1} + (1-\tilde{\lambda})\mathcal{F}(\boldsymbol{\Sigma}_{ba,S})^{-1}$$

$$\boldsymbol{\mu}_{int} = (\boldsymbol{\mu}_{df,S}\tilde{\lambda}\mathcal{F}(\boldsymbol{\Sigma}_{df,S})^{-1} + \boldsymbol{\mu}_{ba,S}(1-\tilde{\lambda})\mathcal{F}(\boldsymbol{\Sigma}_{df,S})^{-1})\boldsymbol{X}^{-1}$$

$$k = 1 - \tilde{\lambda}(1-\tilde{\lambda}) \cdot \frac{1}{n} \sum_{i \in [n]} \frac{c_i^2}{\tilde{\lambda}\sigma_{2,i}^2 + (1-\tilde{\lambda})\sigma_{df}^2}$$

and $\tilde{\lambda} \in [0,1]$ is the value that minimizes the determinant of $\mathbf{\Sigma}_{int,S}$. Specifically,

$$\det(\mathbf{\Sigma}_{int,S}) = k^n \cdot \prod_{i \in [n]} \left(\frac{\tilde{\lambda}}{\sigma_{df}^2} + \frac{1-\tilde{\lambda}}{\sigma_{s_2,i}^2} \right)^{-1}. \tag{4.14}$$

The terms in the product on the right side of equation (4.14) correspond to weighted harmonic means of σ_{df}^2 and $\sigma_{2,i}^2$, for each i. While the harmonic mean tends towards the smaller element, it is at least as large as the minimum of the two values. This is then compensated by multiplication by k, which is always at most 1. However, due to the negative influence of the harmonic mean on the final determinant, we experiment with intersecting only on coordinates i for which the gap between σ_{df}^2 and $\sigma_{2,i}^2$ is not too large.

4.3.3.3 Conditions 1 and 2.

We consider two candidate methods of performing intersection: In the first method, referred to as **Condition 1**, we restrict the intersection to the dimension $n-g$ ellipsoids (where g is the number of guesses) corresponding to the coordinates of the LWE secret (but not the error) that are not guessed. This is essentially equivalent to performing the intersection after guesses are made on the remaining coordinates of the LWE secret. For the remaining coordinates, we follow the baseline approach. In the second method, referred to as **Condition 2**, we restrict the intersection to the dimension n' ellipsoids corresponding to the coordinates i of the LWE secret (but

	Baseline Approach	Combined Hints	
		Condition 1 unknown/known	Condition 2 unkown/known
Original BKZ–β	**268.83**	–	–
DF BKZ–β	**203.02**	–	–
SC BKZ–β before guess	**114.22**	–	–
SC BKZ–β after guess	**68.65**	–	–
$\ln(V_{int}/V_{base})$	–	-26.49/-30.47	-23.46/-32.24
BKZ–β before guesses	**97.86**	95.91/94.83	96.22/94.66
Number of guesses	**190**	190/190	190/190
Guess Success %	**0.76**	0.76/0.76	0.76/0.76
Final BKZ–β	52.20	50.19/ 49.18	50.50/ 49.00

Table 4.2: **Comparison of bikz estimates for FrodoKEM with CCS1 parameters.** Results are the average of 150 randomly generated instances. Starting from the top row, we report the original bikz, the bikz for only the decryption failure attack, and the bikz for only the side channel attack, before and after guesses (throughout we condition on all guesses being correct). We compare the baseline approach (Section 4.3.3.1) with two combined hints approaches using Condition 1 or 2 (Section 4.3.3.3) to select the set of coordinates for intersection. For each, we consider the known and unknown cases (Section 4.3.3.4). We next report the ln of the ratio of the volumes of the intersected and baseline ellipsoids (for the unknown case, these are reported after calibration (Section 4.3.3.4)). For each, we report the bikz without guesses, the number of guesses, the probability that all guesses are correct and the final bikz after guesses.

not the error), for which $\sigma^2_{2,i}$ is in the range $[\frac{\sigma^2_{df}}{5}, \sigma^2_{df}]$. For the remaining coordinates, we again follow the baseline approach.

4.3.3.4 The known and unknown cases

Let $E_{DF,S} = E(\boldsymbol{\mu}_{df,S}, \mathcal{F}(\boldsymbol{\Sigma}_{df,S}))$ and $E_{B,S} = E(\boldsymbol{\mu}_{ba,S}, \mathcal{F}(\boldsymbol{\Sigma}_{ba,S}))$. We restrict ellipsoids $E_{DF,S}$ and $E_{B,S}$ to a set of coordinates $P \subseteq S$ corresponding to Condition 1 or 2, yielding $E_{DF,P}$ and $E_{B,P}$. These are then intersected to yield $E_{int,P}$, and $E_{int,P}$ is substituted for the set of P coordinates in $E_{B,S}$ yielding $E_{int,S}$. To maintain consistency of hardness estimates, we would like to keep $\mathsf{n}_{E_{int,P}} = \mathsf{n}_{\mathsf{E}_{\mathsf{B,P}}}$. Further, ellipsoid/ellipsoid intersection performs best when intersecting two ellipsoids $E_{DF,P}$

and $E_{B,P}$ such that $\mathsf{n}_{E_{DF,P}} = \mathsf{n}_{E_{B,P}} = 1$, since points on the surface of both $E_{DF,P}$ and $E_{B,P}$ also lie on the surface of $E_{int,P}$.

Assuming that $\mathsf{n}_{E_{DF,P}}$ and $\mathsf{n}_{E_{B,P}}$ are known, we scale $E_{DF,P}$ by $\mathsf{n}_{E_{DF,P}}$ and $E_{B,P}$ by $\mathsf{n}_{E_{B,P}}$, so that the correct solution lies on the surface of both scaled ellipsoids, and hence on the surface of $E_{int,P}$. We then scale $E_{int,P}$ by $1/\mathsf{n}_{E_{B,P}}$, to ensure that $\mathsf{n}_{E_{int,P}} = \mathsf{n}_{E_{B,P}}$. This yields the optimal volume reduction while maintaining the norm constraint but requires knowledge of $\mathsf{n}_{E_{DF,P}}$ and $\mathsf{n}_{E_{B,P}}$. We refer to this case as the **known** case.

While $\mathsf{n}_{E_{DF,P}}$ is fairly stable (since the decryption failure ellipsoid is a multivariate Gaussian in our experiments), $\mathsf{n}_{E_{B,P}}$ can fluctuate. We therefore also explore the case in which the adversary is not presumed to know $\mathsf{n}_{E_{DF,P}}$ and $\mathsf{n}_{E_{B,P}}$. We refer to this case as the **unknown** case, and we next describe the algorithm for this case. We find experimentally that with probability at least $1/2$, $\mathsf{n}_{E_{DF,P}} \leq 0.9 \cdot \mathsf{n}_{E_{B,P}}$. We scale $E_{DF,P}$ by 0.9 before intersection. In the case that indeed $\mathsf{n}_{E_{DF,P}} \leq 0.9\mathsf{n}_{E_{B,P}}$, we have by Remark 4.2.3, that $\mathsf{n}_{E_{int,P}} \leq \mathsf{n}_{E_{B,P}}$. In the case that $\mathsf{n}_{E_{DF,P}} > 0.9\mathsf{n}_{E_{B,P}}$, it may be the case that $\mathsf{n}_{E_{int,P}} > \mathsf{n}_{E_{B,P}}$[5] To take into account the fact that $\mathsf{n}_{E_{int,P}}$ can now be smaller or larger than $\mathsf{n}_{E_{B,P}}$, we use Equation (2.11) to calibrate the predicted β value with respect to the entire instance (including error coordinates).

We present our experimental results with decryption failure information modeled as described above, with $\sigma_{df}^2 = 0.25$, and with side-channel data obtained from the single trace attack of Bos et al. [30] on FrodoKEM. As in [41], we incorporate

[5]Note that in our experiments it was always the case that $1/0.9\mathsf{n}_{E_{DF,P}} \leq 1$ so the intersection is always non-empty.

guesses when the side-channel distribution for a secret coordinate allows for a high confidence guess. Table 4.2 displays the predicted hardness (in bikz) of the original and baseline DBDD instances, the intersected instances obtained using Condition 1 and 2, in both the known and unknown cases, both with and without guesses, for the CCS1 parameter set. [6] Our approach lowers the required number of bikz as compared to the baseline approach by 2-3 bikz.

[6]The number of bikz reported in our table for the SCA-only attack differs slightly from the bikz reported in [41], as we use the updated code found here: https://github.com/lducas/leaky-LWE-Estimator/tree/fix_extreme_hints2.

Chapter 5: A Concrete Analysis of Wagner's k-List Algorithm over $\mathbb{Z}_p$

5.1 Introduction

In 2002, Wagner introduced a generalization of the venerable *birthday problem* [143]. This generalized birthday problem tasks solvers with finding $x_i \in L_i$, $i \in \{1, \ldots, k\}$ such that $x_1 + \cdots + x_k = 0 \pmod{p}$, where p is a prime and each L_i is a list of random elements in $\mathbb{Z}_p$. To solve this problem, Wagner introduced a subexponential time algorithm dubbed the k-tree (or k-list) algorithm. This algorithm (and its variants) see use in cryptanalyzing myriad schemes and protocols.

Of particular importance, Wagner's k-list algorithm can be applied to attack the security of a variety of interactive signature schemes such as threshold-, multi-, and blind signature schemes [22, 48, 124, 143].

Currently, the runtime of these attacks depends on the conjecture that Wagner's algorithm [143] outputs a solution with noticeable probability using initial lists of size $O(p^{\frac{1}{1+\ell}})$ where $k = 2^\ell$. Presently, the tightest analysis of the k-list algorithm over $\mathbb{Z}_p$ is due to Shallue [130]. Shallue was able to prove that Wagner's algorithm yields a solution with overwhelming probability with initial lists of size $O(p^{\frac{1}{\ell}})$. While

Shallue's analysis asymptotically matches the conjecture, it hides large polynomial terms and leaves a substantial gap with respect to concrete parameters. For example, the aforementioned attacks of Drijvers et al. [48] and Benhamouda et al. [22] consider attacks on schemes over 256 and 512 bit groups which require anywhere from $k = 4$ to $k = 64$ lists. For these choices of parameters, the $+1$ term in the exponent can easily determine the difference between whether the attacks are feasible. In this work, we revisit Wagner's algorithm and provide the first meaningful analysis of the required initial list sizes for practical values of k.

5.1.1 Contributions

We now give a more detailed overview of our problem statement and our contributions. For clarity, we present the original version of Wagner's k-list algorithm in Figure 5.1. In his beautifully simple algorithm, $k = 2^\ell$ initial lists $L_1^\ell, ..., L_k^\ell$ of m random elements in $\mathbb{Z}_p = I_\ell$ each are merged together in pairs. List elements are given by their representatives in I_ℓ and this (centered) modular reduction is utilized throughout the execution of the algorithm. Note also that for $k \neq 2^\ell$, Wagner suggests invoking the algorithm with k' lists, where $k' = 2^{\lfloor \log k \rfloor}$ [143].

The merging of pairs creates $k/2$ new lists $L_1^{\ell-1}, ..., L_k^{\ell-1}$, where L_i consist of sums of the form $a + b$ where $a \in L_{2i-1}, b \in L_{2i}$ and $a + b \pmod p$ lies in the interval $I_{\ell-1} := \left[\left\lfloor -\frac{p-1}{2 \cdot 2^{\lfloor \log m \rfloor}} \right\rfloor, \left\lfloor \frac{p-1}{2 \cdot 2^{\lfloor \log m \rfloor}} \right\rfloor\right]$. The idea is to shrink the domain in a geometric progression based on the number of initial list elements m (thus balancing the expected size of the merged lists). This step is now recursively repeated over ℓ

Algorithm 5.1.1 k-list Algorithm

Input: $k = 2^\ell$ lists L_i of m random elements in $\mathbb{Z}_p$.
Output: $x_1, ..., x_k$ s.t. $x_i \in L_i$ and $\sum_i x_i = 0 \pmod p$.

for all $j \in [k]$ **do**
    set $L_j^\ell := L_j$.
end for
for all $j \in [\ell]$ **do**
    set $I_j := \left[\left\lfloor -\frac{p-1}{2^{\lfloor \log m \rfloor \cdot (\ell - j) + 1}} \right\rfloor, \left\lfloor \frac{p-1}{2^{\lfloor \log m \rfloor \cdot (\ell - j) + 1}} \right\rfloor\right]$.
end for
for $i = \ell$ down to 1 **do**
    for $j \in [2^{i-1}]$ **do**
        $L_j^{i-1} = \{a + b \pmod p : a \in L_{2j-1}^i, b \in L_{2j}^i, a + b \in I_{i-1} \pmod p\}$
    end for
end for
if $0 \in L_1^0$ **then**
    determine elements $x_1, ..., x_k$ s.t. $\sum_i x_i = 0$ and $x_i \in L_i$.
    return $x_1, ..., x_k$ if such elements exist.
else
    return $\perp$.
end if

Figure 5.1: The original version of Wagner's k-list algorithm over $\mathbb{Z}_p$ [143].

many levels until a list L_1^0 is produced. The algorithm is deemed successful if $0 \in L_1^0$.

Existing Analyses and Their Limitations. Wagner's original analysis of the k-list algorithm provides a heuristic that the algorithm should succeed with constant probability when the initial lists are of size $O(p^{\frac{1}{1+\ell}})$ when elements in the initial lists are sampled from $\mathbb{Z}_p$. The analysis relies on the heuristic invariant that the elements in the lists are sampled uniformly and independently of the increasingly shrinking domains at each level.

However, it is simple to verify that the heuristic is not guaranteed to hold. At each level, the output lists consist of *all* pairs of elements whose sum fits inside the smaller domain $\pmod p$. Consider a merge of a list L_1 with two elements $x_1 = x_1'$

and a list L_2 with an element x_2 such that $x_1 + x_2$ meet the merge criterion. Then, the merged list L_{12} will contain both $x_1 + x_2$ and $x_1' + x_2$. As such, it is possible for multiple merged list elements to share an input element, which can violate independence. The violation of uniformity follows from the act of summation. The sum of two uniform random variables is no longer uniform.

This presents a problem when attempting to analyze the algorithm without using the heuristic assumptions. It is extremely difficult to calculate the explicit probability distribution of output list elements due to the lack of independence (and no guarantees on the exact nature of the dependence between list elements). Shallue worked around some of these constraints by utilizing the theory of Martingales to bound the non-uniformity and dependence of the elements in each merged list [130]. As discussed above, this results in loose bounds on the runtime particularly for practical (smaller) values of k. Thus, his work leaves open the question of giving concrete bounds on the running time of Wagner's algorithm as well as the attacks that build on it.

Our Approach: Analyzing a Degraded Algorithm. In this work, we take a completely different approach to analyzing the running time of Wagner's algorithm. Instead of analyzing his algorithm directly, we instead construct and analyze a variant of the k-list algorithm. Our algorithm can be viewed as an all-around worse version of Wagner's original algorithm (assuming the heuristic analysis holds). Our variant incurs a blow-up in initial list sizes that is polynomial in the number of lists. However, our algorithm provably achieves the uniformity and independence invariants of the

original algorithm and matches its running time and success probability when k is not too large. As such, our algorithm can be viewed as providing a lower-bound on the runtime of Wagner's.

It does so through a modified list merge procedure that we dub the Interval Merge. Like its name suggests, the interval merge procedure first divides the input domain into suitably sized intervals and selects a single element per interval from both input lists (if they exist). Unique sums are then created from these selected elements. Finally, rejection sampling is employed to enforce the uniformity of elements in the merged list. Naturally, this modification does not come for free. We incur (at most) a constant factor loss per level of the tree. We present our main theorem on the runtime of the degraded k-list algorithm.

Theorem 5.1.1. *If the Interval Merge algorithm in Figure 5.4 returns a list of size $c \cdot m_{in}$ for some constant $0 < c \leq 1$ given input lists of size m_{in}, then the degraded k-list algorithm in Figure 5.3 requires initial lists of size*

$$\Theta\left(\alpha^{\frac{-(\ell-1)}{\ell+1}} \cdot c^{\frac{-(\ell-1)(\ell+2)}{2(\ell+1)}} \cdot p^{\frac{1}{\ell+1}}\right),$$

and finds a solution with non-negligible probability, where α is the interval scaling factor input to IntervalMerge *and $k = 2^{\ell}$.*

Here α is a parameter input to the Interval Merge algorithm that affects the number of created intervals (and the maximum achievable c). For the graphs in Figure 5.2 we invoke Theorem 5.1.1 with $c = (1/6)$, and $\alpha = 0.79591$ to calculate the required sizes of the initial lists, and plot the runtimes on the y-axis. We justify

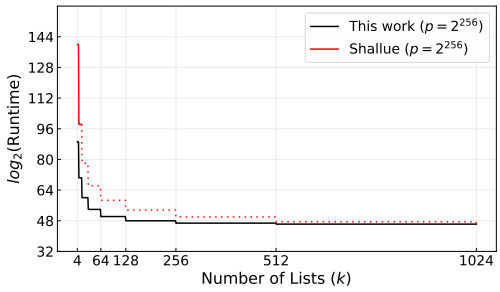
these choices of parameters with our analysis in Section 5.3. Note that we indeed manage to preserve the +1 in the denominator of the exponent(s). This allows our degraded variant of the algorithm to outperform Shallue's analysis [130] in small parameter regimes relevant to a wide variety of attacks on signature schemes.

For example, for a 256-bit prime p, and $k = 8$, our degraded k-list algorithm runs in time $\approx 2^{70.4}$, while Shallue's analysis states that the Wagner's original algorithm runs in time $\approx 2^{98.3}$. Furthermore, note that Shallue's analysis requires a technical assumption[1] that limits the parameter regimes in which it is valid. For a 256-bit p, Shallue's analysis holds only for k up to 8. However, in Figure 5.2, for ease of comparison, we extended the graphs showing the complexity beyond the limit of Shallue's analysis. In combination with Shallue's analysis, our new algorithm substantiates the original performance claims from Wagner's paper for a wide range of parameters for both theory and practice. As such, when accounting for the runtime of the attack by Benhamouda et al. [22], protocol designers would be wise to incorporate the original heuristic runtime of Wagner's algorithm.

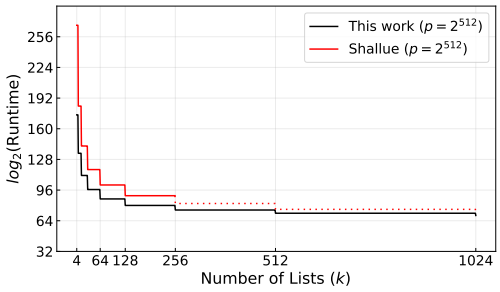
5.1.2 Related Work

Other analyses of the k-list algorithm. Wagner's k-list algorithm [143] has been re-analyzed multiple times since its original publication. In effect, there are two distinct versions of the algorithm, depending on the domain of the input list elements. The first considers lists of elements from an arbitrary finite field $\mathbb{Z}_p$ (which we

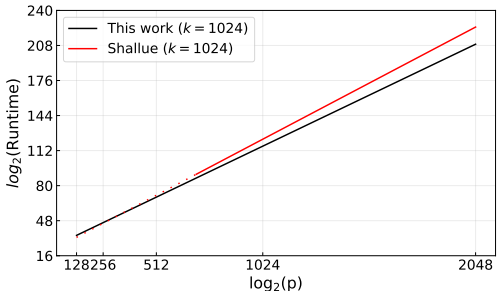
[1] For the range of parameters we are considering, this assumption can be written as $\log p > 70 \log k$.



(a) Comparison for 256-bit p



(b) Comparison for 512-bit p



(c) Comparison for $k = 1024$

Figure 5.2: Runtime comparison between the degraded k-list algorithm (Figure 5.3), and the analysis of Wagner's algorithm [143] by Shallue [130] varying k for $p = \{2^{256}, 2^{512}\}$ (a)(b), and varying p for $k = 1024$ (c). The dotted segments indicate the extension of Shallue's analysis beyond the range where his technical assumption holds.

discuss in this work). The second instead considers initial lists consisting of binary vectors (or equivalently from $\mathbb{Z}_{2^n}$), where the summation operator is replaced by coordinate-wise XOR. In this version, the goal of the k-list algorithm then becomes finding $x_i \in L_i$, $i \in \{0,1\}^n$ such that $x_1 \oplus \cdots \oplus x_k = \mathbf{0}$. The sums depicted in Figure 5.1 are replaced by finding list elements that match on an increasing number of low-order bits. Wagner's original analysis of this variant of the k-list algorithm states that it expects to find a single solution in time $O(k \cdot 2^{\frac{n}{1+\ell}})$ for $k = 2^\ell$.

For this variant of the algorithm, the analysis by Minder and Sinclair [101] is particularly illuminating. Minder and Sinclair were able to prove that Wagner's

algorithm *does* indeed meet its conjectured runtime in the binary vector case and outputs a solution with high probability. Minder and Sinclair's analysis of the failure probability of Wagner's algorithm relies on the fact the bits of random binary vectors are independent of each other. This allows them to prove the uniformity of the elements in all merged lists. With this, Minder and Sinclair are then able to bound the covariance between any two candidate solutions to the algorithm. Thus, they use Chebyshev's inequality to show the number of solutions is tightly concentrated around the expectation.

Note that the techniques used in Minder and Sinclair's analysis does not apply to arbitrary finite fields. In particular, the idea that the bits of input elements are independent of each other. One could certainly represent elements in $\mathbb{Z}_p$ by their bit-decomposition, but an XOR of two field elements in this representation will result in carries, breaking uniformity. We remark that Minder and Sinclair do provide a variant of their analysis for larger finite fields of prime powers $\mathbb{Z}_{p^n}$ for p prime, but this incurs a penalty of $\sqrt{p}$ on the runtime.

For the variant of the k-list algorithm acting on arbitrary finite fields, Lyubashevsky provided an analysis suited to solve the integer subset-sum problem [95]. Similarly to our proposed algorithm, Lyubashevsky only uses a subset of all valid summations during list merging. For this construction, the runtime is approximately $O(k \cdot p^{\frac{2}{1+\ell}})$. Shallue's analysis [130] can be viewed as an improvement to the efficiency of Lyubashevsky's, and thus also incurs a penalty in terms of the runtime ($O(k \cdot p^{\frac{1}{\ell}})$).

Applications of the k-list algorithm. Originally noted by Schnorr in 2001, multiple discrete logarithm-based blind signatures (including Schnorr [124] and Okamoto-Schnorr [110]) implicitly rely on an additional hardness assumption known as the ROS problem [56, 124]. An efficient solver for the ROS problem was noted to immediately produce a one more forgery attack on these signature schemes. Wagner [143] showed that the k-list algorithm for inputs over $\mathbb{Z}_p$, where p is the group modulus of the signature scheme, can be used to solve the ROS problem. In order to craft the proper inputs to the ROS solver, the attacker must use multiple *parallel* signing sessions.

More recently, Benhamouda et al. [22] showed that the ROS problem is solvable in polynomial time given that the dimension of the problem—the number of concurrent signing sessions—is larger than $\log p$. In addition to their polynomial time attack, Benhamouda et al. introduced a generalized variant that offered smooth trade-offs between the number of open sessions and attack difficulty. They did so by first applying Wagner's algorithm (but terminating early) to constrain the size of inputs to the ROS problem, and then applying their original polynomial time attack. As a consequence, the runtime of Benhamouda et al.'s generalized attack relies on the conjectured runtime of Wagner's algorithm over $\mathbb{Z}_p$. In addition, Benhamouda et al. were able to show that their attack techniques can be applied to threshold- and multi-signature schemes such as those found in [58, 86, 97, 135] by refining techniques presented by Drijvers et al. [48].

The k-list algorithm has other cryptanalytic use cases. Many of these applications make use of the binary, i.e. XOR version, of the algorithm. Some direct applications help to break secret-key ciphers, to give a few examples [32, 77, 91]. The

hash function proposed [12] was shown to be vulnerable to a k-list approach in [40], its successor FSB [11] was a SHA-3 candidate and despite additional precautions to withstand such attacks, some vulnerabilities remained as shown in [24].

Extensions of the k-list algorithm. Generalizations of Wagner's algorithm have been considered in numerous directions. The case where the number of lists is not a power-of-two is considered in [104]. Its generalization to quantum computers is considered in [60]. Another extension led to the design of the representation technique giving improved algorithms for the subset-sum problem and the binary decoding problem [13, 19, 21, 23, 27, 50, 72].

5.2 Preliminaries

For any nonnegative interval $I = [a, b]$, $b \geq a \geq 0$, let $-I$ denote the complementary interval $-I := [-b, -a]$. Throughout this chapter, we will need to make reference to the occupancy of complementary pairs of intervals.

Definition 5.2.1. Given a complementary pair of intervals $I, -I$ and two lists L_1, L_2, we say the pair of intervals is *fully occupied* if there exist elements $x_1 \in L_1$ and $x_2 \in L_2$, such that $x_1 \in I$ and $x_2 \in -I$.

Definition 5.2.2. Given a complementary pair of intervals $I, -I$ and two lists L_1, L_2, we say the pair of intervals is *half occupied* if there exists an element $x_1 \in L_1$ and for all $x_2 \in L_2$, $x_1 \in I$ and $x_2 \notin -I$ or there exists an element $x_2 \in L_2$ and for all $x_1 \in L_1$, $x_2 \in -I$ and $x_1 \notin I$.

5.3 Analysis of the Degraded k-List Algorithm

In this section, we present an analysis of a degraded version of the k-list algorithm.

Algorithm 5.3.1 Degraded k-List Algorithm

Input: $k = 2^\ell$ lists L_i of m uniform and independent elements in $\mathbb{Z}_p$, represented as integers in $\left[-\lfloor \frac{p}{2} \rfloor, \lfloor \frac{p}{2} \rfloor\right]$, and interval scaling factor $\alpha > 0$.
Output: $x_1, ..., x_k$ s.t. $x_i \in L_i$ and $\sum_i x_i = 0$.

1: **for** $i = \ell$ down to 2 **do**
2: **for** $j \in [2^{i-1}]$ **do**
3: $L_j^{i-1} = \mathsf{IntervalMerge}(L_{2j-1}^i, L_{2j}^i, \alpha)$
4: **end for**
5: **end for**
6: $L_1^0 = \{a + b : a \in L_1^1, b \in L_2^1, a + b = 0\}$
7: **if** $L_1^0 \neq \emptyset$ **then**
8: determine elements $x_1, ..., x_k$ s.t. $\sum_i x_i = 0$ and $x_i \in L_i$.
9: **return** $x_1, ..., x_k$ if such elements exist.
10: **else**
11: **return** $\perp$
12: **end if**

Figure 5.3: A degraded version of Wagner's k-list algorithm that discards elements to maintain the uniformity and independence of lists at each level.

The performance of our algorithm is degraded (compared to Wagner's original algorithm) by discarding list elements at every level of the tree (except the last). By carefully choosing which elements to discard, the degraded algorithm maintains the uniformity and independence of list elements at every level. We introduce two procedures, IntervalMerge and RejectionSample, that replace the traditional list-merge operation up to the last level of the tree (where the traditional list-merge is sufficient to find the last collision). IntervalMerge preserves the independence of output list

elements by only combining one set of elements per matching "interval" in the input lists. The list elements created by IntervalMerge are sums of two uniform random variables, and as such are no longer uniform. Thus, we employ rejection sampling to further sculpt the output list into uniform and independent elements before inserting them into the output list(s) at each level.

Achieving a comparable success probability to the heuristic version requires the lists at the second to last level retain enough elements to guarantee a collision at the output level with a noticeable probability. As both IntervalMerge and RejectionSample discard list elements, the degraded algorithm will naturally require larger initial lists. Observe that the interval merge can be implemented in time linear in the size of the input lists. As we are only selecting (at most) a single pair of elements per interval, only a linear scan through each list is required. As such, we actually *improve* over Wagner's original algorithm in this regard. The lists need not be sorted nor stored in a hash table. Thus, the runtime of our degraded k-list algorithm is $O(k \cdot m_{in}^{(\ell)})$, where $m_{in}^{(\ell)}$ is the size of the initial input lists. For clarity, we restate our main runtime theorem here:

Theorem 5.1.1. If the Interval Merge algorithm in Figure 5.4 returns a list of size $c \cdot m_{in}$ for some constant $0 < c \leq 1$ given input lists of size m_{in}, then the degraded k-list algorithm in Figure 5.3 requires initial lists of size

$$\Theta\left(\alpha^{\frac{-(\ell-1)}{\ell+1}} \cdot c^{\frac{-(\ell-1)(\ell+2)}{2(\ell+1)}} \cdot p^{\frac{1}{\ell+1}}\right),$$

and finds a solution with non-negligible probability, where α is the interval scaling

factor input to IntervalMerge and $k = 2^\ell$.

Proof. For the degraded k-list algorithm described in Figure 5.3 to obtain a solution, lists at the second-to-last level L_1^1 and L_2^1 must contain enough elements such that a collision between the two lists is found. Thus, we want to analyze the sizes of these lists. The main tool for that is the fact that given input lists of size m_{in}, the IntervalMerge procedure in Figure 5.4 returns a list of size $c \cdot m_{in}$ for some constant $0 < c \leq 1$ with high probability.

Let us denote by $\mathcal{L}_{in}^{(i)}$, the domain of the elements in list $L_j^i \ \forall \ j$. Let us also denote by $m_{in}^{(i)}$, the number of elements in list $L_j^i \ \forall \ j$.

Then, for a collision to occur with constant probability, birthday bounds require

$$(m_{in}^{(1)})^2 = \Theta(|\mathcal{L}_{in}^{(1)}|). \tag{5.1}$$

Observe that $\mathcal{L}_{in}^{(i)}$ is the output range of the Interval Merge procedure in Figure 5.4 as executed on the input lists at level $i+1$. Thus, we have that

$$\mathcal{L}_{in}^{(i)} = \left[- \left\lfloor \left\lfloor \frac{|\mathcal{L}_{in}^{(i+1)}|}{\alpha \cdot m_{in}^{(i+1)}} \right\rfloor /2 \right\rfloor , \left\lfloor \left\lfloor \frac{|\mathcal{L}_{in}^{(i+1)}|}{\alpha \cdot m_{in}^{(i+1)}} \right\rfloor /2 \right\rfloor \right]$$

as the input interval $[-a, a]$ is substituted for $\mathcal{L}_{in}^{(i+1)}$. Then,

$$\begin{aligned} |\mathcal{L}_{in}^{(i)}| &= 2 \cdot \left\lfloor \left\lfloor \frac{|\mathcal{L}_{in}^{(i+1)}|}{\alpha \cdot m_{in}^{(i+1)}} \right\rfloor /2 \right\rfloor \\ &= \Theta \left(\frac{|\mathcal{L}_{in}^{(i+1)}|}{\alpha \cdot m_{in}^{(i+1)}} \right) \end{aligned} \tag{5.2}$$

In addition, as per our initial assumption, $m_{in}^{(i)} = c \cdot m_{in}^{(i+1)}$. Thus $m_{in}^{(i)} = c^{\ell-i} m_{in}^{(\ell)}$. Combining with (5.2), we now obtain a recurrence relation on the size of the input intervals

$$
\begin{aligned}
|\mathcal{L}_{in}^{(i)}| &= \Theta\left(\frac{|\mathcal{L}_{in}^{(i+1)}|}{\alpha c^{\ell-i-1} m_{in}^{(\ell)}}\right) \\
|\mathcal{L}_{in}^{(i)}| &= \Theta\left(\frac{|\mathcal{L}_{in}^{(\ell)}|}{\prod_{j=i}^{\ell-1} \alpha c^{\ell-j-1} m_{in}^{(\ell)}}\right)
\end{aligned} \tag{5.3}
$$

Substituting (5.3) for $|\mathcal{L}_{in}^{(1)}|$ in (5.1), we find that

$$
\begin{aligned}
(m_{in}^{(1)})^2 &= \Theta(|\mathcal{L}_{in}^{(1)}|) \\
(m_{in}^{(1)})^2 &= \Theta\left(\frac{|\mathcal{L}_{in}^{(\ell)}|}{\prod_{j=1}^{\ell-1} \alpha c^{\ell-j-1} m_{in}^{(\ell)}}\right) \\
(c^{\ell-1} m_{in}^{(\ell)})^2 &= \Theta\left(\frac{|\mathcal{L}_{in}^{(\ell)}|}{\prod_{j=1}^{\ell-1} \alpha c^{\ell-j-1} m_{in}^{(\ell)}}\right) \\
(c^{\ell-1} m_{in}^{(\ell)})^2 &= \Theta\left(\frac{p}{\alpha^{\ell-1} c^{(\ell-1)(\ell-2)/2} (m_{in}^{(\ell)})^{\ell-1}}\right)
\end{aligned} \tag{5.4}
$$

as $|\mathcal{L}_{in}^{(\ell)}| = p$. Solving (5.4) for $m_{in}^{(\ell)}$ completes the proof.

$$
m_{in}^{(\ell)} = \Theta\left(\alpha^{\frac{-(\ell-1)}{\ell+1}} \cdot c^{\frac{-(\ell-1)(\ell+2)}{2(\ell+1)}} \cdot p^{\frac{1}{\ell+1}}\right)
$$

□

5.3.1 Analysis of IntervalMerge

The main source of degradation of initial list sizes is due to the IntervalMerge

Algorithm 5.3.2 Interval Merge

Input: Lists L_1, L_2 of m_{in} uniform and independent elements in $[-a, a]$, and interval scaling factor $\alpha > 0$.

Output: List L_{12} of uniform and independent elements in $[-\lfloor\lfloor\frac{2a}{\alpha \cdot m_{in}}\rceil/2\rfloor, \lfloor\lfloor\frac{2a}{\alpha \cdot m_{in}}\rceil/2\rfloor]$.

Let $\mathcal{L} = \left\lfloor \frac{2a}{\alpha \cdot m_{in}} \right\rceil$.
Let $\mathcal{L}_{\frac{1}{2}} = \lfloor \mathcal{L}/2 \rfloor$.
Let $N_{int} = \left\lfloor \frac{2a}{\mathcal{L}} \right\rceil$.
Sample N_{int} random numbers $\in [0, 1]$, $r_1, \ldots, r_{N_{int}}$.
Let f be the linear function s.t. $f(0) = -a$ and $f(N_{int}) = a + 1$.
for $j = 1$ to N_{int} **do**
    Let $I_j = [-a, a] \cap [f(j-1), f(j)] \cap \mathbb{Z}$.
    Let x_1^* be the first element of $L_1 \in I_j$ otherwise $\perp$.
    Let x_2^* be the first element of $L_2 \in -I_j$ otherwise $\perp$.
    if $x_1^* \neq \perp$ AND $x_2^* \neq \perp$ **then**
        if $\mathsf{RejectionSample}(\mathcal{T}_{|I_j|-1}, (x_1^* + x_2^*), \mathcal{L}_{\frac{1}{2}}\ ;\ r_j) = \mathsf{Accept}$ **then**
            Add $(x_1^* + x_2^*)$ to L_{12}.
        end if
    end if
end for
return L_{12}.

Figure 5.4: The interval merge procedure that takes the place of the traditional list merge operation in the degraded k-list algorithm in Figure 5.3.

procedure. The procedure is primarily designed to maintain the independence and uniformity of elements in the merged list. To ensure the degraded k-list algorithm terminates in the same number of iterations as Wagner's original, care must be taken to enforce similar constraints on the range of elements in the merged list. The independence of output elements is ensured by only allowing for sums of unique elements from each input list. This is achieved by first partitioning the input domain into non-overlapping intervals of similar size. If (at most) one element is used (and not reused) in a sum per interval, then the uniqueness of each summand is guaranteed.

To bound the range of the output, we form the sums from complementary

Algorithm 5.3.3 Rejection Sampling

Input: The probability mass function of a discrete triangle distribution $\mathcal{T}_{z-1}$ (Definition 2.2.4), candidate output element x_{12} distributed according to $\mathcal{T}_{z-1}$, a target interval bound $a \leq z-1$, and randomness $r \in [0;1]$.

Output: Accept OR Reject.

1: **if** $x_{12} \notin [-a, a]$ **then**
2: **return** Reject.
3: **end if**
4: Let $p = \frac{\mathcal{T}_{z-1}(a)}{\mathcal{T}_{z-1}(x_{12})}$.
5: **if** $r < p$ **then**
6: **return** Accept
7: **else**
8: **return** Reject
9: **end if**

Figure 5.5: The rejection sampling procedure used during the execution of an interval merge. The procedure is designed to accept with a uniform probability.

intervals symmetric about the origin. Then, to further shrink the range and ensure that elements in the output list are distributed uniformly, we apply rejection sampling to each sum. As such, the size of the merged list is precisely the number of *fully occupied* (see definition 5.2.1) complementary intervals multiplied by the proportion of sums that were accepted. The challenge is to have enough intervals such that we sufficiently shrink the range of output elements (as in the original k-list algorithm), while guaranteeing (with high probability) that enough complementary intervals are fully occupied.

Constructing the Intervals. Defining the interval boundaries is not straightforward. The input list elements are over a subset of the integers. We cannot guarantee that the optimal number of intervals perfectly partitions the input domain. Additionally, the algorithm invariant requires output elements to remain identically distributed. If intervals are allowed to vary in size, we must enforce the identical

distributions of output elements through our rejection sampling procedure.

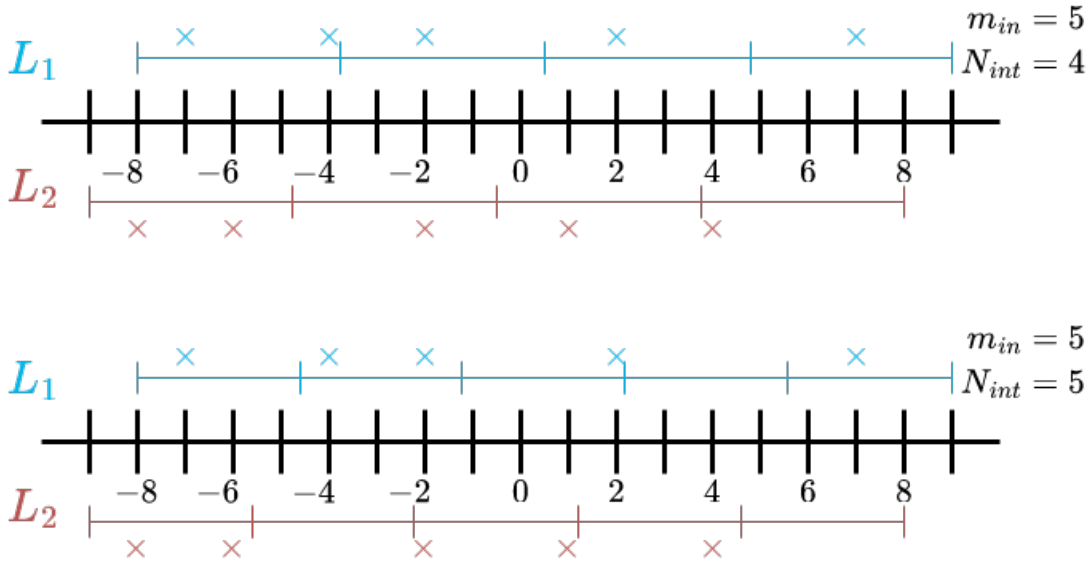


Figure 5.6: Example intervals created during interval merge (Figure 5.4) for $m_{in} = 5$ input elements per list, and $N_{int} = \{4, 5\}$. Input elements are sampled from an input domain of $[-8, 8]$.

More formally, for input elements drawn from $[-a, a]$, IntervalMerge splits the input domain of each list into sub-intervals of size $\mathcal{L}$ and $\mathcal{L} + 1$, where

$$\mathcal{L} = \lfloor (2a)/(\alpha \cdot m_{in}) \rceil. \tag{5.5}$$

Here m_{in} is the number of elements in each input list. Note also that α controls the (approximate) number of intervals we create, since we define the number of intervals as $N_{int} = \lfloor (2a)/\mathcal{L} \rfloor$.

The interval boundaries are set by first defining the linear function $f()$, where $f(0) = -a$ and $f(N_{int}) = a + 1$. Then, each interval $I_j,\ j \in [N_{int}]$ is constructed by taking the set of integers in $[f(j-1), f(j)]$. By inspection, we see that these sets

are intervals of size $\mathcal{L}$ or $\mathcal{L}+1$. See Figure 5.6 for an example.

Output Range and Distribution. The output range and distribution of elements at every level of the degraded k-list algorithm is defined by the rejection sampling procedure in Figure 5.5. We first note the following.

Remark 5.3.1. All elements $x_{12} \in L_{12}$ inserted by the IntervalMerge procedure in Figure 5.4 are independent.

This is fairly straightforward. The independence of output elements is attained by construction, as each output element is formed from unique summands. Each defined subinterval is non-overlapping, and (at most) one element is used per subinterval per list.

The range and (exact) distribution is more involved. Elements in the output list(s) at every level of the degraded k-list algorithm solely consist of elements accepted by the rejection sampling procedure in Figure 5.5. In this procedure, all elements outside the specified symmetric interval are automatically rejected.

For elements inside the specified interval, we flatten the probability distribution and make sure that the final distribution of outcomes is uniform. This is shown by computing the probability that the rejection sampling procedure outputs Accept:

$$\begin{aligned}
&\mathbf{Pr}\left[\mathsf{RejectionSample}(\mathcal{T}_{z-1}, x_{12}, a; r) = \mathsf{Accept}\right] \\
&= \sum_{i=-(z-1)}^{z-1} \mathbf{Pr}\left[\mathsf{Accept} \mid x_{12} = i\right] \cdot \mathbf{Pr}\left[x_{12} = i\right] \\
&= \sum_{i=-a}^{a} \mathbf{Pr}\left[r \leq \frac{\mathcal{T}_{z-1}(a)}{\mathcal{T}_{z-1}(x_{12})}\right] \cdot \mathcal{T}_{z-1}(x_{12}) \qquad (5.6)
\end{aligned}$$

where RejectionSample is the rejection sampling procedure specified in Figure 5.5, and (5.6) is due to the fact that the procedure always returns Reject when $x_{12} \notin [-a, a]$. Finishing the calculation yields

$$= \sum_{i=-a}^{a} \frac{\mathcal{T}_{z-1}(a)}{\mathcal{T}_{z-1}(x_{12})} \cdot \mathcal{T}_{z-1}(x_{12}) \tag{5.7}$$

$$= (2a+1) \cdot \mathcal{T}_{z-1}(a)$$

Observe in (5.7) that for all $i \in [-a, a]$, the probability of accepting given $x_{12} = i$ is $\mathcal{T}_{z-1}(a)$. Therefore, all elements accepted by the rejection sampling procedure are assigned the same probability mass. As such, we can state the following:

Remark 5.3.2. All elements x_{12} accepted by the RejectionSample procedure in Figure 5.5 and inserted into L_{12} by the IntervalMerge procedure in Figure 5.4 are identically distributed. Each element is drawn from the uniform distribution on $[-\mathcal{L}_{\frac{1}{2}}, \mathcal{L}_{\frac{1}{2}}]$ where $\mathcal{L}_{\frac{1}{2}} = \lfloor \mathcal{L}/2 \rfloor$, and $\mathcal{L}$ is as defined in (5.5). We are using the RejectionSample procedure with two different input distributions, $\mathcal{T}_{\mathcal{L}-1}$ or $\mathcal{T}_{\mathcal{L}}$. Yet, in both cases, we have the exact same output distribution.

Here, we are simply replacing the input a in Figure 5.5 with the interval boundary ($\mathcal{L}_{\frac{1}{2}}$) supplied during the Interval Merge procedure in Figure 5.4.

Furthermore, the sums supplied to the rejection sampling procedure are the sums of two independent uniform elements (a random variable that is uniform over an interval is also uniform over any subinterval). Since the elements come from intervals of size either $\mathcal{L}$ or $\mathcal{L}+1$ which are symmetric of each other around 0, the

sums belong to either $[-(\mathcal{L}-1);\mathcal{L}-1]$ or to $[-\mathcal{L};\mathcal{L}]$. The distribution is thus a discrete triangle distribution $\mathcal{T}_{z-1}$ (Definition 2.2.4). Here, the parameter z is set to the size of the complementary intervals that the summands x_1^* and x_2^* inhabit. This can be either $\mathcal{L}$ or $\mathcal{L}+1$ as shown in Figure 5.6.

When the size of the intervals is $\mathcal{L}$, the probability of the rejection sampling procedure returning Accept is

$$\begin{aligned}&(2\mathcal{L}_{\frac{1}{2}}+1)\cdot\mathcal{T}_{\mathcal{L}-1}(\mathcal{L}_{\frac{1}{2}})\\&=(2\lfloor\mathcal{L}/2\rfloor+1)\cdot\frac{\mathcal{L}-\lfloor\mathcal{L}/2\rfloor}{\mathcal{L}^2}.\end{aligned}\tag{5.8}$$

Similarly, when the size of the intervals is $\mathcal{L}+1$, we obtain

$$\begin{aligned}&(2\mathcal{L}_{\frac{1}{2}}+1)\cdot\mathcal{T}_{\mathcal{L}}(\mathcal{L}_{\frac{1}{2}})\\&=(2\lfloor\mathcal{L}/2\rfloor+1)\cdot\frac{(\mathcal{L}+1)-\lfloor\mathcal{L}/2\rfloor}{(\mathcal{L}+1)^2}.\end{aligned}\tag{5.9}$$

To simplify (5.8) and (5.9) we have two cases. One for when $\mathcal{L}$ is even, and another for when $\mathcal{L}$ is odd. We show only the even case here for brevity. When $\mathcal{L}$ is even, (5.8) becomes

$$\begin{aligned}&(2(\mathcal{L}/2)+1)\cdot\frac{(\mathcal{L})-(\mathcal{L}/2)}{(\mathcal{L})^2}\\&=\frac{\mathcal{L}+1}{2\mathcal{L}}\geq\frac{1}{2},\ \forall\ \mathcal{L}\geq 1.\end{aligned}\tag{5.10}$$

Similarly, (5.9) becomes

$$(2(\mathcal{L}/2)+1)\cdot\frac{(\mathcal{L}+1)-(\mathcal{L}/2)}{(\mathcal{L}+1)^2}$$
$$=\frac{\mathcal{L}/2+1}{\mathcal{L}+1}\geq\frac{1}{2},\ \forall\ \mathcal{L}\geq 1. \tag{5.11}$$

The odd case proceeds in the exact same manner, but replace $\lfloor\mathcal{L}/2\rfloor$ with $(\mathcal{L}-1)/2$ instead of $\mathcal{L}/2$. In both cases, the total probability mass for accepted sums is $\geq 1/2$.

5.3.2 Output List Size

In IntervalMerge, for each subinterval, an element is added to L_{12} if and only if that subinterval and its complement are *fully occupied* and the sum of the first elements (in list order) passes the rejection sampling procedure. Therefore, we can view $\mathbf{E}\left[|L_{12}|\right]$ as the expected number of fully occupied pairs of complementary intervals (see Definition 5.2.1) multiplied by the expected loss in probability mass incurred by rejection sampling. To give a lower bound for $|L_{12}|$, we make use of McDiarmid's inequality (Theorem 2.2.5).

Applying the inequality requires defining a function that satisfies the bounded differences property. To this end, we define the function f as a $(2m_{in}+N_{int})$-input function that takes the values of the elements in the lists L_1 and L_2 and the randomness $r_j\ \forall\ j\in[N_{int}]$ supplied to the rejection sampling procedure in Figure 5.5. The return value for f is then set to the number of elements in the list L_{12} output by a single execution of the interval merge procedure in Figure 5.4 for a given interval

scaling factor α.

Note that we must be careful when using the randomness to make sure that adding or removing one doubly occupied interval does not create an unwanted cascade of changes in the other decisions made in subsequent invocations of the interval merge procedure. This can be done by assigning each interval (at every level) its own randomness at the very beginning of the algorithm independently of whether this interval contributes or not. With this precaution in place, any change in the input list only has a localized effect and can change the size of any list that depends on this element by at most 1.

In addition, McDiarmid's inequality is a consequence of constructing a Doob Martingale that tracks the conditional expectation of f as its inputs are sampled. The sums passed to RejectionSample are determined by list order. As such, we ensure that for all $i \in [m_{in}]$, the $(2i-1)$-st input is the i-th element of L_1 and the input $2i$-th input is the i-th element of L_2.

The first step to applying McDiarmid's inequality is to calculate the expected value for f.

Lemma 5.3.3. *Let $\mathcal{X}^* := [-a, a] \subset \mathbb{Z}$, and $\mathcal{U}^* := [0, 1] \subseteq \mathcal{R}$ and let L_1 and L_2 be lists of m_{in} uniform elements from $\mathcal{X}^*$. Further, let $N_{int}^{(\mathcal{L})}$ be the number of sub-intervals of size $\mathcal{L}$, let $N_{int}^{(\mathcal{L}+1)}$ be the number of sub-intervals of size $\mathcal{L}+1$ and let $N_{int} = N_{int}^{(\mathcal{L})} + N_{int}^{(\mathcal{L}+1)}$. Then, the expectation of the function $f : (\mathcal{X}^*)^{2m_{in}} \times (\mathcal{U}^*)^{N_{int}} \mapsto \mathbb{N}$, where the input $x_{2i-1} \in \mathcal{X}^*$ is the i-th element of L_1 and the input $x_{2i} \in \mathcal{X}^*$ is the i-th element of L_2 for all $i \in [m_{in}]$, and the return value is the number of elements*

in the list L_{12} output by the interval merge procedure in Figure 5.4–given the interval scaling factor α–is

$$\mathbf{E}\left[f(x_1,\ldots,x_{2m},r_1,\ldots,r_{N_{int}})\right] \geq \frac{1}{2}\cdot\left(N_{int}^{(\mathcal{L})}\cdot\left(1-\left(1-\frac{\mathcal{L}}{2a+1}\right)^{m_{in}}\right)^2 + N_{int}^{(\mathcal{L}+1)}\cdot\left(1-\left(1-\frac{\mathcal{L}+1}{2a+1}\right)^{m_{in}}\right)^2\right).$$

Proof. The the expected number of output list elements ($\mathbf{E}[f]$) for a single execution of the Interval merge procedure can be found by counting the (expected) number of times the rejection sampling procedure in Figure 5.5 returns Accept. However, the expected number of Accepts cannot be computed directly without first knowing the number of sums $x_1^* + x_2^*$ that were submitted to the rejection sampling procedure. As such, we make use of the law of iterated expectation

$$\begin{aligned}\mathbf{E}\left[f(x_1,\ldots,x_{2m},r_1,\ldots,r_{N_{int}})\right] &= \mathbf{E}\left[\#\ \text{RejectionSample Accepts}\right]\\ &= \mathbf{E}\left[\mathbf{E}\left[\#\ \text{RejectionSample Accepts}\mid N_{sum}\right]\right].\end{aligned} \tag{5.12}$$

where N_{sum} is the number of sums $x_1^* + x_2^*$ submitted to the rejection sampling procedure.

The expected number of Accepts given the number of sums created during the interval merge procedure is covered by our analysis of the rejection sampling procedure in the previous section. By construction, our rejection sampling procedure outputs a *uniform* distribution over a target interval. Each outcome that results in an Accept is assigned a constant probability mass (See (5.7) and Remark 5.3.2).

We state the following claim:

Claim 5.3.4. If the number of sums submitted to the RejectionSample procedure in Figure 5.5 by the IntervalMerge procedure in Figure 5.4 is N_{sum}, where each sum is distributed according to the triangle distributions $\mathcal{T}_{\mathcal{L}-1}$ or $\mathcal{T}_{\mathcal{L}}$, and the target output interval is $\mathcal{L}_{\frac{1}{2}}$, then

$$\mathbf{E}\left[\#\ \textsf{RejectionSample Accepts} \mid N_{sum}\right] \geq (1/2) \cdot N_{sum},$$

where $\mathcal{L}$ is defined in (5.5), and $\mathcal{L}_{\frac{1}{2}} = \lfloor \mathcal{L}/2 \rfloor$.

Claim. The claim follows from (5.10) and (5.11), which show that the rejection sampling procedure outputs Accept with probability $\geq 1/2$, and the linearity of expectation. □

Therefore, we can rewrite (5.12) as

$$\mathbf{E}\left[f(x_1, \ldots, x_{2m}, r_1, \ldots, r_{N_{int}})\right] \geq \mathbf{E}\left[(1/2) \cdot N_{sum}\right]. \tag{5.13}$$

So we now have simplified the expected value calculation of $f()$ to be at least half the (expected) number of sums $x_1^* + x_2^*$ created.

A sum is only created when the pair of intervals at a particular iteration is fully occupied (Definition 5.2.1). Determining the number of fully occupied intervals after sampling all input list elements is an instance of the balls-into-bins problem.

Let $\mathcal{A}_j$ be the event that intervals $I_j, -I_j$ are fully occupied after all $2m_{in}$ input list elements are sampled. Then, we can define the indicators $\mathcal{I}_j$, where $\mathcal{I}_j = 1$ when

event $\mathcal{A}_j$ occurs. Then, the expected number of created sums can be expressed by applying the linearity of expectation to the sum of all $\mathcal{I}_j$,

$$\begin{aligned}\mathbf{E}\left[N_{sum}\right] &= \mathbf{E}\left[\sum_{j=0}^{N_{int}} \mathcal{I}_j\right] = \sum_{j=0}^{N_{int}} \mathbf{E}\left[\mathcal{I}_j\right] = \sum_{j=0}^{N_{int}} \mathbf{Pr}\left[\mathcal{A}_j\right] \\ &= N_{int}^{(\mathcal{L})} \cdot \mathbf{Pr}\left[\mathcal{A}^{(\mathcal{L})}\right] + N_{int}^{(\mathcal{L}+1)} \cdot \mathbf{Pr}\left[\mathcal{A}^{(\mathcal{L}+1)}\right] \end{aligned} \tag{5.14}$$

were $\mathcal{A}^{(\mathcal{L})}$ (resp. $\mathcal{A}^{(\mathcal{L}+1)}$) is shorthand for the event A_j for any given interval I_j of size $\mathcal{L}$ (resp. $\mathcal{L}+1$). Here (5.14) results from each interval (and interval pair) of the same size having equal probability of occupation.

The probability of $\mathcal{A}_j$ occurring is the probability that after sampling $2m_{in}$ elements (m_{in} for each list), interval I_j is occupied by (at least) one element from L_1 and $-I_j$ is occupied by (at least) one element from L_2. As these two sub-events are independent, we can calculate $\mathbf{Pr}\left[\mathcal{A}_j\right]$ as

$$\begin{aligned}\mathbf{Pr}\left[\mathcal{A}_j\right] &= \mathbf{Pr}\left[I_j \text{ is occupied}\right] \cdot \mathbf{Pr}\left[-I_j \text{ is occupied}\right] \\ &= \left(1 - \mathbf{Pr}\left[I_j \text{ is unoccupied after sampling } m_{in} \text{ elements}\right]\right)^2 \\ &= \left(1 - \left(1 - \frac{|I_j|}{2a+1}\right)^{m_{in}}\right)^2 \end{aligned} \tag{5.15}$$

where $1 - |I_j|/(2a+1)$ is the probability of a sampled element landing outside a

particular interval. Combining (5.13) and (5.14) with (5.15) completes the proof.

$$\begin{aligned}
\mathbf{E}\left[f(\mathfrak{r}_1, \ldots, r_{N_{int}})\right] &\geq \mathbf{E}\left[(1/2) \cdot N_{sum}\right] \\
&= \frac{1}{2} \cdot \left(N_{int}^{(\mathcal{L})} \cdot \mathbf{Pr}\left[\mathcal{A}^{(\mathcal{L})}\right] + N_{int}^{(\mathcal{L}+1)} \cdot \mathbf{Pr}\left[\mathcal{A}^{(\mathcal{L}+1)}\right]\right) \\
&= \frac{1}{2} \cdot \left(N_{int}^{(\mathcal{L})} \cdot \left(1 - \left(1 - \frac{\mathcal{L}}{2a+1}\right)^{m_{in}}\right)^2 \right. \\
&\qquad \left. + N_{int}^{(\mathcal{L}+1)} \cdot \left(1 - \left(1 - \frac{\mathcal{L}+1}{2a+1}\right)^{m_{in}}\right)^2\right)
\end{aligned}$$

□

Now, we can state the lower bound on the output list size of IntervalMerge using McDiarmid's inequality.

Applying McDiarmid's Inequality.

Lemma 5.3.5. *Let $L_1, L_2, N_{int}^{(\mathcal{L})}, N_{int}^{(\mathcal{L}+1)}, N_{int}$ be as in Lemma 5.3.3. For any $\epsilon > 0$, the size of the output list L_{12} of the* IntervalMerge *procedure in figure 5.4 is at least*

$$\begin{aligned}
&\frac{1}{2} \cdot \left(N_{int}^{(\mathcal{L})} \cdot \left(1 - \left(1 - \frac{\mathcal{L}}{2a+1}\right)^{m_{in}}\right)^2 \right. \\
&\left. + N_{int}^{(\mathcal{L}+1)} \cdot \left(1 - \left(1 - \frac{\mathcal{L}+1}{2a+1}\right)^{m_{in}}\right)^2\right) - \epsilon
\end{aligned}$$

with probability at least $1 - \exp\left(-2\epsilon^2/(8m_{in} + N_{int})\right)$.

Proof. The proof follows directly from applying McDiarmid's inequality (Theorem 2.2.5) to the expected size of the output list from IntervalMerge (Lemma 5.3.3). To do so, we first make the following claim:

Claim 5.3.6. Let $\mathcal{X}^* := [-a, a] \subset \mathbb{Z}$, and $\mathcal{U}^* := [0, 1] \subseteq \mathcal{R}$. Then, the function $f : (\mathcal{X}^*)^{2m_{in}} \times (\mathcal{U}^*)^{N_{int}} \mapsto \mathbb{N}$, where the input $x_{2i-1} \in \mathcal{X}^*$ is the i-th element of L_1 and the input $x_{2i} \in \mathcal{X}^*$ is the i-th element of L_2 for all $i \in [m_{in}]$, and the return value is the number of elements in the list L_{12} output by the interval merge procedure in Figure 5.4–given the interval scaling factor α–satisfies the bounded differences property in Theorem 2.2.5. Furthermore, this property is satisfied with $c_i = 2$ for all $i \in [2m_{in}]$, and $c_i = 1$ for all $i \in [2m_{in} + 1, 2m_{in} + N_{int}]$.

Claim. This can be seen as follows. Altering the value of a single list element changes which particular interval it occupies. Depending on the index of the element, it may change the value of the sum submitted to the rejection sampling procedure. With this in mind, it is simple to see how changing the value of a single list element can at most change the output of f by ± 2. If the element is contained in a half-occupied (Definition 5.2.2) pair of intervals, then moving the element can make at most one new pair of fully occupied intervals. This would raise the value of f by 1, should the newly created sum pass the rejection sampling procedure.

On the other hand, if the element is contained in a fully occupied pair of intervals, then moving the element can in fact remove two elements from the output list. Moving the element could leave its old interval pair half-occupied, and if the interval the element is moved to is already fully occupied, a different sum for that interval could be submitted to the rejection sampling procedure. This occurs when the element in question appears earlier in its list (thus selected for summation). If the new sum does not pass the rejection sampling (as the randomness remains

unchanged), then this would lower the value of f by 2.

For the remaining N_{int} inputs to the function f, the value of f can change by at most 1 as these inputs govern the randomness for the rejection sampling procedure. As such, modifying these inputs merely alters the decision of the rejection sampling, which adds or subtracts an element from the output list. □

Additionally, note the assumption that the input list elements are sampled uniformly and independently. If we assign each list element to one of the function inputs, then each input is an independent random variable. Therefore, we satisfy all preconditions to applying the inequality. Here we apply the inequality in its single-sided variant, as a lower bound on the size of merged lists suffices for our analysis. Thus,

$$\begin{aligned}
&\mathbf{Pr}[f(r_1, \ldots, r_{N_{int}}) - \mathbf{E}\left[f(r_1, \ldots, r_{N_{int}})\right] \leq -\epsilon] \\
&\qquad \leq \exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^{2m_{in}} 2^2 + \sum_{i=1}^{N_{int}} 1^2}\right) \\
&\qquad \leq \exp\left(-\frac{2\epsilon^2}{8m_{in} + N_{int}}\right)
\end{aligned}$$

Now we can apply Lemma 5.3.3 to the expected value of f, and substitute the

length of the output list L_{12} for the exact value of f.

$$\begin{aligned}\mathbf{Pr}\Bigg[|L_{12}| - \frac{1}{2}\cdot\Bigg(N_{int}^{(\mathcal{L})}\cdot\left(1-\left(1-\frac{\mathcal{L}}{2a+1}\right)^{m_{in}}\right)^2 \\ + N_{int}^{(\mathcal{L}+1)}\cdot\left(1-\left(1-\frac{\mathcal{L}+1}{2a+1}\right)^{m_{in}}\right)^2\Bigg) \leq -\epsilon\Bigg] \\ \leq \exp\left(-\frac{2\epsilon^2}{8m_{in}+N_{int}}\right)\end{aligned}$$

A bit of rearranging completes the proof.

$$\begin{aligned}\mathbf{Pr}\Bigg[|L_{12}| > \frac{1}{2}\cdot\Bigg(N_{int}^{(\mathcal{L})}\cdot\left(1-\left(1-\frac{\mathcal{L}}{2a+1}\right)^{m_{in}}\right)^2 \\ + N_{int}^{(\mathcal{L}+1)}\cdot\left(1-\left(1-\frac{\mathcal{L}+1}{2a+1}\right)^{m_{in}}\right)^2\Bigg) - \epsilon\Bigg] \\ > 1-\exp\left(-\frac{2\epsilon^2}{8m_{in}+N_{int}}\right)\end{aligned} \tag{5.16}$$

□

5.3.3 Determining Constants

In order to make a proper comparison to the analysis by Shallue [130], we need to determine the constants involved. While the analysis in the previous section provides a complete characterization of the number of output elements at every level, the effect of the interval merge algorithm on the number of total initial list elements (and therefore the runtime) is unclear. For starters, the number of elements in each input list m_{in} changes on a per level basis. As such, the size of the subintervals $\mathcal{L}$ (Defined in (5.5)) necessarily changes as well.

We start by simplifying the result of Lemma 5.3.3 at the expense of some tightness.

Corollary 5.3.7. *Let notations be as in Lemma 5.3.3. Then*

$$\mathbf{E}\left[f(x_1,\ldots,x_{2m},r_1,\ldots,r_{N_{int}})\right] \geq \frac{1}{2}\left(N_{int}\cdot\left(1-\left(1-\frac{\mathcal{L}}{2a+1}\right)^{m_{in}}\right)^2\right).$$

This simplification is derived using the fact that

$$1-\left(1-\frac{\mathcal{L}}{2a+1}\right) < 1-\left(1-\frac{\mathcal{L}+1}{2a+1}\right).$$

Using the definition of $\mathcal{L}$ in Equation (5.5), observe that

$$\begin{aligned}1-\left(1-\frac{\mathcal{L}}{2a+1}\right) &= 1-\left(1-\frac{\lfloor(2a)/(\alpha\cdot m_{in})\rceil}{2a+1}\right)\\ &\approx 1-\left(1-\frac{1}{\alpha\cdot m_{in}}\right)\end{aligned}$$

for large enough input element domain $[-a,a]$. Similarly, the number of created intervals N_{int} can be approximated as

$$\begin{aligned}N_{int} &= \left\lfloor\frac{2a}{\mathcal{L}}\right\rfloor\\ &= \left\lfloor\frac{2a}{\lfloor(2a)/(\alpha\cdot m_{in})\rceil}\right\rfloor\\ &\approx \alpha\cdot m_{in}\end{aligned} \tag{5.17}$$

Note that as a continues to grow, the approximation becomes arbitrarily

accurate. Thus, we can restate the above corollary as follows.

Corollary 5.3.8. *Let notations be as in Lemma 5.3.3. Then*

$$\mathbf{E}\left[f(\mathfrak{K}_1 \ldots, r_{N_{int}})\right] \gtrapprox \frac{1}{2}\left(\alpha \cdot m_{in}\left(1-\left(1-\frac{1}{\alpha \cdot m_{in}}\right)^{m_{in}}\right)^2\right).$$

Asymptotically, the relative error of this approximation can be made arbitrarily close to 0.

We have now stated the expected value of f as a function of the quantity $\alpha \cdot m_{in}$. Ideally, we want to state the expected value of f as a fraction of m_{in}. This bounds the amount of elements lost to the interval merge procedure at every level.

Observe that

$$\lim_{m_{in}\to\infty} \frac{1}{2}\left(\alpha\left(1-\left(1-\frac{1}{\alpha \cdot m_{in}}\right)^{m_{in}}\right)^2\right) = \frac{1}{2}\left(\alpha\left(1-e^{-1/\alpha}\right)^2\right) \tag{5.18}$$

Thus, as m_{in} grows large, we can optimize the fraction of remaining elements. Using numerical optimization, we found that (5.18) is maximized when $\alpha \approx 0.79591$, for a maximum value of ≈ 0.20363. With this in mind, we can then restate the corollary for a final time.

Corollary 5.3.9. *Let notations be as in Lemma 5.3.3 and fix* $\alpha = 0.79591$. *Then*

$$\mathbf{E}\left[f(\mathfrak{K}_1 \ldots, r_{N_{int}})\right] \gtrapprox 0.20363 \cdot m_{in}$$

Asymptotically, the relative error of this approximation can be made arbitrarily close

to 0.

Note also that 'm_{in} large enough' is rather small, as convergence is quite quick (when $m_{in} = 10$, the absolute error is 0.014 for $\alpha = 0.8$). If we apply the above corollary to Lemma 5.3.5, we obtain a concrete lower bound on the number of output elements of each execution of the interval merge procedure in Figure 5.4.

Lemma 5.3.10. *Given two input lists L_1, L_2 of size m_{in} large enough, with elements sampled uniformly and independently from $[-a, a] \subset \mathbb{Z}$ large enough, for any $\epsilon > 0$, the size of the output list L_{12} of the* IntervalMerge *procedure in figure 5.4 is at least*

$$0.20363 \cdot m_{in} - \epsilon$$

for interval scaling parameter $\alpha = 0.79591$ with probability at least $1 - \exp\left(-2\epsilon^2/(8.79591 \cdot m_{in})\right)$.

By choosing ϵ to be a small fraction of m_{in}, we can bound the probability that the interval merge procedure returns a list of size $c \cdot m_{in}$ for some constant c. For concrete runtime parameters, we set c to $(1/6) = 0.1\bar{6}$. Thus,

Corollary 5.3.11. *Let notations be as in Lemma 5.3.10. The size of the output list L_{12} of the* IntervalMerge *procedure in figure 5.4 is at least*

$$(1/6) \cdot m_{in}$$

for interval scaling parameter $\alpha = 0.79591$ with probability at least $1 - \exp\left(-0.00031 \cdot m_{in}\right)$.

Now, all that remains is to combine Corollary 5.3.11 with Theorem 5.1.1 to obtain concrete parameters. Theorem 5.1.1 assumes that each input list is of the same size at every level. If we obtain larger lists than desired from the IntervalMerge procedure, then we can simply discard the excess—larger internal list sizes will result in a higher probability of finding a collision at the last level. With this in mind, we can find a lower bound on the probability that our degraded k-list algorithm meets the claimed runtime.

Corollary 5.3.12. *Let notations and input list sizes be as in Theorem 5.1.1 and let* $c = 1/6, \alpha = 0.79591$. *Then the success probability of the degraded k-List algorithm in Figure 5.3 is lower bounded as*

$$\prod_{i=1}^{\ell-1} \left(1 - e^{-0.00031 \cdot (1/6)^{\ell-i-1} \cdot m_{in}^{(\ell)}}\right)^{2^i}.$$

It can be verified that for all parameter settings used for Figure 5.2, this probability is 1 within computer precision. This is because for all relevant parameters, the term $(1/6)^{\ell} \cdot m_{in}^{(\ell)}$ will be much larger than 2^{ℓ}.

Proof. To begin, we need to calculate the joint probability that *all* lists have the requisite number of elements. Let B_j^i be the event that list L_j^i in the degraded k-list algorithm in Figure 5.3 has at least $c^{\ell-i} m_{in}^{(\ell)}$ elements where $m_{in}^{(\ell)}$ is the number of

elements in each input list. Now, we can write the joint probability as

$$\begin{aligned}\mathbf{Pr}\left[\bigcap_{i=1}^{\ell-1}\bigcap_{j=1}^{2^i} B_j^i\right] &= \mathbf{Pr}\left[\bigcap_{h=1}^{2} B_h^1 \,\middle|\, \bigcap_{i=2}^{\ell-1}\bigcap_{j=1}^{2^i} B_j^i\right] \cdot \mathbf{Pr}\left[\bigcap_{i=2}^{\ell-1}\bigcap_{j=1}^{2^i} B_j^i\right] \\ &= \left(\mathbf{Pr}\left[B_1^1 \,\middle|\, \bigcap_{j=1}^{2} B_j^2\right]\right)^2 \cdot \mathbf{Pr}\left[\bigcap_{i=2}^{\ell-1}\bigcap_{j=1}^{2^i} B_j^i\right] \qquad (5.19)\end{aligned}$$

where (5.19) is due to the fact that the sizes of lists at level i are solely dependent on the sizes of the lists at level $(i+1)$. In addition, the sizes of each list at level i are independent and identically distributed when conditioned on the event that lists at level $(i+1)$ are all the same size. Note that i goes to $\ell-1$, as the lists at level ℓ are the initial input lists. Their sizes are set to $m_{in}^{(\ell)}$ by definition.

We can continue decomposing the joint probability in (5.19), making use of the exhibited Markov property. Completing the decomposition yields

$$\prod_{i=1}^{\ell-1}\left(\mathbf{Pr}\left[B_1^i \,\middle|\, \bigcap_{j=1}^{2} B_j^{i+1}\right]\right)^{2^i} \qquad (5.20)$$

To obtain concrete parameters, we want to apply Corollary 5.3.11 to (5.20). Note that Corollary 5.3.11 is calculating the probability that a list at level i has at least $(1/6)\cdot m_{in}^{(i+1)}$ elements given that the input lists at level $(i+1)$ have *exactly* $m_{in}^{(i+1)}$ elements. In this regard, Corollary 5.3.11 provides a lower bound on the probability. Therefore, we can state the following for $c = (1/6)$

$$\mathbf{Pr}\left[\bigcap_{i=1}^{\ell-1}\bigcap_{j=1}^{2^i} B_j^i\right] \geq \prod_{i=1}^{\ell-1}\left(1-e^{-0.00031\cdot m_{in}^{(i+1)}}\right)^{2^i}$$
$$= \prod_{i=1}^{\ell-1}\left(1-e^{-0.00031\cdot(1/6)^{\ell-i-1}\cdot m_{in}^{(\ell)}}\right)^{2^i} \tag{5.21}$$

where the list size at level $(i+1)$, $m_{in}^{(i+1)}$ is substituted for m_{in} in the application of Corollary 5.3.11. Thus, for any fixed number of lists $k = 2^\ell$, the joint probability that the size of the lists at each level of the degraded k-list algorithm depicted in Figure 5.3 shrink by at most $(1/6)$ is overwhelming in the number of inputs to the initial lists at level ℓ. □

Appendix A: Appendix for When NISTPQC FIPS flips

A.1 Additional Attacks against other NIST Lattice KEMs

In this appendix we give additional details concerning how a hypothetical Rowhammer attack against each of Kyber, Saber, NTRU-prime and NTRU might work.

A.1.1 Attacking Kyber

Similar to Frodo, Kyber is an IND-CCA key-capsulation mechanism constructed as a Fujisaki-Okamoto transform of an IND-CPA encryption scheme based on Module-Learning with Error. We defer to Kyber submission available at [132] for details. The design of Kyber follows the same paradigm as Frodo except that Kyber works with polynomials in the ring $R \coloneqq \mathbb{Z}[x]/(x^{256}+1)$ and prime modulus $q = 3329$. The decoding error in Kyber is

$$\boldsymbol{e}^T\boldsymbol{r} + e_2 + c_v - \boldsymbol{s}^T(\boldsymbol{e}_1 + \boldsymbol{c}_u),$$

where $\boldsymbol{e}, \boldsymbol{r}, \boldsymbol{s}, \boldsymbol{e}_1$, and $\boldsymbol{c}_u$ are vectors of polynomials in the ring R, and e_2 and c_v are single polynomials. The correctness condition of Kyber places a bound on the

magnitude of each coefficient of the polynomial expression above. Here, $\boldsymbol{s}$ and $\boldsymbol{e}$ are the secret key and error, analogous to $\boldsymbol{S}$ and $\boldsymbol{E}$ in FrodoPKE. The vectors of polynomials $\boldsymbol{r}$ and $\boldsymbol{e}_1$ and the polynomial e_2 are sampled during encryption. The terms $\boldsymbol{c}_u$ and c_v correspond to additional decoding errors introduced by Kyber's compression and decompression of the ciphertext. Compression encodes an element of $\mathbb{Z}_q$ using fewer than $\log_2(q)$ bits and decompression extracts an element of $\mathbb{Z}_q$ from fewer than $\log_2(q)$ bits.

We note that each polynomial $p \in R$ defines a module endomorphism of R as multiplication by p, which has a matrix representation given a fixed basis of R (e.g. the standard basis $\{1, x, x^2, \dots\}$). Therefore, the decoding error above could be read as a matrix expression similar to Frodo decoding error. So our framework applies. In a similar fashion to our attack on Frodo, row-hammering could potentially induce a change in the distribution of $\boldsymbol{e}$ or $\boldsymbol{s}$. Combining with a malicious distribution of $\boldsymbol{r}$ or $\boldsymbol{e}_1$, the adversary could then obtain an abnormally large decryption rate that could lead to a successful (partial) key recovery.

A.1.2 Attacking Saber

Saber is based on Module Learning with Rounding (M-LWR). The main differences between Saber and FrodoKEM are:

- The matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\ell \times n\ell}$ is structured so that it can be represented as an $\ell \times \ell$ matrix of polynomials of degree $n-1$ in $\mathbb{Z}[x]_q/(x^n+1)$ for M-LWR.

- Rather than adding error vector $\boldsymbol{e}$, sampled explicitly from some distribution,

to $\boldsymbol{As}$ to produce $\boldsymbol{b}$, $\boldsymbol{b}$ is generated by rounding $\boldsymbol{As}$.

As a result, a Rowhammer adversary can only hammer $\boldsymbol{s}$ or $\boldsymbol{b}$. In the reference implementation of Saber, there are a number of calls to SHAKE-128 that may be used for performance degradation similar to what we did with FrodoKEM. While these calls are prior to the generation of $\boldsymbol{s}$ or $\boldsymbol{b}$, they can effectively be used to Rowhammer $\boldsymbol{b}$, since when $\boldsymbol{b}$ is generated, partial sums are added to a 0 vector which is initialized before the SHAKE-128 calls. We therefore expect the 0 vector can be Rowhammered resulting in a constant value being added to certain coefficients of $\boldsymbol{b}$. We did not identify a similarly promising strategy for rowhammering $\boldsymbol{s}$.

A.1.3 Attacking NTRU-LPRime

NTRU-LPRime is similar in construction to FrodoKEM, Kyber and Saber, but based on Ring Learning With Rounding (R-LWR). It uses the polynomial ring $\mathbb{Z}[x]/(x^p - x - 1)$ where p is an inert prime. In NTRU-LPRime's notation, the polynomial $\mathbf{G}$ is analogous to $\mathbf{A}$ in FrodoKEM, the polynomial $\mathbf{a}$ is analogous to FrodoKEM's $\mathbf{S}$. Like Saber, NTRU-LPRime uses rounding instead of an explicit error analogous to FrodoKEM's $\mathbf{E}$. In NTRU-LPRime's notation $\mathbf{A} = \text{Round}_{\mathbf{aG}}$ is analogous to FrodoKEM's B.

The reference implementation of NTRU-LPRime uses AES instead of SHAKE. It seems harder to do performance degradation on AES, since AES has hardware support. If NTRU-LPRime is implemented with other primitives that are more susceptible to performance degradation or if there is a sufficient performance degra-

dation against AES (in hardware), it appears possible to mount a similar decryption failure attack in a very straightforward way. In principle, a or A (in NTRU-LPRime's notation) are possible targets for Rowhammering.

However, in NTRU-LPRime's reference implementation, a is stored in the private key in a compressed way, which means that any large coefficients in a would create a mismatch between the public and private key and cause decryption to fail with overwhelming probability. Also, in the sampling procedure for a, the calls to AES precede the generation of the coefficients which would be the target for Rowhammering. (A similar situation occurs with the polynomial f in Streamlined NTRU Prime.) Attacking A looks a little more promising. Rowhammering must occur before the hash of a slightly compressed encoding of A as part of the private key. An AES-based random number generator is called in this window, but it only generates 256 bits of output which might not be a large enough target for performance degradation.

A.1.4 Attacking NTRU

The design of NTRU is significantly different from that of Frodo, Kyber, and Saber. However, for the purpose of our presentation, it suffices to only consider the correctness condition of NTRU. The correctness condition of NTRU requires that each coefficient of the following polynomial is in the range $[-q/2, q/2)$:

$$3 \cdot \mathbf{r} \cdot \mathbf{g} + \mathbf{f} \cdot \texttt{Lift}(\mathbf{m})) \bmod (x^n - 1), \tag{A.1}$$

where each term is a polynomial having degree at most $n-2$. Here, $(\mathbf{f}, \mathbf{g})$ is the secret sampled during key generation and $(\mathbf{r}, \mathbf{m})$ is sampled during encapsulation. Decapsulation, if successful, recovers $(\mathbf{r}, \mathbf{m})$ and outputs the shared session key $H(\mathbf{r}, \mathbf{m})$. `Lift` is an injection that maps each $\mathbf{m}$ to a polynomial in $\mathbb{Z}[x]$ such that

$$\texttt{Lift}(\mathbf{m}) \bmod (3, (x^n - 1)/(x - 1)) = \mathbf{m}.$$

Again, we may consider (A.1) as a matrix expression and apply our framework. We could potentially row-hammer $(\mathbf{f}, \mathbf{g})$ and maliciously select $(\mathbf{r}, \mathbf{m})$ to try to induce larger decryption failure rate. However, difficulty arises. Algebraic operations in NTRU switch between different polynomial moduli, $x^n - 1, x - 1$, and $(x^n - 1)/(x - 1)$. Row-hammering must ensure the algebraic constraint that $\mathbf{g} \equiv 0 \pmod{(q, x-1)}$ and $\mathbf{f}$ is invertible $\bmod (q, (x^n - 1)/(x - 1))$. Otherwise, the result of key generation will cause high failure probability over all ciphertexts and is unusable. On the other hand, because $\mathbf{r}$ and $\mathbf{m}$ can be selected arbitrarily (subject to some algebraic constraints), we may not need to compute many hashes as in Frodo to find a potentially good ciphertext to query for decryption.

A.1.5 Attacking Streamlined NTRU Prime

Streamlined NTRU Prime has much similarity in its design to NTRU but uses a different polynomial ring $\mathbb{Z}[x]/(x^p - x - 1)$ where p is a prime. Key generation of streamlined NTRU Prime samples two secret polynomials f and g with ternary coefficients, i.e. coefficients in $\{-1, 0, 1\}$. Encapsulation samples a random polynomial r

with ternary coefficients such that exactly w coefficients are non-zero, where w is a parameter of the scheme. For correctness, streamlined NTRU Prime requires that each coefficient of the following polynomial is in the range $(-q/2, q/2)$:

$$g \cdot r + 3 \cdot f \cdot e \bmod (x^p - x - 1),$$

where e is a polynomial with ternary coefficients corresponding to error introduced by a certain rounding operation. Decapsulation, if successful, recovers r and outputs the hash of r (concatenated with other public inputs) as shared session key. If row-hammering induces the polynomial g or f to have malformed coefficients, the adversary could then potentially mount a decryption failure attack with knowledge of many tuples (r, e) that cause the expression above to exceed the threshold.

The reference implementation of streamlined NTRU Prime uses AES instead of SHAKE, and it seems harder to do performance degradation on AES. If Streamlined NTRU Prime is implemented with other primitives that are more susceptible to performance degradation (or if there is a sufficient performance degradation against AES), it appears theoretically possible to mount a similar decryption failure attack to our attack on Frodo. Moreover, similar to the case of NTRU, since the polynomial r can selected arbitrarily without any call to a random oracle, generating a candidate failing ciphertext may require much less computation. The Rowhammering phase of such an attack would need to target g rather than f, because f is stored in the secret key in a compressed form that assumes all the coefficients are small enough to be stored in 2 bits. Also, in the sampling procedure for f, unlike the sampling

procedure for g, the calls to AES precede the generation of the coefficients which would be the target for Rowhammering.

Appendix B: Appendix for Revisiting Security Estimation for LWE with Hints from a Geometric Perspective

B.1 Proof of Theorem 4.2.1

We restate Theorem 4.2.1, followed by the proof.

Theorem B.1.1 (Theorem 4.2.1)**.** *Let $\boldsymbol{c} \in \mathcal{R}^d$ denote the d-dimensional vector that has $c \in \mathcal{R}$ in each position. Let $\sigma_1^2, \sigma_2^2 \in \mathcal{R}$ be such that $\sigma_2^2 < \sigma_1^2$. Consider the rank-scaled ellipsoids $E^{(\mathsf{Rank})}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) = E^{(\mathsf{Rank})}(\mathbf{0}, \sigma_1^2 \boldsymbol{I}_d)$ and $E^{(\mathsf{Rank})}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) = E^{(\mathsf{Rank})}(\boldsymbol{c}, \sigma_2^2 \boldsymbol{I}_d)$. Then the volume of $E^{(\mathsf{Rank})}(\boldsymbol{\mu}', \boldsymbol{\Sigma}')$ is lower than both the volume of $E^{(\mathsf{Rank})}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $E^{(\mathsf{Rank})}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ if and only if $c^2 > \sigma_1^2 - \sigma_2^2$.*

Proof of Theorem 4.2.1. Recall that $E(\boldsymbol{\mu}', \mathcal{F}(\boldsymbol{\Sigma}')) = E^{(\mathsf{Rank})}(\boldsymbol{\mu}', \boldsymbol{\Sigma}')$ is defined as follows:

$$\mathcal{F}(\boldsymbol{\Sigma}') = k\boldsymbol{X}^{-1},$$

$$\boldsymbol{X} = \lambda \mathcal{F}(\boldsymbol{\Sigma}_1)^{-1} + (1-\lambda)\mathcal{F}(\boldsymbol{\Sigma}_2)^{-1}$$

$$\boldsymbol{\mu}' = \frac{(1-\lambda)}{d\sigma_2^2} \boldsymbol{c}\boldsymbol{X}^{-1}$$

$$k = 1 - \lambda(1-\lambda)\frac{c^2}{\lambda\sigma_2^2 + (1-\lambda)\sigma_1^2},$$

for some $\lambda \in [0, 1]$. The determinant of $\mathbf{\Sigma}'$ is

$$k^d \cdot (\frac{\sigma_1^2 \sigma_2^2}{\lambda \sigma_2^2 + (1-\lambda)\sigma_1^2})^d \tag{B.1}$$

Thus, the volume of $E^{(\mathsf{Rank})}(\boldsymbol{\mu}', \mathbf{\Sigma}')$ decreases if and only if there is a setting of $\lambda \in [0, 1]$ for which (B.1) is less than $(\sigma_2^2)^d$, which is the determinant of $\mathbf{\Sigma}_2$.

This constraint is satisfied if and only if

$$k \cdot \frac{\sigma_1^2 \sigma_2^2}{\lambda \sigma_2^2 + (1-\lambda)\sigma_1^2} < \sigma_2^2.$$

Substituting k from above we get the requirement that:

$$\left(1 - \lambda(1-\lambda)\frac{c^2}{\lambda \sigma_2^2 + (1-\lambda)\sigma_1^2}\right) \cdot \frac{\sigma_1^2 \sigma_2^2}{\lambda \sigma_2^2 + (1-\lambda)\sigma_1^2} < \sigma_2^2.$$

Which is true if and only if there exists a $\lambda \in [0, 1]$ such that:

$$f(\lambda) = (\lambda \cdot (\sigma_1^2 \sigma_2^2 - \sigma_1^4 - \sigma_1^2 c^2) + \sigma_1^2 \lambda^2 c^2 + \sigma_1^4) < (\lambda^2 \cdot (\sigma_2^2 - \sigma_1^2)^2 + 2\lambda \cdot (\sigma_1^2 \sigma_2^2 - \sigma_1^4) + \sigma_1^4) = g(\lambda),$$

or equivalently, there exists a $\lambda \in [0, 1]$ such that:

$$g(\lambda) - f(\lambda) = (g - f)(\lambda) > 0. \tag{B.2}$$

Note that when $\lambda = 0$, $(g - f)(\lambda) = 0$, and that when $\lambda = 1$, $(g - f)(\lambda) < 0$. Thus, since $(g - f)(\lambda)$ is a degree-2 function of λ, the condition from equation (B.2) is true

if and only if the derivative of $(g - f)(\lambda)$ is positive at $\lambda = 0$.

We have that:

$$(g - f)'(0) = \sigma_1^2\sigma_2^2 - \sigma_1^4 + \sigma_1^2 c^2.$$

Given the above, $(g - f)'(0) > 0$ if and only if $c^2 > \sigma_1^2 - \sigma_2^2$.

Thus, the volume of $E(\boldsymbol{\mu}', \mathcal{F}(\boldsymbol{\Sigma}'))$ will be smaller than the volume of $E(\boldsymbol{\mu}_2, \mathcal{F}(\boldsymbol{\Sigma}_2))$ if and only if $c^2 > \sigma_1^2 - \sigma_2^2$. This implies that the volume of $E^{(\mathsf{Rank})}(\boldsymbol{\mu}', \boldsymbol{\Sigma}')$ will be strictly less than the volume of $E^{(\mathsf{Rank})}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ if and only if $c^2 > \sigma_1^2 - \sigma_2^2$. This concludes the proof of Theorem 4.2.1. □

B.2 Overview of the Ellipsoid Method

A linear program is an optimization problem of a linear objective function subject to linear equality and linear inequality constraints. The set of feasible solutions (if any exist) correspond to a region contained within a convex body $\mathcal{K}$. For any convex body $\mathcal{K}$, the optimal (i.e. minimum volume) circumscribing ellipsoid is known as the Löwner-John Ellipsoid. In general, these ellipsoids are hard to compute, but special cases, including intersections between ellipsoids and hyperplanes, halfspaces, or spaces between two parallel hyperplanes, have closed form expressions. If the solution of the linear program is initially known to be contained in some ellipsoid, the linear program's constraints can be used to obtain successively smaller volume ellipsoids by computing these Löwner-John ellipsoids in an iterative fashion. This procedure can then be used to determine feasibility: (1) In each iteration, check whether the center of the current ellipsoid satisfies the linear

constraints. (2) If the center satisfies all constraints, then the center is a solution. (3) Otherwise, there is some constraint that is not satisfied by the center. Set the new ellipsoid to be the Löwner-John ellipsoid circumscribing the intersection of the current ellipsoid and the halfspace of the unsatisfied constraint. Continue to the next iteration. (4) If at some point the volume of the ellipsoid becomes sufficiently small, conclude that the linear program is infeasible. The fact that the ellipsoid method is polynomial time is implied by the fact that the volumes of the successive Löwner-John ellipsoids become sufficiently small in a polynomial number of steps.

Bibliography

[1] Alagic, G., Alperin-Sheriff, J., Apon, D., Cooper, D., Dang, Q., Kelsey, J., Liu, Y.K., Miller, C., Moody, D., Peralta, R., Perlner, R., Robinson, A., Smith-Tone, D.: Status report on the second round of the nist post-quantum cryptography standardization process. Tech. Rep. : NIST Internal Report (NISTIR) 8309, U.S. Department of Commerce, Washington, D.C. (2020). https://doi.org/10.6028/NIST.IR.8309

[2] Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T., Kelsey, J., Lichtinger, J., Miller, C., Moody, D., Peralta, R., Perlner, R., Robinson, A., Smith-Tone, D., Liu, Y.K.: Status report on the third round of the nist post-quantum cryptography standardization process. Tech. Rep. : NIST Internal Report (NISTIR) 8413, U.S. Department of Commerce, Washington, D.C. (2022). https://doi.org/10.6028/NIST.IR.8413

[3] Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T., Kelsey, J., Lichtinger, J., Miller, C., Moody, D., Peralta, R., Perlner, R., Robinson, A., Smith-Tone, D., Liu, Y.K.: Status report on the third round of the nist post-quantum cryptography standardization process. Tech. Rep. : NIST Internal Report (NISTIR) 8413, U.S. Department of Commerce, Washington, D.C. (2022). https://doi.org/10.6028/NIST.IR.8413

[4] Albrecht, M., Cid, C., Faugère, J.C., Fitzpatrick, R., Perret, L.: On the complexity of the Arora-Ge Algorithm against LWE. In: SCC 2012 – Third international conference on Symbolic Computation and Cryptography. pp. 93–99. Castro Urdiales, Spain (Jul 2012), https://hal.inria.fr/hal-00776434

[5] Albrecht, M.R., Bai, S., Li, J., Rowell, J.: Lattice reduction with approximate enumeration oracles - practical algorithms and concrete performance. pp. 732–759 (2021). https://doi.org/10.1007/978-3-030-84245-1_25

[6] Albrecht, M.R., Cid, C., Faugère, J.C., Fitzpatrick, R., Perret, L.: On the complexity of the BKW algorithm on LWE. Cryptology ePrint Archive, Report 2012/636 (2012), https://eprint.iacr.org/2012/636

[7] Albrecht, M.R., Göpfert, F., Virdia, F., Wunderer, T.: Revisiting the expected cost of solving uSVP and applications to LWE. pp. 297–322 (2017). https://doi.org/10.1007/978-3-319-70694-8_11

[8] Alkim, E., Bos, J.W., Ducas, L., Longa, P., Mironov, I., Naehrig, M., Nikolaenko, V., Peikert, C., Raghunathan, A., Stebila, D., Easterbrook, K., Brian, L.: Frodokem: Practical quantum-secure key encapsulation from generic lattices (Apr 2022), https://frodokem.org/

[9] Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange - A new hope. pp. 327–343 (2016)

[10] Allan, T., Brumley, B.B., Falkner, K., van de Pol, J., Yarom, Y.: Amplifying side channels through performance degradation. In: Proceedings of the 32nd Annual Conference on Computer Security Applications. p. 422–435. ACSAC '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2991079.2991084, https://doi.org/10.1145/2991079.2991084

[11] Augot, D., Finiasz, M., Manuel, P.G.S., Sendrie, N.: Sha-3 proposal: Fsb (2009), https://www.rocq.inria.fr/secret/CBCrypto/fsbdoc.pdf

[12] Augot, D., Finiasz, M., Sendrier, N.: A fast provably secure cryptographic hash function. Cryptology ePrint Archive, Report 2003/230 (2003), https://eprint.iacr.org/2003/230

[13] Austrin, P., Kaski, P., Koivisto, M., Nederlof, J.: Subset sum in the absence of concentration. In: Mayr, E.W., Ollinger, N. (eds.) 32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany. LIPIcs, vol. 30, pp. 48–61. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2015). https://doi.org/10.4230/LIPICS.STACS.2015.48, https://doi.org/10.4230/LIPIcs.STACS.2015.48

[14] Aweke, Z.B., Yitbarek, S.F., Qiao, R., Das, R., Hicks, M., Oren, Y., Austin, T.: Anvil: Software-based protection against next-generation rowhammer attacks. ACM SIGPLAN Notices **51**(4), 743–755 (2016)

[15] Bagherzandi, A., Cheon, J.H., Jarecki, S.: Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. pp. 449–458 (2008). https://doi.org/10.1145/1455770.1455827

[16] Bai, S., Stehlé, D., Wen, W.: Measuring, simulating and exploiting the head concavity phenomenon in BKZ. pp. 369–404 (2018). https://doi.org/10.1007/978-3-030-03326-2_13

[17] Basso, A., Mera, J.M.B., D'anvers, J.P., Karmakar, A., Roy, S.S., Beirendonck, M.V., Vercauteren, F.: Saber: Mod-lwr based kem (round 3 submission). In: NIST Round 3 Submissions (2021)

[18] Bauer, A., Gilbert, H., Renault, G., Rossi, M.: Assessment of the key-reuse resilience of NewHope. pp. 272–292 (2019). https://doi.org/10.1007/978-3-030-12612-4_14

[19] Becker, A., Coron, J.S., Joux, A.: Improved generic algorithms for hard knapsacks. pp. 364–385 (2011). https://doi.org/10.1007/978-3-642-20465-4_21

[20] Becker, A., Ducas, L., Gama, N., Laarhoven, T.: New directions in nearest neighbor searching with applications to lattice sieving. pp. 10–24 (2016). https://doi.org/10.1137/1.9781611974331.ch2

[21] Becker, A., Joux, A., May, A., Meurer, A.: Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. pp. 520–536 (2012). https://doi.org/10.1007/978-3-642-29011-4_31

[22] Benhamouda, F., Lepoint, T., Loss, J., Orrù, M., Raykova, M.: On the (in)security of ROS **35**(4), 25 (Oct 2022). https://doi.org/10.1007/s00145-022-09436-0

[23] Bernstein, D.J., Jeffery, S., Lange, T., Meurer, A.: Quantum algorithms for the subset-sum problem. pp. 16–33 (2013). https://doi.org/10.1007/978-3-642-38616-9_2

[24] Bernstein, D.J., Lange, T., Niederhagen, R., Peters, C., Schwabe, P.: FSBday. pp. 18–38 (2009)

[25] Bindel, N., Schanck, J.M.: Decryption failure is more likely after success. pp. 206–225 (2020). https://doi.org/10.1007/978-3-030-44223-1_12

[26] Bland, R.G., Goldfarb, D., Todd, M.J.: The ellipsoid method: A survey. Operations research **29**(6), 1039–1091 (1981)

[27] Bonnetain, X., Bricout, R., Schrottenloher, A., Shen, Y.: Improved classical and quantum algorithms for subset-sum. pp. 633–666 (2020). https://doi.org/10.1007/978-3-030-64834-3_22

[28] Bootle, J., Delaplace, C., Espitau, T., Fouque, P.A., Tibouchi, M.: Lwe without modular reduction and improved side-channel attacks against bliss. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 494–524. Springer (2018)

[29] Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehle, D.: Crystals - kyber: A cca-secure module-lattice-based kem. In: 2018 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 353–367 (2018). https://doi.org/10.1109/EuroSP.2018.00032

[30] Bos, J.W., Friedberger, S., Martinoli, M., Oswald, E., Stam, M.: Assessing the feasibility of single trace power analysis of Frodo. pp. 216–234 (2019). https://doi.org/10.1007/978-3-030-10970-7_10

[31] Bosman, E., Razavi, K., Bos, H., Giuffrida, C.: Dedup Est Machina: Memory deduplication as an advanced exploitation vector. In: IEEE SP (2016)

[32] Boura, C., Canteaut, A.: Zero-sum distinguishers for iterated permutations and application to Keccak-f and Hamsi-256. pp. 1–17 (2011). https://doi.org/10.1007/978-3-642-19574-7_1

[33] Brasser, F., Davi, L., Gens, D., Liebchen, C., Sadeghi, A.R.: {CAn't} touch this: Software-only mitigation against rowhammer attacks targeting kernel memory. In: 26th USENIX Security Symposium (USENIX Security 17). pp. 117–130 (2017)

[34] Brown, S.T., Buitrago, P., Hanna, E., Sanielevici, S., Scibek, R., Nystrom, N.A.: Bridges-2: A platform for rapidly-evolving and data intensive research. In: Practice and Experience in Advanced Research Computing, pp. 1–4 (2021)

[35] Bruna, J., Regev, O., Song, M.J., Tang, Y.: Continuous LWE. In: STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021. pp. 694–707 (2021). https://doi.org/10.1145/3406325.3451000, https://doi.org/10.1145/3406325.3451000

[36] Canella, C., Schwarz, M., Haubenwallner, M., Schwarzl, M., Gruss, D.: Kaslr: Break it, fix it, repeat. In: Proceedings of the 15th ACM Asia Conference on Computer and Communications Security. pp. 481–493 (2020)

[37] Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. pp. 13–28 (2003). https://doi.org/10.1007/3-540-36400-5_3

[38] Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better lattice security estimates. pp. 1–20 (2011). https://doi.org/10.1007/978-3-642-25385-0_1

[39] Cojocar, L., Razavi, K., Giuffrida, C., Bos, H.: Exploiting correcting codes: On the effectiveness of ECC memory against Rowhammer attacks. In: IEEE SP (2019)

[40] Coron, J.S., Joux, A.: Cryptanalysis of a provably secure cryptographic hash function. Cryptology ePrint Archive, Report 2004/013 (2004), https://eprint.iacr.org/2004/013

[41] Dachman-Soled, D., Ducas, L., Gong, H., Rossi, M.: LWE with side information: Attacks and concrete security estimation. pp. 329–358 (2020). https://doi.org/10.1007/978-3-030-56880-1_12

[42] D'Anvers, J.P., Guo, Q., Johansson, T., Nilsson, A., Vercauteren, F., Verbauwhede, I.: Decryption failure attacks on IND-CCA secure lattice-based schemes. pp. 565–598 (2019). https://doi.org/10.1007/978-3-030-17259-6_19

[43] D'Anvers, J.P., Rossi, M., Virdia, F.: (One) failure is not an option: Bootstrapping the search for failures in lattice-based encryption schemes. pp. 3–33 (2020). https://doi.org/10.1007/978-3-030-45727-3_1

[44] D'Anvers, J.P., Vercauteren, F., Verbauwhede, I.: On the impact of decryption failures on the security of LWE/LWR based schemes. Cryptology ePrint Archive, Report 2018/1089 (2018), https://eprint.iacr.org/2018/1089

[45] Ding, J., Alsayigh, S., RV, S., Fluhrer, S., Lin, X.: Leakage of signal function with reused keys in RLWE key exchange. Cryptology ePrint Archive, Report 2016/1176 (2016), https://eprint.iacr.org/2016/1176

[46] Ding, J., Alsayigh, S., Saraswathy, R.V., Fluhrer, S., Lin, X.: Leakage of signal function with reused keys in rlwe key exchange. In: 2017 IEEE International Conference on Communications (ICC). pp. 1–6 (2017). https://doi.org/10.1109/ICC.2017.7996806

[47] Ding, J., Fluhrer, S.R., RV, S.: Complete attack on RLWE key exchange with reused keys, without signal leakage. pp. 467–486 (2018). https://doi.org/10.1007/978-3-319-93638-3_27

[48] Drijvers, M., Edalatnejad, K., Ford, B., Kiltz, E., Loss, J., Neven, G., Stepanovs, I.: On the security of two-round multi-signatures. pp. 1084–1101 (2019). https://doi.org/10.1109/SP.2019.00050

[49] Engineering, A.S., (SEAR), A.: imessage with pq3: The new state of the art in quantum-secure messaging at scale (2024), https://security.apple.com/blog/imessage-pq3/

[50] Esser, A., Zweydinger, F.: New time-memory trade-offs for subset sum: Improving ISD in theory and practice. pp. 360–390 (2023). https://doi.org/10.1007/978-3-031-30589-4_13

[51] Fahr Jr, M., Kippen, H., Kwong, A., Dang, T., Lichtinger, J., Dachman-Soled, D., Genkin, D., Nelson, A., Perlner, R., Yerukhimovich, A., et al.: When frodo flips: End-to-end key recovery on frodokem via rowhammer. Cryptology ePrint Archive (2022)

[52] Fluhrer, S.: Cryptanalysis of ring-lwe based key exchange with key share reuse. Cryptology ePrint Archive, Report 2016/085 (2016), https://eprint.iacr.org/2016/085

[53] Fluhrer, S.: Cryptanalysis of ring-LWE based key exchange with key share reuse. Cryptology ePrint Archive, Report 2016/085 (2016), https://eprint.iacr.org/2016/085

[54] Frigo, P., Giuffrida, C., Bos, H., Razavi, K.: Grand pwning unit: Accelerating microarchitectural attacks with the GPU. In: IEEE SP. pp. 195–210 (2018)

[55] Frigo, P., Vannacci, E., Hassan, H., van der Veen, V., Mutlu, O., Giuffrida, C., Bos, H., Razavi, K.: TRRespass: Exploiting the Many Sides of Target Row Refresh. In: S&P (May 2020), https://comsec.ethz.ch/wp-content/files/trrespass_sp20.pdf, best Paper Award, Pwnie Award for the Most Innovative Research, Honorable Mention in IEEE MICRO Top Picks

[56] Fuchsbauer, G., Plouviez, A., Seurin, Y.: Blind schnorr signatures and signed ElGamal encryption in the algebraic group model. pp. 63–95 (2020). https://doi.org/10.1007/978-3-030-45724-2_3

[57] Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. pp. 537–554 (1999). https://doi.org/10.1007/3-540-48405-1_34

[58] Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems **20**(1), 51–83 (Jan 2007). https://doi.org/10.1007/s00145-006-0347-3

[59] Gras, B., Razavi, K., Bosman, E., Bos, H., Giuffrida, C.: Aslr on the line: Practical cache attacks on the mmu. In: NDSS. vol. 17, p. 26 (2017)

[60] Grassi, L., Naya-Plasencia, M., Schrottenloher, A.: Quantum algorithms for the k-xor problem. pp. 527–559 (2018). https://doi.org/10.1007/978-3-030-03326-2_18

[61] Grötschel, M., Lovász, L., Schrijver, A.: The Ellipsoid Method, pp. 64–101. Springer Berlin Heidelberg, Berlin, Heidelberg (1988). https://doi.org/10.1007/978-3-642-97881-4_4, https://doi.org/10.1007/978-3-642-97881-4_4

[62] Gruss, D., Lipp, M., Schwarz, M., Genkin, D., Juffinger, J., O'Connell, S., Schoechl, W., Yarom, Y.: Another flip in the wall of Rowhammer defenses. In: IEEE SP. pp. 245–261 (2018)

[63] Gruss, D., Maurice, C., Fogh, A., Lipp, M., Mangard, S.: Prefetch side-channel attacks: Bypassing smap and kernel aslr. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. pp. 368–379 (2016)

[64] Gruss, D., Maurice, C., Mangard, S.: Rowhammer.js: A remote software-induced fault attack in JavaScript. In: DIMVA. pp. 300–321 (2016)

[65] Güler, O., Gürtuna, F.: Symmetry of convex sets and its applications to the extremal ellipsoids of convex bodies. Optimization Methods and Software **27**(4-5), 735–759 (2012)

[66] Guo, Q., Johansson, T., Nilsson, A.: A generic attack on lattice-based schemes using decryption errors with application to ss-ntru-pke. Cryptology ePrint Archive, Report 2019/043 (2019), https://eprint.iacr.org/2019/043

[67] Gupte, A., Vafa, N., Vaikuntanathan, V.: Continuous LWE is as hard as LWE & applications to learning gaussian mixtures. Cryptology ePrint Archive, Report 2022/437 (2022), https://eprint.iacr.org/2022/437

[68] Hanebeck, U.D., Horn, J.: Fusing information simultaneously corrupted by uncertainties with known bounds and random noise with known distribution. Information Fusion **1**(1), 55–63 (2000)

[69] Herold, G., Kirshanova, E., Laarhoven, T.: Speed-ups and time-memory trade-offs for tuple lattice sieving. pp. 407–436 (2018). https://doi.org/10.1007/978-3-319-76578-5_14

[70] Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. pp. 341–371 (2017). https://doi.org/10.1007/978-3-319-70500-2_12

[71] Howe, J., Khalid, A., Martinoli, M., Regazzoni, F., Oswald, E.: Fault attack countermeasures for error samplers in lattice-based cryptography. In: 2019 IEEE International Symposium on Circuits and Systems (ISCAS). pp. 1–5. IEEE (2019)

[72] Howgrave-Graham, N., Joux, A.: New generic algorithms for hard knapsacks. pp. 235–256 (2010). https://doi.org/10.1007/978-3-642-13190-5_12

[73] für Sicherheit in der Informationstechnik, B.: Bsi tr-02102-1: "cryptographic mechanisms: Recommendations and key lengths" version: 2022-1 (Jan 2022), https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf

[74] Islam, S., Mus, K., Singh, R., Schaumont, P., Sunar, B.: Signature correction attack on dilithium signature scheme. CoRR **abs/2203.00637** (2022). https://doi.org/10.48550/arXiv.2203.00637, https://doi.org/10.48550/arXiv.2203.00637

[75] Islam, S., Mus, K., Singh, R., Schaumont, P., Sunar, B.: Signature correction attack on dilithium signature scheme (2022). https://doi.org/10.48550/ARXIV.2203.00637, https://arxiv.org/abs/2203.00637

[76] Jattke, P., van der Veen, V., Frigo, P., Gunter, S., Razavi, K.: Blacksmith: Scalable Rowhammering in the Frequency Domain. In: S&P (May

2022), Paper=https://comsec.ethz.ch/wp-content/files/blacksmith_sp22.pdfURL=https://comsec.ethz.ch/research/dram/blacksmith

[77] Joux, A.: Cryptanalysis of the EMD mode of operation. pp. 1–16 (2003). https://doi.org/10.1007/3-540-39200-9_1

[78] Jr., H.W.L.: Integer programming with a fixed number of variables. Math. Oper. Res. **8**(4), 538–548 (1983). https://doi.org/10.1287/moor.8.4.538, https://doi.org/10.1287/moor.8.4.538

[79] Kalman, R.E.: A new approach to linear filtering and prediction problems (1960)

[80] Khachiyan, L.G.: A polynomial algorithm in linear programming. In: Doklady Akademii Nauk. vol. 244, pp. 1093–1096. Russian Academy of Sciences (1979)

[81] Kim, Y., Daly, R., Kim, J., Fallin, C., Lee, J.H., Lee, D., Wilkerson, C., Lai, K., Mutlu, O.: Flipping bits in memory without accessing them: An experimental study of dram disturbance errors. In: 2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA). pp. 361–372 (2014). https://doi.org/10.1109/ISCA.2014.6853210

[82] Kirkwood, D., Lackey, B.C., McVey, J., Motley, M., Solinas, J.A., Tuller, D.: Failure is not an option: Standardization issues for post-quantum key agreement (2015), https://csrc.nist.gov/csrc/media/events/workshop-on-cybersecurity-in-a-post-quantum-world/documents/presentations/session7-motley-mark.pdf

[83] Kocher, P., Horn, J., Fogh, A., Genkin, D., Gruss, D., Haas, W., Hamburg, M., Lipp, M., Mangard, S., Prescher, T., Schwarz, M., Yarom, Y.: Spectre attacks: exploiting speculative execution. Commun. ACM **63**(7), 93–101 (2020)

[84] Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. pp. 104–113 (1996). https://doi.org/10.1007/3-540-68697-5_9

[85] Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. pp. 388–397 (1999). https://doi.org/10.1007/3-540-48405-1_25

[86] Komlo, C., Goldberg, I.: Frost: Flexible round-optimized schnorr threshold signatures **Version from "January 7, 2020"** (2020), https://crysp.uwaterloo.ca/software/frost/frost-extabs.pdf

[87] Konoth, R.K., Oliverio, M., Tatar, A., Andriesse, D., Bos, H., Giuffrida, C., Razavi, K.: {ZebRAM}: Comprehensive and compatible software protection against rowhammer attacks. In: 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18). pp. 697–710 (2018)

[88] Kurzhanski, A., Vályi, I.: Ellipsoidal calculus for estimation and control. Springer (1997)

[89] Kwong, A., Genkin, D., Gruss, D., Yarom, Y.: Rambleed: Reading bits in memory without accessing them. In: 41st IEEE Symposium on Security and Privacy (S&P) (2020)

[90] Laarhoven, T.: Search problems in cryptography: from fingerprinting to lattice sieving. PhD thesis (2015)

[91] Levieil, É., Fouque, P.A.: An improved LPN algorithm. pp. 348–359 (2006). https://doi.org/10.1007/11832072_24

[92] Lindner, R., Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. pp. 319–339 (2011). https://doi.org/10.1007/978-3-642-19074-2_21

[93] Lipp, M., Schwarz, M., Gruss, D., Prescher, T., Haas, W., Horn, J., Mangard, S., Kocher, P., Genkin, D., Yarom, Y., Hamburg, M., Strackx, R.: Meltdown: reading kernel memory from user space. Commun. ACM **63**(6), 46–56 (2020). https://doi.org/10.1145/3357033, https://doi.org/10.1145/3357033

[94] Lipp, M., Schwarz, M., Raab, L., Lamster, L., Aga, M.T., Maurice, C., Gruss, D.: Nethammer: Inducing rowhammer faults through network requests. In: 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). pp. 710–719. IEEE (2020)

[95] Lyubashevsky, V.: The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In: International Workshop and International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (2005), https://api.semanticscholar.org/CorpusID:7748280

[96] Marazzi, M., Jattke, P., Solt, F., Razavi, K.: ProTRR: Principled yet Optimal In-DRAM Target Row Refresh. In: S&P (May 2022), Paper=https://comsec.ethz.ch/wp-content/files/protrr_sp22.pdfURL=https://comsec.ethz.ch/research/dram/protrr, patent pending, ETH Spark Award Nomination

[97] Maxwell, G., Poelstra, A., Seurin, Y., Wuille, P.: Simple schnorr multi-signatures with applications to bitcoin. Cryptology ePrint Archive, Report 2018/068 (2018), https://eprint.iacr.org/2018/068

[98] McCann, D., Oswald, E., Whitnall, C.: Towards practical tools for side channel aware software engineering: 'grey box' modelling for instruction leakages. pp. 199–216 (2017)

[99] McDiarmid, C.: On the method of bounded differences, p. 148–188. London Mathematical Society Lecture Note Series, Cambridge University Press (1989). https://doi.org/10.1017/CBO9781107359949.008

[100] Micciancio, D., Regev, O.: Lattice-based cryptography. In: Post-quantum cryptography, pp. 147–191. Springer (2009)

[101] Minder, L., Sinclair, A.: The extended k-tree algorithm **25**(2), 349–382 (Apr 2012). https://doi.org/10.1007/s00145-011-9097-y

[102] Mus, K., Islam, S., Sunar, B.: QuantumHammer: A practical hybrid attack on the LUOV signature scheme. pp. 1071–1084 (2020). https://doi.org/10.1145/3372297.3417272

[103] Mutlu, O., Kim, J.S.: Rowhammer: A retrospective. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **39**(8), 1555–1571 (2020). https://doi.org/10.1109/TCAD.2019.2915318

[104] Nikolic, I., Sasaki, Y.: Refinements of the k-tree algorithm for the generalized birthday problem. pp. 683–703 (2015). https://doi.org/10.1007/978-3-662-48800-3_28

[105] NIST: Submission requirements and evaluation criteria for the post-quantum cryptography standardization process (Dec 2016), https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf

[106] Ohta, K., Okamoto, T.: A digital multisignature scheme based on the Fiat-Shamir scheme. pp. 139–148 (1993). https://doi.org/10.1007/3-540-57332-1_11

[107] Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. pp. 31–53 (1993). https://doi.org/10.1007/3-540-48071-4_3

[108] Peikert, C.: A decade of lattice cryptography. Foundations and Trends in Theoretical Computer Science **10**(4), 283–424 (2016). https://doi.org/10.1561/0400000074, http://dx.doi.org/10.1561/0400000074

[109] Pessl, P., Bruinderink, L.G., Yarom, Y.: To bliss-b or not to be: Attacking strongswan's implementation of post-quantum signatures. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1843–1855 (2017)

[110] Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures **13**(3), 361–396 (Jun 2000). https://doi.org/10.1007/s001450010003

[111] Pordes, R., Petravick, D., Kramer, B., Olson, D., Livny, M., Roy, A., Avery, P., Blackburn, K., Wenaus, T., Würthwein, F., et al.: The open science grid. In: Journal of Physics: Conference Series. vol. 78, p. 012057. IOP Publishing (2007)

[112] Qin, Y., Cheng, C., Zhang, X., Pan, Y., Hu, L., Ding, J.: A systematic approach and analysis of key mismatch attacks on lattice-based nist candidate kems. In: ASIACRYPT 2021, Tibouchi and H. Wang (Eds.). pp. 92–121 (2021). https://doi.org/https://doi.org/10.1007/978-3-030-92068-5_4

[113] Qin, Y., Cheng, C., Zhang, X., Pan, Y., Hu, L., Ding, J.: A systematic approach and analysis of key mismatch attacks on lattice-based NIST candidate KEMs. Cryptology ePrint Archive, Report 2021/123 (2021), https://eprint.iacr.org/2021/123

[114] Ravi, P., Jhanwar, M.P., Howe, J., Chattopadhyay, A., Bhasin, S.: Side-channel assisted existential forgery attack on Dilithium - A NIST PQC candidate. Cryptology ePrint Archive, Report 2018/821 (2018), https://eprint.iacr.org/2018/821

[115] Ravi, P., Jhanwar, M.P., Howe, J., Chattopadhyay, A., Bhasin, S.: Side-channel assisted existential forgery attack on dilithium-a nist pqc candidate. Cryptology ePrint Archive (2018)

[116] Ravi, P., Jhanwar, M.P., Howe, J., Chattopadhyay, A., Bhasin, S.: Exploiting determinism in lattice-based signatures: practical fault attacks on pqm4 implementations of nist candidates. In: Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security. pp. 427–440 (2019)

[117] Ravi, P., Jhanwar, M.P., Howe, J., Chattopadhyay, A., Bhasin, S.: Exploiting determinism in lattice-based signatures: Practical fault attacks on pqm4 implementations of NIST candidates. pp. 427–440 (2019). https://doi.org/10.1145/3321705.3329821

[118] Razavi, K., Gras, B., Bosman, E., Preneel, B., Giuffrida, C., Bos, H.: Flip feng shui: Hammering a needle in the software stack. In: USENIX Security. pp. 1–18 (2016)

[119] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. pp. 84–93 (2005). https://doi.org/10.1145/1060590.1060603

[120] Research, G.: Half-double: Next-row-over assisted rowhammer (Aug 2021), https://github.com/google/hammer-kit/blob/main/20210525_half_double.pdf

[121] de Ridder, F., Frigo, P., Vannacci, E., Bos, H., Giuffrida, C., Razavi, K.: SMASH: Synchronized Many-sided Rowhammer Attacks from JavaScript. In: USENIX Security (Aug 2021), Paper=https:

//comsec.ethz.ch/wp-content/files/smash_sec21.pdfURL=https://comsec.ethz.ch/research/dram/smash, pwnie Nomination for the Most Underhyped Research

[122] Ros, L., Sabater i Pruna, A., Thomas, F.: An ellipsoid calculus based on propagation and fusion. IEEE Transactions on Systems Man and Cybernetics Part B (Cybernetics) **32**, 430–443 (08 2002). https://doi.org/10.1109/TSMCB.2002.1018763

[123] Saito, T., Xagawa, K., Yamakawa, T.: Tightly-secure key-encapsulation mechanism in the quantum random oracle model. pp. 520–551 (2018). https://doi.org/10.1007/978-3-319-78372-7_17

[124] Schnorr, C.P.: Security of blind discrete log signatures against interactive attacks. pp. 1–12 (2001)

[125] Schnorr, C., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Math. Program. **66**, 181–199 (1994)

[126] Seaborn, M., Dullien, T.: Exploiting the DRAM Rowhammer bug to gain kernel privileges. https://googleprojectzero.blogspot.com/2015/03/exploiting-dram-rowhammer-bug-to-gain.html (2015)

[127] Sepulveda, J., Zankl, A., Mischke, O.: Cache attacks and countermeasures for ntruencrypt on mpsocs: post-quantum resistance for the iot. In: 2017 30th IEEE International System-on-Chip Conference (SOCC). pp. 120–125. IEEE (2017)

[128] Sepulveda, J., Zankl, A., Mischke, O.: Cache attacks and countermeasures for ntruencrypt on mpsocs: Post-quantum resistance for the iot. In: 2017 30th IEEE International System-on-Chip Conference (SOCC). pp. 120–125 (2017). https://doi.org/10.1109/SOCC.2017.8226020

[129] Sfiligoi, I., Bradley, D.C., Holzman, B., Mhashilkar, P., Padhi, S., Wurthwein, F.: The pilot way to grid resources using glideinwms. In: 2009 WRI World congress on computer science and information engineering. vol. 2, pp. 428–432. IEEE (2009)

[130] Shallue, A.: An improved multi-set algorithm for the dense subset sum problem. In: Algorithmic Number Theory: 8th International Symposium, ANTS-VIII Banff, Canada, May 17-22, 2008 Proceedings 8. pp. 416–429. Springer (2008)

[131] Snow, K.Z., Monrose, F., Davi, L., Dmitrienko, A., Liebchen, C., Sadeghi, A.R.: Just-in-time code reuse: On the effectiveness of fine-grained address space layout randomization. In: 2013 IEEE Symposium on Security and Privacy. pp. 574–588. IEEE (2013)

[132] of Standards, N.I., (NIST), T.: Post-quantum cryptography - round 3 submissions (Apr 2022), https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions

[133] of Standards, N.I., (NIST), T.: Post-quantum cryptography standardization (Apr 2022), https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization

[134] of Standards, N.I., (NIST), T.: Pqc standardization process: Announcing four candidates to be standardized, plus fourth round candidates (Jul 2022), https://csrc.nist.gov/News/2022/pqc-candidates-to-be-standardized-and-round-4

[135] Syta, E., Tamas, I., Visher, D., Wolinsky, D.I., Jovanovic, P., Gasser, L., Gailly, N., Khoffi, I., Ford, B.: Keeping authorities "honest or bust" with decentralized witness cosigning. pp. 526–545 (2016). https://doi.org/10.1109/SP.2016.38

[136] Tatar, A., Konoth, R.K., Athanasopoulos, E., Giuffrida, C., Bos, H., Razavi, K.: Throwhammer: Rowhammer Attacks over the Network and Defenses. In: USENIX ATC (Jul 2018), https://comsec.ethz.ch/wp-content/files/throwhammer_atc18.pdf, pwnie Award Nomination for the Most Innovative Research

[137] Tibouchi, M., Wallet, A.: One bit is all it takes: a devastating timing attack on bliss's non-constant time sign flips. Journal of Mathematical Cryptology **15**(1), 131–142 (2021)

[138] Tobah, Y., Kwong, A., Kang, I., Genkin, D., Shin, K.G.: Spechammer: Combining spectre and rowhammer for new speculative attacks. In: 43rd IEEE Symposium on Security and Privacy (S&P) (2022)

[139] TSUNOO, Y.: Crypt-analysis of block ciphers implemented on computers with cache. Proc. ISITA2002, Oct. (2002), https://ci.nii.ac.jp/naid/10026863967/en/

[140] Van Der Veen, V., Fratantonio, Y., Lindorfer, M., Gruss, D., Maurice, C., Vigna, G., Bos, H., Razavi, K., Giuffrida, C.: Drammer: Deterministic Rowhammer attacks on mobile platforms. In: CCS. pp. 1675–1689 (2016)

[141] Villanueva-Polanco, R.: Cold boot attacks on bliss. In: International Conference on Cryptology and Information Security in Latin America. pp. 40–61. Springer (2019)

[142] Villanueva-Polanco, R.: Cold boot attacks on Bliss. pp. 40–61 (2019). https://doi.org/10.1007/978-3-030-30530-7_3

[143] Wagner, D.: A generalized birthday problem. pp. 288–303 (2002). https://doi.org/10.1007/3-540-45708-9_19

[144] Wang, Z., Shen, X., Zhu, Y.: On equivalence of major relaxation methods for minimum ellipsoid covering intersection of ellipsoids. Automatica **103**, 337–345 (2019). https://doi.org/https://doi.org/10.1016/j.automatica.2019.02.001, https://www.sciencedirect.com/science/article/pii/S0005109819300445

[145] Westerban, B., Evans, W.: Post-quantum crypto should be free, so we're including it for free, forever (Mar 2023), https://blog.cloudflare.com/post-quantum-crypto-should-be-free/

[146] Xiao, Y., Zhang, X., Zhang, Y., Teodorescu, R.: One bit flips, one cloud flops: Cross-VM row hammer attacks and privilege escalation. In: USENIX Security (2016)

[147] Yarom, Y., Falkner, K.: {FLUSH+ RELOAD}: A high resolution, low noise, l3 cache {Side-Channel} attack. In: 23rd USENIX security symposium (USENIX security 14). pp. 719–732 (2014)