



# LUND UNIVERSITY

## Secure on Paper: Attacking Post-Quantum Cryptography in Practice

Nabokov, Denis

2026

[Link to publication](#)

*Citation for published version (APA):*

Nabokov, D. (2026). *Secure on Paper: Attacking Post-Quantum Cryptography in Practice*. Electrical and Information Technology, Lund University.

*Total number of authors:*

1

### General rights

Unless other specific re-use rights are stated the following general rights apply:

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: <https://creativecommons.org/licenses/>

### Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117  
221 00 Lund  
+46 46-222 00 00



# Secure on Paper: Attacking Post-Quantum Cryptography in Practice

Denis Nabokov



**LUND**  
UNIVERSITY

ISBN 978-91-90202-15-9 (print)  
ISBN 978-91-90202-14-2 (electronic)  
Series of licentiate and doctoral theses  
No. 194  
ISSN 1654-790X

Denis Nabokov  
Department of Electrical and Information Technology  
Lund University  
Box 118  
SE-221 00 Lund  
Sweden

Typeset using L<sup>A</sup>T<sub>E</sub>X.

Cover image illustrating that the real implementation might not always fit the model. Generated by the ChatGPT Images 2.0.

Printed in Sweden by Tryckeriet i E-huset, Lund, 2026.

© 2026 Denis Nabokov

*Published articles have been reprinted with permission from the respective copyright holder.*

## Abstract

The transition to post-quantum cryptography is well underway. Driven by the recognition that Shor’s algorithm would render all widely deployed asymmetric cryptosystems insecure in the presence of a sufficiently capable quantum computer, the cryptographic community has spent the past decade designing, evaluating, and standardizing a new generation of public-key primitives. In 2024, ML-KEM became the first standardized post-quantum Key Encapsulation Mechanism; HQC was selected as the primary backup candidate shortly thereafter. These schemes are now being deployed in production systems.

A cryptographic scheme, however, is not merely an algorithm on paper. It is a piece of software running on physical hardware, and physical hardware is not abstract. Execution time, power consumption, and electromagnetic radiation all depend on the data being processed, and an adversary who can observe these physical side channels has access to information that no standard security model (such as commonly used IND-CCA model) is designed to account for. The migration to post-quantum cryptography therefore raises a question that is as urgent as the theoretical one: are these new schemes secure in practice, once the physical realities of implementation are taken into account?

This thesis addresses that question. Its central contribution is a collection of novel key-recovery attacks against post-quantum KEMs and related primitives, each grounded in concrete implementation vulnerabilities and validated on real hardware. A unifying thread is the SCA-LDPC framework, first introduced here and progressively refined across several papers: the core idea is to model a large set of carefully crafted chosen-ciphertext side-channel queries as a sparse parity-check code, and to recover the secret key via belief-propagation decoding. This approach brings information-theoretic rigour to a problem that has previously been addressed by largely ad hoc methods, and it yields substantial improvements in attack efficiency across ML-KEM, HQC, NTRU, and NTRU Prime.

Beyond side-channel attacks on KEMs, the thesis also investigates implementation security in a different setting: approximate homomorphic encryption. We demonstrate that the noise-flooding countermeasures used to secure the CKKS scheme against passive key-recovery attacks are only as strong as the noise estimation procedure they rely upon. When that estimation is performed in a non-worst-case manner, an adversary can recover the full secret key from a single decryption output. The vulnerability is shown to be structural and cannot be mitigated by increasing the statistical security parameter alone.

Across all contributions, the attacks are accompanied by information-theoretic analyses that characterize how close they come to fundamental limits, and by open-source release of all attack code. The results collectively suggest that the gap between provable security and practical security in post-quantum cryptography remains significant, and that closing it demands both improved countermeasures and a more careful understanding of what existing protections actually guarantee.



---

## Acknowledgements

Many people contributed to the journey of writing this Ph.D. thesis in ways both big and small, and I am happy to acknowledge and thank them here.

First and foremost, I am deeply grateful to my supervisor and co-supervisor, Qian and Thomas, for their gentle guidance, stimulating discussions and thoughtful comments that helped to steer the work in the right direction. Thank you for always being there and remaining ready to help, even though I was not always the best at reaching out when I should have.

My sincere thanks also go to my co-authors, without whom this thesis would simply not exist in its current form. Our collaborations were not only productive but genuinely fun, and I came away from each one having learned something new; among other things, I honed my communication skills and learned the art of working with others.

I am also grateful to my opponent and committee members for taking the time to read the thesis carefully and for providing helpful feedback. I am equally grateful to the anonymous reviewers of the submitted papers whose comments and questions made each paper better.

It has been a pleasure to be surrounded by such wonderful colleagues and seniors at the department. The lunch breaks, the occasional after-work activities, and the countless impromptu conversations in the corridor or the office made the day-to-day of Ph.D. life something I will look back on fondly. I wish to especially thank Vu, Dachao, and Maggie for the laughs and the endless stream of interesting discussions, technical and otherwise, as well as for being pleasant company during our travels together to conferences. There are many others who made the department such a warm place to work, including old Ph.D. students (at least, in my eyes): Syafiq, Jing, Alexander, Martin, Joakim, Pegah, Rohon; as well as newer additions: Karim, Shouran, Robert, Markus, Arthur, Haochen. And then there is Erik, who defies this categorization entirely. My appreciation also goes to the senior researchers, Paul, Elena, Christian, Sara, and Debo, for being excellent role models. Paul and Sara, your infectious enthusiasm for teaching made me a better teacher.

To my family and relatives, whom the events of recent years kept me from seeing nearly as much as I would have liked, I am so grateful for your love and support across the distance. I always appreciate our conversations, where over a call or the rare visit in person. To Mom and Dad, thank you for everything. Your encouragement and your belief in me laid the very road that eventually led to this thesis. To my brother Vanya and cousin Nastya, I always value your perspective, more than you might know.

To Mike, Nadya, Nastya, and Roma, thank you for being such wonderful friends and for giving me that irreplaceable feeling of home wherever we were. The trips, the laughter, and the warmth you brought made all the difference. To

Simon and Emil, thank you for that silly energy that helped me recharge during the busiest and hardest stretches of this journey.

There are many others not mentioned here who made this path easier, and who helped me grow, both personally and professionally. You know who you are, and I am grateful to each of you.

*Denis Nabokov*  
Lund, May 2026

---

## Contribution Statement

The following papers are included in this thesis:

- Paper I** Qian Guo, Denis Nabokov, Alexander Nilsson, Thomas Johansson. “SCA-LDPC: A Code-Based Framework for Key-Recovery Side-Channel Attacks on Post-Quantum Encryption Schemes”. In: *Advances in Cryptology – ASIACRYPT 2023*. Ed. by J. Guo and R. Steinfeld. Lecture Notes in Computer Science, vol 14441. Springer, Singapore, pp. 203-236
- Paper II** Qian Guo, Denis Nabokov, Elias Suvanto, Thomas Johansson. “Key Recovery Attacks on Approximate Homomorphic Encryption with Non-Worst-Case Noise Flooding Countermeasures”. In: *33rd USENIX Security Symposium (USENIX Security 24)*. USENIX Association, Philadelphia, PA, USA, 2024, pp. 7447-7461
- Paper III** Xiaofei Tong, Denis Nabokov, Qian Guo. “LDPC Strikes Again: New Key-Recovery Chosen-Ciphertext Side-Channel Attacks on NTRU-based KEMs”. Part of the work appears in: *IEEE 43rd International Conference on Computer Design (ICCD)*. Richardson, TX, USA, 2025, pp. 528-531
- Paper IV** Haiye Dong, Qian Guo, Denis Nabokov. “Single-Trace Key Recovery Attacks on HQC Using Valid and Invalid Ciphertexts”. To appear in: *Advances in Cryptology – EUROCRYPT 2026*.
- Paper V** Qian Guo, Denis Nabokov, Thomas Johansson. “Unlocking the True Potential of Decryption Failure Oracles: A Hybrid Adaptive-LDPC Attack on ML-KEM Using Imperfect Oracles”. To appear in: *35th USENIX Security Symposium (USENIX Security 26)*.

The table below summarizes the responsibilities Denis had in each paper:

| <i>Paper</i> | <i>Writing</i> | <i>Concepts</i> | <i>Implementation</i> | <i>Evaluation</i> |
|--------------|----------------|-----------------|-----------------------|-------------------|
| Paper I      | ●              | ◐               | ●                     | ●                 |
| Paper II     | ●              | ●               | ■                     | ●                 |
| Paper III    | ●              | ■               | ●                     | ■                 |
| Paper IV     | ●              | ○               | ●                     | ◐                 |
| Paper V      | ●              | ●               | ■                     | ■                 |

In the table, ■ indicates roles where Denis took primary or sole responsibility for the given role while ● indicates the responsibility was shared with one or more co-authors. In contrast, ◐ indicates contribution without taking responsibility for the role. In all papers, writing was split evenly between all the co-authors.

In Paper I, the concept was introduced by the senior co-authors, Denis' contribution to the concept was limited to his part of the paper. Denis took primary responsibility of one half of the implementation. Evaluation was shared between all co-authors.

In Paper II, Denis took primary responsibility for the implementation of the attack while the work on the concepts and evaluation was shared between co-authors.

In Paper III, the basic concept of applying already established framework from Paper I to slightly different scheme came from the senior co-author, Denis then took primary responsibility to develop concepts to make it work. The implementation was shared with another co-author, while Denis took primary responsibility to evaluate the results and change the approach accordingly.

In Paper IV, Denis was introduced to the project after the main idea was formulated, but was asked to help with implementation. Denis took primary responsibility of the physical part of the implementation. The evaluation of this part was shared.

In Paper V, the basic concept was introduced by the senior co-authors, but it did not work. Denis then proposed a slightly different concept that resulted in the final work and showed that it works in simulations. The idea for the application resulted in the attack on the physical device came from the senior co-authors. Denis took full responsibility for implementation and evaluation.

More comprehensive descriptions of each paper's contributions are available in Section 6.1.

# Contents

---

|                                                     |            |
|-----------------------------------------------------|------------|
| <b>Abstract</b>                                     | <b>iii</b> |
| <b>Acknowledgements</b>                             | <b>v</b>   |
| <b>Contribution Statement</b>                       | <b>vii</b> |
| <b>Contents</b>                                     | <b>ix</b>  |
| <br>                                                |            |
| <b>Overview of the Research Field</b>               | <b>1</b>   |
| <br>                                                |            |
| <b>1 Introduction</b>                               | <b>3</b>   |
| 1.1 Notations and Conventions . . . . .             | 5          |
| 1.2 Cryptographic Primitives . . . . .              | 5          |
| 1.3 Post-Quantum Cryptography . . . . .             | 13         |
| 1.4 Moving from Theory to Practice . . . . .        | 15         |
| 1.5 Outline . . . . .                               | 15         |
| <br>                                                |            |
| <b>2 Code-based Cryptography and Decoding</b>       | <b>17</b>  |
| 2.1 Error Correcting Codes . . . . .                | 17         |
| 2.2 Low-Density Parity-Check (LDPC) Codes . . . . . | 19         |
| 2.3 Hamming Quasi-Cyclic (HQC) . . . . .            | 23         |
| <br>                                                |            |
| <b>3 Lattice-based Cryptography</b>                 | <b>27</b>  |
| 3.1 Lattices and Hard Problems . . . . .            | 27         |
| 3.2 ML-KEM (Kyber) . . . . .                        | 28         |
| 3.3 NTRU and NTRU Prime . . . . .                   | 29         |

|           |                                                                                                                      |            |
|-----------|----------------------------------------------------------------------------------------------------------------------|------------|
| <b>4</b>  | <b>Side-Channel Analysis</b>                                                                                         | <b>33</b>  |
| 4.1       | Side-Channel Leakages . . . . .                                                                                      | 33         |
| 4.2       | Micro-architectural Leakages . . . . .                                                                               | 34         |
| 4.3       | Classification of Oracles . . . . .                                                                                  | 35         |
| 4.4       | The Key Recovery Challenge . . . . .                                                                                 | 36         |
| <b>5</b>  | <b>Security of Approximate Homomorphic Encryption</b>                                                                | <b>39</b>  |
| 5.1       | Approximate Homomorphic Encryption and CKKS . . . . .                                                                | 39         |
| 5.2       | Security Models for HE . . . . .                                                                                     | 41         |
| 5.3       | Countermeasures . . . . .                                                                                            | 43         |
| 5.4       | Noise Estimation Pitfalls . . . . .                                                                                  | 45         |
| <b>6</b>  | <b>Contributions and Conclusions</b>                                                                                 | <b>47</b>  |
| 6.1       | Contributions . . . . .                                                                                              | 47         |
| 6.2       | Topic relevance . . . . .                                                                                            | 55         |
| 6.3       | Lessons learned . . . . .                                                                                            | 55         |
| 6.4       | Looking forward . . . . .                                                                                            | 56         |
|           | <b>References</b>                                                                                                    | <b>59</b>  |
|           | <b>Included Publications</b>                                                                                         | <b>65</b>  |
| <b>I</b>  | <b>SCA-LDPC: A Code-Based Framework for Key-Recovery Side-Channel Attacks on Post-Quantum Encryption Schemes</b>     | <b>67</b>  |
| 1         | Introduction . . . . .                                                                                               | 68         |
| 2         | Preliminaries . . . . .                                                                                              | 74         |
| 3         | The SCA-LDPC Attack Framework . . . . .                                                                              | 79         |
| 4         | Application to Kyber . . . . .                                                                                       | 83         |
| 5         | Application to HQC . . . . .                                                                                         | 87         |
| 6         | Experiments . . . . .                                                                                                | 92         |
| 7         | Concluding Remarks and Future Work . . . . .                                                                         | 98         |
|           | References . . . . .                                                                                                 | 100        |
| <b>II</b> | <b>Key Recovery Attacks on Approximate Homomorphic Encryption with Non-Worst-Case Noise Flooding Countermeasures</b> | <b>105</b> |
| 1         | Introduction . . . . .                                                                                               | 106        |
| 2         | Preliminaries . . . . .                                                                                              | 109        |
| 3         | New key-recovery attacks . . . . .                                                                                   | 115        |
| 4         | On deterministic Gaussian noise for fresh ciphertext . . . . .                                                       | 126        |
| 5         | Concluding remarks . . . . .                                                                                         | 130        |
|           | References . . . . .                                                                                                 | 131        |

|            |                                                                                                                                    |            |
|------------|------------------------------------------------------------------------------------------------------------------------------------|------------|
| <b>III</b> | <b>LDPC Strikes Again: New Key-Recovery Chosen-Ciphertext Side-Channel Attacks on NTRU-based KEMs</b>                              | <b>135</b> |
| 1          | Introduction . . . . .                                                                                                             | 136        |
| 2          | Preliminary . . . . .                                                                                                              | 139        |
| 3          | KR-CCA-SCA-LDPC on NTRU . . . . .                                                                                                  | 146        |
| 4          | KR-CCA-SCA-LDPC on NTRU Prime . . . . .                                                                                            | 151        |
| 5          | Key-recovery Attack on a Masked NTRU Implementation . . .                                                                          | 158        |
| 6          | Discussion . . . . .                                                                                                               | 161        |
| 7          | Conclusion . . . . .                                                                                                               | 163        |
|            | References . . . . .                                                                                                               | 164        |
| <b>IV</b>  | <b>Single-Trace Key Recovery Attacks on HQC Using Valid and Invalid Ciphertexts</b>                                                | <b>169</b> |
| 1          | Introduction . . . . .                                                                                                             | 169        |
| 2          | Preliminaries . . . . .                                                                                                            | 174        |
| 3          | VC-KRA-MDPC: Single-Trace Key Recovery Attack with Valid Ciphertexts via MDPC Decoding . . . . .                                   | 177        |
| 4          | OT-FDA-CK: Chosen-Ciphertext Attack with Invalid Ciphertexts Using Combined Key Components . . . . .                               | 181        |
| 5          | Post-Processing with a New ISD Variant . . . . .                                                                                   | 185        |
| 6          | Validations . . . . .                                                                                                              | 191        |
| 7          | Discussion . . . . .                                                                                                               | 195        |
| 8          | Concluding Remarks . . . . .                                                                                                       | 197        |
|            | References . . . . .                                                                                                               | 198        |
|            | Appendix A Additional Figures . . . . .                                                                                            | 202        |
|            | Appendix B Sum-Product BP Algorithm . . . . .                                                                                      | 203        |
|            | Appendix C Broad Applicability of the VC-KRA-MDPC Attack . . .                                                                     | 205        |
| <b>V</b>   | <b>Unlocking the True Potential of Decryption Failure Oracles: A Hybrid Adaptive-LDPC Attack on ML-KEM Using Imperfect Oracles</b> | <b>207</b> |
| 1          | Introduction . . . . .                                                                                                             | 208        |
| 2          | Preliminaries . . . . .                                                                                                            | 213        |
| 3          | New Key-Recovery Approach for ML-KEM . . . . .                                                                                     | 216        |
| 4          | Application to GoFetch Attack on Kyber . . . . .                                                                                   | 228        |
| 5          | Discussions . . . . .                                                                                                              | 234        |
| 6          | Conclusion and Future Works . . . . .                                                                                              | 235        |
|            | References . . . . .                                                                                                               | 237        |
|            | Appendix A Extended Analysis of Encoding Strategies . . . . .                                                                      | 242        |



---

# Overview of the Research Field

---



# Introduction

---

The security of modern digital infrastructure rests on a compact set of mathematical assumptions. The public-key cryptography that underpins secure communication derives its hardness from the computational difficulty of factoring large integers and computing discrete logarithms. For decades, this foundation has proven robust: despite sustained cryptanalytic effort, no classical algorithm has been found that breaks these problems at security-relevant scales.

That foundation is, however, contingent on an assumption that is now under serious threat. In 1994, Shor demonstrated that a sufficiently large quantum computer could solve both the Integer Factorisation Problem and the Discrete Logarithm Problem in polynomial time [Sho94]. The practical consequence is stark: the existence of a *Cryptographically Relevant Quantum Computer* (CRQC) would render RSA, Diffie-Hellman, and Elliptic Curve Cryptography simultaneously and irreparably broken. While such machines do not yet exist, the trajectory of quantum hardware development has made their eventual arrival a concrete engineering question rather than a distant theoretical concern.

Critically, the threat does not require a CRQC to materialise before action is necessary. Encrypted traffic intercepted and stored today can be decrypted retroactively once a capable quantum computer becomes available, a strategy known as *harvest now, decrypt later*. This threat model imposes a hard deadline on the migration to a safer alternative: any data that must remain confidential for a decade or more is already at risk. The cryptographic community must therefore act well in advance, a planning horizon that is measured in years and is, by all accounts, already underway.

The primary response to this threat is *Post-Quantum Cryptography* (PQC): the design of classical asymmetric cryptosystems whose security rests on mathematical problems believed to be intractable for both classical and quantum computers. Unlike Quantum Key Distribution, which demands specialised hardware and remains constrained by range and implementation imperfections, PQC operates on existing hardware and slots into current protocol architectures with manageable engineering effort.

In December 2016, the National Institute of Standards and Technology (NIST) launched a formal standardization process [NIS16], soliciting post-quantum candidates from the global research community. Over subsequent years, dozens of submissions were subjected to intense public scrutiny spanning cryptanalysis, performance profiling, and implementation review. The process culminated in two primary standards for Key Encapsulation Mechanisms (KEMs): the lattice-based scheme ML-KEM (previously known as CRYSTALS-Kyber), standardized in August 2024 [Ava+19], and the code-based scheme Hamming Quasi-Cyclic (HQC), selected as the primary backup candidate in March 2025 [Gab+25], with its formal standard pending at the time of writing. The diversity of mathematical foundations is deliberate: it hedges against the possibility that a breakthrough in one family might otherwise compromise the entire ecosystem.

With standardization advancing, the focus has shifted from the question of which algorithms are theoretically secure to the question of how they can be deployed securely in practice. This transition reveals a second, qualitatively different class of vulnerabilities. A cryptographic scheme is specified as a mathematical object; its security proofs are statements about abstract algorithms operating on abstract data. A real implementation, however, must run on physical hardware, and physical hardware is not abstract. Execution time, power consumption, and electromagnetic radiation all vary as a function of the data being processed. These physical observables (also called *side channel* information) can leak information about secret state in ways that no security proof accounts for, because no security proof models the physics of computation.

An algorithm that is perfectly secure on paper may be trivially broken when the adversary can observe the time taken to process a single query, the power trace of a decapsulation call, or the electromagnetic emissions of an embedded device. This is not a hypothetical concern: timing attacks on implementations of classical schemes have been known since Kocher’s seminal 1996 work [Koc96a], and the PQC transition has, if anything, sharpened the problem. The new algorithms involve operations that are both structurally different from their classical counterparts and, in some cases, genuinely difficult to implement without data-dependent timing or power variation.

This thesis is situated precisely at this intersection. It captures the central tension: the algorithms examined here are all provably secure in the sense demanded by modern cryptographic standards, yet each admits concrete, practical attacks once the physical realities of implementation are taken into account. The contributions span key-recovery side-channel attacks on ML-KEM, HQC, NTRU, and NTRU Prime, as well as an investigation of security vulnerabilities in approximate homomorphic encryption. Together, they demonstrate that implementation security is not a secondary concern to be addressed after standardization, but an integral and ongoing component of the migration to post-quantum cryptography.

## 1.1 Notations and Conventions

Let  $\mathbb{Z}$ ,  $\mathbb{Q}$ ,  $\mathbb{R}$ , and  $\mathbb{C}$  represent the sets of integers, rationals, reals, and complex numbers, respectively. For integers  $a < b$ , the set of integers from  $a$  to  $b - 1$  is represented by  $[a..b]$ . For a finite set  $\mathcal{I}$ , the symbol  $\#\{\mathcal{I}\}$  denotes the number of elements in  $\mathcal{I}$ . For two functions  $f$  and  $g$ , we write  $f = O(g)$  if  $f$  is bounded above by  $g$  up to a constant factor asymptotically, and  $f = o(g)$  if  $f$  grows strictly slower than  $g$ , i.e.,  $f(n)/g(n) \rightarrow 0$  as  $n \rightarrow \infty$ . We write  $f = \omega(g)$  if  $g = o(f)$ , meaning  $f$  grows strictly faster than  $g$  asymptotically.

We denote matrices by bold uppercase letters (e.g.,  $\mathbf{A}$ ) and vectors by bold lowercase letters (e.g.,  $\mathbf{v}$ ). The Hamming weight of a vector  $\mathbf{v}$  is its number of non-zero elements, denoted by  $w_H(\mathbf{v})$ . For a vector  $\mathbf{v}$ ,  $\mathbf{v}[i]$  refers to the entry at index  $i$ .

The dot product of two vectors is written as  $\langle \mathbf{a}, \mathbf{b} \rangle$ . Let  $\lfloor \cdot \rfloor$  denote the operation of rounding to the nearest integer. The security parameter is represented by  $\lambda$ . A function  $f$  is deemed negligible in  $\lambda$  if  $f(\lambda) = \lambda^{-\omega(1)}$ .

We work with the ring of integers modulo  $q$ , denoted  $\mathbb{Z}_q$ . When  $q$  is odd, it is convenient to represent elements of  $\mathbb{Z}_q$  in the *centered* range  $\{-\lfloor q/2 \rfloor, \dots, \lfloor q/2 \rfloor\}$  rather than the standard  $\{0, \dots, q - 1\}$ ; this choice is purely a matter of convention but will simplify notation later. When  $q$  is prime,  $\mathbb{Z}_q$  is in fact a field, which we then write as  $\mathbb{F}_q$ . We also work with polynomial rings of the form  $\mathcal{R}_q = \mathbb{Z}_q[x]/(f(x))$ , where  $f(x)$  is a polynomial whose choice depends on the cryptographic scheme at hand, and where all arithmetic on coefficients is performed modulo  $q$ .

The notation  $x \xleftarrow{\$} \mathcal{S}$  indicates a uniform random sampling of an element  $x$  from a given set  $\mathcal{S}$ . When a value  $x$  is assigned according to some function or probability distribution  $\mathcal{D}$ , we use  $x \leftarrow \mathcal{D}$ .

## 1.2 Cryptographic Primitives

### 1.2.1 Symmetric and Asymmetric Cryptography

Symmetric cryptography constitutes the foundational layer of secure communication, enabling confidentiality between two parties, provided they share a pre-established secret key. In this setting, the sender utilizes the shared secret to transform a plaintext message into a ciphertext. Ideally, this ciphertext appears as a random string to any observer and reveals no meaningful information about the underlying message, apart from its length, to parties lacking the key. The legitimate recipient subsequently recovers the original message by applying the inverse decryption function using the same shared key.

However, the requirement of a pre-shared secret poses a significant logistical challenge, particularly in modern, large-scale networks where parties often communicate without prior contact. Public Key Cryptography (PKC), or asymmetric

cryptography, resolves this by employing a pair of mathematically related keys rather than a single shared secret.

**Public Key Encryption (PKE)** In a PKE scheme, users generate a key pair  $(pk, sk)$ . The *public key* ( $pk$ ) is shared openly, while the *private key* ( $sk$ ) is kept secret. To transmit a message, a sender encrypts it using the recipient’s public key. The resulting ciphertext can only be decrypted by the holder of the corresponding private key, effectively decoupling encryption capabilities from decryption capabilities.

Formally, a PKE scheme is defined by a triplet of algorithms  $(KeyGen, Enc, Dec)$ :

- $(pk, sk) \leftarrow KeyGen(1^\lambda)$ : A probabilistic algorithm that takes a security parameter  $\lambda$  and outputs a public key  $pk$  and a secret key  $sk$ .
- $c \leftarrow Enc(pk, m)$ : A probabilistic encryption algorithm that takes the public key  $pk$  and a message  $m$  from the message space  $\mathcal{M}$ , and outputs a ciphertext  $c$ .
- $m' \leftarrow Dec(sk, c)$ : A deterministic decryption algorithm that recovers the message  $m'$  from the ciphertext  $c$  using the secret key  $sk$ . Correctness requires that  $Dec(sk, Enc(pk, m)) = m$ .

The security of such schemes relies on the existence of *trapdoor one-way functions*. A one-way function is a mathematical function that is easy to compute in one direction but believed to be computationally infeasible to invert without specific auxiliary information. In the context of PKE, the public key  $pk$  defines a function that anyone can compute (encryption), while the private key  $sk$  serves as the “trapdoor” — a piece of auxiliary information that makes the inverse operation (decryption) feasible. Common examples of such hardness assumptions include the factorization of large integers (RSA), the discrete logarithm problem (Diffie-Hellman), and the elliptic curve discrete logarithm problem (ECC).

A fundamental challenge in asymmetric cryptography is the authentication of public keys. If an adversary intercepts the initial key exchange and substitutes their own public key for the recipient’s, they can successfully decrypt intercepted messages (a Man-in-the-Middle attack). This issue is typically addressed via Digital Signature schemes and a Public Key Infrastructure (PKI), which binds public keys to verified identities. As the scope of this thesis is primarily concerned with encryption and key encapsulation mechanisms, we assume the existence of an authenticated channel for public key distribution and do not discuss digital signatures further.

**Key Encapsulation Mechanisms (KEMs)** While PKE solves the key distribution problem, it is generally computationally intensive and inefficient for encrypting large volumes of data compared to symmetric primitives. Consequently, modern cryptographic protocols adopt a hybrid<sup>1</sup> approach known as the KEM/DEM con-

<sup>1</sup>Not to be confused with a hybrid classical and post-quantum scheme.

struction. In this paradigm, a Key Encapsulation Mechanism (KEM) uses asymmetric techniques to securely establish a short, shared symmetric key. This key is then handed off to a Data Encapsulation Mechanism (DEM), which is typically represented by a symmetric block or stream cipher. DEM then encrypts the actual message payload efficiently.

Formally, a KEM is defined by a triplet of algorithms (KeyGen, Encaps, Decaps):

- $(pk, sk) \leftarrow \text{KeyGen}()$ : A probabilistic algorithm that outputs a public key  $pk$  and a secret key  $sk$ .
- $(c, K) \leftarrow \text{Encaps}(pk)$ : A probabilistic algorithm that generates a random key  $K \xleftarrow{\$} \mathcal{K}$  and its encapsulation (ciphertext)  $c$ . The set  $\mathcal{K}$  denotes the key space of the shared secret.
- $K \leftarrow \text{Decaps}(sk, c)$ : A deterministic algorithm that recovers the shared secret  $K$  from the encapsulation  $c$  using the secret key  $sk$ .

It is worth noting the distinct difference between PKE and KEM interfaces. In a standard PKE scheme, the algorithm  $c \leftarrow \text{Enc}(pk, m)$  takes a specific message  $m$  as an explicit input. In contrast, a KEM implicitly generates the shared secret  $K$  randomly alongside the ciphertext during the encapsulation process. This random generation is central to the security of KEMs, as the resulting key  $K$  is used for subsequent symmetric encryption.

**Homomorphic Encryption** The primitives discussed thus far primarily address the confidentiality of data during transmission. Cryptographic research has also expanded to address the processing of data while it remains encrypted. Homomorphic Encryption (HE) is a primitive that allows computation on ciphertexts directly. Unlike PKE or KEMs, where the goal is solely secure transport, HE enables the evaluation of a function  $f$  over encrypted inputs such that decrypting the result yields  $f$  applied to the original plaintexts. This capability allows for secure offloading of computation to untrusted servers without compromising the privacy of the input data. Depending on the class of functions that can be evaluated, one distinguishes between *partially* homomorphic schemes, which support only a limited type of operation, *somewhat* homomorphic schemes, which support a bounded number of mixed operations, and *fully* homomorphic schemes (FHE), which support arbitrary computation. Chapter II covers a practically important instance called *approximate* homomorphic encryption and its security properties.

### 1.2.2 Security Models

Having established the fundamental goals of cryptography, such as confidentiality, we now turn to rigorous definitions. Ideally, an encryption scheme would achieve

*information-theoretic security*, revealing absolutely no information about the plaintext other than its length. Such perfect secrecy comes at a steep price: the secret key must be at least as long as the message and used only once making it impractical for most applications, a typical example of such encryption is the one-time pad. In practice, we therefore rely on *computational security*, which quantifies the infeasibility for a resource-constrained adversary to learn meaningful information about the plaintext.

The standard notion for confidentiality in public-key encryption is *Indistinguishability under Chosen Plaintext Attack* (IND-CPA). This model formalizes the security requirements by treating the scenario as a game played between a challenger and an adversary.

**IND-CPA Security** Consider a PKE scheme  $\Pi = (\text{KeyGen}, \text{Enc}, \text{Dec})$ . The IND-CPA security game proceeds as follows:

1. **Setup:** The Challenger generates a key pair  $(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$  and sends the public key  $pk$  to the Adversary  $\mathcal{A}$ .
2. **Query Phase 1:** The Adversary  $\mathcal{A}$  is granted access to an encryption oracle  $\mathcal{O}_{\text{Enc}}(\cdot)$ , which, upon input  $m$ , returns  $\text{Enc}(pk, m)$ . While  $\mathcal{A}$  holds  $pk$  and can ostensibly perform encryption locally, the oracle formalism strictly defines the “chosen plaintext” capability. The adversary may perform any number of computations and/or oracle queries, provided the total work remains polynomial in  $\lambda$ .
3. **Challenge:** The Adversary  $\mathcal{A}$  outputs two distinct messages of equal length,  $m_0$  and  $m_1$ , from the message space  $\mathcal{M}$ . The Challenger selects a bit  $b \xleftarrow{\$} \{0, 1\}$  uniformly at random, computes the challenge ciphertext  $c^* \leftarrow \text{Enc}(pk, m_b)$ , and sends  $c^*$  to  $\mathcal{A}$ .
4. **Query Phase 2:** The Adversary  $\mathcal{A}$  continues to perform computations and to access  $\mathcal{O}_{\text{Enc}}(\cdot)$ .
5. **Guess:** The Adversary  $\mathcal{A}$  outputs a guess  $b' \in \{0, 1\}$ .

The Adversary wins the game if  $b' = b$ . The advantage of the adversary in breaking the IND-CPA security of the scheme  $\Pi$  is defined as the distance between the adversary’s success probability and random guessing:

$$\text{Adv}_{\Pi}^{\text{IND-CPA}}(\mathcal{A}) = \left| \Pr[b' = b] - \frac{1}{2} \right|. \quad (1.1)$$

A scheme is considered IND-CPA secure if this advantage is negligible in the security parameter  $\lambda$ . Intuitively, this means that a computationally bounded adversary performs no better than a random guess when attempting to distinguish

between the encryptions of two messages of their own choosing. The fact that  $\mathcal{A}$  selects both  $m_0$  and  $m_1$  is deliberate: by allowing the adversary to pick the most favorable pair of messages, the definition provides a worst-case security guarantee that captures all possible relationships between plaintexts. This formal definition captures the essence of *semantic security*, ensuring that the ciphertext leaks no partial information about the plaintext beyond its length—not even a single bit, such as the least significant one. Consequently, observing a ciphertext provides the adversary with no new knowledge about the underlying message that they did not already possess prior to interception.

**The Necessity of Randomization** A critical requirement for IND-CPA security is that the encryption algorithm  $\text{Enc}$  must be probabilistic. If  $\text{Enc}$  were deterministic, an adversary could trivially win the game by computing  $c' = \text{Enc}(\text{pk}, m_0)$  (using  $\text{pk}$  or the oracle) and comparing it to the challenge ciphertext  $c^*$ . If  $c' = c^*$ , the adversary knows  $b = 0$ ; otherwise,  $b = 1$ . Therefore, a secure encryption scheme must introduce randomness during the encryption process, ensuring that encrypting the same message multiple times yields different ciphertexts with overwhelming probability.

**Beyond IND-CPA: Active Attacks** While IND-CPA captures security against passive eavesdroppers, it is often insufficient in real-world scenarios where an adversary can do more than simply observe ciphertexts. Consider an attacker who is able to submit carefully crafted ciphertexts to a legitimate party and observe the outcome—whether a decryption succeeds, fails, or produces a particular error. Such *chosen ciphertext* capabilities allow an adversary to extract information far beyond what passive eavesdropping permits, and IND-CPA provides no protection against them. This motivates the need for a stronger security model: *Indistinguishability under Chosen Ciphertext Attack* (IND-CCA).

**IND-CCA Security** The IND-CCA game extends the IND-CPA game by granting the adversary access to a decryption oracle  $\mathcal{O}_{\text{Dec}}$ . This oracle accepts a ciphertext  $c$  and returns the corresponding plaintext  $m \leftarrow \text{Dec}(\text{sk}, c)$ .

1. **Setup:**  $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(1^\lambda)$ . The Challenger sends  $\text{pk}$  to  $\mathcal{A}$ .
2. **Query Phase 1:**  $\mathcal{A}$  has access to the decryption oracle  $\mathcal{O}_{\text{Dec}}(\cdot)$ .  $\mathcal{A}$  may query any ciphertext  $c$ .
3. **Challenge:**  $\mathcal{A}$  submits distinct  $m_0, m_1$ . The Challenger computes  $c^* \leftarrow \text{Enc}(\text{pk}, m_b)$  for  $b \xleftarrow{\$} \{0, 1\}$  and returns  $c^*$  to  $\mathcal{A}$ .
4. **Query Phase 2:**  $\mathcal{A}$  continues to have access to  $\mathcal{O}_{\text{Dec}}(\cdot)$ , with the strict restriction that  $\mathcal{A}$  may not query the challenge ciphertext  $c^*$  itself.

5. **Guess:**  $\mathcal{A}$  outputs a guess  $b'$ .

The advantage is calculated identically to the IND-CPA setting:

$$\text{Adv}_{\Pi}^{\text{IND-CCA}}(\mathcal{A}) = \left| \Pr[b' = b] - \frac{1}{2} \right|. \quad (1.2)$$

If a scheme is IND-CCA secure, it implies immunity to attacks where the adversary can manipulate ciphertexts or observe decryption results of related ciphertexts. There are in fact two variants of this notion. In *IND-CCA1* (also called non-adaptive chosen ciphertext security), the adversary may query the decryption oracle only before seeing the challenge ciphertext  $c^*$ . In the stronger *IND-CCA2* notion (adaptive chosen ciphertext security), the adversary retains access to the decryption oracle even after receiving  $c^*$ , subject solely to the restriction that  $c^*$  itself is not submitted. IND-CCA2 is the standard notion used in modern cryptography, and throughout this thesis we use IND-CCA to refer to IND-CCA2 unless explicitly stated otherwise.

**Malleability** Closely related to IND-CCA security is the concept of *malleability*. An encryption scheme is malleable if, given a ciphertext  $c$  corresponding to a plaintext  $m$ , it is possible to generate a different ciphertext  $c'$  that decrypts to a related plaintext  $f(m)$  for some known function  $f$ , without learning  $m$  or the secret key  $sk$ . In the context of PKE, this property is generally undesirable as it compromises integrity; an adversary could potentially modify a message without detection.

It is a standard result that IND-CCA security implies non-malleability [Bel+98]: intuitively, if an adversary could produce a meaningful transformation of the challenge ciphertext, they could submit it to the decryption oracle and exploit the response to break ciphertext indistinguishability.

An important exception to the requirement for non-malleability is Homomorphic Encryption which is malleable by definition and intention: the goal is to allow computations on ciphertexts (e.g., addition or multiplication) that correspond to operations on the underlying plaintexts. Consequently, HE schemes cannot achieve IND-CCA security. They are typically evaluated under the IND-CPA model, as the ability to modify ciphertexts is a feature, not a vulnerability, in that specific context.

**Security for Key Encapsulation Mechanisms** As modern protocols often utilize a KEM/DEM paradigm, we must adapt the IND-CCA definition for KEMs. The primary difference lies in the challenge phase. In a KEM, the encapsulation algorithm  $\text{Encaps}$  implicitly generates the shared secret.

In the IND-CCA game for a KEM, instead of choosing between two plaintexts, the adversary receives a pair  $(K^*, c^*)$ . The challenger generates this pair such that  $c^*$  is the encapsulation of a key  $K$ . With probability  $1/2$ ,  $K^*$  is the real key

$K$ ; with probability  $1/2$ ,  $K^* \xleftarrow{\$} \mathcal{K}$  is a random key drawn uniformly from the key space. The adversary's goal is to distinguish whether  $K^*$  is the real shared secret or a uniformly random string, and the advantage is defined similarly to the PKE setting. A KEM is IND-CCA secure if no probabilistic polynomial-time (PPT) adversary can distinguish the two cases with advantage non-negligible in  $\lambda$ —that is, the adversary does no better than a random guess.

**From IND-CPA to IND-CCA: The Fujisaki-Okamoto Transform** It is common practice to construct IND-CCA secure KEMs from IND-CPA secure PKE schemes using the Fujisaki-Okamoto (FO) transform [FO99] (or variants thereof). This transformation is ubiquitous in modern KEMs, including those discussed later in this thesis.

The FO transform takes a randomized IND-CPA secure PKE scheme and produces an IND-CCA secure KEM. The key idea is to make encryption *deterministic* by deriving the random coins from the message itself—typically via a hash function, often also binding in the public key—rather than sampling them uniformly at random. This allows the decapsulation procedure to re-encrypt the decrypted message using the same derived coins and verify that the result matches the received ciphertext. Any tampering with or malformation of the ciphertext will cause this check to fail, providing the ciphertext integrity that IND-CCA security requires. The core mechanism of the FO transform is as follows:

1. The recipient decrypts the ciphertext  $c$  to obtain a candidate message  $m' \leftarrow \text{Dec}(\text{sk}, c)$ .
2. The recipient re-encrypts  $m'$  using the deterministic encryption function (re-generating the specific random coins from  $m'$ ) to obtain a candidate ciphertext  $c' \leftarrow \text{Enc}(\text{pk}, m')$ .
3. The recipient compares  $c \stackrel{?}{=} c'$ .

If  $c' = c$ , the ciphertext was generated correctly, and the shared secret is derived as intended. If  $c' \neq c$ , the ciphertext is considered invalid—either malformed or tampered with. In this case, the scheme performs an *implicit rejection*: rather than returning an explicit error, it returns a pseudorandom value derived deterministically from the session transcript (typically involving the ciphertext). Crucially, this pseudorandom value is indistinguishable from a legitimate shared secret, so the adversary receives no useful signal about why decapsulation failed. This design ensures that every ciphertext the adversary can submit to the decryption oracle is, in effect, one they could have produced themselves via a valid encryption call—the decryption oracle yields no additional leverage. Consequently, the adversary's capability is reduced to that of a CPA attacker, which is precisely what the underlying IND-CPA secure PKE scheme is designed to resist.

**The Quantum Threat** Classical computers process information as *bits*, each taking a value of either 0 or 1. A quantum computer, by contrast, operates on *qubits*, which exploit the principles of quantum mechanics to exist in a *superposition* of 0 and 1 simultaneously. When multiple qubits are combined, their joint state can encode an exponentially large number of configurations at once. Through carefully designed quantum algorithms, this parallelism can be harnessed to solve certain computational problems dramatically faster than any classical machine.

A second key resource is *quantum entanglement*, a phenomenon in which the states of two or more qubits become correlated in ways that have no classical analogue. Together, superposition and entanglement allow quantum algorithms to constructively amplify probability amplitudes leading to correct answers while destructively cancelling those leading to wrong ones. This interference-based computation is not universally faster than classical computation, but for a specific class of structured problems, the speedup is profound.

In 1994, Shor published his seminal work demonstrating that a sufficiently large quantum computer could efficiently solve the mathematical problems that serve as the centerpiece of currently deployed asymmetric cryptography [Sho94]. The security of the most ubiquitous public-key encryption and digital signature schemes relies on the assumed intractability of two specific problems: the Integer Factorization Problem (IFP), which underpins RSA, and the Discrete Logarithm Problem (DLP), which underpins Diffie-Hellman and Elliptic Curve Cryptography (ECC).

Shor's algorithm solves these problems in polynomial time on a quantum computer, directly undermining the security of RSA, Diffie-Hellman, and ECC. Concretely, given an  $n$ -bit integer  $N$ , the algorithm recovers its prime factors in time polynomial in  $n$ ; the same algorithmic ideas, applied to the discrete logarithm problem, equally threaten Diffie-Hellman and elliptic curve based schemes. The existence of a *Cryptographically Relevant Quantum Computer* (CRQC)—a quantum machine of sufficient scale and reliability to run Shor's algorithm against real-world key sizes—would therefore render essentially all currently deployed asymmetric cryptography insecure.

**Grover's Algorithm** While Shor's algorithm dismantles the security of widely used asymmetric primitives, Grover's algorithm represents the primary quantum threat to symmetric cryptography, such as block ciphers and hash functions. Formally, Grover's algorithm is designed to search for a specific entry in an unsorted database of  $N$  elements. In the context of cryptography, this equates to a brute-force search for a secret key within a key space of size  $N = 2^\lambda$ , where  $\lambda$  is the security parameter (key length in bits).

Unlike classical brute-force attacks, which require  $O(N)$  operations on average, Grover's algorithm provides a quadratic speedup, finding the target in  $O(\sqrt{N})$  quantum operations. For a block cipher like AES-128, this reduces the effective security level from 128 to 64 bits: whereas a classical adversary must perform on the

order of  $2^{128}$  operations to exhaust the key space, a quantum adversary running Grover's algorithm requires only around  $2^{64}$ —a reduction that brings the attack within a far more conceivable computational budget. The conventional mitigation is straightforward: doubling the key length. AES-256, for instance, retains approximately 128 bits of security against a quantum adversary, which is generally considered sufficient.

**Practical Impact on Symmetric Cryptography** Despite the theoretical efficacy of Grover's algorithm, there is a growing consensus that the threat to symmetric cryptography is severely overstated in practice. Unlike Shor's algorithm, which yields an exponential speedup sufficient to break asymmetric primitives outright, Grover's quadratic speedup is subject to physical and algorithmic constraints that render it effectively negligible for well-chosen key sizes.

Three compounding factors drive this conclusion. First, quantum error correction requires each logical qubit to be encoded in thousands of physical qubits, substantially inflating hardware requirements. Second, Grover's algorithm parallelises badly: splitting the search across  $P$  machines reduces runtime by only  $\sqrt{P}$ , while total operation count grows, making large-scale parallel attacks far more expensive than the naive  $2^{64}$  figure suggests. Third, and most concretely, when accounting for realistic circuit depths and error-correction overhead, the number of physical qubits required to attack AES-128 would, at 2 microns per qubit, cover the surface of the Moon [Jaq24], and the energy cost is bounded below by the annual output of a nuclear power plant.

Consequently, while the theoretical security reduction from  $2^{128}$  to  $2^{64}$  is real, the practical threat to AES-128 is still far from being urgent. The quantum crisis is therefore immediate and existential only for asymmetric cryptography, where Shor's algorithm offers a genuine exponential speedup.

## 1.3 Post-Quantum Cryptography

Post-Quantum Cryptography constitutes the definitive response to the existential threat posed by Cryptographically Relevant Quantum Computers. The primary objective of PQC is the design and rigorous analysis of modern asymmetric schemes capable of resisting cryptanalysis by adversaries possessing both classical and quantum computational capabilities. An alternative approach, Quantum Key Distribution (QKD), attempts to address the problem at the physical layer by using quantum mechanical properties to establish shared secrets. While theoretically appealing, QKD requires specialised hardware infrastructure, is limited in range, and in practice falls short of the security guarantees due to technical limitations, e.g., the description of the algorithm assumes only one photon with special polarization is sent, but currently we do not know how to achieve that; intercepting the extra photon(s) breaks the perfect theoretical security. PQC, by contrast, is

grounded in mathematical hardness assumptions, runs on existing classical hardware, and is therefore the practical and scalable path forward.

To ensure long-term security, the cryptographic community has explored diverse mathematical foundations. This approach mitigates the risk that a breakthrough in one specific area might compromise the entire ecosystem. The principal families of hard problems currently under active investigation include the following:

- **Code-based cryptography**, which relies on the hardness of decoding random linear codes;
- **Lattice-based cryptography**, founded on problems such as Learning With Errors (LWE) and Module-LWE;
- **Hash-based cryptography**, primarily used for digital signatures;
- **Multivariate cryptography**, based on solving systems of multivariate polynomial equations;
- **Isogeny-based cryptography**, which utilizes the structure of supersingular elliptic curve isogeny graphs.

In December 2016, the National Institute of Standards and Technology (NIST) formally launched a public call for proposals, effectively kickstarting the transition to a post-quantum world. This initiative mobilized the global cryptographic community. Through multiple rounds of elimination, dozens of initial submissions were subjected to relentless cryptanalysis and performance profiling, leaving only the most resilient algorithms as finalists.

Regarding KEMs, the standardization effort has yielded two primary finalists to date. As established, NIST has formally standardized the lattice-based ML-KEM and selected the code-based HQC as the primary backup candidate to provide necessary mathematical diversity. As of early 2026, at the time of writing this thesis, the formal standard for HQC has not yet been released by NIST. While the NIST PQC project also encompasses digital signature schemes, this thesis restricts its scope to key encapsulation mechanisms. Consequently, we do not discuss digital signatures further.

In addition to the standardized finalists, several other strong KEM schemes remain relevant to the research presented in this thesis. Specifically, we analyze the lattice-based schemes NTRU and NTRU Prime, both of which were strong contenders in the third round of the NIST competition. These schemes offer distinct algebraic structures that provide valuable insights when examining side-channel leakage and implementation security.

## 1.4 Moving from Theory to Practice

With the formal standardization of ML-KEM and the upcoming release of the HQC standard, the cryptographic landscape is shifting from theoretical exploration to industrial deployment. While the cryptanalysis of the underlying hardness assumptions remains a critical and ongoing work, the research focus has inevitably shifted to embrace the challenges of engineering. At this point, the community operates with a carefully measured confidence that the selected algorithms offer adequate theoretical security.

However, transitioning these mathematical constructs into production-grade software introduces a distinct class of vulnerabilities. The first hurdle is implementation correctness; while software bugs are inevitable, they are generally detectable and reparable through rigorous testing and community review. Far more insidious, however, is the threat of Side-Channel Analysis (SCA), which forms the core focus of Chapter 4 and this thesis. Unlike logical bugs, SCA exploits the physical reality that cryptographic algorithms must eventually execute on physical hardware.

When a cryptographic primitive runs on a device, it unwittingly broadcasts information through physical observables. Variations in execution time, dynamic power consumption, and electromagnetic radiation can leak correlations to the internal state of the processor. This leakage provides an adversary with auxiliary information that is not accounted for in standard security models (such as IND-CCA). Consequently, a scheme that is mathematically perfect in the abstract may be trivially insecure in a real-world scenario. To bridge this gap, implementations must be fortified with specific countermeasures that go beyond the theoretical specification of the protocol.

## 1.5 Outline

This thesis is organized as follows. Part I contains the “Kappa”, Chapters 2 to 6. Part II contains the included publications, referenced as papers I to V. Finally Part III contains the Popular Science Summary.

The “Kappa” starts from Chapter 1, where we are now. Chapter 2 delves into code-based cryptography, Chapter 3 follows with lattice-based cryptography. Chapter 4 finally gets close to the topic of this thesis and presents the formulation of the main problem. Chapter 5 is then covers another relevant part of the practical security connected to homomorphic encryption. Chapter 6 is the final chapter and summarizes the entire project and the included publications.

**General Thesis Focus** Chapters 2 to 5 will give a discussion of the background information necessary to comprehend this thesis in a meaningful way. The idea is that these chapters will make the included publications easier to follow, compared to reading the individual papers independently. This means that mathematical strictness is intentionally sacrificed in favor of readability, and in-depth discussions

are sometimes sacrificed in favor of putting the work in a larger context. It is my hope that this initial intent survives contact with the reality that is your experience of the printed/electronic book in front of you.

# Code-based Cryptography and Decoding

---

## 2.1 Error Correcting Codes

Historically, coding theory was developed to ensure reliable communication over noisy channels. Consider a scenario where multiple Wi-Fi networks broadcast in the same physical area. The overlapping signals can cause physical interference, leading to corrupted or entirely dropped bits when a message is received over the air.

The fundamental mechanism to resolve this unreliability is the introduction of redundant information. Rather than transmitting a raw  $k$ -bit message directly, we encode it into an  $n$ -bit string, where  $n > k$ . This longer string is called a codeword. The ratio  $R = k/n$  is known as the code rate, representing the proportion of actual information transmitted per symbol.

In practice, we typically deal with linear codes. For a message vector  $\mathbf{m} \in \mathbb{F}_2^k$ , the corresponding codeword  $\mathbf{c} \in \mathbb{F}_2^n$  is generated using a *generator matrix*  $\mathbf{G} \in \mathbb{F}_2^{k \times n}$  via the matrix multiplication:

$$\mathbf{c} = \mathbf{m}\mathbf{G}. \quad (2.1)$$

A defining feature of any such code is its minimum distance  $d$ , which is the smallest Hamming distance between any two valid codewords in the code space. The distance dictates the robustness of the code. Specifically, a code can guarantee the correction of up to  $t = \lfloor (d - 1)/2 \rfloor$  errors.

The optimal choice of a code depends heavily on the physical characteristics of the communication medium. For the purposes of this thesis, our primary model is the Binary Symmetric Channel (BSC). The BSC is a memory-less channel characterized by a single crossover probability  $\rho$ . For every transmitted bit, regardless of its position, the channel flips its value with probability  $\rho$  and leaves it intact

with probability  $1 - \rho$ . Consequently, the only source of noise in this model is this independent bit-flipping behavior.

Because the code is linear, the sum of any two valid codewords yields another valid codeword. Furthermore, the  $k$ -dimensional subspace of valid codewords can be alternatively defined by its null space using a *parity-check matrix*  $\mathbf{H} \in \mathbb{F}_2^{r \times n}$ , where  $r = n - k$ . For every valid codeword  $\mathbf{c}$ , the following condition holds:

$$\mathbf{c}\mathbf{H}^T = \mathbf{0}. \quad (2.2)$$

When a codeword is transmitted over a noisy channel, the receiver obtains a noisy vector  $\mathbf{v} = \mathbf{c} + \mathbf{e}$ , where  $\mathbf{e}$  represents the error vector introduced by the channel. To detect if errors occurred, the receiver calculates the syndrome  $\mathbf{s}$  of the received vector:

$$\mathbf{s} = \mathbf{v}\mathbf{H}^T = (\mathbf{c} + \mathbf{e})\mathbf{H}^T = \mathbf{e}\mathbf{H}^T. \quad (2.3)$$

If  $\mathbf{s} = \mathbf{0}$ , the received vector is assumed to be a valid codeword. A non-zero syndrome directly points to unsatisfied parity checks, indicating the presence of errors.

This brings us to the core computational challenge: the Decoding Problem. Given a received noisy codeword  $\mathbf{v}$  and the code parameters, the goal is to find the closest valid codeword  $\mathbf{c}$ . Note that finding  $\mathbf{c}$  is strictly equivalent to finding the exact noise vector  $\mathbf{e}$  and removing it.

An equivalent formulation is the Syndrome Decoding Problem. Here, the task is to find a minimal-weight error vector  $\mathbf{e}$  that satisfies  $\mathbf{e}\mathbf{H}^T = \mathbf{s}$  for a given syndrome  $\mathbf{s}$ . These two problems are functionally identical, as one can efficiently convert a generator matrix  $\mathbf{G}$  into a parity-check matrix  $\mathbf{H}$  and vice versa.

The hardness of these problems is what makes code-based cryptography viable. For large, randomly generated linear codes, both the Decoding and Syndrome Decoding problems are NP-hard [BMV78]. While general-purpose algorithms like Information Set Decoding (ISD) exist to solve them, their runtime scales exponentially even on a quantum computer, with Grover's algorithm yielding at most a quadratic speedup that is manageable by choosing sufficiently large parameters. This quantum resistance is one of the most attractive features of code-based cryptography. Furthermore, the field has benefited from nearly half a century of intense cryptanalysis. This extensive scrutiny makes code-based cryptography one of the oldest, most mature, and best-understood candidates for post-quantum security.

The central design idea is to exploit the gap between *structured* and *random* codes. Efficient polynomial-time decoding algorithms exist, but only for codes with a very specific algebraic structure. A code-based cryptosystem is therefore built around a deliberate asymmetry: the secret key  $\mathbf{sk}$  encodes a structured code together with its efficient decoding algorithm, while the public key  $\mathbf{pk}$  is a carefully scrambled representation of the same code that appears random to an outside observer. Legitimate users who know  $\mathbf{sk}$  can decode in polynomial time; an ad-

versary holding only  $pk$  is left with what looks like a random linear code and must therefore face the full exponential hardness of the Decoding Problem.

**The Duality of Coding: Source Coding** Coding theory is most famously associated with channel coding, specifically, by adding redundancy to protect against noise, but it has an equally important dual: *source coding*, or data compression. Where channel coding adds structure to a message, source coding removes it. If a data source exhibits statistical regularity, its output carries more redundancy than entropy, and can therefore be represented using fewer bits than a naive encoding would require. Shannon’s source coding theorem formalises this precisely, establishing the entropy of the source as the fundamental lower bound on the average number of bits needed to represent it.

This duality turns out to be a remarkably powerful lens in the context of side-channel analysis. Physical devices leak information about their internal state through observable side channels such as power consumption or electromagnetic radiation. Rather than treating each measurement as a hard, binary decision about a secret variable, one can model the leakage as a noisy communication channel and collect *soft* probabilistic information across multiple observations. As we explore in Paper I, this reframes key recovery as a joint source coding and error correction problem: by combining soft leakage information with the algebraic structure of the secret, advanced decoding algorithms can reconstruct the secret key from far fewer measurements than classical hard-decision attacks require.

## 2.2 Low-Density Parity-Check (LDPC) Codes

As established previously, a linear code can be represented by either a generator matrix or a parity-check matrix. Here, we focus on the latter. To build an intuition, consider the following parity-check matrix  $\mathbf{H} \in \mathbb{F}_2^{3 \times 6}$

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \quad (2.4)$$

Note that a row of a parity-check matrix  $\mathbf{H}$  defines a subset of all codeword symbols, and by the construction, the sum of this subset has to be zero for a noise-free codeword. For instance, reading the first row of  $\mathbf{H}$ , any valid codeword must satisfy the following constraint

$$c_0 + c_1 + c_3 + c_5 = 0.$$

We could represent parity-check matrix in a more intuitive geometric perspective. A Tanner graph, illustrated in Figure 2.1, is a bipartite graph that visually rep-

resents the parity-check matrix  $\mathbf{H}$  from Equation (2.4). It consists of two distinct sets of vertices:

- **Variable nodes** represent the individual symbols of the codeword, corresponding to the columns of  $\mathbf{H}$ .
- **Check nodes** represent the parity-check equations, corresponding to the rows of  $\mathbf{H}$ .

An edge connects a variable node to a check node if and only if that specific variable is involved in that specific parity-check equation.

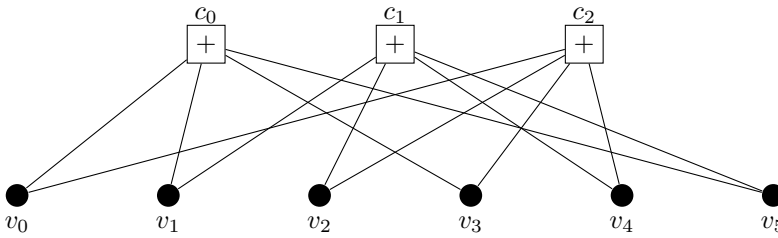


Figure 2.1: Tanner graph corresponding to the parity-check matrix  $\mathbf{H}$ . Variable nodes ( $v_i$ ) are connected to check nodes ( $c_j$ ) based on the non-zero entries of the matrix.

This thesis heavily relies on Low-Density Parity-Check (LDPC) codes, first introduced by Gallager [Gal62]. An LDPC code is fundamentally defined by its parity-check matrix. The defining characteristic of this matrix is that it is sparse, i.e., it contains a small fraction of ones. It's where the “low-density” part of the name comes from. If the number of ones (the Hamming weight) is strictly constant across all rows and columns of the matrix, the code is classified as regular, such matrix is given in Equation (2.4) as regular matrix with row weight 4 and column weight 2. If these weights vary, the code is irregular.

There exists an efficient decoder for LDPC codes. The standard approach relies on a family of message-passing iterative algorithms known collectively as Belief Propagation (BP). The core intuition is as follows: nodes pass information back and forth along the edges of the Tanner graph, continuously updating their local “beliefs” about the true value of the variable nodes based on the constraints of the check nodes. Gallager's original proposal included Bit-Flip decoding algorithm.

1. Compute the syndrome  $\mathbf{s}$  according to Equation (2.3). If  $\mathbf{s}$  is the zero vector, all parity checks are satisfied, and the algorithm outputs the valid codeword.
2. For each variable node, count the number of its connected parity-check equations that are currently failing.

3. Flip every variable node for which this count exceeds a threshold, which is typically a majority of its connections. The rationale is that a bit involved in many failing checks is more likely to be the erroneous one than the checks themselves.
4. Repeat the process until the syndrome is zero (a successful decode) or a predefined maximum number of iterations is reached (a decoding failure).

**Belief Propagation with Soft Information** To improve upon Bit-Flip decoding, we abandon hard decisions entirely and instead track, for each variable node  $v$ , a probability mass function  $P_v(x)$  over the alphabet  $\mathcal{A}$  of possible symbol values and use an iterative message-passing algorithm known as the *Sum-Product* algorithm. Two families of messages flow along each edge of the Tanner graph:  $P_{v \rightarrow c}(x)$  carries the belief of variable node  $v$  about its own value  $x \in \mathcal{A}$ , given its channel observation and all constraints *except*  $c$ ; and  $P_{c \rightarrow v}(x)$  carries the extrinsic evidence that check node  $c$  provides about  $v$ , based on all other variable nodes connected to  $c$ .

**Step 1: Initialization.** For every variable node  $v$  and each adjacent check node  $c$ , outgoing messages are seeded from the channel observation:

$$P_{v \rightarrow c}^{(0)}(x) = P_{\text{ch}}(x).$$

**Step 2: Iterative update.** For iterations  $t = 1, 2, \dots$ , the following two updates are applied.

*Check node update.* Check node  $c$  collects the incoming messages from all neighboring variables except  $v$ , i.e., considering  $\mathcal{N}(c)$  to be the set of variable nodes connected to  $c$ , we iterate over  $\mathcal{N}(c) \setminus \{v\}$ . Fixing  $v = x$ , the algorithm takes all possible assignments  $(x_{v'})_{v' \in \mathcal{N}(c) \setminus \{v\}}$  to the other variables leading to the satisfied parity check. The general idea is to answer the question: “*given what all my other neighbors believe, how likely is it that you take the value  $x$ , so that the parity constraint is satisfied?*” Thus, we have:

$$P_{c \rightarrow v}^{(t)}(x) \propto \sum_{\text{valid } (x_{v'})_{v' \in \mathcal{N}(c) \setminus \{v\}}} \prod_{v' \in \mathcal{N}(c) \setminus \{v\}} P_{v' \rightarrow c}^{(t-1)}(x_{v'}). \quad (2.5)$$

*Variable node update.* Variable node  $v$  combines its channel prior with all incoming check-to-variable messages except the one from  $c$ :

$$P_{v \rightarrow c}^{(t)}(x) \propto P_{\text{ch}}(x) \prod_{c' \in \mathcal{N}(v) \setminus \{c\}} P_{c' \rightarrow v}^{(t)}(x). \quad (2.6)$$

**Step 3: Termination and decision.** The algorithm iterates until all parity checks are satisfied or a maximum number of iterations is reached. At any point, the posterior belief at  $v$  is

$$b_v(x) \propto P_{\text{ch}}(x) \prod_{c \in \mathcal{N}(v)} P_{c \rightarrow v}^{(t)}(x), \quad (2.7)$$

and a hard decision is extracted as  $\hat{x}_v = \arg \max_x b_v(x)$ . Because soft beliefs are maintained throughout, reaching the iteration limit is not a total failure: the resulting distributions often retain enough refined information to recover partial data through post-processing.

**Implementation Considerations: Log Domain, Min-Sum, and Scheduling** Computing Equations (2.6) and (2.7) as written requires multiplying many small probabilities, which rapidly causes numerical underflow on standard hardware. In practice, BP is therefore carried out in the *logarithmic domain*, replacing each probability  $P(x)$  with a log-likelihood  $\lambda(x) = \log P(x)$  and converting products into sums. To further reduce the computational complexity of the check nodes, a popular alternative known as the *Min-Sum* algorithm is often used. It approximates the complex log-domain check-node computations by replacing the sum over combinations with a simple minimum operation over the LLR magnitudes. This trade-off sacrifices some decoding accuracy for a significant acceleration in computational speed.

The order in which updates are scheduled also affects convergence. A straightforward approach is to use a *layered* schedule, where each iteration updates all check nodes followed by updates of the variable nodes, as described above. Other strategies are possible, for example, a priority-based *node-wise* schedule, where each iteration updates a single edge of the Tanner graph currently carrying the most uncertainty.

The earlier paper (Paper I) in this thesis uses Min-Sum approach with layered scheduling. Our library with LDPC decoder was made public and was later improved. Thus, later on we use different approach, e.g., for the Paper V, we use Sum-Product algorithm with node-wise scheduling, providing better convergence and error correction performance.

**Cycles and Asymptotic Optimality** The assumption underlying the exactness of Belief Propagation is that the Tanner graph is a perfect tree [RU01]. In reality, practically all LDPC graphs contain cycles, which are closed loops of edges that start and end at the same node. If a graph contains short cycles, messages loop back on themselves too rapidly during the decoding process. A node ends up indirectly receiving its own prior belief as “new” evidence, which artificially inflates its certainty and severely degrades the overall decoding performance.

Because a Tanner graph is bipartite, the shortest possible cycle has a length of four. Consequently, a primary objective in LDPC code design is to maximize the *girth* (the length of the shortest cycle). While eliminating cycles of length four is straightforward, systematically avoiding cycles of length six or more becomes a highly complex combinatorial problem, especially when designing codes of arbitrary lengths.

For very large code lengths, Belief Propagation on properly constructed LDPC codes is demonstrably optimal, capable of operating vanishingly close to the Shannon limit [RSU02]. In other words, as the code length grows, they can reliably correct errors at the absolute theoretical maximum noise threshold of the communication channel. Despite the optimality is shown asymptotically, LDPC codes provide excellent error correction performance in practice.

Finally, while classical LDPC codes were designed to correct binary bits over  $\mathbb{F}_2$ , the underlying message-passing framework generalizes in a straightforward manner to larger finite fields or even rings of integers. Instead of tracking binary probabilities, the variable nodes maintain probability mass functions over a small set of possible values. This generalization will prove to be extremely important in this thesis, allowing us to model and correct the noisy side-channel leakages from lattice-based schemes.

## 2.3 Hamming Quasi-Cyclic (HQC)

Hamming Quasi-Cyclic (HQC) [Gab+25] is a prominent code-based Key Encapsulation Mechanism (KEM) and a successful candidate in the NIST post-quantum standardization process. Its security is based on the hardness of decoding a random quasi-cyclic code in the Hamming metric.

All polynomial operations in HQC take place in the polynomial ring  $\mathcal{R} = \mathbb{F}_2[x]/(x^n - 1)$  for a prime  $n$ . Working in this ring is highly efficient. Multiplying two polynomials in  $\mathcal{R}$  is mathematically equivalent to multiplying a vector by a circulant matrix. This quasi-cyclic structure avoids the massive key matrices required by traditional Goppa-code systems, drastically reducing key sizes.

Before generating keys, the communicating parties must agree on a public error-correcting code  $\mathcal{C}$  with a known generator matrix  $\mathbf{G}$ . In HQC,  $\mathcal{C}$  is a two-layer concatenated code: the outer layer is a shortened Reed-Solomon (RS) code operating over bytes, while the inner layer is a duplicated Reed-Muller (RM) code operating over bits. The details of these codes go beyond the scope of this thesis; what matters here is their role in the scheme. The code  $\mathcal{C}$  is entirely public and plays no role in hiding the message. Its sole purpose is to ensure that a legitimate recipient can reliably recover the plaintext despite the noise introduced during encryption.

The foundation of HQC is an IND-CPA secure public-key encryption scheme. It consists of three primary operations:

**Key Generation.** To construct a keypair, we begin by sampling a uniformly random polynomial  $h \xleftarrow{\$} \mathcal{R}$ . Next, we sample two private polynomials  $x, y \xleftarrow{\$} \mathcal{R}$  such that they have a strictly low Hamming weight  $\omega$ . The secret key is defined as  $\text{sk} = (x, y)$ . We then compute the syndrome  $s = x + h \cdot y$ . The public key is published as  $\text{pk} = (h, s)$ . For an adversary to recover the secret polynomials  $x$  and  $y$  from  $\text{pk}$ , they would need to solve a structured variant of the Syndrome Decoding Problem — Quasi-Cyclic Syndrome Decoding Problem which is believed to be intractable.

**Encryption.** To encrypt a message vector  $\mathbf{m}$ , the sender first encodes it using the public generator matrix to produce a codeword  $\mathbf{mG}$  which is interpreted as a polynomial in  $\mathcal{R}$ . Next, the sender generates three sparse error polynomials to act as cryptographic noise:  $r_1$  and  $r_2$  of weight  $\omega_r$ , and  $e$  of weight  $\omega_e$ . The ciphertext is constructed as a pair of polynomials  $(u, v)$ :

$$u = r_1 + h \cdot r_2 \quad (2.8)$$

$$v = \mathbf{mG} + s \cdot r_2 + e \quad (2.9)$$

The structure of this ciphertext mirrors the public key construction and is best understood by analogy. The component  $u = r_1 + h \cdot r_2$  plays the same role as the syndrome  $s = x + h \cdot y$  in key generation: it is a noisy linear combination involving  $h$  that commits the sender to the ephemeral randomness  $r_2$  without revealing it. The component  $v$  then uses this same  $r_2$  to entangle the encoded message with the public key: substituting  $s = x + h \cdot y$  gives  $v = \mathbf{mG} + (x + h \cdot y) \cdot r_2 + e$ , so  $v$  masks the codeword with a term that only the holder of  $\text{sk} = (x, y)$  can relate back to  $u$  and ultimately remove.

**Decryption.** The legitimate recipient possesses the secret polynomial  $y$  and uses it to unmask the message by computing  $v - u \cdot y$ . If we expand this expression using the definitions from encryption, we get:

$$(\mathbf{mG} + (x + h \cdot y) \cdot r_2 + e) - (r_1 + h \cdot r_2) \cdot y,$$

cancelling the terms we are left with:

$$\mathbf{mG} + x \cdot r_2 - r_1 \cdot y + e. \quad (2.10)$$

The result is the original codeword  $\mathbf{mG}$  perturbed by an effective error polynomial  $e' = x \cdot r_2 - r_1 \cdot y + e$ . Because the polynomials  $x, y, r_1, r_2$ , and  $e$  were all deliberately chosen to be sparse, the total Hamming weight of  $e'$  remains small. The recipient simply feeds this noisy codeword into the public decoder for  $\mathcal{C}$ . As long as the weight of  $e'$  falls within the error-correction capacity of the concatenated RS and RM codes, the decoder successfully strips away the noise and outputs the original message  $\mathbf{m}$ .

The IND-CCA security of the KEM version of HQC is achieved by the Hofheinz-Hövelmanns-Kiltz (HHK) transform [HHK17], which could be seen

---

as a modern variant of the Fujisaki-Okamoto transform.



# Lattice-based Cryptography

---

## 3.1 Lattices and Hard Problems

Intuitively, a lattice could be thought as a regular, infinitely repeating grid of points in space. Formally, a lattice  $\mathcal{L}$  is a set of points in  $m$ -dimensional Euclidean space, generated by taking all integer linear combinations of  $n$  linearly independent basis vectors  $\mathbf{b}_1, \dots, \mathbf{b}_n \in \mathbb{R}^m$ . This can be written as:

$$\mathcal{L} = \left\{ \sum_{i=1}^n c_i \mathbf{b}_i \mid c_i \in \mathbb{Z} \right\}$$

The integer  $n$  is the *rank* of the lattice, and  $m$  is its *dimension*. We typically work with full-rank lattices where  $n = m$ , and the vectors are defined over  $\mathbb{Z}_q$  for a prime  $q$ .

**Hard Problems** While navigating a two-dimensional grid on a piece of paper is trivial, the complexity of this space explodes as we move into hundreds or thousands of dimensions. This dimensional complexity gives rise to mathematically hard problems. The most famous of these is the Shortest Vector Problem (SVP). The challenge is simple to state: given a completely arbitrary basis for a lattice, find the non-zero vector in the lattice that is closest to the origin.

While SVP is shown to be NP-hard problem (although under randomized reductions) [Ajt98], it is not as useful for building practical cryptographic schemes. In 2005, Regev introduced the famous Learning With Errors (LWE) problem [Reg05]. LWE (or its variants) lies as a basis of most modern lattice-based cryptography. To understand LWE, consider a classic high school algebra problem: solving a system of linear equations. If you are given a public matrix  $\mathbf{A}$  and a public vector  $\mathbf{b}$  such that  $\mathbf{b} = \mathbf{A}\mathbf{s} \bmod q$ , you can easily find the secret vector  $\mathbf{s}$  using standard Gaussian elimination.

The problem becomes difficult when we introduce a small error vector  $\mathbf{e}$  to the system:

$$\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q,$$

where the matrix  $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$  and error  $\mathbf{e}$  comes from a specific noise distribution  $\chi$ . In the original formulation, the secret  $\mathbf{s}$  was set randomly, but in the modern formulation it is also sampled from  $\chi$ .

For an adversary  $\mathcal{A}$  observing only the public values  $(\mathbf{A}, \mathbf{b})$ , extracting the secret  $\mathbf{s}$  is believed to be computationally intractable for both classical and quantum computers.

Standard LWE comes with a significant practical drawback: because  $\mathbf{A}$  is a large matrix of random integers, the resulting public key  $\mathbf{pk}$  is large. To reduce the key size and drastically improve computational performance, the new variant of LWE was introduced as Ring-LWE (RLWE) [LPR10]. Instead of relying on matrices and vectors of integers, RLWE operates over ring  $\mathbb{Z}_q[x]/f(x)$ , where  $f(x)$  is typically a cyclotomic polynomial  $f(x) = x^n + 1$ . In this setting, a single polynomial  $a$  replaces the entire matrix  $\mathbf{A}$ , compressing the key size significantly. The secret and the error are also polynomials. Since  $f(x)$  is cyclotomic, for correctly chosen parameters, the multiplication could be computed fast using Number Theoretic Transform which could be seen as a Fourier Transform over  $\mathbb{Z}_q$ .

However, cryptography is always a game of trade-offs. The algebraic structure that makes RLWE so efficient also introduces a potential vulnerability. Because cyclotomic rings possess intense mathematical structure, it opens up the possibility that an adversary might one day discover a highly efficient, specialized mathematical attack that reduces the security considerably or in the worst-case breaks the scheme.

To mitigate the structural risks of RLWE without returning to the undesirably slow performance of standard LWE, the cryptographic community found a middle ground: Module-LWE (MLWE). MLWE acts as a structural compromise. Instead of using a single massive matrix of integers or a single massive polynomial, MLWE uses a small matrix where each entry is a moderately sized polynomial. This provides a highly tunable trade-off between performance and security. As we will see in the following sections, this module framework is the very foundation of ML-KEM, the newly standardized post-quantum key encapsulation mechanism.

### 3.2 ML-KEM (Kyber)

ML-KEM (previously known as CRYSTALS-Kyber [Sch+22d]) is the primary post-quantum key encapsulation mechanism (KEM) standardized by NIST. It is build upon MLWE problem and achieves practical public key sizes and high computational efficiency.

Operations in ML-KEM are defined over the polynomial ring  $\mathbb{Z}_q[X]/(X^{256} + 1)$  with a specific prime modulus  $q = 3329$ . During key generation, the secret key

vector  $\mathbf{s}$  and the error vector  $\mathbf{e}$  are sampled from a centered binomial distribution  $\mathcal{B}_\eta$ . This distribution ensures their polynomial coefficients remain bounded and small. A public matrix  $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{k \times k}$  is generated uniformly at random.

As most KEMs, ML-KEM achieves IND-CCA security by designing IND-CPA public key encryption and applying FO transform. To encrypt a message  $m$ , the sender generates their own ephemeral randomness by sampling a secret vector  $\mathbf{r}$  and two small error terms  $\mathbf{e}_1$  and  $e_2$  from the centered binomial distribution  $\mathcal{B}_\eta$ . The ciphertext consists of two components,  $\mathbf{u}$  and  $v$ , computed as:

$$\mathbf{u} = \mathbf{A}^T \mathbf{r} + \mathbf{e}_1 \quad (3.1)$$

$$v = \mathbf{t}^T \mathbf{r} + e_2 + m. \quad (3.2)$$

Before being added to  $v$ , each bit of the message  $m \in \{0, 1\}$  is encoded by mapping it to a coefficient in  $\mathbb{Z}_q$ : a 0 maps to 0 and a 1 maps to  $\lfloor q/2 \rfloor$ . Geometrically, this places each message bit at one of two maximally separated points on the integer circle modulo  $q$ . The noise added by  $e_2$  and  $\mathbf{t}^T \mathbf{r}$  perturbs these points by a small amount, but as long as the total noise remains bounded by  $\lfloor q/4 \rfloor$ , rounding the noisy coefficient back to the nearest of the two reference points recovers the original bit exactly.

To decrypt, the receiver utilizes the secret key  $\mathbf{s}$  to strip away the cryptographic masking applied during encryption, computing a noisy message  $m'$ :

$$m' = v - \mathbf{s}^T \mathbf{u}. \quad (3.3)$$

Because  $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}$ , expanding this equation reveals that the large  $\mathbf{A}^T \mathbf{r}\mathbf{s}$  terms cancel out. What remains is the upscaled message  $m$  combined with a sum of small error terms (specifically,  $e_2 + \mathbf{e}^T \mathbf{r} - \mathbf{s}^T \mathbf{e}_1$ ). As long as the cumulative noise is sufficiently small, it is naturally discarded when the receiver rounds and downscales the coefficients back to their original binary values, successfully recovering the plaintext.

### 3.3 NTRU and NTRU Prime

While the Learning With Errors (LWE) problem dominates much of the post-quantum landscape, it is not the only foundation for lattice-based cryptography. Introduced in 1996 by Hoffstein, Pipher, and Silverman[HPS98], NTRU is a venerable cryptosystem that predates LWE by nearly a decade. Though unrelated to the early Ajtai-Dwork lattice schemes[AD97], NTRU relies on the hardness of finding the shortest or closest vectors in high-dimensional integer lattices. Over the years, numerous improvements have been proposed, culminating in multiple submissions to the NIST PQC standardization effort, notably NTRUEncrypt [Zha+17], NTRU-HRSS-KEM [Sch+17], and NTRU Prime [Ber+17]. The

simplified foundational version is often denoted NTRU-HPS. A comparison of the NTRU-based NIST-PQC submissions can be found in [Sch18].

NTRU operates over polynomial rings. Classically, all operations are defined over a polynomial ring  $\mathcal{R}_q = \mathbb{F}_q[x]/(x^N - 1)$ . The scheme is parameterized by coprime positive integers  $(N, q, p)$ . Here,  $N$  represents the maximum degree of the polynomials,  $q$  is a large modulus (often a power of two), and  $p$  is a small modulus (typically a prime).

During key generation, two small polynomials  $f$  and  $g$  are randomly sampled from strictly defined sampling spaces. The polynomial  $f$  serves as the secret key  $\text{sk}$ , and it must be invertible modulo both  $p$  and  $q$ . We compute its inverses  $f_p$  and  $f_q$  such that  $ff_p = 1 \bmod p$  and  $ff_q = 1 \bmod q$ . The public key  $\text{pk}$ , denoted  $h$ , is constructed by essentially masking  $g$  with the inverse of  $f$ :

$$h = pf_qg \bmod q.$$

To encrypt, a message is first encoded into a polynomial  $m$  of degree at most  $N - 1$  with coefficients limited to modulo  $p$ . A blinding polynomial  $r$  is randomly selected to act as noise. The ciphertext  $c$  is computed by multiplying  $r$  with the public key  $h$  and adding the message  $m$ :

$$c = rh + m \bmod q.$$

To decrypt, the recipient multiplies  $c$  by the secret polynomial  $f$  modulo  $q$ :

$$\begin{aligned} fc &= f(rh + m) \bmod q \\ &= frpf_qg + fm \bmod q \\ &= prg + fm \bmod q \end{aligned}$$

using the fact that  $ff_q = 1 \bmod q$ . Provided the parameters are chosen so that the coefficients of  $fc$  lie within a bounded range, one can lift this result to  $\mathbb{Z}$  and reduce modulo  $p$ , causing the term  $prg$  to vanish entirely and leaving  $fm \bmod p$ . A final multiplication by  $f_p$  then recovers  $m$ .

**NTRU Prime** NTRU Prime represents a conservative evolutionary branch of the NTRU family, motivated by concerns about the algebraic structure of the rings used in classical NTRU and Ring-LWE schemes. Traditional constructions operate in cyclotomic rings such as  $\mathbb{F}_q[x]/(x^n - 1)$  or  $\mathbb{F}_q[x]/(x^n + 1)$ , whose rich algebraic structure, while computationally convenient, has been identified as a potential attack surface for specialized lattice attacks that exploit this structure.

To mitigate these risks, NTRU Prime replaces the cyclotomic ring with a non-cyclotomic field defined by an irreducible polynomial:  $\mathcal{R}_q = \mathbb{F}_q[x]/(x^n - x - 1)$ , where  $n$  is prime. This polynomial has no special algebraic symmetries, removing the structural footholds that certain cryptanalytic techniques rely on. Arithmetic

in this ring remains efficient, but the security argument rests on more conservative ground.

The encryption mechanism in the Streamlined NTRU Prime variant also departs from the classical additive noise approach. Most importantly, it is explicitly designed as a KEM rather than a traditional encryption scheme. In this paradigm, there is no arbitrary message polynomial  $m$  to encrypt. Instead, the sender randomly generates a short polynomial  $r$  with strictly defined weight constraints. This  $r$  acts as the encapsulated plaintext, which will ultimately be hashed by both parties to derive a symmetric shared key.

To encapsulate  $r$ , the algorithm first computes a masked value  $v = hr \bmod q$  and then applies a deterministic rounding function  $\text{Round}$ , which maps each coefficient to the nearest multiple of 3. The ciphertext  $c$  is simply this rounded polynomial. Decryption recovers the polynomial  $r$  by multiplying the ciphertext by the secret key and reducing modulo 3, effectively inverting the rounding step. This rounding-based design avoids the small modulus  $p$  of classical NTRU entirely, simplifying the security analysis.

Furthermore, to achieve IND-CCA security, strict parameter checks are enforced during decapsulation. If the recovered intermediate polynomial does not satisfy the expected weight conditions, or if re-encrypting the recovered  $r$  does not perfectly match the received ciphertext  $c$ , the algorithm outputs a fixed pseudorandom dummy key rather than failing outright. This constant-time implicit rejection provides robust resistance against decryption failure attacks.



# Side-Channel Analysis

---

Cryptographic algorithms are mathematically proven secure in the black-box model, where an adversary only has access to the inputs and outputs of the scheme. However, physical implementations of these algorithms run on real hardware, which consumes power, emits electromagnetic radiation, and takes measurable amounts of time to execute. Side-Channel Analysis (SCA), first introduced by Kocher [Koc96b], exploits these physical and operational leakages to extract secret information, entirely bypassing the mathematical security of the algorithms.

As Post-Quantum Cryptography undergoes standardization and deployment, evaluating its resilience against physical and implementation-level threats is critical. This chapter defines the attack vectors and oracle models used throughout the appended papers, establishing the theoretical foundation for our key-recovery frameworks.

## 4.1 Side-Channel Leakages

Whenever a device processes sensitive data, its physical properties fluctuate in ways that depend on its internal state. By capturing and analyzing these fluctuations, an adversary can infer the underlying secret keys. Broadly speaking, these physical attacks fall into two categories: active and passive. Active attacks involve deliberate interference with the device. For example, in fault injection attacks, an adversary manipulates voltage, clock frequency, or laser pulses to induce computational errors that leak information about the secret. Passive attacks, conversely, rely solely on observing the device's natural emissions without altering its operation. In this thesis, we focus strictly on passive side-channel analysis, specifically targeting the two most classical and widely studied leakage sources:

- **Power and Electromagnetic (EM) Analysis:** Every operation a processor performs draws electrical current from its power supply. Because the current drawn depends on both the instruction being executed and the data values being processed, the instantaneous power consumption of a device varies in a data-dependent manner. Electromagnetic emanations carry essentially the

same information, arising from the same underlying physical activity, but can be captured non-invasively and at a distance using a suitable antenna.

- **Timing Analysis:** Cryptographic implementations frequently contain operations whose execution time depends on secret data such as through data-dependent conditional branches, non-constant-time arithmetic, or variable-latency instructions. An adversary who can repeatedly trigger a cryptographic operation and precisely measure its execution time can infer which control-flow paths were taken and thereby recover information about the secret. Beyond direct timing measurements, microarchitectural attacks such as cache-based attacks exploit shared hardware resources: by observing which cache lines are accessed during a computation, an adversary can reconstruct secret-dependent memory access patterns even without measuring total execution time directly.

**Disclaimer on Scope.** The physical acquisition of these signals via high-resolution oscilloscopes, as well as the signal processing techniques required to build templates and filter noise, is a deep and complex field in its own right. However, this thesis is not a primer on signal processing. Instead, we primarily treat the leakage as an algorithmic abstraction. Our focus is on the mathematical and information-theoretic challenge of how to optimally exploit this leakage once it has been detected and digitized.

## 4.2 Micro-architectural Leakages

While traditional power and EM attacks require close physical proximity to the target device, modern computing environments may introduce remote attack vectors via shared hardware. In cloud servers, mobile devices, and multi-core processors, several processes continuously share underlying micro-architectural resources.

**Cache Attacks.** The CPU cache is a prime example of a shared resource. Because accessing the main memory is slow, processors store recently used data in a fast, shared cache. In an attack such as Prime+Probe[OST06a], an adversary fills the cache with their own data (Prime), waits for the victim to perform a cryptographic operation, and then measures the time it takes to access their data again (Probe). If the victim's operation accessed specific memory lines, the adversary's data is evicted, resulting in a measurable delay. This reveals the victim's memory access patterns, which can leak secret keys if the cryptographic implementation uses secret-dependent lookups.

**Prefetchers and the DMP.** Beyond caches, modern CPUs employ hardware prefetchers that attempt to predict which memory addresses will be needed next, loading

them into the cache proactively to reduce latency. A particularly aggressive variant is the Data Memory-Dependent Prefetcher (DMP), present in several modern microarchitectures. Unlike conventional stride-based prefetchers that predict addresses from access patterns, the DMP inspects the *contents* of memory and, if it encounters a value that resembles a valid pointer, speculatively dereferences it and fetches the corresponding memory line. This behaviour creates a subtle but powerful side channel: if a cryptographic computation causes a secret-dependent value to reside in memory, the DMP may act on it, leaving an observable trace in the cache state or timing profile that an adversary can exploit. We investigate this microarchitectural leakage mechanism and its implications for post-quantum cryptographic implementations in Paper V.

### 4.3 Classification of Oracles

To systematically study and optimize key-recovery chosen-ciphertext attacks, we abstract the complex physical and micro-architectural leakages into a conceptual black box called an *oracle*. This abstraction provides a clear separation between the physical discovery of a leakage and its mathematical exploitation.

In our threat model, an adversary (Mallory) sends a carefully crafted chosen ciphertext  $c'$  to the victim (Alice). Alice runs the decapsulation algorithm  $\text{Dec}(\text{sk}, c')$ . By observing the side-channel leakage emitted during this execution, Mallory queries the conceptual oracle to extract information about the decapsulated message  $m'$ , which is mathematically entangled with the secret key  $\text{sk}$ .

Depending on the nature of the physical leakage, we can classify these oracles into several distinct categories [Rav+20a; GJN20a; Uen+22a].

#### 4.3.1 One-Bit-Bound Oracles

The most common side-channel leakages in lattice-based cryptography provide, at most, a single bit of information per chosen-ciphertext query. We refer to these collectively as one-bit-bound oracles.

- **Binary Plaintext-Checking (PC) Oracle:** This oracle answers a binary question about the decrypted plaintext. The adversary crafts  $c'$  such that the decrypted message  $m'$  is equal to one of the two predefined options, either  $m_0$  or  $m_1$ . For example, the oracle reveals whether  $m' = \mathbf{0}$  or  $m'$  is a sparse unit vector containing a single 1.
- **General PC Oracle:** A more general primitive is an oracle that, given a ciphertext  $c'$  and a reference message  $m$ , reveals whether the decrypted message  $m'$  is equal to  $m$ . Binary PC oracle could be seen as a special case of general PC oracle with a promise that the decrypted message is either equal to the reference message or deviates from it in one specific way.

- **Decryption Failure (DF) Oracle:** This oracle reveals whether decrypted message is equal to  $m$ . Typically (although it is not required for DF oracle), the ciphertext  $c'$  that will be decrypted is built by adding additional noise of the special form to the ciphertext  $c$ , which is a valid encryption of  $m$ . In the context of FO-transformed KEMs, the general PC oracle and the DF oracle effectively collapse into the same side-channel observation [Uen+22a] and can be treated as mathematically equivalent. They provide more flexibility during the attack compared to a binary PC oracle, which in turn, as shown in Paper V, could greatly improve the key recovery efficiency.

### 4.3.2 Multi-Value and Full Decryption Oracles

While one-bit-bound oracles are the most prevalent due to their ease of instantiation, stronger physical leakages can yield far more powerful oracles.

- **Full Decryption (FD) Oracles:** An FD oracle is the most powerful variant. It returns the entire decrypted message vector  $m'$  in a single query [Ngo+21a]. While incredibly efficient for key recovery, FD oracles require very strong, high-resolution leakages, severely limiting their applicability in heavily noise-protected environments.
- **Multi-Value PC Oracles:** These oracles act as a middle ground between FD and binary PC oracles. By applying multi-class classification to the side-channel traces, an adversary can extract a few bits of information from a single decapsulation call [Raj+23].

## 4.4 The Key Recovery Challenge

The classification of oracles brings us to a central problem in applied side-channel analysis: *How do we optimally construct chosen ciphertexts to efficiently extract the secret key?*

Historically, attacks relying on one-bit-bound (binary PC or DF) oracles follow a naive, localized approach. The attacker crafts a ciphertext designed to isolate and test a single coefficient of the secret key  $sk$ . Because physical measurements are inherently noisy, the constructed oracle is rarely perfect; it has an *accuracy*  $\rho < 1$  — probability to output true value. To compensate for this noise, standard attacks rely on majority voting, meaning the attacker must query the exact same secret coefficient several times until a statistical consensus is reached.

This coefficient-by-coefficient recovery is exceptionally inefficient. If a KEM has hundreds of secret coefficients, recovering the full key requires an immense number of traces, making the attack highly impractical against targets with trace limits or dynamic keys. Furthermore, this method entirely ignores the non-uniform distribution from which the secret symbols are often sampled.

**Soft Information and the LDPC Framework.** To break through this inefficiency, we shift from localized hard-decision recovery to global soft-decision decoding. The key insight is that a single query need not target a single coefficient in isolation. Instead, we can craft ciphertexts that simultaneously expose information about several secret coefficients at once, in the form of a *linear combination* of those coefficients.

To build intuition, consider first the simplest case: learning the sum  $sk_i + sk_j$  of two secret coefficients. Even this provides joint information that neither coefficient alone reveals. More generally, consider a linear combination  $c = \sum_{j \in \mathcal{J}} \alpha_j sk_j$  for some index set  $\mathcal{J}$  and weights  $\alpha_j$ . If the values this combination can take are all distinct, then learning  $c$  exactly reveals complete information about the individual coefficients that contribute to it. In practice we do not learn  $c$  exactly (and the values are not necessarily unique), but we obtain a noisy observation from which we can compute a probability distribution over the possible values of  $c$ . This is *soft information*: rather than a hard binary answer, each query yields a probability distribution that updates our prior belief about the joint state of the participating coefficients.

By structuring many such queries carefully, we can model the entire secret key as a codeword within a Low-Density Parity-Check (LDPC) code. Each linear combination  $c_i = \sum_{j \in \mathcal{J}_i} \alpha_j sk_j$  corresponds to a parity-check equation: the extended constraint  $\sum_{j \in \mathcal{J}_i} \alpha_j sk_j - c_i = 0$  defines a row of a parity-check matrix. Collecting all such equations yields a matrix of the form

$$\mathbf{H} = (\mathbf{H}' \quad -\mathbf{I}),$$

where  $\mathbf{H}'$  encodes the positions  $\mathcal{J}_i$  and weights  $\alpha_j$  of each linear combination, and  $-\mathbf{I}$  ties each row to its corresponding observed value  $c_i$ . Concretely, the variable nodes of the resulting Tanner graph are the union of the secret coefficients  $sk_j$  and the observed combination values  $c_i$ , and the codeword to be recovered is their concatenation. Ultimately, however, our objective is solely the successful extraction of  $sk$ . Each check node enforces one constraint of the form  $\sum_{j \in \mathcal{J}_i} \alpha_j sk_j - c_i = 0$ , which must evaluate to zero for a valid codeword.

The role of the oracle is to initialize the variable nodes  $c_i$  with soft information: rather than fixing  $c_i$  to a hard value, each noisy query yields a probability distribution over the possible values of  $c_i$ , which serves as the prior for that variable node. Belief Propagation then propagates this soft information *through* the check nodes, whose zero-sum constraint couples each  $c_i$  to its neighboring secret coefficients. BP then iteratively refines the distributions of the  $sk_j$  variable nodes until the secret key is recovered or a maximum number of iterations is reached. Because LDPC codes decoded via BP operate near the Shannon limit (as established in Section 2.2), this framework simultaneously exploits the non-uniform prior distribution of the secret coefficients (source coding) and tolerates oracle noise natively without majority voting (error correction).

This paradigm is the core contribution of the appended papers. In Paper I, we introduce the foundational SCA-LDPC framework for Kyber and HQC. In Paper III and Paper IV, we extend the methodology to the non-cyclotomic structures of NTRU Prime and achieve single-trace recovery on HQC using Moderate-Density Parity-Check (MDPC) decoding. Finally, in Paper V, we deploy a hybrid adaptive-LDPC framework to maximize the information extracted from the microarchitectural GoFetch vulnerability, reducing the required number of queries by an order of magnitude.

# Security of Approximate Homomorphic Encryption

---

As introduced in Section 1.2, Homomorphic Encryption (HE) allows a third party to evaluate functions over ciphertexts such that the result, once decrypted, equals the function applied to the original plaintexts. The server learns nothing about the underlying data.

For many years, HE remained a theoretical curiosity. Partial schemes supporting only addition or only multiplication existed, but supporting both simultaneously for arbitrary depth was an open problem. Gentry's breakthrough in 2009 [Gen09] settled this question by constructing the first Fully Homomorphic Encryption (FHE) scheme, capable in principle of evaluating any Boolean circuit. While Gentry's original construction was far too inefficient for practical deployment, it established a blueprint based on a technique called *bootstrapping* to periodically refresh the accumulated noise. This has since inspired a generation of more efficient schemes.

## 5.1 Approximate Homomorphic Encryption and CKKS

A key observation driving practical progress is that many real-world applications do not require exact arithmetic. Scientific computing, machine learning inference, and statistical analysis all operate naturally in the realm of floating-point numbers, where a small computational error is expected and acceptable. This observation motivated Cheon, Kim, Kim, and Song to introduce the CKKS scheme [Che+17], the first *approximate* homomorphic encryption scheme and arguably the most practically relevant HE scheme today.

The central design philosophy of CKKS is to treat the inherent cryptographic noise of lattice-based encryption not as an error to be eliminated, but as a natural analogue of floating-point rounding. This reframing enables substantial efficiency gains: rather than working around the noise, CKKS absorbs it into the computation.

**Message Encoding** CKKS operates over the polynomial ring  $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^N + 1)$  for a power-of-two degree  $N$ . Rather than encrypting a single value, CKKS encodes an entire vector  $\mathbf{z} \in \mathbb{C}^{N/2}$  of complex numbers into a single polynomial via a variant of the inverse discrete Fourier transform scaled by a factor  $\Delta$ , called the *scaling factor*. Concretely, encoding maps  $\mathbf{z}$  to a polynomial whose evaluations at the  $2N$ -th roots of unity approximate  $\Delta \cdot \mathbf{z}$ . This packing allows all  $N/2$  slots to be processed simultaneously in a SIMD fashion, amortizing the computational cost over many values at once.

**Encryption and Homomorphic Operations** The structure of CKKS closely mirrors that of ML-KEM, but is built on the Ring Learning With Errors (RLWE) problem rather than its module variant. The secret key is a sparse polynomial  $s \in \mathcal{R}_q$  with small coefficients, and the public key is a pair  $(a, b)$  with  $b = a \cdot s + e$  for a small error polynomial  $e$ , an RLWE instance that is computationally indistinguishable from uniform. In addition to  $(pk, sk)$ , key generation produces an *evaluation key*  $ek$ , which encodes an encryption of  $s^2$  and is needed to carry out homomorphic multiplications without growing the ciphertext size.

To encrypt a plaintext polynomial  $m$  (the encoded message vector), the sender samples a small ephemeral polynomial  $r$  and two small error terms  $e_0, e_1$ , and forms the ciphertext  $(a', b') = r \cdot (a, b) + (e_1, m + e_0)$ . Decryption computes  $b' - a' \cdot s$ , which expands to  $r(b - as) + m + e_0 - e_1s = re + m + e_0 - e_1s \approx m$ , recovering the message up to a small residual noise term. This noise is the unavoidable cost of RLWE-based security and is precisely what CKKS embraces rather than eliminates.

Homomorphic operations act directly on ciphertext pairs. Addition is simply componentwise:  $(a_0, b_0) + (a_1, b_1) \bmod q$ , with noise terms accumulating additively. Multiplication is considerably more involved: the product of two ciphertexts naturally produces a degree-three object involving  $s^2$ , which must be *relinearised* back to a degree-two ciphertext using the evaluation key  $ek$ .

For the decryption to be correct, the noise must never grow large enough to corrupt the encoded message. During the homomorphic operations, this noise grows which somewhat limits the number of operations to be performed. The exception is bootstrapping operation that allows to reset the noise, but it is computationally expensive. Therefore, the noise must be carefully tracked.

**Approximate Correctness** Unlike exact HE schemes, CKKS does not satisfy the standard correctness definition requiring decryption to return exactly  $\mathfrak{C}(m_1, \dots, m_k)$ . Instead, it satisfies an *approximate* variant: decryption yields a value close to the true result, with the error bounded by a quantity that depends on the circuit depth and scheme parameters:

$$\text{Dec}_{sk}(\text{Eval}_{ek}(\mathfrak{C}, c_1, \dots, c_k)) \approx \mathfrak{C}(m_1, \dots, m_k).$$

For applications such as machine learning inference or statistical analysis, this approximation is entirely acceptable.

## 5.2 Security Models for HE

**The Incompatibility of IND-CCA and the Role of IND-CPA** The foundational security baseline for most modern public-key encryption is IND-CCA, as introduced in Section 1.2.2. However, it is well known that homomorphic encryption schemes fundamentally cannot achieve IND-CCA security. The defining feature of HE is malleability: the ability to take a ciphertext and apply a homomorphic operation to produce a new, valid ciphertext without knowing the secret key. An adversary could easily intercept a challenge ciphertext, homomorphically add a certain constant to it, and submit the resulting ciphertext to a decryption oracle. By subtracting the same constant from the oracle's response, the adversary trivially recovers the original plaintext.

Because of this inherent vulnerability to active chosen-ciphertext attacks, IND-CPA is universally adopted as the standard security notion for HE schemes. For standard PKE, IND-CPA guarantees that a passive adversary observing only ciphertexts gains no information about the underlying plaintexts. For *exact* homomorphic encryption, this guarantee extends naturally to scenarios where final decryption results are shared, an important scenario mentioned a bit later. If an adversary knows an initial message  $m$  and a public homomorphic function  $f$ , it can easily compute the final result  $f(m)$  on its own. Because observing the decrypted output yields no new information, the IND-CPA definition is perfectly equivalent to IND-CPA with a decryption oracle when applied to exact encryption schemes.

**The Gap in IND-CPA for Approximate HE** For approximate schemes like CKKS, the assumption that decrypted outputs provide no new information breaks down entirely. The core issue is that approximate decryption does not simply return  $f(m)$ ; rather, it returns an approximation that inherently contains small noise. This noise is directly influenced by the secret key and the randomness used during encryption.

Because the result of decryption is not a simple deterministic function of the inputs and the evaluated circuit, observing it provides additional information that the adversary cannot easily simulate on its own. This leakage is not captured by the traditional IND-CPA model, meaning an approximate homomorphic encryption scheme can technically satisfy IND-CPA security while remaining completely vulnerable to passive key-recovery attacks.

**The Shared Decryption Scenario** This theoretical gap poses a severe threat in practical deployments, particularly in systems where decryption results are made

publicly available or disclosed to selected parties. A natural example is privacy-preserving data sharing and aggregation for medical data. In this setting, hospitals encrypt sensitive records and upload them to a cloud service. A researcher acts as an investigator, querying the cloud server to homomorphically compute aggregate statistics (e.g., mean or variance). Finally, the result is decrypted and returned to the researcher.

Even if the server verifies that the query is legitimate, a malicious researcher acts as a passive adversary: they choose the input distribution, specify the computation, and observe the decrypted result. As demonstrated by recent attacks, this single decrypted query result can be enough to recover the secret decryption key in full, thereby exposing the entire encrypted database.

**The Li–Micciancio Attack** Li and Micciancio [LM21] demonstrated that this seemingly harmless noise leakage has devastating consequences. To understand the core mechanism of their attack without delving into the complexities of message encoding, consider the underlying Ring-LWE ciphertext structure. Suppose an adversary asks a server to encrypt a known message, such as  $m = 0$ , and then homomorphically evaluates the identity function (i.e., performs no computation at all). The resulting ciphertext is a pair  $ct = (a, b)$ , where  $b = a \cdot s + m + e \bmod q$  and  $e$  is the small encryption noise. Because  $m = 0$ , this simplifies to  $b = a \cdot s + e \bmod q$ .

When the server performs the approximate decryption and returns the result, the adversary receives the noisy plaintext  $m' = m + e$ , which in this case is exactly the error term  $e$ . Since the adversary already holds the ciphertext component  $b$  and now knows the exact noise  $e$  from the decryption oracle, it can simply compute:

$$b - m' = a \cdot s \bmod q$$

This yields a direct linear equation in the secret key  $s$ . In the ring setting, a single such equation is often sufficient to completely recover  $s$  by computing the ring inverse of  $a$ , giving  $s = a^{-1} \cdot (b - m') \bmod q$ . If  $a$  is not invertible, collecting a small number of ciphertexts provides a solvable system of linear equations.

The attack is not restricted to the zero message or the identity function: any known message and any known homomorphic computation allows the adversary to isolate the error and form the exact same structural equation. Li and Micciancio implemented this attack against all major CKKS libraries of the time (HEAAN, SEAL, HELib, and PALISADE) and demonstrated complete secret key recovery in a matter of seconds, even at the highest security parameter settings.

Crucially, this is not an implementation vulnerability. The libraries correctly implement the CKKS specification. The problem lies with the security definition itself: CKKS satisfies IND-CPA under standard RLWE hardness assumptions, but IND-CPA simply does not model the precise error information leaked by approximate decryption.

**The IND-CPA<sup>D</sup> Security Model** To accurately capture this threat, Li and Micciancio proposed the IND-CPA<sup>D</sup> (IND-CPA with Decryption oracles) security model. The key insight is that a passive adversary in the homomorphic setting has three distinct capabilities: they can influence which messages get encrypted, they can specify the homomorphic computation to be performed, and they can observe the decrypted result. The IND-CPA<sup>D</sup> model grants the adversary formal oracle access to all three, as illustrated in Section 5.2.

**Definition 1** (IND-CPA<sup>D</sup> Security [LM21]). *An adversary  $\mathcal{A}$  is given access to a shared state  $S$  containing sequences of message pairs and ciphertexts  $(m_0, m_1, c)$ . The adversary interacts with three oracles:*

- **Encryption Oracle**  $E_{pk}(m_0, m_1)$ : *Computes  $c \leftarrow \text{Enc}_{pk}(m_b)$ , appends  $(m_0, m_1, c)$  to the state  $S$ , and returns  $c$ .*
- **Evaluation Oracle**  $H_{ek}(g, J)$ : *Given a deterministic function  $g$  and a list of indices  $J$ , homomorphically evaluates  $g$  on the corresponding ciphertexts in  $S$ , appends the result to  $S$ , and returns the evaluated ciphertext.*
- **Decryption Oracle**  $D_{sk}(j)$ : *Given an index  $j$ , checks that the underlying messages satisfy  $m_0 = m_1$ . If so, it returns  $\text{Dec}_{sk}(c)$ ; otherwise it returns  $\perp$ .*

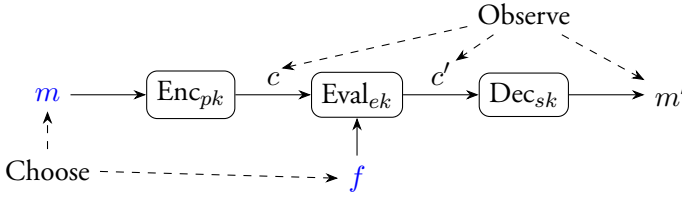
*The scheme is IND-CPA<sup>D</sup> secure if no efficient adversary can distinguish the hidden bit  $b$  with non-negligible advantage.*

The restriction  $m_0 = m_1$  in the decryption oracle is essential: it prevents the adversary from trivially winning by asking to decrypt a ciphertext where the two challenge messages produce observably different outputs. In other words, the adversary may only observe decryptions of honestly generated ciphertexts whose underlying messages are already identical—precisely the shared decryption scenario. Importantly, IND-CPA<sup>D</sup> is a conservative extension of IND-CPA: for exact encryption schemes, the two notions are provably equivalent, because seeing the decryption of  $f(m)$  yields nothing the adversary could not compute from  $m$  and  $f$  alone. For approximate schemes, IND-CPA<sup>D</sup> is strictly stronger.

## 5.3 Countermeasures

The vulnerabilities exposed by the IND-CPA<sup>D</sup> model necessitate robust countermeasures. If the approximate decryption output inherently leaks the key-dependent noise, the logical defense is to mathematically obscure that noise before it is revealed to the user.

**Noise Flooding and Differential Privacy** To thwart passive key-recovery attacks, modern FHE libraries have adopted a technique commonly referred to as noise



**Figure 5.1:** Passive attacker scenario based on Li and Micciancio’s model [LM21]. In this context, the attacker may choose/know the plaintext  $m$  and the homomorphic function  $f$ . The attacker can eavesdrop on the ciphertexts  $ct$ ,  $ct'$  and decryption result  $m'$ , but is not allowed to tamper with (or inject) ciphertexts or alter the computation’s final outcome.

flooding [Li+22]. The core intuition is straightforward: before the final decryption result is handed over, the evaluating party deliberately post-processes the output by adding a massive, statistically independent error term (the “flood”) to the ciphertext.

By applying concepts from Differential Privacy (DP), cryptographers can guarantee that the final decryption result is statistically indistinguishable regardless of the specific key-dependent noise hidden underneath. In the context of approximate homomorphic encryption, the DP mechanism protects the secret key by masking the exact approximation error. Li et al. demonstrated that adding properly calibrated Gaussian noise to the output of the CKKS decryption function is sufficient to provably achieve IND-CPA<sup>D</sup> security [Li+22].

**The Two-Parameter Security Model:  $(c, s)$ -Security** While noise flooding is mathematically sound, its practical implementation introduces a severe penalty. Perturbing the output of the decryption function lowers the output quality, making the already approximate result significantly less accurate. The FHE parameters must be expanded to accommodate this massive artificial noise, degrading performance.

If cryptographers treat the required statistical DP noise with the exact same security margin as the underlying computational lattice hardness, the resulting flood must be overwhelmingly massive. To mitigate this precision loss without sacrificing safety, Li et al. proposed replacing the rigid security boundary with a finer-grained, two-parameter security model known as  $(c, s)$ -bit security.

Instead of a single metric, this model conceptually separates the computational security parameter ( $c$ ) from the statistical security parameter ( $s$ ). The intuition is that while a high computational bound (e.g.,  $c = 128$  bits) is necessary to protect the underlying RLWE problem against future algorithmic advances or supercomputers, the DP noise relies on pure, unconditional statistical masking. Therefore, the statistical security parameter can safely be set to a lower, yet practically secure bound (e.g.,  $s = 64$ ). This separation significantly reduces the sheer volume

of Gaussian noise that must be injected, saving dozens of bits of precision while maintaining a rigorous defense against passive adversaries.

## 5.4 Noise Estimation Pitfalls

The effectiveness of the noise flooding countermeasure relies entirely on one critical assumption: the evaluating party must know precisely how large the key-dependent noise has grown during the homomorphic evaluation to ensure the flooding noise is large enough to mask it. This requirement forces library developers to implement internal noise estimation routines.

There are two primary paradigms for estimating this error growth, each with vastly different operational and security implications: static estimation and dynamic estimation.

**Static Estimation and Computational Integrity** Static noise estimation calculates the error bound publicly, based solely on the abstract sequence of homomorphic operations (the circuit) and the error bounds of the inputs. Because this bound relies purely on the planned computational path, it avoids touching sensitive variables during the estimation process. Alternatively, the noise bound could be determined experimentally by randomly generating a new secret key, running the desired fixed circuit and inspecting the output, repeating this a few times for different secret keys.

However, relying purely on static estimation introduces a practical mismatch with the realities of the IND-CPA<sup>D</sup> security model. Static estimation inherently assumes that the adversary honestly follows the pre-defined computational path to the letter. In a real-world scenario, an active or semi-honest adversary could deviate from this path by submitting maliciously crafted ciphertexts that do not conform to the expected noise profile. Unless advanced cryptographic tools, such as Zero-Knowledge Proofs (ZKPs), are employed to strictly enforce the integrity of the ciphertext and the computation, static bounds can be silently violated, leading to insufficient noise flooding and key recovery.

**Dynamic Estimation: Worst-case vs. Average-case** To circumvent the rigidity of static offline bounds, many libraries implement dynamic, on-the-fly noise tracking. Instead of pre-computing a single global bound for the entire expected circuit, the library attaches a noise estimate (as metadata) directly to each ciphertext object. As the server evaluates the homomorphic circuit, every applied gate updates this metadata to reflect the expected noise expansion. Upon receiving the final ciphertext, the decrypting party reads this dynamically tracked estimate and scales the differential privacy (DP) flooding noise accordingly.

When updating these dynamic bounds, developers must choose between two methodologies:

- **Worst-case (Pessimistic) Estimation:** The library assumes the maximum possible noise expansion through every homomorphic gate. While mathematically safe, this approach forces the system to calculate massive DP noise margins, severely degrading performance.
- **Average-case (Empirical) Estimation:** To optimize performance, libraries may dynamically estimate noise based on average-case, empirical assumptions. In this model, noise accumulation is treated as a random walk, allowing the system to track the variance of the noise rather than the absolute worst-case bound. This results in much tighter bounds and better output precision.

**The Danger of Non-worst-case Dynamic Estimation** Opting for average-case dynamic estimation is fundamentally flawed in an adversarial environment. In the IND-CPA<sup>D</sup> scenario, the adversary dictates the homomorphic function that the server evaluates.

As we show in Paper II, an attacker can meticulously craft a malicious evaluation circuit that completely violates the statistical assumptions of the average-case noise model. By forcing the internal noise to grow deterministically rather than randomly, the attacker inflates the actual noise far beyond the library’s empirical dynamic estimate. We successfully demonstrated this vulnerability in practice, executing full key-recovery attacks against the dynamic noise estimation routines of major real-world libraries, specifically OpenFHE and HELib.

Consequently, the server calculates an inadequate amount of flooding noise. Because the DP mechanism applies noise dependent on the artificially deflated estimate, it adds drastically different amounts of noise in different scenarios. When the final result is shared, the insufficient flood fails to mask the inflated key-dependent noise. This complete breakdown of the differential privacy countermeasure allows the adversary to perfectly distinguish the outputs and extract the secret key.

In response to these findings, the cryptographic community has sought more resilient noise-bounding paradigms. A notable follow-up work introduces an “application-aware” noise estimation framework for configuring approximate homomorphic encryption [Ale+26]. By adopting this application-aware approach, OpenFHE has since adapted its codebase to mitigate our proposed attack, moving away from vulnerable average-case assumptions to safely configure FHE parameters for practical, shared-decryption use cases.

# Contributions and Conclusions

---

This thesis aims to investigate the practical security of post-quantum cryptographic schemes that have been standardized or were promising candidates for standardization by NIST. Only by identifying concrete implementation vulnerabilities early, and by demonstrating that they can be exploited in realistic settings, can the true security guarantees of these schemes be understood and countermeasures be developed before widespread deployment.

## 6.1 Contributions

This thesis sits at the intersection of cryptographic implementation security, cryptanalysis, and coding theory, with post-quantum key encapsulation mechanisms and homomorphic encryption as its primary subjects. Let us summarize the different areas of the included papers below.

**Side-channel attacks** Papers I, III to V. These papers construct and evaluate side-channel attacks against real or realistic implementations of post-quantum KEMs. Papers I and III target masked software implementations on ARM Cortex-M4 hardware, demonstrating practical key recovery under cross-device profiling conditions. Paper IV introduces the first single-trace attacks achieving full long-term key recovery on a protected HQC implementation. Paper V validates a new attack framework against a microarchitectural side channel on Apple Silicon hardware.

**Cryptanalysis** Papers I to V. All included papers advance the state of the art in the cryptanalysis of post-quantum schemes. Papers I, III to V improve upon existing key-recovery attacks with new techniques rooted in coding theory. Paper II establishes novel key-recovery attacks against approximate homomorphic encryption libraries, showing that deployed noise-flooding countermeasures fail under a non-worst-case noise estimation assumption.

**Coding theory** Papers I, III to V. These papers use coding theory as their central analytical and algorithmic tool. Papers I, III and V model the key-recovery problem as LDPC decoding over secret-key parity checks, leveraging belief propagation for efficient recovery. Paper IV introduces an MDPC decoding formulation for side-channel attacks and a new soft-information variant of information set decoding applicable to code-based cryptography more broadly.

**Code-based cryptography** Papers I and IV. The HQC scheme is the subject of dedicated attacks in both papers. Paper I targets HQC via a timing oracle, while Paper IV introduces fundamentally new passive and active attack strategies that together reduce the number of required side-channel traces to one.

**Lattice-based cryptography** Papers I, III and V. Lattice-based KEMs form an important sub-topic of this thesis. The Kyber scheme (now standardized as ML-KEM) and NTRU-style schemes are subject to key-recovery side-channel attacks in Papers I and III respectively, while Paper V revisits ML-KEM and demonstrates that decryption failure oracles are far more powerful than previously understood.

**Homomorphic encryption** Paper II. The CKKS approximate homomorphic encryption scheme and its open-source implementations, in particular OpenFHE, are the subject of this paper. It demonstrates that relying on average-case or empirical noise estimation in noise-flooding countermeasures is insufficient for KRD security, even when the differential privacy bounds of Li et al. are nominally implemented.

### 6.1.1 Paper I: SCA-LDPC: A Code-Based Framework for Key-Recovery Side-Channel Attacks on Post-Quantum Encryption Schemes

This paper introduces a generic framework for improving key-recovery side-channel attacks on CCA-secure post-quantum KEMs. An important observation is that prior attacks waste most of the information available from each side-channel measurement by targeting one secret coefficient at a time. By instead constructing chosen ciphertexts whose leakage reveals a sparse linear combination of multiple secret coefficients simultaneously (linear combination considers only the sum of coefficients), one obtains a parity-check equation over the secret key. Collecting many such equations across different subsets of coefficients yields a low-density parity-check (LDPC) code, from which the full secret key is recovered via belief propagation using soft information from the side channel.

The framework yields two compounding advantages. First, a *source coding gain* arises because a sum of secret coefficients is distributed more uniformly than any individual coefficient, making each oracle query more informative. This gain is

especially pronounced for code-based schemes such as HQC, whose sparse secret vectors have very low entropy. Second, an *error correction gain* is realized through the LDPC structure itself: correctly recovered positions propagate through the parity-check graph to correct erroneous ones, making the attack robust to inaccurate oracles.

The framework is instantiated and experimentally validated in two settings. The first is a full power analysis attack on a first-order masked implementation of Kyber-768, targeting the `masked_poly_frommsg` function on an ARM Cortex-M4 CPU using the ChipWhisperer platform in a cross-device profiling scenario. An FD oracle is constructed from neural network classifiers with an average accuracy of approximately 95%, and the full long-term secret key is recovered with an average of only 12 power traces. The best prior method using majority voting requires 99 traces under the same conditions, representing an improvement of 86%.

The second instantiation is a timing attack on HQC, using the rejection-sampling timing vulnerability of [Guo+22a] as the underlying PC oracle. Here the LDPC framework operates in source coding mode, exploiting the extremely low entropy of the HQC secret vector to extract far more information per oracle call than was previously possible. For the idealized timing oracle, the new attack requires approximately 10 000 oracle calls to recover the full secret key of hqc-128, compared to 866 000 calls in prior work — an improvement factor of 86.6. A real-world validation on a laptop confirms that full key recovery is feasible within  $2^{18}$  decapsulation calls.

Both attacks are accompanied by simulation results that closely predict real-world performance, and a non-rigorous information-theoretic lower bound analysis shows that the Kyber attack nearly achieves the Shannon limit. The full attack software is released as open source.

### 6.1.2 Paper II: Key Recovery Attacks on Approximate Homomorphic Encryption with Non-Worst-Case Noise Flooding Countermeasures

This paper presents novel key-recovery attacks on Approximate Homomorphic Encryption (AHE) schemes, specifically targeting the CKKS scheme, when noise-flooding countermeasures are implemented using non-worst-case noise estimation. The work builds upon and extends the seminal attack by Li and Micciancio [LM21], which demonstrated that the standard IND-CPA security notion is insufficient for approximate encryption when decryption results are shared, and introduced the stronger IND-CPA<sup>D</sup> security notion to capture the threat of passive adversaries who can observe decryption outputs.

As it turns out, the noise-flooding countermeasures (which adds carefully sized Gaussian noise before releasing a decryption result) are only as secure as the noise bound they are calibrated against. When the noise bound is estimated in a

non-worst-case manner, such as through average-case estimation using the central limit theorem or through empirical measurements, an adversary operating within the IND-CPA<sup>D</sup> model can deliberately violate the assumptions underlying that estimate. Specifically, by submitting a ciphertext that is a  $t$ -fold homomorphic addition of a single fresh ciphertext  $c_0$  to the evaluation oracle, the adversary causes the actual noise to grow as  $t \cdot e$ , while the library’s CLT-based tracker, which assumes independent inputs, estimates it as only  $\sqrt{t} \cdot e$ . The resulting gap renders the flooding noise negligible relative to the true signal, exposing the secret key.

The attack is instantiated and experimentally validated against OpenFHE, the first major open-source FHE library to claim an implementation of the differential privacy countermeasure from [Li+22]. Despite this claim, OpenFHE’s use of average-case noise estimation makes it vulnerable: the full secret key is recovered from a single shared decryption output in under three seconds of oracle computation. The analysis further shows that the attack complexity scales only linearly with the statistical security parameter  $\nu$ , meaning the vulnerability cannot be patched simply by increasing  $\nu$ .

A second line of attack addresses implementations that use empirical noise estimation, such as HELib when the `ctxt.bumpNoiseBound` function is invoked. Here the adversary, unconstrained by the assumptions built into the user’s empirical calibration, can choose a far larger value of  $t$  than was used during estimation and recover the key with equal efficiency.

As an additional finding, the paper demonstrates that replacing random noise with deterministic noise derived from a pseudorandom generator seeded by the ciphertext provides no meaningful additional protection; such countermeasure was suggested in prior work and implemented in HELib. By querying decryption of the family of ciphertexts  $\{i \cdot c_0\}$  rather than  $c_0$  itself, the adversary obtains multiple independent approximate samples of the secret noise  $e$  and averages them to cancel the deterministic flooding term, requiring only  $\Theta(\eta^2)$  decryption queries to recover  $\eta$  bits of the key. The complete attack code is released publicly. The paper concludes that worst-case noise estimation, as prescribed by the differential privacy bounds of [Li+22], is a necessary condition for secure decryption sharing in approximate homomorphic encryption.

### 6.1.3 Paper III: LDPC Strikes Again: New Key-Recovery Chosen-Ciphertext Side-Channel Attacks on NTRU-based KEMs

This paper extends the SCA-LDPC framework from Paper I to NTRU-style Key Encapsulation Mechanisms, a class of schemes whose unique algebraic structure makes such an adaptation non-trivial. While the framework was shown to be highly effective against LWE-based schemes, the decryption procedures of NTRU and NTRU Prime differ fundamentally from those of Kyber and HQC, requiring the derivation of entirely new ciphertext construction techniques before the LDPC machinery can be applied.

The central technical challenge is extracting parity-check information about the secret key polynomial  $\mathbf{f}$  from side-channel measurements during NTRU decapsulation. The approach builds on the collision-based PC oracle construction of Ravi et al. [Rav+22a], which crafts ciphertexts inducing a single collision in the decryption intermediate  $\mathbf{a} = \mathbf{f} \cdot \mathbf{c}$  at a chosen index  $i$ . The new contribution is to augment these ciphertexts with an additional polynomial term  $\ell_3 \cdot (x^{u_1} - x^{u_2} + \dots - x^{u_w})$ , whose interaction with  $\mathbf{f}$  during decryption produces a linear combination  $\beta_i(u_1, \dots, u_w) = \sum_{j=1}^w (-1)^{j-1} \mathbf{f}[i - u_j \bmod n]$  as a sparse parity check over secret key coefficients. By varying the indices  $u_1, \dots, u_w$  across many queries and modeling the collected oracle responses as an LDPC code, the full key polynomial is recovered via belief propagation. Adapting this construction to NTRU Prime requires additional care due to the non-cyclotomic ring  $\mathbb{Z}_q[x]/(x^n - x - 1)$  and the rounding operation in encryption, which introduces an internal noise term that must be carefully bounded to keep false-positive and false-negative collision rates low.

Simulation results across all relevant parameter sets demonstrate substantial improvements over prior work. For `ntruhs2048509` with a perfect oracle, the attack requires 31% fewer traces than [Rav+22a], and 55% fewer traces than [Uen+22a] at oracle accuracy  $\rho = 0.96$ . For NTRU Prime, the improvements are even more pronounced: at  $\rho = 0.96$ , the attack on `sntrup653` and `sntrup1277` requires approximately 59% and 58% fewer traces, respectively, compared to [Uen+22a]. As with Paper I, the relative gain increases as the oracle becomes noisier, reflecting the superior error-correction capability of the LDPC decoder in low-accuracy settings.

The paper also presents the first documented power analysis attack on a first-order masked NTRU implementation [Cor+23a], conducted in a realistic cross-device setting where the profiling device differs from the target device. A neural network classifier is trained on 20 000 power traces of the `masked_pack_S3` function on device  $D_1$ , targeting a leakage in the arithmetic-to-boolean masking conversion `convert_A2B_CGV14`. When applied to an independent device  $D_2$ , the model achieves an oracle accuracy of  $\rho = 0.995$ . The full secret key of `ntruhs2048509` is recovered in 97 out of 100 independent experiments with an average of approximately 1 250 power traces, closely matching simulation predictions. This constitutes the first practical power or electromagnetic side-channel attack against any masked NTRU-style KEM in a cross-device scenario, and demonstrates that first-order masking alone is insufficient protection against this class of attacks. The attack code is publicly available.

#### 6.1.4 Paper IV Single-Trace Key Recovery Attacks on HQC Using Valid and Invalid Ciphertexts

This paper presents the first single-trace side-channel attacks achieving full long-term secret key recovery on a protected implementation of HQC, a code-based

KEM recently selected by NIST for standardization. Prior state-of-the-art practical attacks on the long-term key required a minimum of two traces [DG25a; Pai+25], while single-trace investigations were either limited to simulations [Bai+25] or required as many as 83 traces on a real platform [Mai+25]. The paper closes this gap with two fundamentally distinct attacks, validated on an ARM Cortex-M4 implementation protected by codeword masking.

The first attack, VC-KRA-MDPC, is a passive single-trace method that operates on a single valid ciphertext. Its central contribution is a reframing of the key recovery problem as decoding a moderate-density parity-check (MDPC) code. A parity-check matrix is constructed from the publicly known vectors  $r_1$  and  $r_2$  of the ciphertext, and the unknown secret key components  $x$  and  $y$  are concatenated with their corresponding syndrome to form a codeword. A noisy observation of the syndrome is obtained by exploiting power leakage from the `expand_and_sum` function inside the Reed-Muller decoder, modeled as a binary symmetric channel. A sum-product belief propagation decoder is then initialized with prior LLRs from the known sparsity of the secret key and channel LLRs from the side-channel measurements, producing soft posterior beliefs over all secret key bits. This MDPC framing is substantially more information-theoretically efficient than prior algebraic approaches: the attack requires only approximately 1 000 bits of side-channel information to succeed, compared to roughly 17 000 bits for [Mai+25] and 7 000 bits per trace for [Pai+25].

The second attack, OT-FDA-CK, is an active chosen-ciphertext attack that improves upon the two-trace Full-Decryption (FD) oracle attack of [DG25a]. Rather than probing the secret key components  $x$  and  $y$  in separate queries, the new method crafts a single invalid ciphertext with  $r_1 = X^k$  and  $r_2 = X^l$ , causing the decoder input to depend on the linear combination  $x + y$  simultaneously. The resulting FD oracle query has an output entropy of approximately 191 bits for `hqc-1`, compared to 128 bits per query in prior work, which eliminates the need for a second trace. An offline phase using a genetic algorithm identifies optimal error patterns that maximize the entropy of the Reed-Muller decoding output, and offline templates map each decoder output to conditional probabilities over the combined secret  $x + y$ .

Both attacks share a common post-processing backend: a new ISD variant that exploits the soft reliability information produced by the preceding phase. Rather than selecting information sets uniformly at random, the algorithm restricts sampling to a candidate pool of the most zero-likely bit positions, using a weighted scheme so that more reliable positions are selected with higher probability. This modification yields a substantially better per-trial success probability without increasing the cost of individual trials, and the independence of trials gives a clean analytical model for the total success probability as a function of the computational budget.

Practical validation is performed on the most recent HQC reference implementation [Gab+25] compiled at -O3 on a Cortex-M4 board. Codeword mask-

ing is defeated as a preliminary step by targeting the Galois field multiplications in the Reed-Solomon encoder via Fisher’s linear discriminant analysis, recovering the random mask  $m'$  with negligible error probability. For VC-KRA-MDPC, the full secret key is recovered from a single trace in all 100 independent trials. For OT-FDA-CK, a perfect partial FD oracle is constructed by targeting the `compute_syndromes` function, achieving a 90% key recovery rate after  $2^{10}$  ISD trials and 98% after  $2^{15}$  trials. Both attacks are inherently immune to detection-based countermeasures, and the results demonstrate that codeword masking provides no meaningful protection against either method. The attack source code is publicly available.

### 6.1.5 Paper V: Unlocking the True Potential of Decryption Failure Oracles: A Hybrid Adaptive-LDPC Attack on ML-KEM Using Imperfect Oracles

This paper revisits chosen-ciphertext side-channel attacks on ML-KEM that exploit the Decryption Failure (DF) oracle, challenging the prevailing view that DF oracles are fundamentally less efficient than their binary Plaintext-Checking (PC) counterparts. While prior work required 7 000 perfect oracle queries for a full key recovery on ML-KEM-768 — a factor of 4.49 above the information-theoretic lower bound — this paper demonstrates that the gap between DF oracle attacks and the Shannon limit is not inherent to the oracle type, but rather a consequence of suboptimal query construction.

The central observation is that targeting a linear combination of multiple secret coefficients simultaneously yields more balanced and informative oracle queries than targeting a single coefficient. This idea was already established in Paper I. However, that framework operates on the *sum* of coefficients, which introduces an unavoidable aliasing loss: distinct secret vectors with identical sums, such as  $(0, -1)$  and  $(-1, 0)$ , are indistinguishable. This paper eliminates that loss by targeting the *joint distribution* of a vector of  $\omega$  secret coefficients directly, allowing every possible outcome to be distinguished in principle. The increased granularity of the support — up to  $|\mathcal{B}_{\eta_1}|^\omega$  values for  $\omega$  coefficients — enables substantially more balanced encodings, but comes at the cost of an exponentially larger key-recovery computation. A second key contribution is the *full rotation encoding*, which exploits the negacyclic ring structure of  $R_q = \mathbb{Z}_q[x]/(x^n + 1)$  to obtain  $\omega$  inequalities over the *same* set of  $\omega$  secret coefficients from a single oracle query. By selecting nonzero positions in  $\mathbf{u}'$  and the noise term  $\delta$  at symmetrically spaced indices, all resulting decryption-failure conditions are coupled to an identical block of key coefficients via negacyclic rotation. Finally, the paper exploits a structural advantage of the DF oracle over the binary PC oracle: whereas a PC oracle encodes a single hyperplane predicate, a DF oracle naturally implements a logical OR over multiple inequalities at different ciphertext positions, yielding encodings that are far less structured and therefore easier to balance.

The key-recovery strategy is structured as a two-phase hybrid algorithm. In the first phase, a precomputed set of 7-bit multibit full rotation encodings is applied to disjoint blocks of four secret coefficients, each block queried seven times with independently chosen balanced encodings. Initial posterior probability mass functions over each coefficient are then computed via maximum likelihood. In the second phase, the attack proceeds in adaptive batches: at each iteration, new single-bit *linking* encodings are constructed over adaptively chosen sets of four coefficients, selected to bridge the disjoint groups from the first phase and to maximally reduce remaining uncertainty. After each batch, all accumulated oracle responses coming from the initial phase and all subsequent batches are jointly processed by a belief propagation decoder following the SCA-LDPC framework, which simultaneously integrates new information and corrects errors from noisy queries. The attack terminates with a meet-in-the-middle post-processing step that resolves any remaining coefficients in under one second for a Hamming distance of at most 4.

The information-theoretic analysis formalizes the optimality of balanced encodings via a lemma showing that the mutual information  $I(S; Y) = H_2(r) - H_2(\rho)$  is maximized if and only if the observable bit  $Y$  is balanced (i.e.,  $r = 1/2$ ), establishing that each query can extract at most  $1 - H_2(\rho)$  bits. Simulation results demonstrate that the attack achieves 4.67 bits of information per 7 queries for ML-KEM-768 at  $\rho = 0.95$ , close to the theoretical maximum of 4.995 bits. Full key recovery requires only 2 950 oracle queries for ML-KEM-768 with  $\rho = 0.95$ , a ratio of 1.35 to the Shannon lower bound, outperforming all prior binary PC and DF oracle attacks.

The framework is experimentally validated against the GoFetch microarchitectural side-channel attack, which exploits the Data Memory-Dependent Prefetcher (DMP) present in Apple M-series CPUs to construct a DF oracle. The ciphertext construction is adapted to the DMP's pointer-dereference behavior: the oracle fires when the decrypted message contains a specific target pointer, and the secret-dependent predicate is encoded in whether one bit of the decrypted pointer matches the DMP's sensitive region. To support full rotation encodings in this setting, a workaround is developed for an empirically observed DMP behavior that suppresses prefetches on arrays of repeating pointer values, by constructing the intended decrypted message as an array of four distinct but cache-equivalent pointers. Compared to the original GoFetch attack on ML-KEM-512, which required 100 352 traces and a subsequent 7.5-hour lattice reduction step on a high-performance server, the new attack requires approximately 6 000 traces and no post-processing, achieving a full key recovery in around 25 minutes. Furthermore, whereas the original GoFetch attack cannot scale beyond ML-KEM-512 due to DMP sensitivity constraints leaving too many coefficients unrecoverable for a tractable lattice reduction, the new framework recovers all secret coefficients directly and extends straightforwardly to ML-KEM-768 and ML-KEM-1024. The attack code is publicly available.

## 6.2 Topic relevance

The great truism of cryptanalysis “attacks always get better, they never get worse” is nowhere more consequential than in the context of post-quantum standardization. The schemes studied in this thesis are not academic curiosities; ML-KEM has already been standardized by NIST, and HQC has recently joined it. The window between standardization and widespread deployment is precisely the moment when identifying concrete implementation vulnerabilities matters most, and the efficiency improvements demonstrated in Papers I, III, IV and V directly inform how much protection deployed implementations actually need. Design and implementation issues can only be fixed once they are known, and the value of these results lies as much in what they imply for future hardening as in the attacks themselves.

The choice to validate every attack on real hardware grounds the contributions in practical reality rather than idealized models. This is especially important given that the cryptographic community is simultaneously trying to understand both the theoretical security guarantees of new schemes and the gap between those guarantees and what is achievable in practice. The results of this thesis suggest that gap remains substantial, and that closing it will require both improved countermeasures and a more conservative interpretation of what masking and noise-flooding actually provide.

## 6.3 Lessons learned

During this research project I, the author, have learned advanced and up-to-date specialized knowledge in the fields of side-channel attacks, cryptanalysis, coding theory, code- and lattice-based cryptography, homomorphic encryption as listed in Section 6.1 and evidenced by the papers in Part II.

In a more general sense, I have learned how to conduct research and how to navigate the different methodologies of my research area. I now know how to build up a theoretical framework (Papers I and V), how to apply it to a specific problem domain, and how to generate, aggregate, and select results of relevance for publication. I have learned how to implement simulations and real-world attack experiments in Python, C and a bit of Rust, and how to disseminate those results in a way that is both reproducible and honest about the gap between idealized models and practical conditions. The public release of attack code for every paper in this thesis reflects a commitment to that standard. I have also learned how the cryptographic research community structures its papers: what counts as a contribution, how comparisons with prior work are conducted, and how theoretical claims are supported by simulation and experiment. This kind of knowledge is difficult to acquire from courses alone, and I have learned much of it from my co-authors, from reading a great many papers, and from going through the peer-review process on both sides of the table.

The progression in independence across the five papers has been one of the most formative aspects of the PhD. Paper I was a largely collaborative effort, while Papers III and V required me to take primary responsibility for the research direction, ciphertext constructions, and experimental validation. This journey from contributor, to independent lead, to framework proposer represents what I consider the most important single thing a PhD is supposed to produce. That said, none of the papers would have reached their final form without the experience, feedback, and course-corrections of my supervisors and co-authors, and I do not understate that.

Through not a trivial process of personal growth, I have learned to present talks and communicate about my research orally in English. I have presented at large conferences (Asiacrypt 2023, USENIX Security 2024) and been invited to present at seminars at other universities. I have made successful collaborations with international colleagues.

Teaching as a TA and co-supervising Master's thesis students has improved me as a PhD student in unexpected ways. Explaining ideas to students with varying backgrounds translates directly into giving clearer research talks and writing better paper introductions. Trying to convey a complicated matter in the cryptography or computer security areas in a simple terms certainly raises some questions for yourself, allowing to find and learn new aspects within the foundations.

The question of research ethics has been one I have had to think about seriously rather than abstractly, especially thanks to the USENIX Security conference and their experiment, introducing a special ethics review process for conference in 2026 and later. The idea of deliberately constructing attacks against cryptographic implementations being deployed to protect real data is uncomfortable on the face of it. The appropriate level of trust in a system can only be established by researchers who have genuinely and publicly tried to break it. Vulnerabilities can only be fixed once they are known. Still, this requires handling responsibly: in Paper V we conducted responsible disclosure to Apple and to the designers of ML-KEM prior to publication. Throughout all papers I have relied exclusively on data generated by my own simulations and experiments — a deliberate choice that reflects both scientific integrity and awareness of the responsibilities that come with working in implementation security.

## 6.4 Looking forward

Post-quantum cryptography is no longer a research topic confined to academic papers and standardization committees. The transition is happening now. NIST finalized the first three post-quantum standards in 2024 with another one coming soon. US and the European Union has adopted a coordinated migration roadmap requiring governmental and critical infrastructure operators to begin transitioning their cryptographic systems within the coming years. The schemes studied in this thesis will within a decade be running on billions of devices, from smartphones

and smart cards to cloud servers and automotive control units. The question of whether their implementations are secure is therefore not a theoretical one.

In this light, the results of this thesis should be read not only as a set of attacks, but as an early-warning system. Implementation vulnerabilities in newly standardized schemes are best discovered now, while countermeasures can still be designed into deployment guidelines, before widespread rollout has locked in specific hardware and software choices. The efficiency of the attacks demonstrated here suggests that the margin between a secure and an insecure implementation is narrower than the community may have appreciated.

The SCA-LDPC framework introduced in Paper I and substantially refined in Paper V now achieves a ratio of 1.35 to the Shannon lower bound for ML-KEM-768, which is a meaningful improvement over all prior work. Still, the current attack construction is in part empirical and ad hoc: the encoding strategy is tuned manually, and the framework does not yet adapt automatically to arbitrary noise levels. It would be both theoretically satisfying and practically useful to close the remaining gap more rigorously, perhaps to within 10% of the lower bound, and to develop a principled method for optimizing encodings without manual fine-tuning. This seems like a tractable and worthwhile direction.

Microarchitectural side channels are a direction I expect to grow considerably in importance as post-quantum schemes are deployed. The DMP-based attack in Paper V is one instance of a broader class of hardware-level vulnerabilities that emerge whenever software with secret-dependent memory access patterns runs on processors optimized aggressively for performance. As post-quantum KEMs reach new platforms (e.g., mobile application processors, server-grade CPUs, automotive SoCs) each new microarchitecture brings its own set of hardware features that may interact unexpectedly with cryptographic code. Systematically characterizing which microarchitectural behaviors can be weaponized, and how, is a research agenda that has barely begun.

Trusted Execution Environments (TEEs) present a particularly interesting and critically vulnerable target in this transition. Architectures such as ARM TrustZone, Intel SGX, and RISC-V Keystone are increasingly proposed as the deployment environment for sensitive cryptographic operations, including post-quantum key exchange and decryption. However, evaluating TEEs in a post-quantum context reveals a twofold problem. First, the foundational security mechanisms of current TEEs (e.g., remote attestation and secure channel establishment) still rely almost exclusively on classical asymmetric cryptography, such as ECDSA. This presents a systemic weakness: an attacker with a cryptographically relevant quantum computer could forge attestation quotes to spoof a trusted enclave, or retrospectively decrypt the channels used to provision enclave secrets. Second, even as TEE architectures are updated to support PQC natively, they continue to share microarchitectural resources (including caches, prefetchers, and branch predictors) with untrusted code running on the same chip. Attacks that cross the TEE boundary through microarchitectural side channels represent a qualitatively more dan-

gerous class of vulnerability than conventional power analysis, because they can potentially be launched remotely, without physical access to the device, and at scale. There are open and pressing questions whether the post-quantum schemes examined in this thesis, implemented inside a TEE, are truly safe from such leakage and whether the TEEs themselves can securely transition their attestation mechanisms.

The broader and harder question is protection against power and electromagnetic side-channel attacks. Some schemes are inherently masking-unfriendly (e.g., HQC). But even for schemes where masking is more straightforward in principle, such as ML-KEM, Papers I, III and the broader literature show that first-order masking in software does not provide adequate protection against a determined attacker with cross-device profiling capabilities. Higher-order masking carries a steep performance cost, and hardware masking for post-quantum schemes remains an open engineering and research problem. How to protect these schemes efficiently and provably, on the full range of hardware they will eventually run on, is perhaps the most important practical question left open by this thesis.

# References

---

- [AD97] M. Ajtai and C. Dwork. “A Public-Key Cryptosystem with Worst-Case/Average-Case Equivalence”. In: *29th ACM STOC*. ACM Press, May 1997, pp. 284–293.
- [Ajt98] M. Ajtai. “The Shortest Vector Problem in L2 is NP-hard for Randomized Reductions (Extended Abstract)”. In: *30th ACM STOC*. ACM Press, May 1998, pp. 10–19.
- [Ale+26] A. Alexandru et al. “Application-Aware Approximate Homomorphic Encryption: Configuring FHE for Practical Use”. In: *IACR Communications in Cryptology* 2.4 (Jan. 8, 2026).
- [Ava+19] R. Avanzi et al. “CRYSTALS-Kyber algorithm specifications and supporting documentation”. In: *NIST PQC Round 2.4* (2019), pp. 1–43.
- [Baï+25] C. Baisse et al. “Secret and shared keys recovery on hamming quasi-cyclic with SASCA”. In: *DCC* 93.6 (2025), pp. 2137–2157.
- [Bel+98] M. Bellare et al. “Relations among notions of security for public-key encryption schemes”. In: *Annual International Cryptology Conference*. Springer. 1998, pp. 26–45.
- [Ber+17] D. J. Bernstein et al. *NTRU Prime*. Tech. rep. available at <https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions>. National Institute of Standards and Technology, 2017.
- [BMV78] E. Berlekamp, R. McEliece, and H. Van Tilborg. “On the inherent intractability of certain coding problems (corresp.)” In: *IEEE Transactions on Information theory* 24.3 (1978), pp. 384–386.

- [Che+17] J. H. Cheon et al. “Homomorphic Encryption for Arithmetic of Approximate Numbers”. In: *ASIACRYPT 2017, Part I*. Ed. by T. Takagi and T. Peyrin. Vol. 10624. LNCS. Springer, Cham, Dec. 2017, pp. 409–437.
- [Cor+23a] J.-S. Coron et al. “High-order masking of NTRU”. In: *IACR TCHES 2023.2* (2023), pp. 180–211.
- [DG25a] H. Dong and Q. Guo. “Multi-Value Plaintext-Checking and Full-Decryption Oracle-Based Attacks on HQC from Offline Templates”. In: *IACR TCHES 2025.4* (2025), pp. 254–289.
- [FO99] E. Fujisaki and T. Okamoto. “Secure Integration of Asymmetric and Symmetric Encryption Schemes”. In: *CRYPTO’99*. Ed. by M. J. Wiener. Vol. 1666. LNCS. Springer, Berlin, Heidelberg, Aug. 1999, pp. 537–554.
- [Gab+25] P. Gaborit et al. *Hamming Quasi-Cyclic (HQC)*. <https://pqc-hqc.org>. Version 2025/08/22. HQC Submission Team, Aug. 2025. eprint: 2508.022.
- [Gal62] R. Gallager. “Low-density parity-check codes”. In: *IRE Transactions on information theory* 8.1 (1962), pp. 21–28.
- [Gen09] C. Gentry. “Fully homomorphic encryption using ideal lattices”. In: *41st ACM STOC*. Ed. by M. Mitzenmacher. ACM Press, May 2009, pp. 169–178.
- [GJN20a] Q. Guo, T. Johansson, and A. Nilsson. “A Key-Recovery Timing Attack on Post-quantum Primitives Using the Fujisaki-Okamoto Transformation and Its Application on FrodoKEM”. In: *CRYPTO 2020, Part II*. Ed. by D. Micciancio and T. Ristenpart. Vol. 12171. LNCS. Springer, Cham, Aug. 2020, pp. 359–386.
- [Guo+22a] Q. Guo et al. “Don’t Reject This: Key-Recovery Timing Attacks Due to Rejection-Sampling in HQC and BIKE”. In: *IACR TCHES 2022.3* (2022), pp. 223–263.
- [HHK17] D. Hofheinz, K. Hövelmanns, and E. Kiltz. “A Modular Analysis of the Fujisaki-Okamoto Transformation”. In: *TCC 2017, Part I*. Ed. by Y. Kalai and L. Reyzin. Vol. 10677. LNCS. Springer, Cham, Nov. 2017, pp. 341–371.
- [HPS98] J. Hoffstein, J. Pipher, and J. H. Silverman. “NTRU: A Ring-Based Public Key Cryptosystem”. In: *Third Algorithmic Number Theory Symposium (ANTS)*. Vol. 1423. LNCS. Springer, June 1998, pp. 267–288.

- [Jaq24] S. Jaques. *Quantum Attacks on AES*. Invited talk at CHES 2024 (IACR Workshop on Cryptographic Hardware and Embedded Systems), Halifax, Canada. Slides: [https://ches.iacr.org/2024/Jaques\\_CHEs\\_2024.pdf](https://ches.iacr.org/2024/Jaques_CHEs_2024.pdf). Sept. 2024.
- [Koc96a] P. C. Kocher. “Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems”. In: *Annual international cryptology conference*. Springer. 1996, pp. 104–113.
- [Koc96b] P. C. Kocher. “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems”. In: *CRYPTO’96*. Ed. by N. Koblitz. Vol. 1109. LNCS. Springer, Berlin, Heidelberg, Aug. 1996, pp. 104–113.
- [Li+22] B. Li et al. “Securing Approximate Homomorphic Encryption Using Differential Privacy”. In: *CRYPTO 2022, Part I*. Ed. by Y. Dodis and T. Shrimpton. Vol. 13507. LNCS. Springer, Cham, Aug. 2022, pp. 560–589.
- [LM21] B. Li and D. Micciancio. “On the Security of Homomorphic Encryption on Approximate Numbers”. In: *EUROCRYPT 2021, Part I*. Ed. by A. Canteaut and F.-X. Standaert. Vol. 12696. LNCS. Springer, Cham, Oct. 2021, pp. 648–677.
- [LPR10] V. Lyubashevsky, C. Peikert, and O. Regev. “On Ideal Lattices and Learning with Errors over Rings”. In: *EUROCRYPT 2010*. Ed. by H. Gilbert. Vol. 6110. LNCS. Springer, Berlin, Heidelberg, May 2010, pp. 1–23.
- [Mai+25] N. Maillet et al. “Key Recovery from Side-Channel Power Analysis Attacks on Non-SIMD HQC Decryption”. In: *CRYPTO 2025, Part V*. Ed. by Y. T. Kalai and S. F. Kamara. Vol. 16004. LNCS. Springer, Cham, Aug. 2025, pp. 70–102.
- [Ngo+21a] K. Ngo et al. “A Side-Channel Attack on a Masked IND-CCA Secure Saber KEM Implementation”. In: *IACR TCHES 2021.4* (2021), pp. 676–707.
- [NIS16] NIST. *NIST Post-Quantum Cryptography Standardization*. <https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization>. 2016.
- [OST06a] D. A. Osvik, A. Shamir, and E. Tromer. “Cache Attacks and Countermeasures: The Case of AES”. In: *CT-RSA 2006*. Ed. by D. Pointcheval. Vol. 3860. LNCS. Springer, Berlin, Heidelberg, Feb. 2006, pp. 1–20.

- [Pai+25] T. B. Paiva et al. “Et tu, Brute? Side-Channel Assisted Chosen Ciphertext Attacks Using Valid Ciphertexts on HQC KEM”. In: *Post-Quantum Cryptography - 16th International Workshop, PQCrypto 2025, Part II*. Ed. by R. Niederhagen and M.-J. O. Saarinen. Springer, Cham, Apr. 2025, pp. 294–321.
- [Raj+23] G. Rajendran et al. “Pushing the Limits of Generic Side-Channel Attacks on LWE-based KEMs - Parallel PC Oracle Attacks on Kyber KEM and Beyond”. In: *IACR TCHES 2023.2* (2023), pp. 418–446.
- [Rav+20a] P. Ravi et al. “Generic Side-channel attacks on CCA-secure lattice-based PKE and KEMs”. In: *IACR TCHES 2020.3* (2020), pp. 307–335.
- [Rav+22a] P. Ravi et al. “Will You Cross the Threshold for Me? Generic Side-Channel Assisted Chosen-Ciphertext Attacks on NTRU-based KEMs”. In: *IACR TCHES 2022.1* (2022), pp. 722–761.
- [Reg05] O. Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In: *37th ACM STOC*. Ed. by H. N. Gabow and R. Fagin. ACM Press, May 2005, pp. 84–93.
- [RSU02] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke. “Design of capacity-approaching irregular low-density parity-check codes”. In: *IEEE transactions on information theory* 47.2 (2002), pp. 619–637.
- [RU01] T. J. Richardson and R. L. Urbanke. “The capacity of low-density parity-check codes under message-passing decoding”. In: *IEEE Transactions on information theory* 47.2 (2001), pp. 599–618.
- [Sch+17] J. M. Schanck et al. *NTRU-HRSS-KEM*. Tech. rep. available at <https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions>. National Institute of Standards and Technology, 2017.
- [Sch+22d] P. Schwabe et al. *CRYSTALS-KYBER*. Tech. rep. available at <https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022>. National Institute of Standards and Technology, 2022.
- [Sch18] J. M. Schanck. *A Comparison of NTRU Variants*. Cryptology ePrint Archive, Report 2018/1174. 2018.
- [Sho94] P. W. Shor. “Algorithms for Quantum Computation: Discrete Logarithms and Factoring”. In: *35th FOCS*. IEEE Computer Society Press, Nov. 1994, pp. 124–134.

- [Uen+22a] R. Ueno et al. “Curse of Re-encryption: A Generic Power/EM Analysis on Post-Quantum KEMs”. In: *IACR TCHES* 2022.1 (2022), pp. 296–322.
- [Zha+17] Z. Zhang et al. *NTRUEncrypt*. Tech. rep. available at <https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions>. National Institute of Standards and Technology, 2017.



---

## Included Publications

---



# SCA-LDPC: A Code-Based Framework for Key-Recovery Side-Channel Attacks on Post-Quantum Encryption Schemes

---

## Abstract

Whereas theoretical attacks on standardized cryptographic primitives rarely lead to actual practical attacks, the situation is different for side-channel attacks. Improvements in the performance of side-channel attacks are of utmost importance.

In this paper, we propose a framework to be used in key-recovery side-channel attacks on CCA-secure post-quantum encryption schemes. The basic idea is to construct chosen ciphertext queries to a plaintext checking oracle that collects information on a set of secret variables in a single query. Then a large number of such queries is considered, each related to a different set of secret variables, and they are modeled as a low-density parity-check code (LDPC code). Secret variables are finally determined through efficient iterative decoding methods, such as belief propagation, using soft information. The utilization of LDPC codes offers efficient decoding, source coding, and error correction benefits. It has been demonstrated that this approach provides significant improvements compared to

previous work by reducing the required number of queries, such as the number of traces in a power attack.

The framework is demonstrated and implemented in two different cases. On one hand, we attack implementations of HQC in a timing attack, lowering the number of required traces considerably compared to attacks in previous work. On the other hand, we describe and implement a full attack on a masked implementation of Kyber using power analysis. Using the ChipWhisperer evaluation platform, our real-world attacks recover the long-term secret key of a first-order masked implementation of Kyber-768 with an average of only 12 power traces.

**Keywords:** Lattice-based cryptography, code-based cryptography, side-channel attacks, NIST post-quantum cryptography standardization, low-density parity-check codes.

## 1 Introduction

NIST [NIS16] is running a standardization process (referred to as the NIST PQ project) for post-quantum public-key cryptographic algorithms (PQC schemes), which are supposed to be secure even against attacks from quantum computers. This is not the case for most public-key algorithms in use today [Sho94]. The project started in 2017 and just recently the first choices for standardization were announced. The project is ongoing and round 4 will involve a further examination of additional schemes. All of the schemes in the NIST PQ project are based on a variety of hard problems that are believed to be intractable for quantum computers, and many of them can be categorized as either Public-key Encryption (PKE) or Key Encapsulation Mechanisms (KEMs). These PKE/KEM schemes are based on either the Learning with Errors (LWE) problem as introduced by Regev [Reg05] in 2005 or on code-based problems, initiated in [McE78].

Two such schemes will be considered in this paper. One is CRYSTALS-Kyber [Sch+20], selected by NIST as the candidate for standardization for KEMs. The security of Kyber is based on the Module LWE problem and has strong confidence in its theoretical security, while also offering a good performance. The other scheme is HQC [Agu+20], a code-based round 4 candidate. Other code-based round 4 candidates are BIKE [Ara+20] and Classic McEliece [Alb+20]. NIST has stated that one of the schemes HQC or BIKE may be standardized.

LWE- or code-based PKE/KEMs are usually built to be secure against chosen plaintext attacks (IND-CPA secure) and then transformed to be secure against adaptive chosen ciphertext attacks (IND-CCA secure) by applying some CCA conversion method, such as the Fujisaki-Okamoto (FO) transform. The FO transform involves a re-encryption after decryption, which enables the detection of invalid ciphertexts and correspondingly return failure. Invalid chosen ciphertexts that are not proper encryptions of a message will almost always be rejected by the decryption/decapsulation.

Side-Channel Attacks (SCA) were introduced by Kocher [Koc96b] and are a separate area of research today. For PQC schemes, it is a major concern and NIST also in the later rounds encouraged more research on the security of PQC schemes against side-channel cryptanalysis. In relation to this, there has been great research interest in developing new side-channel attacks on all relevant NIST candidates as well as studying efficient side-channel protection techniques.

There are many different approaches to SCA on PQC schemes. Following previous work, we may roughly classify attacks into two main categories. The first category includes attacks that require either a single trace or at least only a few traces to perform key recovery or message recovery and targets very precise leakages in an implementation. The second category includes attacks of a more generic type, exploiting arbitrary leakages in the implementation of the algorithm, but typically requiring the collection of many traces in the attack phase. These more generic attacks are modeled by instantiating a side-channel oracle for chosen ciphertexts. The oracle is explained in more detail in the following.

## 1.1 Related works

Key-recovery chosen-ciphertext side-channel attacks (KR-CCA-SCA) are attacks where the adversary recovers the secret key in the scheme by using chosen ciphertext calls to the decryption or decapsulation algorithm and getting measurement data from some side-channel.

KR-CCA-SCA attacks on PQC encryption schemes are a well-established research field, as evidenced by numerous publications [DAn+19a; Rav+20a; Ngo+21a; Xu+22; GJN20a; Guo+22a; Ham+21a; GJJ22a; Sch+22a; GLG22a; Rav+22b; She+23a]. These attacks can be classified depending on where the information leakages are detected. The first type of KR-CCA-SCAs [GJN20a; Guo+22a; Rav+20a] exploits leakages from the two added procedures, the re-encryption and ciphertext comparison, of the FO transform, since these two components in the FO transform depend on the decrypted message vector. There are also KR-CCA-SCAs [Ngo+21a; Ham+21a; GJJ22a; Sch+22a; GLG22a] that exploit side-channel leakages from the CPA-secure decryption, where parts of the decryption procedure will directly use the secret key.

In [RR21], Ravi et al. classified side-channel-assisted CCA attacks on lattice-based KEMs into three main categories, plaintext-checking (PC) oracle based attacks [DAn+19a; Rav+20a], decryption-failure (DF) oracle based attacks [GJN20a], and full-domain (FD) oracle based attacks [Ngo+21a; Xu+22]. The classification depends on what kind of answer the oracle gives. In a DF oracle, the oracle answer is simply whether the chosen ciphertext decodes/decrypts to a valid message or not. On the other hand, a PC oracle and an FD oracle require message recovery before key recovery can take place. In a PC oracle, the response of the oracle is whether the chosen ciphertext results in a specific given message upon decryption. In an FD Oracle, the oracle returns the full message that has been decrypted. As

a result, in a PC oracle based attack, it is possible to recover a maximum of one bit of secret information from a single side-channel measurement; however, if the message is of  $m$ -bit length (where  $m$  is 256 for Kyber), it is possible to recover  $m$  bits of secret information with an FD oracle based attack.

Recently, Tanaka et al. in [Tan+22] and Rajendran et al. in [Raj+22] have independently proposed a new type of oracle called multi-values PC oracle. This oracle can extract 8-12 bits of information from a single decapsulation oracle call through multi-class classification. The multi-values PC oracle can be considered as a compromise between the PC oracle and the FD oracle, although it is still much less efficient than the latter.

For general PQC schemes, Ueno et al. [Uen+22b] have shown that all round-3 NIST KEM candidates except for Classic McEliece are vulnerable to KR-CCA-SCAs. However, it was later established in [Sch+22a] that the attack detailed in [Uen+22b] is only applicable to earlier versions of the HQC proposal and not to the recent Reed-Muller-Reed-Solomon (RMRS) version. Schamberger et al. in [Sch+22a] and Goy et al. in [GLG22a] very recently proposed new power side-channel attacks on the RMRS version of the HQC scheme, but their attack only applied to power analysis with leakages from the CPA decryption. In [Guo+22a] a generic PC oracle based attack on the RMRS version of the HQC scheme has been proposed, presented in the format of timing attacks.

One central problem in KR-CCA-SCAs is *identifying a generic approach to optimize the selection of chosen ciphertexts, in order to efficiently extract secret keys from various types of oracles*. The main obstacles arise from two primary sources: (1) the inaccuracies that may occur in the construction of oracles and (2) the non-uniform distribution from which secret symbols are generated. To overcome these challenges, one may need to incorporate concepts from coding theory, particularly in the areas of source coding and error correction. Several early research efforts are made to address these challenges, as documented in [Ngo+21a; Qin+21; She+23a]. But the existing solutions are limited in scope, either because they are restricted to a specific oracle or because they are applicable only to particular types of side-channel leakages. Finally, there is ample room for improvement in terms of attack efficiency.

## 1.2 Contributions

In this paper, we propose a framework named SCA-LDPC to improve the key-recovery side-channel attacks on CCA-secure PQC encryption schemes. The basic idea is to construct chosen ciphertext queries to an oracle that collects information on a set of secret variables in a single query. Then a large number of such queries are considered, each related to a different set of secret variables, and they are modeled as a low-density parity-check code (LDPC code). The secret variables are then determined through efficient iterative decoding methods, such as belief propagation (BP), using soft information.

**New concepts.** The concept of designing chosen-ciphertexts to gather side-channel information in a linear parity check is novel. This approach has the potential to provide both source coding and error correction simultaneously. The reason for this is that the combination of multiple secret entries is typically more closely aligned with the uniform distribution, which allows for more effective extraction of information from a single side-channel measurement. This effect echoes the concept of source coding in information theory, leading us to term it as the *source coding gain* throughout the paper. This gain can result in a substantial improvement for proposals with extremely low entropy secret symbols, as demonstrated for HQC, as well as a noticeable improvement for lattice-based schemes. The error correction gain is realized through the utilization of linear parity checks, which enable the utilization of correctly recovered coefficients to rectify incorrect decisions. The implementation of these linear checks in the form of an LDPC code was selected due to its efficient decoding capabilities and its well-known near-optimal performance from an information-theoretical perspective.

The new framework has significantly transformed the design philosophy of prior methods for source coding and error correction, as documented in [Ngo+21a; Qin+21; She+23a]. The previously proposed methods aimed to achieve full key recovery with higher accuracy by introducing additional measurements for each individual secret symbol, thereby increasing the success rate of symbol recovery. The new framework, however, proposes a novel approach by allowing for fewer measurements on the secret symbols, leading to a higher level of symbol-level errors, which are subsequently corrected by the specially designed LDPC codes through inter-symbol parity checks.

We emphasize that the framework is generic in nature and can be applied to both code-based and lattice-based schemes, across adaptive and non-adaptive attack models, and in a multitude of side-channel leakage scenarios, including timing, cache-timing, power, and electromagnetic leakages. To demonstrate the applicability of the framework, we have instantiated it in two relevant applications: an adaptive timing attack on an HQC implementation with PC oracles, and a non-adaptive power attack on a Kyber implementation with FD oracles. The choice of Kyber and HQC as the primary targets was motivated by their significance, with Kyber being selected as the primary KEM/PKE algorithm for standardization by NIST and HQC still being considered for standardization at the end of round-4. In the full version of the paper [Guo+23a], we extend our discussions to include the methods of attacking Saber and FrodoKEM to further enhance the generality of the framework.

**New results.** We list the contributions of the paper in the following.

- We introduce a code design method in designing capacity-approaching LDPC/QC-LDPC codes over binary and non-binary alphabets. Our method establishes a relationship between oracle calls and parity checks in the LDPC

code, leading to substantial improvements over previous methods in both noiseless and noisy real-world scenarios. The prior improvement is primarily attributed to source coding, while the latter improvement is the result of a combination of source coding and error correction.

- We simulate the performance with different noise levels and characterize the performance of the new approach through a simulation method. The simulated gains are substantial. For example, when the oracle accuracy is 100%, as is the case for the key misuse oracle or an oracle constructed from highly reliable side-channels such as cache-timing leaks on an Intel-SGX platform [Hua+23b], we can recover the secret key of hqc-128 with approximately 9,000 traces in the PC oracle. Using the same oracle setting as the ideal oracle in [Guo+22a], we have achieved an improvement factor of 86.6, as we only need about 10,000 traces, compared to the 866,000 traces reported in [Guo+22a]. This significant improvement is due to the fact that the HQC secret entries are sampled from a distribution with extremely low entropy and the previously known methods (e.g., in [Guo+22a]) ignore the potential source coding gain. In the scenario of perfect FD oracles, the number of traces required to recover Kyber-768 is only 7, which meets the Shannon lower bound.
- We perform actual attacks on two target algorithms, Kyber and HQC, in real-world scenarios. The results of our study demonstrate a close alignment, or even an improvement, of the real attack performance when compared to the simulation results. The first attack is a full power analysis on a masked implementation of Kyber-768. The attack was carried out using the ChipWhisperer framework on the open-source mkm4 library in [Hei+22a] with the profiling and attack phases performed on two distinct boards, both equipped with ARM-Cortex-M4 CPUs. In the real-world scenario, we obtained FD oracles with varying accuracy levels based on their positions. The average accuracy was estimated to be approximately 95%. The full secret key was successfully recovered with an average of 12 traces. In comparison, the simulation required roughly 17 traces for the same oracle accuracy. The better performance in the real-world scenario can be attributed to the availability of soft information and the possibility of some secret symbols having a high accuracy since they are related to a high-accuracy oracle, which in turn helps in the correct decoding of other positions through the parity checks of the specially designed LDPC codes.

The second attack is a full timing attack simulation on HQC, validated with a real-world timing oracle. The real-life attack performance highly depends on the targeted platform. On our laptop with an Intel Core i5 CPU, we can achieve full key recovery against hqc-128 with  $2^{18}$  decapsulation calls.

- The software for attack and simulation is made open-source<sup>1</sup>.

It is important to note that, in accordance with previous research, our approach focuses on recovering the entire secret vector through side-channel leakages. The sample complexity can be reduced by performing additional post-processing procedures, such as information set decoding and lattice reduction [Dac+20], to recover a portion of the secret entries. The specific reduction in the number of traces depends on the permissible amount of computation for post-processing.

**Comparison with previous studies in [Ngo+21a; Qin+21; She+23a].** In [Qin+21], Qin et al. presented an efficient PC oracle based attack on lattice-based schemes by adaptively choosing a new ciphertext for decryption based on the side-channel information obtained from previous power/electromagnetic measurements. Their approach is similar to the well-known Huffman coding method and can result in a source coding gain. Shen et al. in [She+23a] further extended this work by proposing a detection coding method to identify incorrectly recovered positions and to send additional measurements for those secret positions. Note that these studies are limited to PC oracle based attacks on lattice-based schemes and operate in adaptive mode, resulting in a more restrictive attack model and lower efficiency compared to other attacks based on more powerful oracles. For example, when the oracle accuracy is 95%, it was reported in [She+23a] that 3874 traces are required to attack the Kyber-512 scheme; in contrast, the new attack based on the FD oracle presented in this paper only requires 12 traces in a real attack (or 17 traces in simulation) to attack the Kyber-768 scheme. Furthermore, from a viewpoint of information theory, LDPC codes are attractive due to their near-optimal performance. As a result, they are expected to provide improved performance in scenarios where the oracle accuracy is low, compared to the detection codes proposed in [She+23a].

A relevant study [Ngo+21a] has proposed the extended Hamming coding method to enhance the FD oracle based power attack on the masked Saber, a round-3 NIST PQ KEM candidate, in the non-adaptive attack model. However, this approach does not provide any source coding gain and its error correction is limited to an inner-symbol style, resulting in a lack of inter-symbol connections and a less potent error correction mechanism. A more detailed comparison of our work with [Ngo+21a] can be found in Section 6.3.

**Differences from SASCA.** The new framework connects to soft-analytic side-channel attacks (SASCAs) proposed by Veyrat-Charvillon et al. in [VGS14] since both employ iterative decoding. A coding-theoretical treatment on SASCA is presented in [Guo+20]. SASCA has been employed for attacking lattice-based scenarios by exploiting leakages from the number-theoretical transform (NTT) of secret polynomials (e.g., in [PPM17; Ham+21a]).

<sup>1</sup><https://github.com/atneit/SCA-LDPC>

While both attacks use iterative coding, the new SCA-LDPC attack differs from SASCA in several ways. First, the applicability of SASCA extends to both symmetric and asymmetric cryptography; in contrast, the SCA-LDPC attacks fall into the category of PC oracle (and its variants) based KR-CCA-SCAs for lattice-based and code-based systems, presenting unique advantages for these types of systems. Our SCA-LDPC framework employs oracles, thereby being capable of utilizing various types of leakages, including those related to timing that are seldom explored in SASCA. SASCA, mainly used to extract information from physical leakages, can offer an alternative means for generating oracles in contrast to the template attack or the machine learning approaches. The second primary distinction is that SASCA builds its code using pre-existing connections formed by the intrinsic structure of the cipher or implementation, whereas SCA-LDPC provides the attacker with greater flexibility in choosing ciphertexts to create new parity check variables and establish new connections, ultimately constructing a linear code with near-optimal decoding efficiency.

**Relations to concurrent and subsequent works.** The new SCA-LDPC framework is generic in the investigation of side-channel security for post-quantum PKE/KEMs. Given the current relevance of this subject, numerous concurrent and subsequent studies are emerging, such as [Bac+22; Dub+23; Hua+23b]. We detail the connections between these works in the full version of the paper [Guo+23a].

### 1.3 Organization

The remaining parts of the paper are organized as follows. In Section 2, we present the necessary background information. In Section 3, we present a general description of the new attacking framework. We apply the new attack ideas towards Kyber and HQC, in Section 4 and Section 5, respectively. Then, we present the extensive computer simulation results and real-world attacks in Section 6. We finally conclude the paper and present future directions in Section 7.

## 2 Preliminaries

We present the necessary background in this section. We first provide the employed notations and terminology in coding theory, followed by a description of the two KEM candidates, Kyber and HQC. Finally, we conclude this section by outlining the threat model.

### 2.1 Notations and coding terminology

**Notation.** For a finite set  $\mathcal{I}$ , the symbol  $\#\{\mathcal{I}\}$  denotes the number of elements in  $\mathcal{I}$ . Let  $\mathbb{F}_q$  be the finite field of size  $q$ ,  $\lceil x \rceil$  the rounding function, and  $\mathcal{H}$ ,  $\mathcal{G}$ , and  $\mathcal{K}$  three cryptographic hash functions. Let  $\mathcal{R}_q$  be a polynomial ring

$\mathbb{F}_q[x]/(x^{256} + 1)$ . The central binomial distribution  $B_\mu$  outputs  $\sum_{i=1}^\mu (a_i - b_i)$ , where  $a_i$  and  $b_i$  are independently and uniformly randomly sampled from  $\{0, 1\}$ . The Bernoulli distribution  $\text{Ber}_\eta$  defines a random variable from  $\{0, 1\}$ , which is 1 with probability  $\eta$  and 0 otherwise. The notation  $\mathbf{a} \xleftarrow{\$} \mathcal{U}$  denotes that the entries in  $\mathbf{a}$  are randomly sampled from the distribution  $\mathcal{U}$ , where  $\mathbf{a}$  is a vector or polynomial. For a set  $\mathcal{I}$ ,  $\mathbf{a} \xleftarrow{\$} \mathcal{I}$  means that the entries in  $\mathbf{a}$  are uniformly sampled from the set  $\mathcal{I}$  at random. For a vector or polynomial  $\mathbf{a}$ ,  $\mathbf{a}[i]$  refers to the coefficient of  $\mathbf{a}$  at the index  $i$ . The Shannon's binary entropy function of a random variable  $X$  is defined as  $H(X) = -\sum_{x \in \mathcal{X}} \Pr[X = x] \log_2 \Pr[X = x]$ .

**Linear codes.** The Hamming weight of a vector  $\mathbf{x}$  is its number of non-zero elements, denoted by  $w_H(\mathbf{x})$ . We define an  $[n, k, d]_q$  linear code  $\mathcal{C}$  as a linear subspace over  $\mathbb{F}_q$  of length  $n$ , dimension  $k$ , and minimum distance  $d$ . Here minimum distance is defined as the minimum Hamming weight of its non-zero elements. Since a linear code  $\mathcal{C}$  is a subspace, we can define it as the image of a matrix  $\mathbf{G}$ , called a *generator matrix*. We can also define the code  $\mathcal{C}$  as the kernel of a matrix  $\mathbf{H} \in \mathbb{F}_q^{(n-k) \times n}$ . Here  $\mathbf{H}$  is called a parity-check matrix of  $\mathcal{C}$ .

**LDPC codes.** *Low-density parity-check* (LDPC) codes are linear codes with a sparse parity-check matrix first introduced in [Gal62]. LDPC codes can be considered sparse graph codes because they can be decoded efficiently using iterative decoding (such as belief propagation [Pea82]) on the *Tanner graph*, a bipartite graph with edges corresponding to non-zero elements in the parity-check matrix  $\mathbf{H}$ .

**Concatenated codes.** Forney [For65] in 1965 firstly proposed the concatenated code construction approach of combining two simple codes called an *inner code* and an *outer code*, respectively, to achieve good error-correcting capability with reasonable decoding complexity. Let the inner code  $\mathcal{C}_{in} : \mathcal{A}^k \rightarrow \mathcal{A}^n$ , the outer code  $\mathcal{C}_{out} : \mathcal{B}^K \rightarrow \mathcal{B}^N$ , and  $\#\{\mathcal{B}\} = \#\{\mathcal{A}\}^k$ . The concatenated code is a code  $\mathcal{C}_{con} : \mathcal{A}^{kK} \rightarrow \mathcal{A}^{nN}$ . The key of the concatenated code construction method is that the decoding can be done sequentially by passing first the inner code decoder and then the outer code decoder. Typically in the inner code decoding, one can use a maximum-likelihood decoding approach, while the outer code allows efficient decoding in polynomial time (e.g. by employing an LDPC code).

## 2.2 Kyber

Kyber [Sch+20], the KEM version of the Cryptographic Suite for Algebraic Lattices (CRYSTALS), is based on the module Learning with Errors (MLWE) problem and has been solicited as the KEM/PKE standard in the NIST PQ project.

Kyber achieves the IND-CCA security through a tweaked Fujisaki-Okamoto transform [FO99] transforming an IND-CPA-secure PKE  $\text{KYBER.CPAPKE}$  to

Table 1: Parameter sets for Kyber [Sch+20]

|            | $n_{\text{mod}}$ | $d$ | $q$  | $\mu_1$ | $\mu_2$ | $(d_u, d_v)$ |
|------------|------------------|-----|------|---------|---------|--------------|
| Kyber-512  | 256              | 2   | 3329 | 3       | 2       | (10,4)       |
| Kyber-768  | 256              | 3   | 3329 | 2       | 2       | (10,4)       |
| Kyber-1024 | 256              | 4   | 3329 | 2       | 2       | (11,5)       |

an IND-CCA-secure KEM  $\text{KYBER.CCAKEM}$ . The description algorithms of  $\text{KYBER.CPAPKE}$  and  $\text{KYBER.CCAKEM}$  can be found in [Sch+20]. We include a simplified description of these functions in the full version of paper [Guo+23a].

In the following, we define the compression function and the decompression function, i.e.,  $\text{Comp}_q(x, d)$  and  $\text{Decomp}_q(x, d)$ , respectively.

**Definition 1.** *The Compression function is defined as:  $\mathbb{Z}_q \rightarrow \mathbb{Z}_{2^d}$*

$$\text{Comp}_q(x, d) = \left\lfloor \frac{2^d}{q} \cdot x \right\rfloor \pmod{2^d}. \quad (1)$$

**Definition 2.** *The Decompression function is defined as:  $\mathbb{Z}_{2^d} \rightarrow \mathbb{Z}_q$*

$$\text{Decomp}_q(x, d) = \left\lfloor \frac{q}{2^d} \cdot x \right\rfloor. \quad (2)$$

The compression and decompression function can be done coefficient-wise if the input is a polynomial or a vector of polynomials  $\mathbf{x} \in \mathcal{R}_q^d$ . The procedure  $\text{KDF}(\cdot)$  denotes a key-derivation function.

The security parameter sets for the three versions of Kyber, Kyber-512, Kyber-768, and Kyber-1024 are shown in Table 1. In Kyber  $q$  is a prime 3329. Let  $\mathbf{H}_0$  be a *negacyclic* matrix from a vector  $\mathbf{h}_0$ , i.e. the first row is  $\mathbf{h}_0$ , subsequent rows are cyclically shifted, when the value is moved from the last column to the first one, it is multiplied by -1. Let  $d$  denote the rank of the module, set to be 2, 3, and 4, respectively, for Kyber-512, Kyber-768, and Kyber-1024. When sampling from central binomial distribution  $B_\mu$ , Kyber also has two parameters  $(\mu_1, \mu_2)$ , set to be (3, 2) for Kyber-512 and (2, 2) for Kyber-768 and Kyber-1024.

### 2.3 HQC

HQC (Hamming Quasi-Cyclic) [Agu+20] is one of the main code-based IND-CCA-secure KEMs in the NIST PQ project, which has advanced to the fourth round. Its security is based on the hardness of decoding a random quasi-cyclic code in the Hamming metric. In HQC, the base field is  $\mathbb{F}_2$  and  $\mathcal{R}_2$  denotes the polynomial ring  $\mathbb{F}_2[x]/(x^n - 1)$ . The multiplication of two polynomials  $\mathbf{u}, \mathbf{v} \in \mathcal{R}_2$  can be

represented as a vector and a *circulant matrix*, induced from a vector in  $\mathbb{F}_2^n$ . Given  $\mathbf{y} = (y_1, y_2, \dots, y_n) \in \mathbb{F}_2^n$ , its corresponding circulant matrix is defined as

$$\mathbf{rot}(\mathbf{y}) = \begin{pmatrix} y_1 & y_n & \cdots & y_2 \\ y_2 & y_1 & \cdots & y_3 \\ \vdots & \vdots & \ddots & \vdots \\ y_n & y_{n-1} & \cdots & y_1 \end{pmatrix}.$$

We can write the multiplication of  $\mathbf{u}\mathbf{v}$  as  $\mathbf{u} \cdot \mathbf{rot}(\mathbf{v})^\top$  or  $\mathbf{v} \cdot \mathbf{rot}(\mathbf{u})^\top$ . The transpose of the circulant matrix is the counterpart of the negacyclic matrix.

The detailed description of the IND-CPA-secure PKE version of HQC and the IND-CCA-secure KEM version can be found in the HQC reference document [Agu+20]. We list them in the full version of paper [Guo+23a]. The procedure  $\text{KeyGen}(\cdot)$  randomly generates two private vectors  $\mathbf{x}, \mathbf{y} \in \mathcal{R}_2$  with a low Hamming weight  $w$  as the private key. It also generates a random public vector  $\mathbf{h} \in \mathcal{R}_2$ , computes  $\mathbf{s} = \mathbf{x} + \mathbf{h} \cdot \mathbf{y}$ , and returns  $(\mathbf{h}, \mathbf{s})$  as the public key. The scheme employs a linear code  $\mathcal{C}$  with a generator matrix  $\mathbf{G}$  and generates noise  $\mathbf{e}, \mathbf{r}_1, \mathbf{r}_2 \in \mathcal{R}_2$  with low Hamming weight in the encryption. The encryption function computes  $\mathbf{u} = \mathbf{r}_1 + \mathbf{h} \cdot \mathbf{r}_2$  and  $\mathbf{v} = \mathbf{m}\mathbf{G} + \mathbf{s} \cdot \mathbf{r}_2 + \mathbf{e}$  and returns  $(\mathbf{u}, \mathbf{v})$  as the ciphertext. In decryption, the secret vector  $\mathbf{y}$  is an input and it computes

$$\mathbf{v} - \mathbf{u} \cdot \mathbf{y} = \mathbf{m}\mathbf{G} + \underbrace{\mathbf{s} \cdot \mathbf{r}_2 - \mathbf{u} \cdot \mathbf{y}}_{\hat{\mathbf{e}}} + \mathbf{e}. \quad (3)$$

Since  $w_H(\hat{\mathbf{e}})$  is small, the decryption function inputs  $\mathbf{v} - \mathbf{u} \cdot \mathbf{y}$  to the decoder of  $\mathcal{C}$  and can succeed with high probability.

The parameter sets of HQC are shown in Table 2. In the recent version published in June 2021, HQC employs a concatenation of outer  $[n_1, k_1, n_1 - k_1 + 1]_{256}$  Reed-Solomon (RS) codes and inner duplicated Reed-Muller (RM) codes built from the first-order  $[128, 8, 64]_2$  Reed-Muller code. The encoding procedure first encodes a message  $\mathbf{m} \in \mathbb{F}_2^{8k_1}$  to a codeword  $\hat{\mathbf{m}} \in \mathbb{F}_2^{n_1}$  of the employed shortened Reed-Solomon codes. It then maps each byte of  $\hat{\mathbf{m}}$  to a codeword of the first-order RM and repeats the RM codeword for 3 or 5 times depending on the security level to obtain a duplicated RM codeword in  $\mathbb{F}_2^{n_2}$ . In summary, we employ a linear code  $\mathbf{m}\mathbf{G} \in \mathbb{F}_2^{n_1 n_2}$ . The HQC proposal makes all computations in the ambient space  $\mathbb{F}_2^n$  and truncates the remaining  $n - n_1 n_2$  useless bits.

The IND-CCA security of the KEM version of HQC is achieved by the Hofheinz-Hövelmanns-Kiltz (HHK) transform [HHK17].

## 2.4 Threat model

We consider a side-channel-assisted chosen-ciphertext attack on a KEM's decapsulation algorithm, where the attacker selects ciphertexts and observes specific

**Table 2:** The HQC parameter sets [Agu+20]. The inner code is the duplicated Reed-Muller code defined by the first-order  $[128, 8, 64]_2$  Reed-Muller code.

| Instance | RS-S  |       |          | Duplicated RM |       |          | $n_1 n_2$ | $n$    | $\omega$ | $\omega_r = \omega_e$ |
|----------|-------|-------|----------|---------------|-------|----------|-----------|--------|----------|-----------------------|
|          | $n_1$ | $k_1$ | $d_{RS}$ | Mult.         | $n_2$ | $d_{RM}$ |           |        |          |                       |
| hqc-128  | 46    | 16    | 31       | 3             | 384   | 192      | 17 664    | 17 669 | 66       | 75                    |
| hqc-192  | 56    | 24    | 33       | 5             | 640   | 320      | 35 840    | 35 851 | 100      | 114                   |
| hqc-256  | 90    | 32    | 49       | 5             | 640   | 320      | 57 600    | 57 637 | 131      | 149                   |

side-channel data, such as timing [GJN20a], cache-timing [Hua+23b], or power/electromagnetic leakages [Rav+20a], from the targeted device, which can be a high-end CPU, low-end CPU (e.g., ARM cortex-M4), or hardware device.

Specifically, we assume that a communication party Alice is using her device for key establishment. An adversary called Mallory sends selected ciphertexts to Alice to recover Alice’s long-term secret key. Alice runs the decapsulation algorithm and Mallory will fail if the used KEM algorithm is IND-CCA secure. However, the designed side-channel-assisted CCAs can make the attack successful after a few such attempts, using the observed side-channel leakages.

This side-channel-assisted CCA attack model is well-established – it is stated in [Uen+22b] that all the NIST round-3 KEM candidates except for Classic McEliece are vulnerable to such attacks exploring leakages from FO transform. The basic idea is to construct a PC oracle outputting whether  $\mathbf{Dec}(c') \stackrel{?}{=} m$ , where  $c'$  is the chosen ciphertext and  $m$  is a message vector.

Finally, the attacker recovers the long-term secret keys based on the output of the PC oracle. Since the PC oracle is generally built from measurements of side-channel leakages, it cannot be 100% correct, in practice. We denote the accuracy of the constructed PC oracle  $\rho$ , i.e., the oracle outputs the right decision with probability  $\rho$  and the wrong one with probability  $1 - \rho$ .

Note that we are discussing general methods for near-optimal CCA SCAs. This new coding-theoretical approach for reduced sample (trace) complexity can be applied in various side-channel attacks on various platforms, while the starting oracle accuracy  $\rho$  can be different. A PC oracle with 100% correctness ( $\rho = 1$ ) can also be connected to a key misuse attack model, as described in [Qin+21].

**Profiled power/EM attacks.** Specific to power/EM attacks, we mainly consider a profiled setting that the adversary has a similar but different device to perform training activities. Though the adversary has no access to the secret key in the targeted device, the secret key in the training device can be freely set. We can also apply the new idea to non-profiled attacks that can build the required abstract oracles online, but the sample complexity analysis will be different.

**Comparison with the adaptive model in power/EM attacks.** The studies [Qin+21; Raj+22; She+23a] proposed efficient adaptive KR-CCA-SCAs on lattice-based proposals. This attack model allows the adversary to select a new chosen ciphertext based on information obtained from previous power/EM traces, which can be employed for source coding on secret coefficients. This approach can result in reduced sample complexity close to the lower Huffman or Shannon bounds. However, this attack model is strong for many practical (say IoT) applications since the adversary needs to have good connections with the device measuring the power/EM leakages and good computation capability to instantly process the obtained traces. We highlight that our new SCA-LDPC attack framework eliminates the requirement and offers source coding gain in a non-adaptive attack model.

### 3 The SCA-LDPC Attack Framework

This section presents a new idea of incorporating LDPC codes and soft information to design chosen ciphertexts and improve previously established KR-CCA-SCAs for CCA-secure post-quantum Key Encapsulation Mechanisms (KEMs) and encryption schemes. We propose a novel technique to extract, from a single side-channel measurement, information regarding a low-weight parity check of the secret coefficients, as opposed to information regarding a single coefficient in previous methods. The sparse system is then solved using iterative decoding methods, such as belief propagation. This new approach enables the attainment of both source coding benefits and error correction advantages. This is due to the combination of several secret coefficients, which leads to a more uniform extraction of information from a single trace. Additionally, the correct recovery of coefficients facilitates the correction of erroneous decisions through spare parity-check relations. The adoption of this new method significantly reduces the number of necessary side-channel measurements. We call the new attack strategy a framework as it is generic and can be applied to both code-based and lattice-based schemes, in a multitude of side-channel leakage scenarios including timing, cache-timing, power, and electromagnetic leakages.

We start this section by assuming the availability of a well-designed LDPC code with specific dimensions and proceed to explain its utilization for improved side-channel information extraction. We then in Section 3.2 present a simple method for constructing such linear codes, the effectiveness of which will be demonstrated through experiments in Section 6. In addition, we broaden the framework by introducing a concatenated construction, where the LDPC codes are utilized as the outer code. This construction is particularly efficient for lattice-based schemes that feature a large alphabet size or for scenarios where the accuracy of the oracle constructed from side-channel measurements is limited.

### 3.1 New attack idea

Given a good linear code with a sparse parity-check matrix  $\mathbf{H}_{r \times n}$ , there are  $k = n - r$  secret positions to recover. In lattice-based and code-based KEM proposals, the value  $k$  is usually divided into  $b$  blocks, each of which has the size of  $k/b$ . We add the constraint that  $\mathbf{H}$  should have the form of

$$\mathbf{H} = [\mathbf{H}_{r \times k} \mid -\mathbf{I}_{r \times r}].$$

Our goal is to recover the first  $k$  secret entries  $s_i$ . One parity-check equation, i.e., one row in the parity-check matrix  $\mathbf{H}$ , will introduce one check variable  $c_i$  for  $i \in \{k+1, \dots, k+r\}$ . We can rewrite each parity-check equation as  $c_i = \sum_{j \in \mathcal{I}} s_j$  and the size of  $\#\{\mathcal{I}\}$  is small since the matrix  $\mathbf{H}$  is of low density.

The secret entries  $s_i$  are typically generated according to a certain secret distribution. For example, in the lattice-based scheme Kyber, the secret entries are generated from the central binomial distribution  $B_\mu$ ; in the code-based scheme HQC, the secret vector is very sparse and each secret entry can be viewed as a Bernoulli variable  $\text{Ber}_\eta$ , where  $\eta$  is a small positive number. The secret distribution can be utilized as the prior information for  $s_i$ . Moreover, additional information can be obtained through the implementation of side-channel measurements of  $s_i$ , which subsequently updates the relevant distribution. This approach is particularly beneficial in lattice-based scenarios. We then design new ciphertexts to obtain side-channel leakages of sparse linear combination  $c_i$  of  $s_j$  for  $j \in \mathcal{I}$ . The side channel information could reveal an empirical probability of  $c_i$ . The problem of recovering all  $s_i$  for  $i \in \{1, \dots, k\}$  is transformed into a coding problem through a noisy discrete channel. Note that the design method for ciphertexts that can reveal partial information of  $c_i$ , is unique to each proposed scheme and differs between lattice-based and code-based schemes. This ciphertext design is one main technical challenge in the proposed attack framework.

**Explanation.** The attack idea is illustrated in Figure 1. We assume that six secret coefficients or variables  $s_i$  for  $1 \leq i \leq 6$  need to be recovered. For each  $s_i$ , we can use the a priori distribution (e.g., in the HQC case), or we have more traces or oracle calls to get a better knowledge of its distribution (e.g., in the Kyber case). We show 4 parity checks in this example, and each check connects to a new variable  $v_i$ . From side-channel measurements or oracle calls, we got additional information about these variables. Thus, we could assign the corresponding distribution to these variables and build a Tanner graph as in Figure 1. With this sparse bipartite Tanner graph, we perform iterative decoding to recover the desired secret coefficients  $s_i$  for  $1 \leq i \leq 6$ .

**The gain of using LDPC codes.** It is essential to select a sparse graph code that facilitates efficient decoding and renders the key recovery procedure computationally feasible. Therefore, it is natural to examine LDPC codes that have favorable

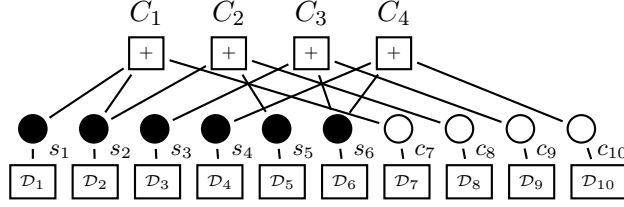


Figure 1: The Tanner graph explanation.

characteristics from an information-theoretic point of view. We introduce the variables  $c_i$ , which are sparse linear combinations of the secret coefficients  $s_j$ , thereby facilitating a more efficient extraction of information from a single side-channel measurement. This is due to the fact that the distribution of  $c_i$  is typically closer to a uniform distribution compared to the distribution of  $s_i$ , resulting in substantial source coding gains, particularly in the case of HQC and to a significant extent in Kyber. Further discussions regarding these source coding gains will be presented in Section 6. Finally, LDPC codes can offer close to optimal error correction performance, rendering the attack framework efficient in terms of the number of side-channel measurements required, even when the oracle constructed from side-channel leakages is highly inaccurate.

**Example 1** (The source coding gain for hqc-128). *In hqc-128, the length of  $\mathbf{y}$  is  $n = 17669$  and the Hamming weight of  $\mathbf{y}$  is  $w_H(\mathbf{y}) = 66$ . Hence, we can approximate each position of  $\mathbf{y}$  as a Bernoulli distribution  $\text{Ber}_\eta$ , where  $\eta \approx 0.0037$ . Assume that we have a perfect oracle to inform us of the value of one position from one oracle call. If we try to recover a bit in  $\mathbf{y}$  by one oracle call, with Shannon's binary entropy function, the obtained information is bounded by 0.0352 bit. If we xor 50 i.i.d. secret positions (as we did later in Section 6) and try to recover the new random bit from one oracle call, then we can instead obtain 0.6255 bit of information. Thus, from an information-theoretical perspective, the new framework is much more advantageous.*

**The ciphertext selection.** If PC oracle variants can be developed to directly reveal information about a single secret symbol (see [Uen+22b] for example), the framework can be utilized to extract information about linear combinations of multiple symbols. Therefore, the new attack framework can be applied to enhancing attacks on LWE/LWR-based schemes, NTRU-based schemes, and HQC-like schemes.

In the case of LWE/LWR-based schemes and HQC-like schemes, the ciphertexts are made up of two components, one of which is directly multiplied by the secret key in decryption. Prior attack strategies select a single non-zero position in this component (a matrix or a vector) to extract information about a single

secret symbol. We could instead select a small number of non-zero positions to recover information about linear combinations. The specific implementations for NTRU-based schemes are more intricate, necessitating further research.

### 3.2 Code generation

It has been demonstrated in previous research [RU08] that random sparse linear codes exhibit superior decoding performance and specific classes of Low-Density Parity-Check (LDPC) codes, such as [RSU01], can attain error-correction capabilities that approach the Shannon capacity. In this work, we present a straightforward code construction method that has shown remarkable results in our experiments.

We first borrow the concept of distance spectrum from [GJS16].

**Definition 3** (Distance Spectrum [GJS16]). *For a binary vector  $\mathbf{h} \in \mathbb{F}_2^{n_0}$ , we define its distance spectrum  $D(\mathbf{h})$  as*

$$D(\mathbf{h}) = \{d : 1 \leq d \leq \lfloor n_0/2 \rfloor, d \text{ classified as existing in } \mathbf{h}\},$$

where "existing in  $\mathbf{h}$ " means there are two ones in  $\mathbf{h}$  with distance  $d$  or  $(n_0 - d)$  inbetween. A distance  $d$  can appear many times in the distance spectrum of a given bit pattern  $\mathbf{h}$ . We call this number the multiplicity of  $d$ .

In our new attack, we first generate QC-LDPC codes with  $mb$  blocks of the parity-check matrix

$$\mathbf{H}_{\text{ini}} = \begin{bmatrix} \mathbf{H}_{11} & \cdots & \mathbf{H}_{1b} \\ \vdots & \ddots & \vdots \\ \mathbf{H}_{m1} & \cdots & \mathbf{H}_{mb} \end{bmatrix},$$

where  $\mathbf{H}_{ij}$  is the circulant matrix (or the negacyclic matrix in the  $q$ -ary case) generated from a binary vector  $\mathbf{h}_{ij}$  for  $1 \leq i \leq m, 1 \leq j \leq b$  with a low Hamming weight. We generate the vectors  $\mathbf{h}_{ij}$  randomly with the constraint that only distances of multiplicity 1 are allowed in its distance spectrum. This can be done with high probability since the constructed LDPC codes are sparse. The key point in the design is that a length-4 cycle occurs in the associated Tanner graph if the multiplicity of a distance in the distance spectrum is larger than 2. By avoiding such patterns in a block, we can avoid many length-4 cycles; such attempts can improve the decoding performance as short cycles can substantially hurt the decoding performance [RU08].

We select  $r$  rows of  $\mathbf{H}_{\text{ini}}$  (randomly or according to certain rules) to form a sub-matrix  $\mathbf{H}'$  and append  $-\mathbf{I}_{r \times r}$ , where  $\mathbf{I}_{r \times r}$  is the identity matrix. Thus, the parity-check matrix of the final generated code is

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}'_{r \times n_0 b} & -\mathbf{I}_{r \times r} \end{bmatrix} \quad (4)$$

**Concatenated code construction.** The LDPC codes generated from the above simple approach can serve as the outer code in the concatenated construction. The inner code can be any linear code such as a repetition code, the extended Hamming codes, and a further concatenation of the extended Hamming codes and repetition codes in [Ngo+21a]. Moreover, we can include a soft-input-soft-output decoder (e.g., in [JZ98]) to utilize the soft-information. Note that in the soft-decoding procedure (e.g., the BP algorithm) of the outer code, only a distribution of each secret coefficient random variable is required; we could thus employ a code with an efficient maximum likelihood decoding procedure as the inner code allowing an efficient calculation of the soft output of the coefficient distribution.

In summary, such concatenated code construction enhances decoding capability and also balances decoding complexity, as the decoding of both outer and inner codes is efficient. This construction is particularly effective for lattice-based proposals or when the side-channel oracle exhibits a low level of accuracy.

## 4 Application to Kyber

In this section, we outline the details of how the new SCA-LDPC framework can be applied to Kyber. The attack is more effective for Kyber if we have side-channel leakages for both  $s_i$  and  $c_j$ . We demonstrate how to obtain these leakages, construct inner codes for them, and apply the outer LDPC decoder.

### 4.1 Basic key recovery attack

In the following, we explain the basic attack to obtain side-channel information about secret coefficients  $s_i$ . We focus on Kyber-768 mostly because the new protected implementation [Hei+22a] that we target supports only this set of parameters. For Kyber-768, the secret key is  $\mathbf{s} = (s_0, s_1, s_2)$ , a ciphertext is a pair  $(\mathbf{u}', \mathbf{v}')$ . To decrypt a ciphertext, one computes  $\mathbf{m} = \mathbf{Comp}_q(\mathbf{v} - \mathbf{s}^T \mathbf{u}, 1)$ , where  $\mathbf{u} = \mathbf{Decomp}_q(\mathbf{u}', d_u) = (\mathbf{u}_0, \mathbf{u}_1, \mathbf{u}_2)$ ,  $\mathbf{v} = \mathbf{Decomp}_q(\mathbf{v}', d_v)$ . The common practice [Rav+20a] is to choose a ciphertext that leads to  $\mathbf{m} = (0, 0, \dots, 0)$  or  $\mathbf{m} = (1, 0, \dots, 0)$ . In other words, all bits of the message are fixed to 0 except the first one. This can be done, for example, by setting  $\mathbf{u}_0 = (k_u, 0, \dots, 0)$ ,  $\mathbf{u}_1 = \mathbf{u}_2 = \mathbf{0}$  and  $\mathbf{v} = (k_v, 0, \dots, 0)$ , where  $k_u, k_v$  are some numbers modulo  $q$ . In this case, the message bits are subject to the following equation.

$$\mathbf{m}[i] = \begin{cases} \mathbf{Comp}_q(k_v - k_u \cdot s_0[0], 1), & i = 0 \\ \mathbf{Comp}_q(k_u \cdot s_0[i], 1), & i \geq 1 \end{cases} \quad (5)$$

By choosing appropriate values for  $k_u$  and  $k_v$ , it is possible to force  $\mathbf{m}[i]$  to always be zero for  $i \geq 1$ , while the value of  $\mathbf{m}[0]$  depends on the first secret coefficient  $s_0[0]$ . Since secret coefficients for Kyber-768 are taken from the range  $[-2, \dots, 2]$ , some of the coefficients are encoded as 0, while others are encoded

as 1. We can use several such ciphertexts with (possibly) different  $k_v$  and/or  $k_u$  to get an inner code of longer length. This way, using an oracle that distinguishes message  $(1, 0, \dots, 0)$  from  $(0, 0, \dots, 0)$ , the attacker can get the distribution of a secret coefficient closer to the real value the more ciphertexts he uses.

There are restrictions for the values  $k_u$  and  $k_v$ : (1) these values are taken from the image of  $\mathbf{Decomp}_q$ ; (2)  $k_u$  is chosen in a way such that  $\mathbf{Comp}_q(k_u \cdot s, 1) = 0$  for any secret coefficient  $s$  (follows from Equation (5)). Thus, one cannot use any code; even though it is possible to encode each secret coefficient with only  $\lceil \log_2(5) \rceil = 3$  bits, for any fixed in-advance combination of 3 ciphertexts one cannot fully determine an arbitrary secret coefficient even with perfect oracle.

One way to solve this problem [She+23a] is to choose ciphertexts adaptively based on the output of the oracle, but we take a different approach. Consider an FD oracle based attack, i.e., assume that we have a set of oracles  $(\mathcal{O}_i)_{i \in (0..n-1)}$ , where  $n$  is the length of the message. Given a ciphertext, the oracle  $\mathcal{O}_i$  says if  $\mathbf{m}[i] = 1$  or not. Essentially, the attacker calls all of these oracles at once, giving them the same ciphertext, this way he can get information about the whole message to be decrypted, the scenario is the same as in [Ngo+21a]. The attacker can create a ciphertext in the following way, set  $\mathbf{u}_0 = (k_u, 0, \dots, 0)$ ,  $\mathbf{u}_1 = \mathbf{u}_2 = \mathbf{0}$ , and  $\mathbf{v} = (k_v, k_v, \dots, k_v)$ , then

$$\mathbf{m}[i] = \mathbf{Comp}_q(k_v - k_u \cdot s_0[i], 1), \quad (6)$$

i.e.,  $i^{\text{th}}$  bit of the message depends on  $s_0[i]$ . Thus, from one ciphertext the information about the block of 256 coefficients  $s_0$  can be obtained. Since there is no more restriction on  $\mathbf{m}[i] = 0$  for  $i \geq 1$ , the amount of possible inner codes increases greatly. Table 3 shows an inner code from three ciphertexts built from  $(k'_u, k'_v)$  pairs, this code can be used to fully determine 256 secret coefficients with perfect oracles. Note that to create an actual ciphertext  $(\mathbf{u}', \mathbf{v}')$  we need a pair  $(k'_u, k'_v)$  that maps to  $(k_u, k_v)$  with coefficient-wise function  $\mathbf{Decomp}_q$ . The next block of secret coefficients  $s_1$  can be retrieved by setting  $\mathbf{u}_0 = \mathbf{0}$ ,  $\mathbf{u}_1 = (k_u, 0, \dots, 0)$ ,  $\mathbf{u}_2 = \mathbf{0}$  and so on. Note that the attacker could choose different values in  $\mathbf{v}$ , this way different encodings can be used for different message bits (although all those encodings should have the same  $k_u$ ) and this potentially opens up the possibility of the adaptive attack. However, such an attack is more complicated since the set of allowed encodings given the fixed  $k_u$  is quite limited, and the attacker has to choose the same  $k_u$  for all 256 coefficients. We leave it as a potential follow-up work and focus on the situation where for all message bits there is a fixed in-advance encoding to be used.

The common approach in the literature is to use just an inner code for secret coefficients (without outer code) that makes the probability of getting the wrong coefficient to be very small (with real imperfect oracles), such that the probability to get all secret coefficients correctly is close to 1. In our approach, however, we use a shorter inner code that is not sufficient by itself, for example in our real attack

**Table 3:** Example of an inner code for the secret coefficients. Each value from the range  $[-2, \dots, 2]$  is encoded with 3 bits (columns of the table), therefore, the secret coefficient could be fully determined with just 3 oracle calls given that the oracle is perfect.

| $(k'_u, k'_v)$ | Secret coefficient |    |   |   |   |
|----------------|--------------------|----|---|---|---|
|                | -2                 | -1 | 0 | 1 | 2 |
| (630, 14)      | 0                  | 1  | 0 | 1 | 1 |
| (706, 6)       | 0                  | 0  | 1 | 1 | 0 |
| (706, 10)      | 0                  | 1  | 1 | 0 | 0 |

from Section 6.1 we encode each secret coefficient with only 2 bits and encode the values  $-2$  and  $2$  the same way, i.e., with only inner code it is impossible to differentiate between these values.

**How to choose inner code.** For the fixed in-advance code length  $\ell$  we want to create an inner code  $C_\ell$  that maximizes the information we get from the oracles with accuracy  $\rho$ . We solve this problem by considering the entropy of secret coefficients. Initially, each of them is distributed according to  $B_\mu$ , whose entropy is  $H(B_\mu) \approx 2.03$ , for  $\mu = 2$ . Each value  $s \in B_\mu$  is encoded as  $C_\ell(s)$  – a binary string of length  $\ell$ . Given an output string  $\mathbf{y}$  of length  $\ell$  from an oracle (note that  $\mathbf{y}$  can be different from every  $C_\ell(s)$ ,  $s \in B_\mu$ ), consider the probability  $\Pr[B_\mu = s \mid \mathbf{y}]$  for each  $s \in B_\mu$ . As an example from Table 3,  $\Pr[B_\mu = 0 \mid 011] = 1$  for the perfect oracle, but it is less than 1 for an oracle with  $\rho < 1$  since we could have reached this  $\mathbf{y}$  from another coefficient.

To avoid ambiguity, we denote  $\mathbf{y}_\rho$  as the output of the oracle with the accuracy  $\rho$ . The conditional distribution  $B_\mu \mid \mathbf{y}_\rho$  can be naturally defined as  $\Pr[(B_\mu \mid \mathbf{y}_\rho) = s] = \Pr[B_\mu = s \mid \mathbf{y}_\rho]$ . Now, the difference between the entropy values  $H(B_\mu) - H(B_\mu \mid \mathbf{y}_\rho)$  shows how much information the output  $\mathbf{y}_\rho$  gives. To assess how good the code is, we can compute the expectation of this information as

$$I(C_\ell) = \sum_{\mathbf{y}_\rho \in \{0,1\}^\ell} (H(B_\mu) - H(B_\mu \mid \mathbf{y}_\rho)) \cdot \Pr[Y = \mathbf{y}_\rho],$$

where  $Y$  is a random variable that describes the output of an oracle with accuracy  $\rho$  on a random secret coefficient. The probability of the specific oracle's output is computed as follows.

$$\Pr[Y = \mathbf{y}_\rho] = \sum_{x \in \text{supp}(B_\mu)} \rho^{d(\mathbf{y}_\rho, C_\ell(x))} (1 - \rho)^{\ell - d(\mathbf{y}_\rho, C_\ell(x))} \Pr[B_\mu = x],$$

where  $d(\cdot, \cdot)$  is the Hamming distance. To decode a received word  $\mathbf{y}_\rho$ , one computes conditional probability  $B_\mu \mid \mathbf{y}_\rho$  of secret coefficient, i.e. we use maximum-

likelihood decoding approach.

## 4.2 Improving the attack using LDPC

The basic attack allows us to compute the conditional distribution for each secret coefficient using the inner code. Now, following our framework, we create an outer LDPC code. For it to work, we also need a way to get information about parity checks  $c_i$ . Let us describe how to create a ciphertext corresponding to a parity check. Consider an example: Let  $\mathbf{u}_1$ ,  $\mathbf{u}_2$  and  $\mathbf{v}$  be as above, but  $\mathbf{u}_0 = k_u + k_u x^2$ , then

$$\mathbf{s}^T \mathbf{u} = k_u \left( (s_0[0] - s_0[n-2]) + (s_0[1] - s_0[n-1])x + (s_0[2] + s_0[0])x^2 + \dots \right).$$

Looking at the first message bit

$$\mathbf{m}[0] = \text{Comp}_q(k_v - k_u \cdot (s_0[0] - s_0[n-2]), 1)$$

and comparing it to Equation (6), one can recover  $c_0 \leftarrow s_0[0] - s_0[n-2]$  using a similar approach as in recovering  $s_0[0]$  with  $\mathcal{O}_0$ . However,  $c_0$  lies in the range  $[-4, \dots, 4]$ , therefore the recovery process is more complicated. However, we still use several different ciphertexts to get an inner code for the check variables. In other words, there are two inner codes: one for secret coefficients, and another one for check variables. Each of them helps us to compute conditional distributions, which we use with outer LDPC code.

Now, let us represent  $c_0$  as a vector  $\mathbf{h}_0$  with values from  $\{-1, 0, 1\}$  such that  $c_0 = \mathbf{h}_0^T (s_0[0], \dots, s_0[n-1])$ . In general, if  $\mathbf{u}_0 = k_u \cdot \sum_{j=1}^w x^{i_j}$ , then  $\mathbf{h}_0$  is a vector with  $w$  nonzero entries at the positions  $(-i_j) \bmod n$ , where the entry is 1 if and only if  $i_j = 0$ . Let  $\mathbf{H}_0$  be a negacyclic matrix of the vector  $\mathbf{h}_0$ . With this ciphertext, the  $i^{\text{th}}$  message bit is connected to the  $i^{\text{th}}$  row of  $\mathbf{H}_0 (s_0[0], \dots, s_0[n-1])^T$ . Note that, unlike in Section 3.1,  $c_i$  is the sum of secret coefficients, possibly multiplied by -1. However, this does not significantly affect the result since from the distribution of the coefficient it is trivial to obtain the distribution of the negative coefficient and vice versa. Thus, we still call  $c_i$  the sum of secret coefficients.

Let  $\mathbf{u}_r = k_u \cdot \sum_{j=1}^w x^{i_j^{(r)}}$ ,  $r \in \{0, 1, 2\}$ . Ciphertext  $(\mathbf{u}, \mathbf{v})$  with the help of oracles  $\mathcal{O}_i$  reveals information about 256 parity checks. The parity-check matrix of the outer LDPC code in this case is of the form

$$\mathbf{H}_{\text{ini}} = [\mathbf{H}_0 | \mathbf{H}_1 | \mathbf{H}_2],$$

where  $\mathbf{H}_j$  is the negacyclic matrix obtained from the vector connecting  $c_0$  and  $s_j$ . Due to the FD oracle, parity checks  $c_1, \dots, c_{n-1}$  must be negacyclic shifts of  $c_0$ . We only demonstrated the parity-check matrix for the outer LDPC code

consisting of block of 256 checks, but there could be several such blocks. Note that in general, the polynomials  $\mathbf{u}_r$  do not have to use the same  $w$ .

There are three main ways to increase the success probability of the attack.

1. Increase the length of the inner code for the secret coefficients. Querying oracles as in Section 4.1 leads to a more accurate distribution for each coefficient.
2. Similarly, increase the length of the inner code for the check variables, i.e., fix the indexes  $i_j^{(r)}$  and use different  $(k_u, k_v)$ .
3. Increase the number of check blocks. The resulting parity-check matrix of the LDPC code  $\mathbf{H}_{\text{ini}}$  consists of  $3 \times m$  blocks of negacyclic matrices, where  $m$  is the number of “unique” parity checks  $c_0, c_n, c_{2n}, \dots$

Creating the best inner code for the check variables that maximizes the amount of information is a challenging task. An educated guess would be the most accurate way to describe our approach to tackling this problem.

## 5 Application to HQC

In this section, we describe the detailed attack on HQC.  $\mathcal{O}_{\text{HQC}}$  denotes a general side-channel-based PC oracle for HQC, referenced prior-art assumes timing leakage, but this is not required. We treat a key-misuse oracle as a chosen-ciphertext side-channel oracle with 100% oracle accuracy.

### 5.1 Key-recovery attack with $\mathcal{O}_{\text{HQC}}$

In [Guo+22a] the authors presented a plaintext checking (PC) oracle based on timing information due to the use of rejection sampling. In this section, we describe how the PC attack works and then explain how we can improve it by using our new SCA-LDPC framework, which is based on coding theory.

Currently, HQC makes use of so-called rejection sampling in the CPA secure encryption function [Agu+20; Guo+22a]. The rejection sampling algorithm is used to construct random vectors with a specific Hamming weight  $\omega$ . It works by random sampling of bit positions in the vector, and if some positions are sampled twice, they are rejected. Straight-forwardly implemented, this algorithm leaks timing information due to the inherently random number of rejections that occur. The HQC implementations tested in [Guo+22a] leak timing information mainly through the use of so-called “seedexpander” calls. The output of the seedexpander function is deterministic pseudo-randomness given by an eXtendable Output Function (XOF). The rejection sampling algorithm uses the seedexpander function to generate relatively large blocks of randomness, at a time. The timing distribution, therefore, is highly dependent on the number of seedexpander calls

needed. The minimum number of seedexpander calls occurs when there are no rejections in the rejection sampling algorithm. In practice, we classify timing measurements based on the number of *additional* seedexpander calls. They are each related to one of the four<sup>2</sup> distributions  $\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$ , listed in increasing order of rarity.

Prior to the publication of the referenced work, it was believed that this randomness was only dependent on values known to the attacker, in this case, the plaintext  $\mathbf{m}$ . The assumption then was that constant time implementation was not needed for the rejection sampling algorithm. Certainly, it was shown in [Guo+22a] that this assumption is problematic. Although  $\mathbf{m}$  is indeed known to the attacker, the result of the implicitly carried out comparison  $\mathbf{m}' \stackrel{?}{=} \mathbf{m}$  is not. Here  $\mathbf{m}' = \text{decode}(\mathbf{c} + \mathbf{e}')$  and  $\mathbf{e}'$  is a extra noise supplied by the attacker.

The authors showed a key-recovery attack where, by using the timing information due to rejection sampling, knowledge of  $\mathbf{m}' \stackrel{?}{=} \mathbf{m}$  is leaked. The attack required 866,000 so-called “idealized oracle” ( $\mathcal{O}_{\text{HQC}}^{\text{ideal}}$ ) queries for the 128-bit security setting. The idealized oracle assumes a noise-free environment where a single timing measurement is sufficient to determine the membership of  $\mathcal{S}_j$  (where  $j = 3$  in [Guo+22a]). Unfortunately, this is not sufficient for a 100% correct oracle, due to reasons explained in the following paragraph.

**What follows is a high-level summary of the referenced attack;** A plaintext  $\mathbf{m}$  is selected according to some criteria useful for the distinguisher. In the case of timing leakage, the distinguishing property is such that the selected  $\mathbf{m}$  results in the timing distribution  $\mathcal{S}_3$ , since it is the one most easily distinguished. The probability of for any random  $\mathbf{m}'$ , where  $\mathbf{m}' \neq \mathbf{m}$ , resulting in the same  $\mathcal{S}_3$  timing distribution is low (0.58% per [Guo+22a]). In other words,  $\mathbf{m}' \stackrel{?}{=} \mathbf{m}$  can be distinguished with a high, yet-not-complete, advantage.

A ciphertext  $\mathbf{c}' = (\mathbf{u}, \mathbf{v})$  is crafted in the next step such that  $\mathbf{r}_1$  is  $\mathbf{1} \in \mathcal{R}$  and  $\mathbf{r}_2$  and  $\mathbf{e}$  is  $\mathbf{0} \in \mathcal{R}$ . By Equation (3) this results in

$$\mathbf{v} - \mathbf{u} \cdot \mathbf{y} = \mathbf{mG} + \mathbf{s} \cdot \mathbf{r}_2 + \mathbf{e} - (\mathbf{r}_1 + \mathbf{h} \cdot \mathbf{r}_2) \cdot \mathbf{y} = \mathbf{mG} - \mathbf{r}_1 \cdot \mathbf{y} = \mathbf{mG} - \mathbf{y} \quad (7)$$

which makes  $\mathbf{y}$  the only remaining error for the decoder to correct. Note too that by knowledge of  $-\mathbf{y} = \mathbf{y}$  it is a simple computation to find the rest of the private key, since  $\mathbf{x} = \mathbf{s} - \mathbf{h} \cdot \mathbf{y}$ . Calculating  $\mathbf{x}$  is quite unnecessary, however, since it is not used in decapsulation.

Plainly, this crafted ciphertext is invalid and will be rejected in the ciphertext comparison step of the decapsulation. However, a valid ciphertext is not required due to the timing leakage in the XOF via the non-constant time rejection sampling algorithm. The reencryption step immediately preceding the comparison derives

<sup>2</sup>Strictly, there is no upper bound, but the practical benefit of finding a value for  $\mathcal{S}_{\geq 4}$  is not worth the exponential effort required [Guo+22a].

the values of  $\mathbf{r}_1$ ,  $\mathbf{r}_2$  and  $\mathbf{e}$  from the XOF seeded by  $\mathbf{m}$ . The single bit information  $\mathbf{m}' \stackrel{?}{=} \mathbf{m}$  leaks prior to the ciphertext comparison step.

Hall et al. proposed in [HGS99] a way to recover  $\mathbf{y}$ ; An additional error vector  $\mathbf{e}'$  is added to  $\mathbf{c}'$ .  $\mathbf{e}'$  is of just sufficient weight to cause a decoding failure (i.e.  $\mathbf{m}' \neq \mathbf{m}$  is leaked). The basic attack then simply iterates through each bit  $0 \leq i \leq N$  of  $\mathbf{e}'$  not already flipped to find those positions that if flipped would result in a decoding success. If this is the case for any value of  $i$  this indicates that the bit was already flipped in  $\mathbf{y}$  in the ciphertext.

However, this technique alone is not sufficient to provide decisions on all bits in the ciphertext. The reason is twofold. First, unflipping a bit in the error pattern given to the RMRS decoder does not guarantee a decoding success, and secondly due to the possibility that both  $\mathbf{m}'$  and  $\mathbf{m}$  result in the timing distribution  $\mathcal{S}_3$ , even though  $\mathbf{m}' \neq \mathbf{m}$ . This is modeled by  $\mathcal{O}_{\text{HQC}}^{\text{ideal}}$ , the idealized oracle from [Guo+22a], which though noise-free, is not 100% correct.

The first problem is solved by using many different error patterns  $\mathbf{e}'$ . The second was solved by majority voting, i.e. by gathering three or more decisions for every bit. Both of these solutions drive up the number of required oracle calls, even in the ideal timing leakage setting. For the 128-bit security level, this number adds up to 866,000 oracle calls [Guo+22a].

## 5.2 New improved attack using LDPC codes.

What follows is a description of the new attack listed in Figure 2, a PC oracle  $\mathcal{O}_{\text{HQC}}$  is assumed. Like in the original attack [Guo+22a] we select a plaintext with good side-channel detection properties (in the original case this is a timing property).

The next step is to construct a  $N \times N$  regular cyclic LDPC parity-check matrix  $\mathbf{H}_{\text{ini}}$  without cycles of length 4, with a good decoding performance.  $\mathbf{H}_{\text{ini}}$  has a row-weight of  $W$ . This construction is detailed in Section 3.2, with  $(m = 1, b = 1)$ . The first row of  $\mathbf{H}_{\text{ini}}$  is the vector  $\mathbf{h}_{\text{ini}}$ .

We craft a special ciphertext  $\mathbf{c}'$  where  $\mathbf{r}_2 = \mathbf{0}$ ,  $\mathbf{e} = \mathbf{0}$  and  $\mathbf{r}_1 = \mathbf{h}_{\text{ini}}$ . Similarly to the case given by Equation (7) above, this results in

$$\mathbf{v} - \mathbf{u} \cdot \mathbf{y} = \dots = \mathbf{m}\mathbf{G} - \mathbf{r}_1 \cdot \mathbf{y} = \mathbf{m}\mathbf{G} - \mathbf{h}_{\text{ini}}\mathbf{y} \quad (8)$$

which makes the added noise that the decoder has to correct equal to  $\mathbf{h}_{\text{ini}}\mathbf{y}$ . In other words, each bit position  $i$  in  $\mathbf{c}'$  correspond to the result of a parity-check equation over  $\mathbf{y}$ , given by  $\mathbf{h}_{\text{ini}} \gg i$  (cyclic shift by  $i$  steps) due to the cyclic nature of our LDPC code.

The Reed-Muller (RM) and Reed-Solomon (RS) concatenated (RMRS) decoder, used in HQC, can be attacked in two stages. First we select  $(d_{\text{RS}} - 1)/2$  outer RM blocks (each RM block decodes to one RS symbol) to flip in  $\mathbf{c}'$  (by XOR with  $\mathbf{e}'$ ). This results in a state where if one more block is flipped it will re-

**Input:**  $\mathcal{O}_{\text{HQC}}$ , public key

**Output:**  $y$

```

1: Select plaintext  $\mathbf{m}$   $\triangleright$  With good side channel distinguishing properties
2: Generate sparse vector  $\mathbf{h}_{\text{ini}}$   $\triangleright$  According to Section 3.2
3: Construct  $\mathbf{H}_{\text{ini}}$   $\triangleright$  From  $\mathbf{h}_{\text{ini}}$  by cyclic shifts
4: Craft  $\mathbf{c}'$  with  $\mathbf{r}_2 = \mathbf{0}$ ,  $\mathbf{e} = \mathbf{0}$  and  $\mathbf{r}_1 = \mathbf{h}_{\text{ini}}$ 
5:  $\mu \leftarrow 0^N$   $\triangleright$  Initialize message
6: loop
7:    $\mathbf{e}' \leftarrow 0^N$ ,
8:    $\mathcal{B}' \leftarrow$  random subset of size  $(d_{RS} - 1)/2$  from  $\{0, \dots, n_1 - 1\}$ 
9:   for each  $B' \in \mathcal{B}'$  do  $\triangleright$  Flip  $(d_{RS} - 1)/2$  RM-blocks
10:     Flip block  $B'$  in  $\mathbf{e}'$ 
11:   end for
12:    $B \xleftarrow{\$} \{0, \dots, n_1\} \setminus \mathcal{B}'$   $\triangleright$  Select a random unflipped block
13:    $\mathcal{I}^B \leftarrow \{Bn_2, \dots, B(n_2 + 1) - 1\}$ 
14:    $\mathcal{I}_{\mathbf{e}'}^B, \mathcal{I}_0^B, \mathcal{I}_1^B \leftarrow \emptyset, \emptyset, \emptyset$ ,
15:   while  $\mathcal{O}_{\text{HQC}}^{0=\text{repeat}}(\mathbf{c}' + \mathbf{e}')$  do  $\triangleright$  Find an initial error pattern for block  $B$ 
16:      $\mathcal{I}_{\mathbf{e}'}^B \leftarrow \mathcal{I}_{\mathbf{e}'}^B \cup \{i\}$ , where  $i \xleftarrow{\$} \mathcal{I}^B$ 
17:      $\mathbf{e}'[i] \leftarrow 1$ 
18:   end while
19:   for each  $i \in \mathcal{I}_{\mathbf{e}'}^B$  do  $\triangleright$  Minimize the error pattern
20:      $\mathbf{e}'[i] \leftarrow 0$   $\triangleright$  Unflip bit in error pattern
21:     if  $\mathcal{O}_{\text{HQC}}^{1=\text{repeat}}(\mathbf{c}' + \mathbf{e}')$  then
22:        $\mathcal{I}_0^B \leftarrow \mathcal{I}_0^B \cup \{i\}$   $\triangleright$  Satisfied parity check, add  $i$  to  $\mathcal{I}_0^B$ 
23:        $\mathbf{e}'[i] \leftarrow 1$   $\triangleright$  Restore bit in error pattern
24:     end if
25:   end for
26:   for each  $i \in (\mathcal{I}^B \setminus \mathcal{I}_{\mathbf{e}'}^B)$  do  $\triangleright$  Find unsatisfied parity checks
27:     if  $\mathcal{O}_{\text{HQC}}(\mathbf{c}' + \mathbf{e}')$  then
28:        $\mathcal{I}_1^B \leftarrow \mathcal{I}_1^B \cup \{i\}$   $\triangleright$  If found, store in  $\mathcal{I}_1^B$ 
29:     end if
30:   end for
31:   Select rows  $i \in (\mathcal{I}_0^B \cup \mathcal{I}_1^B)$  from  $\mathbf{H}_{\text{ini}}$  and add to  $\mathbf{H}'$ 
32:   Construct  $\mathbf{H} = [\mathbf{H}' | \mathbf{I}]$ 
33:    $\mu \leftarrow \mu \mid 0^{\#\{\mathcal{I}_0^B\}} \mid 1^{\#\{\mathcal{I}_1^B\}}$ 
34:    $\mathbf{y} \leftarrow \text{Decode}_{\mathbf{H}}(\mu)[0..n]$   $\triangleright$  Decoder returns the error vector
35:   if  $\mathbf{y}$  correct then  $\triangleright$  Try to decrypt a valid message
36:     return  $\mathbf{y}$ 
37:   end if
38: end loop

```

**Figure 2:** HQC new attack algorithm.  $\mathcal{O}_{\text{HQC}}^{0=\text{repeat}}$  denotes a PC oracle which is repeated as necessary (determined by empirical study) to achieve better than nominal error rate in the case of decoding failure; decoding successes are never repeated.  $\mathcal{O}_{\text{HQC}}^{1=\text{repeat}}$  works in a similar but opposite fashion.

sult in a decoding error in the RS decoder. A decoding failure such as that would be detected by  $\mathcal{O}_{\text{HQC}}$ . We randomly select another block which we denote  $B$ .

The next stage is to find which bits  $\mathcal{I}_{\mathbf{e}'}^B$  to flip in the block  $B$  that results in a decoding failure. We do this by flipping bits  $i \in \mathcal{I}_{\mathbf{e}'}^B$  such that  $\mathbf{e}'[i] = 1$  in block  $B$  until a RM decoding failure occurs. This propagates as a failure symbol to the RS decoder which is already on the brink of being overwhelmed. This results in a state where  $\mathbf{c}' + \mathbf{e}'$  fails to decode due to too much additional noise in the block  $B$  partition of  $\mathbf{e}'$ .

**An aside on oracle accuracy.** The LDPC code helps with recovery from bad oracle decisions. However, the stateful nature of the new algorithm can cause certain poor oracle decisions to propagate and result in the algorithm ending up in a bad state. Such errors occur naturally more often for less accurate oracles. We compensate for these effects by introducing extra confirmation calls to those oracle decisions which are most sensitive. These are denoted in Figure 2 by  $\mathcal{O}_{\text{HQC}}^{r=\text{repeat}}$ , where  $r \in \{0, 1\}$  indicates which Oracle outputs are repeated for confirmation.  $\mathcal{O}_{\text{HQC}}^{0=\text{repeat}}$  means decoding failures are confirmed but decoding successes are not. The number of repeated oracle calls is determined by empirical study.

After finding an error pattern resulting in decoding failure, the next step is to reduce the number of flipped bits, in block  $B$ . The goal is to find the minimal pattern that still results in a decoding failure. We do this by unflipping each of the flipped bits  $i \in \mathcal{I}_{\mathbf{e}'}^B$  in block  $B$ . This results in one of two cases:

1. If we get a decoding success we record it in  $\mathcal{I}_0^B$  for later use, undo the flip and then move on to select another bit  $i \in \mathcal{I}_{\mathbf{e}'}^B$ .
2. If we still get a decoding failure we try again with another flipped bit  $i \in \mathcal{I}_{\mathbf{e}'}^B$ .

Once we have run out of flipped bits in  $\mathcal{I}_{\mathbf{e}'}^B$  to check, we have achieved a minimal bit pattern in  $\mathcal{I}_0^B$  for block  $B$  that results in decoding failure. That is, the set  $\mathcal{I}_0^B$  contains those bits that result in a decoding success if any are unflipped. Conversely, when flipped, they have been unambiguously shown to increase the noise for the RMRS decoder. All bits in  $\mathcal{I}_0^B$  can therefore reliably be assumed to correspond to a satisfied parity check. So, for each bit  $i \in \mathcal{I}_0^B$  we construct<sup>3</sup> our sub matrix  $\mathbf{H}'$  by the selection of row  $i$  of  $\mathbf{H}_{\text{ini}}$ .

Working from the minimal decoding failure pattern ( $\mathbf{e}'[i] = 1 \forall i \in \mathcal{I}_0^B$  and  $\mathbf{e}'[i] = 0 \forall i \notin \mathcal{I}_0^B$ ) for RM block  $B$  we can now flip bits that so far have been left untouched ( $i \notin \mathcal{I}_{\mathbf{e}'}^B$ ), one at a time. For each flip, if it results in a decoding success, then we record it in  $\mathcal{I}_1^B$ . Such a bit must mean that by flipping it we reduce the noise that the RMRS decoder has to handle. Therefore, this bit can be reliably assumed to correspond to an unsatisfied parity-check equation, or a '1' in the vector  $\mathbf{h}_{\text{ini}}\mathbf{y}$ . When all bits have been tested we extend our sub matrix  $\mathbf{H}'$  by the selection of all rows  $i \in \mathcal{I}_1^B$  of  $\mathbf{H}_{\text{ini}}$ .

<sup>3</sup>or extend if this is not the first selected block/iteration of the algorithm

At this time in the algorithm,  $r$  number of parity-check equations have been collected in  $\mathbf{H}'$ . The remaining step is to construct parity-check matrix  $\mathbf{H} = [\mathbf{H}'_{r \times n} | \mathbf{I}_{r \times r}]$  and a message vector

$$\mu = \left[ \mathbf{0}^n | \mathbf{0}^{\#\{\mathcal{I}_0^B\}} | \mathbf{1}^{\#\{\mathcal{I}_1^B\}} \dots | \mathbf{0}^{\#\{\mathcal{I}_0^t\}} | \mathbf{1}^{\#\{\mathcal{I}_1^t\}} \right] \quad (9)$$

in such a way that we have  $n$  zeroes, each representing an unknown bit-value of  $\mathbf{y}$  to be recovered. The message is appended by the following redundancies: a single 0 for each satisfied parity-check equation hitherto selected ( $i \in \mathcal{I}_0^B$ ) and a 1 for each unsatisfied parity check ( $i \in \mathcal{I}_1^B$ ). We do this for all  $t$  blocks  $B$  that have so far been selected.

We try to decode the message  $\mu$  and recover  $\mathbf{y}$  from the first  $n$  bits. We use  $\mathbf{H}$  as input and a suitable decoder such as sum-product or the min-sum approximation.

If the decoding is not successful we unflip all bits in block  $B$  and unflip all other blocks. Then we restart the algorithm (using the same ciphertext) and select another block. The old  $\mathcal{I}_0^B$  and  $\mathcal{I}_1^B$  are saved and re-used in the next decoding attempt. We continue until successful.

In some cases (for less accurate oracles) one might still fail to decode even after all outer RM blocks have been exhausted. In such cases, one can simply save  $\mu$  and  $\mathbf{H}'$  and continue extending them by restarting the algorithm.

## 6 Experiments

In this section, we show the results of simulations and real-world experiments for Kyber and HQC.

### 6.1 Masked Kyber

#### Software simulations

We introduce software simulations, where we fix the accuracy  $\rho$  of each oracle  $\mathcal{O}_i$  to be the same.

The attack improves as the weight of the rows in the parity matrix increases. However, the decoding time increases exponentially with it. In the course of experiments, we found that the value  $w = 2$  works best, i.e., the parity-check matrix consists of negacyclic matrices with row weight 2. For Kyber-768, this means that each check variable is the sum of 6 secret coefficients.

The three main parameters of the attack are  $m_0$ ,  $m_1$  and  $m_2$ , where  $m_0$  and  $m_2$  are the lengths of the inner code for the secret coefficients and the check variables, resp.,  $m_1$  is the number of blocks of check variables. Recall that Kyber-768 has 3 blocks of 256 secret coefficients, and we assume that from one power trace we get information about all 256 message bits. This means that we need

**Table 4:** Comparison with the majority voting for full-key recovery.  $t$  is the number of votes cast, values in the brackets are  $m_0$ ,  $m_1$  and  $m_2$ , resp.

|                              |                  |                          |
|------------------------------|------------------|--------------------------|
| $\rho = 0.995$               | Number of traces | Average number of errors |
| Majority Voting ( $t = 3$ )  | 27 (ref)         | 0.21/768                 |
| Our Method (2, 1, 4)         | 10 (−63%)        | 0.37/768                 |
| $\rho = 0.95$                | Number of traces | Average number of errors |
| Majority Voting ( $t = 7$ )  | 63 (ref)         | 0.47/768                 |
| Our Method (3, 4, 2)         | 17 (−73%)        | 0.16/768                 |
| $\rho = 0.9$                 | Number of traces | Average number of errors |
| Majority Voting ( $t = 11$ ) | 99 (ref)         | 0.67/768                 |
| Our Method (4, 3, 4)         | 24 (−75.8%)      | 0.46/768                 |

$3m_0$  and  $m_1 \cdot m_2$  traces to get the distributions for secret coefficients and check variables, respectively. The interested reader is referred to the full version of the paper [Guo+23a] for the actual codes used in the simulation and in the real attack.

We evaluate our methodology against the majority voting technique, a conceptually simple coding approach that can be considered as a repetition code. Majority voting is a typical approach to ensure that a single secret coefficient can be recovered with high accuracy. This approach has been selected as the baseline attack method due to its relevance as the most frequently used coding scheme for attacking Kyber in previous literature (e.g., in [She+23a]). For majority voting, we choose the code as in Table 3 and use  $t$  votes, i.e., the actual code is repeated  $t$  times. We run 1000 tests and compute the average number of wrong secret coefficients, the attack is considered successful if this number is less than 1. For our approach, we choose  $m_0$ ,  $m_1$ , and  $m_2$  such that the total number of traces is minimized and the average number of errors is close to majority voting. We run 100 tests, and all tests are done with randomly generated secret keys. The results for a wide range of accuracy levels are shown in Table 4.

### Real-world experiments

We conduct our experiments in the ChipWhisperer toolkit, where we use the ChipWhisperer-Lite board, the CW308 UFO board, and the CW308T-STM32F4 target board with a 32-bit ARM Cortex-M4 CPU. We target the mkm4 library<sup>4</sup> in [Hei+22a] implementing a first-order masked version of Kyber. The library is compiled using the -O3 optimization level, which is typically harder to attack [She+23a; Ngo+21a]. The target board is run at 24 MHz, and the traces are sampled at 24 MHz.

We attacked the function `masked_poly_tomsg` in the first draft of the work and it was the first power analysis attack on an open-source masked implemen-

<sup>4</sup><https://github.com/masked-kyber-m4/mkm4>

```

masked_poly_frommsg(uint16_t poly[2][256], uint8_t msg[2][32])
1: ... /* initialization */
2: for i = 0 to 31 do
3:   for j = 0 to 7 do
4:     mask = -((msg[0][i] >> j) & 1)
5:     poly[0][8*i+j] += mask & ((KYBER_Q+1)/2)
6:   end for
7: end for
8: for i = 0 to 31 do
9:   for j = 0 to 7 do
10:    mask = -((msg[1][i] >> j) & 1)
11:    poly[1][8*i+j] += mask & ((KYBER_Q+1)/2)
12:   end for
13: end for
14: ...

```

Figure 3: The attacked function in `KYBER.CPAPKE.Enc()` (from [Hei+22a])

tation of Kyber, as far as we know. Then we switched to the function `masked_poly_frommsg` similarly to [Dub+23]. With this approach, real oracles from side-channel leakages have better accuracy, leading to a lower amount of traces.

The function `masked_poly_frommsg` (shown in Figure 3) maps each masked polynomial coefficient to a corresponding message bit during decapsulation. In one loop the function works on the message bits XORed with random bits; on the other loop it works with these random bits themselves. Obtaining a power trace for these two loops allows us to retrieve information about all message bits and implement the FD oracles  $\mathcal{O}_i$ .

The attack scenario is the same as in [Ngo+21a]. First, there is the profiling stage during which, using the profiling device  $D_1$ , we collect 100,000 power traces of the function `masked_poly_frommsg`. It is done by generating a random message which is encrypted using the device’s public key, the resulting ciphertext is passed to the measured by the ChipWhisperer decapsulation function. Each byte of the message is computed in the same way, and the power traces corresponding to each byte are similar. Thus, we can train only 8 neural network models, one for each bit of the byte. Models are trained for up to 100 epochs. The interested reader is referred to the full version of the paper [Guo+23a] for the architecture of the model.

Each of the 8 models simulates the 32 oracles  $\mathcal{O}_{i+8j}$ ,  $j = 0, \dots, 31$ , with some accuracy  $\rho_i$ . The oracle behaves like a binary symmetric channel with success probability  $\rho_i$ , but the model provides soft values, which can be treated as the probabilities of output being 1 or 0 from the model’s perspective. Thus, the real-world attack is more powerful since there is more information we can work with.

**Table 5:** Accuracy of recovering particular bit for models. Device  $D_1$  is the profiling device, and  $D_2$  is the device to be attacked.

| Device | $\rho_0$ | $\rho_1$ | $\rho_2$ | $\rho_3$ | $\rho_4$ | $\rho_5$ | $\rho_6$ | $\rho_7$ |
|--------|----------|----------|----------|----------|----------|----------|----------|----------|
| $D_1$  | 0.9651   | 0.9986   | 0.9985   | 0.9985   | 0.9992   | 0.9995   | 1.0000   | 1.0000   |
| $D_2$  | 0.9390   | 0.9811   | 0.9923   | 0.9023   | 0.9654   | 0.8940   | 0.9404   | 0.9873   |

**Table 6:** Real-world attack results on the first-order masked Kyber-768. We performed 100 runs of the attack with a random secret key for each run.

|                              | Number of traces | Average number of errors |
|------------------------------|------------------|--------------------------|
| Majority Voting ( $t = 11$ ) | 99               | 0.34/768                 |
| Our Method (2, 2, 3)         | 12               | 0.82/768                 |

After the profiling stage, there is the attacking stage. The assumption is that the attacker has access for a (relatively) short period of time to a similar device  $D_2$ . After collecting power traces for decapsulation on chosen ciphertexts, the attacker’s goal is to recover the key using the trained models. The Table 5 shows the accuracy  $\rho_i$  of recovering  $i^{\text{th}}$  bit for devices  $D_1$  and  $D_2$ .

The experimental results (shown in Table 6) with the average oracle accuracy of 0.9502 are better than the simulation results with an accuracy of 0.95. There are two reasons for this: (1) real models provide soft values, making the attack more powerful; (2) In the simulation, the accuracy of each bit is the same, but for our LDPC approach, it is more beneficial for some bits to be more reliable than others.

On the other hand, the success of majority voting approach depends on the worst bit position. In other words, in the real world majority voting works worse since the bottleneck is the worst bit. The real attack with accuracy from Table 5 uses  $t = 11$  votes, i.e. in total we need 99 traces (instead of 63 as in Table 4). In this case, our framework uses 86% fewer traces.

## 6.2 HQC

In order to test the new attack strategy against HQC it is advantageous to make as close to an apples-to-apples comparison as possible against the results of [Guo+22a]. To this end, we model the PC oracle as follows; The success probability for an oracle query is determined by  $\rho_0$  and  $\rho_1$ , which are the probabilities of correctly classifying decoding failures and decoding successes, respectively. For the case of the ideal HQC timing oracle used in [Guo+22a] these values are listed in Table 7 and correspond to  $\rho_0 = \rho_f$  and  $\rho_1 = \rho_s$ . We label the ideal oracle  $\mathcal{O}_{\text{HQC}}^{\text{ideal}}$ .

Simulating real-world attacks with noisy measurements can be done by selecting other values of  $\rho_0$  and  $\rho_1$ . For simplicity, we introduce  $\rho$  as a single representa-

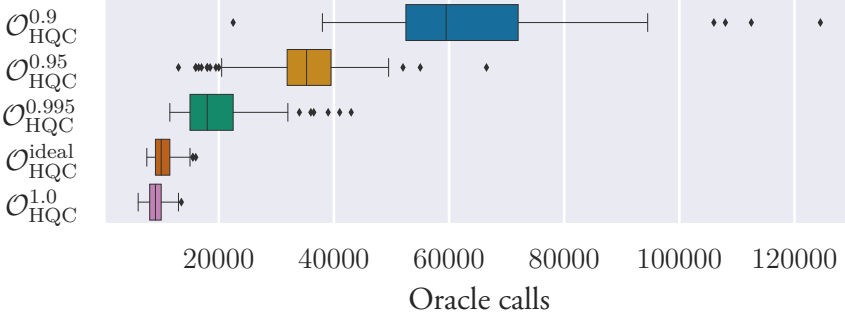
**Table 7:** Ideal HQC timing oracle,  $\mathcal{O}_{\text{HQC}}^{\text{ideal}}$ , as modelled with  $\rho_f$  and  $\rho_s$ .

| Real \ Reported as | decoding failure  | decoding success      |
|--------------------|-------------------|-----------------------|
|                    | $\rho_f = 0.9942$ | $1 - \rho_f = 0.0058$ |
| decoding failure   | $\rho_f = 0.9942$ | $1 - \rho_f = 0.0058$ |
| decoding success   | $1 - \rho_s = 0$  | $\rho_s = 1.0$        |

tive value of PC oracle accuracy, where  $\rho = \rho_0 = \rho_1$ . We label the corresponding oracle  $\mathcal{O}_{\text{HQC}}^\rho$ .

By empirical study we have selected a row weight of  $W = 50$  in the constructed LDPC code (for hqc-128). This is close to the upper limit of our code generation algorithm. Using a bigger  $W$  would occasionally require a more advanced algorithm with backtracking of the random walk. Regardless, the decoding appears to suffer in reliability for values  $W > 50$ . Smaller values of  $W$  require more parity checks and thus make the attack slower.

Some interesting  $\rho$  values, corresponding to real attacks, are  $\{1.0, 0.995, 0.95, 0.9\}$ . In Figure 4 we show the results of simulations using the various oracle models we have described so far. The results for  $\mathcal{O}_{\text{HQC}}^{\text{ideal}}$  indicate an 86.6 times improvement over the original attack [Guo+22a].



**Figure 4:** Experiment for hqc-128. The median number of oracle calls for successful key recovery, are 59500, 35250, 18000, 10000, and 9000 respectively for the listed oracles. For each oracle model, 100 key-recovery simulations ran to completion.

We have validated our attack by running a real timing oracle on a Ubuntu 20.04 LTS laptop with Intel Core i5-7200@2.50GHz. Measurement noise was reduced by turning off hyper-threading and by running in recovery mode. We used  $2^{18}$  measurements to generate a profile, first of a decoding success and again of a decoding failure. Measuring 8 decapsulations resulted in an oracle accuracy of  $\rho_{\mathcal{O}_{\text{HQC}}^{\text{real}}} = 0.951$  as determined by 1000 trials. The simulated results for  $\mathcal{O}_{\text{HQC}}^{0.95}$  indicate a real-life key recovery attack of hqc-128 can be done by measuring  $2^3 \times 35250 \approx 2^{18}$  decapsulation calls.

### 6.3 Discussions

In this section, we present discussions on the performance of the new SCA-LDPC framework, including its information-theoretical advantages and limitations. Furthermore, we compare the SCA-LDPC framework with the inner-symbol error correction method proposed in [Ngo+21a] and highlight the advantages of the former.

**A non-rigorous information theoretical bound.** Assuming that a single side-channel measurement provides a certain amount of information (denoted by  $l$  bits), and considering the fact that there are  $k$  secret symbols that are independently generated from a distribution with entropy  $E$  bits, it is possible to calculate a lower bound for the number of measurements required. This can be accomplished by dividing  $k \cdot E$  by  $l$ . Estimation of  $l$  can be performed by considering each recovered message bit as a Bernoulli variable, with a specified probability  $\rho_i$  of being correct. It is noteworthy that the value of  $\rho_i$  may vary for different secret positions. This information-theoretical estimation is approximate in nature. It is subject to limitations arising from the simplicity of the Bernoulli model. Additionally, near-optimal source coding and channel coding are required to match the derived lower bound. Notwithstanding these limitations, the estimation suggests the possibility of improvement, though the extent of such improvement may be constrained.

The aforementioned lower bound is equivalent to the well-known Shannon source coding bound when the accuracy of the oracle is 100%, which can be used to characterize the source coding gain. The results obtained from the FD oracle based attack on the scheme Kyber-768 exactly meet the lower bound of 7 traces. Conversely, for the PC oracle based attack on hqc-128, 1324 parity checks were required, a factor of 2.1 times the lower bound of 628 checks.

It has been observed that the difference between the simulated results and the lower bound increases as the oracle accuracy decreases. For example, in the case of the PC oracle based attack on hqc-128, when the oracle accuracy drops to 0.95, the ratio of the simulated parity checks to the lower bound increases to approximately 2.7, as calculated by  $2396/880$ . For the FD oracle based real-world attack on Kyber-768, based on the message recovery accuracy data presented in the second row of Table 5, the lower bound was determined to be 9, which is slightly lower than the 12 traces utilized in the actual attack.

**Limitations.** Despite the remarkable reduction of necessary side-channel measurements, a gap remains between actual performance and our non-rigorous information-theoretical lower bound. This gap may be attributed to the requirement of an extremely long codeword, potentially in the range of tens of millions of bits, for the LDPC codes to approach optimality. Additionally, it may be a result of the simplicity and inadequacy of our current code-construction method. More sophisticated

LDPC code construction techniques could further reduce the required number of measurements.

**Our method vs. inner-symbol error correction.** The new SCA-LDPC framework utilizes a system of sparse parity checks to interconnect all the secret symbols. As a result, accurately determined symbols can be utilized to rectify incorrectly determined symbols, categorizing this method as inter-symbol error correction. On the other hand, the method presented in [Ngo+21a] falls under the category of inner-symbol error correction, as the utilization of extended Hamming codes increases the possibility of recovering individual secret symbols, which all must be recovered independently.

Both methods can be applied to the FD oracle based attack on lattice-based schemes in a non-adaptive attack model. However, the inner-symbol error correction method presented in [Ngo+21a] offers no source coding gain and has inferior error correction capabilities. For instance, it is demonstrated in [Ngo+21a] that for a platform with an average message bit recovery rate of 0.972, 216 traces, or  $9 \times 24$ , are required to recover the long-term secret key of a masked Saber implementation. We utilize the detailed message bit recovery rates recorded in Table 20 of [Ngo+21a] to calculate the corresponding lower bound, which is determined to be 10 traces. This demonstrates a significant gap of 21.6 between the actual performance in a real-world scenario and the calculated lower bound. While there is no guarantee that the non-rigorous lower bound will always be attainable, the small ratio of 1.33, or 12/9, for our SCA-LDPC attack on Kyber, illustrates the superior efficiency of our method in terms of the required number of traces.

The substantial improvement of the new SCA-LDPC framework can be attributed to various factors, such as the utilization of soft information in the real-world attack that we conducted. The dominant reason is that all the secret symbols are interconnected and correlated, and redundant symbols are introduced, allowing for the effective handling of a significant number of symbol-level errors. On the contrary, in the inner-symbol error correction method, all the symbols (e.g., 768 symbols in the Saber case) are independent and needs to be successfully recovered, thus precluding the tolerance of any symbol-level errors. Last, in a real-world attack scenario, several secret positions typically have a higher chance of containing errors, which can be effectively corrected through the inter-symbol approach, but may prove to be a bottleneck for the inner-symbol method where all symbols must be correctly identified independently.

## 7 Concluding Remarks and Future Work

From coding theory, we have presented a generic framework for key-recovery side-channel attacks on CCA-secure post-quantum encryption/KEM schemes. Our design philosophy is to employ randomly generated LDPC codes with efficient decoding to connect secret coefficients, which introduces additional benefits of

source coding and error correction. We presented simulation results and real-world experiments on the main lattice-based KEM Kyber and the code-based KEM HQC. The new attack framework can significantly improve the state-of-the-art in terms of the required number of side-channel measurements. Our improvement has great practical impact. For instance, in real-world attack scenarios against a masked Kyber implementation, we require just 12% of the traces compared to the majority voting method. As our simulation results indicate that the gain would increase as the oracle accuracy decreases, this significant reduction could feasibly turn a previously impractical attack into a practical one. Also, we have a remarkable gain in the HQC timing attack instance. An explanation for our substantial improvements is that LDPC codes are considered to have near-optimal performance from an information-theoretic standpoint.

The sample complexity of the new attack framework can be improved further by (i) employing a more advanced code-construction method with improved decoding performance or by (ii) heavy post-processing such as lattice-reduction or information-set decoding. An intriguing area of study is to utilize sophisticated coding-theoretical methods [RU08], such as density evolution or EXIT charts, to carry out efficient and precise security assessments against proposed attacks.

While the new attack framework is general, the specific attack instances can vary due to the diversity of schemes, oracle types, and leakages. We note that schemes selected as new standards such as Kyber should be given primary consideration for investigation. From a research standpoint, it would be engaging to explore the framework's precise implementation and performance in NTRU [Che+20] and NTRU prime [Ber+20]. Additionally, future research could explore whether this framework could be extended to target Classic McEliece and BIKE.

Last, the new attack framework shows the need for countermeasures such as constant-time implementations or higher-order masked implementations. As noted in [Uen+22b], masked hardware exhibits much higher security compared to masked software when considering PC oracle-based side-channel attacks. As our simulations indicate that the gain of the new attack framework increases with less accurate oracles, it would be fascinating to examine the exact improvements in this attack scenario. Furthermore, it would also be worthwhile to investigate whether SASCA could deliver more accurate PC oracles than a deep learning-based approach in this context.

## Acknowledgement

We thank F.X. Standaert for his helpful comments. This work was supported by the Swedish Research Council (grant numbers 2019-04166 and 2021-04602); the Swedish Civil Contingencies Agency (grant number 2020-11632); the Swedish Foundation for Strategic Research (Grant No. RIT17-0005); and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut

and Alice Wallenberg Foundation. The computations and simulations were partly enabled by resources provided by LUNARC.

## References

- [Agu+20] C. Aguilar-Melchor et al. *HQC*. Tech. rep. available at <https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions>. National Institute of Standards and Technology, 2020.
- [Alb+20] M. R. Albrecht et al. *Classic McEliece*. Tech. rep. available at <https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions>. National Institute of Standards and Technology, 2020.
- [Ara+20] N. Aragon et al. *BIKE*. Tech. rep. available at <https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions>. National Institute of Standards and Technology, 2020.
- [Bac+22] L. Backlund et al. *Secret Key Recovery Attacks on Masked and Shuffled Implementations of CRYSTALS-Kyber and Saber*. Cryptology ePrint Archive, Paper 2022/1692. <https://eprint.iacr.org/2022/1692>. 2022.
- [Ber+20] D. J. Bernstein et al. *NTRU Prime*. Tech. rep. available at <https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions>. National Institute of Standards and Technology, 2020.
- [Che+20] C. Chen et al. *NTRU*. Tech. rep. available at <https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions>. National Institute of Standards and Technology, 2020.
- [Dac+20] D. Dachman-Soled et al. “LWE with Side Information: Attacks and Concrete Security Estimation”. In: *CRYPTO 2020, Part II*. Ed. by D. Micciancio and T. Ristenpart. Vol. 12171. LNCS. Springer, Cham, Aug. 2020, pp. 329–358.
- [DAn+19a] J.-P. D’Anvers et al. “Timing Attacks on Error Correcting Codes in Post-Quantum Schemes”. In: *Proceedings of ACM Workshop on Theory of Implementation Security Workshop*. TIS’19. London, United Kingdom: Association for Computing Machinery, 2019, pp. 2–9.

- [Dub+23] E. Dubrova et al. “Breaking a Fifth-Order Masked Implementation of CRYSTALS-Kyber by Copy-Paste”. In: *Proceedings of the 10th ACM Asia Public-Key Cryptography Workshop*. APKC’23. Melbourne, VIC, Australia: Association for Computing Machinery, 2023, pp. 10–20.
- [FO99] E. Fujisaki and T. Okamoto. “Secure Integration of Asymmetric and Symmetric Encryption Schemes”. In: *CRYPTO’99*. Ed. by M. J. Wiener. Vol. 1666. LNCS. Springer, Berlin, Heidelberg, Aug. 1999, pp. 537–554.
- [For65] G. D. Forney. “Concatenated codes.” In: *Technical Report 440, MIT* (1965).
- [Gal62] R. Gallager. “Low-density parity-check codes”. In: *IRE Transactions on information theory* 8.1 (1962), pp. 21–28.
- [GJJ22a] Q. Guo, A. Johansson, and T. Johansson. “A Key-Recovery Side-Channel Attack on Classic McEliece Implementations”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2022.4 (2022), pp. 800–827.
- [GJN20a] Q. Guo, T. Johansson, and A. Nilsson. “A Key-Recovery Timing Attack on Post-quantum Primitives Using the Fujisaki-Okamoto Transformation and Its Application on FrodoKEM”. In: *CRYPTO 2020, Part II*. Ed. by D. Micciancio and T. Ristenpart. Vol. 12171. LNCS. Springer, Cham, Aug. 2020, pp. 359–386.
- [GJS16] Q. Guo, T. Johansson, and P. Stankovski. “A Key Recovery Attack on MDPC with CCA Security Using Decoding Errors”. In: *ASIACRYPT 2016, Part I*. Ed. by J. H. Cheon and T. Takagi. Vol. 10031. LNCS. Springer, Berlin, Heidelberg, Dec. 2016, pp. 789–815.
- [GLG22a] G. Goy, A. Loiseau, and P. Gaborit. “A New Key Recovery Side-Channel Attack on HQC with Chosen Ciphertext”. In: *Post-Quantum Cryptography*. Ed. by J. H. Cheon and T. Johansson. Cham: Springer International Publishing, 2022, pp. 353–371.
- [Guo+20] Q. Guo et al. “Modeling Soft Analytical Side-Channel Attacks from a Coding Theory Viewpoint”. In: *IACR TCHES* 2020.4 (2020), pp. 209–238.
- [Guo+22a] Q. Guo et al. “Don’t Reject This: Key-Recovery Timing Attacks Due to Rejection-Sampling in HQC and BIKE”. In: *IACR TCHES* 2022.3 (2022), pp. 223–263.
- [Guo+23a] Q. Guo et al. *SCA-LDPC: A Code-Based Framework for Key-Recovery Side-Channel Attacks on Post-Quantum Encryption Schemes*. Cryptology ePrint Archive, Paper 2023/294. <https://eprint.iacr.org/2023/294>. 2023.

- [Ham+21a] M. Hamburg et al. “Chosen Ciphertext k-Trace Attacks on Masked CCA2 Secure Kyber”. In: *IACR TCHES* 2021.4 (2021), pp. 88–113.
- [Hei+22a] D. Heinz et al. *First-Order Masked Kyber on ARM Cortex-M4*. Cryptology ePrint Archive, Paper 2022/058. 2022.
- [HGS99] C. Hall, I. Goldberg, and B. Schneier. “Reaction Attacks against several Public-Key Cryptosystems”. In: *ICICS 99*. Ed. by V. Varadharajan and Y. Mu. Vol. 1726. LNCS. Springer, Berlin, Heidelberg, Nov. 1999, pp. 2–12.
- [HHK17] D. Hofheinz, K. Hövelmanns, and E. Kiltz. “A Modular Analysis of the Fujisaki-Okamoto Transformation”. In: *TCC 2017, Part I*. Ed. by Y. Kalai and L. Reyzin. Vol. 10677. LNCS. Springer, Cham, Nov. 2017, pp. 341–371.
- [Hua+23b] S. Huang et al. *Cache-timing attack against HQC*. Cryptology ePrint Archive, Paper 2023/102. <https://eprint.iacr.org/2023/102>. 2023.
- [JZ98] T. Johansson and K. S. Zigangirov. “A Simple One-Sweep Algorithm for Optimal APP Symbol Decoding of Linear Block Codes”. In: *IEEE Trans. Inf. Theory* 44.7 (1998), pp. 3124–3129.
- [Koc96b] P. C. Kocher. “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems”. In: *CRYPTO’96*. Ed. by N. Koblitz. Vol. 1109. LNCS. Springer, Berlin, Heidelberg, Aug. 1996, pp. 104–113.
- [McE78] R. J. McEliece. “A public-key cryptosystem based on algebraic coding theory”. In: *DSN Progress Report* 42-44 (1978), pp. 114–116.
- [Ngo+21a] K. Ngo et al. “A Side-Channel Attack on a Masked IND-CCA Secure Saber KEM Implementation”. In: *IACR TCHES* 2021.4 (2021), pp. 676–707.
- [NIS16] NIST. *NIST Post-Quantum Cryptography Standardization*. <https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization>. 2016.
- [Pea82] J. Pearl. “Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach”. In: *AAAI*. 1982.
- [PPM17] R. Primas, P. Pessl, and S. Mangard. “Single-Trace Side-Channel Attacks on Masked Lattice-Based Encryption”. In: *CHES 2017*. Ed. by W. Fischer and N. Homma. Vol. 10529. LNCS. Springer, Cham, Sept. 2017, pp. 513–533.

- [Qin+21] Y. Qin et al. “A Systematic Approach and Analysis of Key Mismatch Attacks on Lattice-Based NIST Candidate KEMs”. In: *ASIACRYPT 2021, Part IV*. Ed. by M. Tibouchi and H. Wang. Vol. 13093. LNCS. Springer, Cham, Dec. 2021, pp. 92–121.
- [Raj+22] G. Rajendran et al. *Pushing the Limits of Generic Side-Channel Attacks on LWE-based KEMs - Parallel PC Oracle Attacks on Kyber KEM and Beyond*. Cryptology ePrint Archive, Paper 2022/931. 2022.
- [Rav+20a] P. Ravi et al. “Generic Side-channel attacks on CCA-secure lattice-based PKE and KEMs”. In: *IACR TCHES 2020.3* (2020), pp. 307–335.
- [Rav+22b] P. Ravi et al. “Will You Cross the Threshold for Me? Generic Side-Channel Assisted Chosen-Ciphertext Attacks on NTRU-based KEMs”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2022.1 (2022), pp. 722–761.
- [Reg05] O. Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In: *37th ACM STOC*. Ed. by H. N. Gabow and R. Fagin. ACM Press, May 2005, pp. 84–93.
- [RR21] P. Ravi and S. S. Roy. *Side-Channel Analysis of Lattice-based PQC Candidates*. <https://csrc.nist.gov/CSRC/media/Projects/post-quantum-cryptography/documents/round-3/seminars/mar-2021-ravi-sujoy-presentation.pdf>. Accessed: 2022-09-29. 2021.
- [RSU01] T. Richardson, M. Shokrollahi, and R. Urbanke. “Design of capacity-approaching irregular low-density parity-check codes”. In: *IEEE Transactions on Information Theory* 47.2 (2001), pp. 619–637.
- [RU08] T. Richardson and R. Urbanke. *Modern Coding Theory*. USA: Cambridge University Press, 2008.
- [Sch+20] P. Schwabe et al. *CRYSTALS-KYBER*. Tech. rep. available at <https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions>. National Institute of Standards and Technology, 2020.
- [Sch+22a] T. Schamberger et al. *A Power Side-Channel Attack on the Reed-Muller Reed-Solomon Version of the HQC Cryptosystem*. Cryptology ePrint Archive, Paper 2022/724. 2022.
- [She+23a] M. Shen et al. “Find the Bad Apples: An efficient method for perfect key recovery under imperfect SCA oracles - A case study of Kyber”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2023.1 (2023), pp. 89–112.

- [Sho94] P. W. Shor. “Algorithms for Quantum Computation: Discrete Logarithms and Factoring”. In: *35th FOCS*. IEEE Computer Society Press, Nov. 1994, pp. 124–134.
- [Tan+22] Y. Tanaka et al. *Multiple-Valued Plaintext-Checking Side-Channel Attacks on Post-Quantum KEMs*. Cryptology ePrint Archive, Paper 2022/940. 2022.
- [Uen+22b] R. Ueno et al. “Curse of Re-encryption: A Generic Power/EM Analysis on Post-Quantum KEMs”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2022.1 (2022), pp. 296–322.
- [VGS14] N. Veyrat-Charvillon, B. Gérard, and F.-X. Standaert. “Soft Analytical Side-Channel Attacks”. In: *ASIACRYPT 2014, Part I*. Ed. by P. Sarkar and T. Iwata. Vol. 8873. LNCS. Springer, Berlin, Heidelberg, Dec. 2014, pp. 282–296.
- [Xu+22] Z. Xu et al. “Magnifying Side-Channel Leakage of Lattice-Based Cryptosystems With Chosen Ciphertexts: The Case Study of Kyber”. In: *IEEE Transactions on Computers* 71.9 (2022), pp. 2163–2176.

# Key Recovery Attacks on Approximate Homomorphic Encryption with Non-Worst-Case Noise Flooding Countermeasures

---

## Abstract

In this paper, we present novel key-recovery attacks on Approximate Homomorphic Encryption schemes, such as CKKS, when employing noise-flooding countermeasures based on non-worst-case noise estimation. Our attacks build upon and enhance the seminal work by Li and Micciancio at EUROCRYPT 2021. We demonstrate that relying on average-case noise estimation undermines noise-flooding countermeasures, even if the secure noise bounds derived from differential privacy as published by Li et al. at CRYPTO 2022 are implemented. This study emphasizes the necessity of adopting worst-case noise estimation in Approximate Homomorphic Encryption when sharing decryption results.

We perform the proposed attacks on OpenFHE, an emerging open-source FHE library garnering increased attention. We experimentally demonstrate the ability to recover the secret key using just one shared decryption output. Furthermore, we investigate the implications of our findings for other libraries, such as IBM's HELib library, which allows experimental estimation of the noise bounds.

Finally, we reveal that deterministic noise generation utilizing a pseudorandom generator fails to provide supplementary protection.

## 1 Introduction

Fully homomorphic encryption (FHE) is a cryptographic primitive that supports arbitrary computations on encrypted data without using the decryption key and without revealing information about plaintexts. This problem appeared already in the 70s [RAD+78] but a solution did not appear until 2009, when Gentry [Gen09] constructed the first such schemes. Many homomorphic encryption schemes [BV11; FV12; GSW13; DM15; Chi+16] have now appeared and most of them are based on the security of the Learning with Errors (LWE) problem or some of its variants. In 2022, Brakerski, Gentry, and Vaikuntanathan received the prestigious Gödel prize for their transformative contributions to cryptography by constructing efficient FHE schemes.

FHE is a very powerful cryptographic tool and can be a solution to many security problems, particularly for distributed or outsourced computation. A future application area is also in AI and privacy-preserving machine learning.

A very popular FHE branch is *approximate* homomorphic encryption such as CKKS [Che+17] and its variants [Che+19; Che+18]. FHE on approximate numbers, as proposed by Cheon, Kim, Kim and Song [Che+17], has attracted much attention from the fact that it improves the efficiency of computing on encrypted data in many applications where approximate results are acceptable, like privacy-preserving machine learning. FHE based on LWE builds ciphertexts that contain noise and as operations on ciphertexts are performed, the noise is increasing. One has to control the noise and make sure it does not grow to be too large, in order to ensure correct decryption. The main contribution in CKKS was the observation that it can be tolerable for decryption to be approximate so that the plaintext from the decrypted ciphertext is only approximately the correct one. Typically, this is the case in applications where small errors can occur. The CKKS scheme then supports real-valued plaintexts, making it efficient and attractive for certain applications, allowing for some small errors in the least significant bits. CKKS has since been implemented in many well-known open-source libraries for homomorphic encryption and also been extensively examined and optimized.

The CKKS scheme as most other lattice-based FHE schemes meets the basic security notion of indistinguishability under chosen plaintext attack (IND-CPA), assuming the hardness of some well-studied problems, like the average-case hardness of the Learning With Errors (LWE) problem which in turn relates to the worst-case complexity of some computational problems on lattices. Achieving a stronger security notion – indistinguishability under chosen ciphertext attack (IND-CCA), which presumes an active adversary capable of tampering with (or injecting) arbitrary ciphertexts – is challenging, due to the inherent nature of FHE operations.

Li and Micciancio [LM21] have recently shown that the traditional formulation of IND-CPA security does not adequately capture the security of approximate encryption against passive attacks. Specifically, in the context of a homomorphic encryption scheme, a passive adversary has the capability to select or know the homomorphic function that is being executed. Such an adversary can also observe the outcome of the decryption process. Despite being able to eavesdrop, this adversary remains categorized as passive, as it is not permitted to manipulate, inject, or modify ciphertexts, nor alter the final computational result. Li and Micciancio have shown that the CKKS scheme is vulnerable to an efficient key recovery attack under these conditions.

They also proposed a new, enhanced formulation of IND-CPA security that they called IND-CPA with decryption oracles ( $\text{IND-CPA}^D$ ). This notion captures better the threats of a passive attacker. It is proven to be identical to IND-CPA security for encryption schemes with exact decryption, but the situation is different when allowing approximate decryption. The primary distinction lies in the utility of observing decryption outcomes. In encryption schemes with exact decryption, an adversary gains no additional information from observing the decryption result. This is because the adversary, who already knows the original message and evaluation function, can calculate the final outcome on its own. Conversely, for approximate encryption schemes, observing the decryption output could yield invaluable information not readily computable by the adversary. Such information may be exploited by a passive adversary for effective key recovery.

A relevant security notion is  $\text{KR}^D$  security against key recovery attacks, a notion that, while implied by  $\text{IND-CPA}^D$  security, is significantly weaker. This weakness arises because key recovery attacks constitute a more potent form of attack compared to distinguishing attacks.

Li and Micciancio also suggested some ideas of countermeasures to avoid the proposed attack, and the open-source libraries implementing CKKS include different but similar countermeasures to thwart the attack in [LM21]. Subsequently, Li, Micciancio, Schultz, and Sorrell further examined the use of differential privacy to assure  $\text{IND-CPA}^D$  security in [Li+22]. Their method necessitates the inclusion of a considerable amount of additional noise.

FHE schemes, and CKKS-like schemes in particular, involve different parameters and a major task for the designer of the scheme is the choice of parameters that should give the best trade-off between efficiency and security. Both these aspects relate to the connected noise growth in the scheme. For approximate FHE, encoding noise and encryption noise should be considered together. Also, we have to keep track of the level of ciphertexts as in traditional FHE schemes as well as a so-called scaling factor. CKKS-like schemes consider the precision of the scheme and one has to track how it changes through homomorphic operations and its influence on the encrypted data. The scale parameter is typically a way for the user to control the desired precision.

The CKKS paper provided an open-source implementation in the “Homo-

morphic Encryption for Arithmetic on Approximate Numbers”, Heaan [Heaan], other implementations of the scheme have been included in essentially all mainstream libraries for secure computation on encrypted data, like Microsoft’s “Simple Encrypted Arithmetic Library” SEAL [Seal], IBM’s “Homomorphic Encryption” library HELib [Helib], the lattice cryptography libraries PALISADE [Palisade] and Lattigo [Lattigo]. Recently, the PALISADE authors along with a subset of authors from the other libraries released a new library in July 2022, under the name OpenFHE [Bad+22]. OpenFHE provides implementations of all main FHE schemes.

These libraries are also used as a part of other implemented tools and applications. A well-known example is Intel’s nGraph-HE compiler [Boe+19] for secure machine learning applications. Other implemented applications include the encrypted computation of logistic regression [Han+19], security-preserving support vector machine [Par+19], homomorphic evaluation of neural networks and tensor programs [Dat+19], and clustering over encrypted data [CKP19].

A recent trend in the implementation of CKKS-like schemes involves using a non-worst-case noise estimation, such as an empirical or average-case noise estimation, as seen in sources like [Helib; Bad+22; Cos+22]. In this context, a worst-case noise bound refers to a bound that tracks the maximum possible noise throughout each step of homomorphic evaluation. Conversely, empirical noise estimation involves setting the noise limit based on empirical tests, while average-case noise estimation models the noise as a Gaussian distribution, keeping tracks of the variance during each homomorphic encryption operation. Opting for a non-worst-case approach can enhance performance, since the non-worst-case noise bound can be significantly smaller than the worst-case noise bound, thus substantially improving the scheme’s efficiency. This is particularly appealing when considering the efficiency loss incurred while protecting against the Li-Micciancio attack.

To our knowledge, OpenFHE is the first major open-source library to claim the implementation of the differential privacy countermeasure as proposed in [Li+22], yet they suggest the use of a non-worst-case noise estimation. In this paper, we explore the potential impact of such noise estimations on security, when the decryption results need to be shared.

## 1.1 Contributions

The main contributions of this work can be summarized as follows. Firstly, using non-worst-case estimation methods, such as relying on the central limit theorem (CLT) for average-case noise estimation [Cos+22] or experimentally determining the noise bound, can introduce significant security vulnerabilities. For instance, even when adopting the secure noise bounds suggested in [Li+22] to achieve IND-CPA<sup>D</sup> security through differential privacy, KR<sup>D</sup> security may not be ensured if the noise size is estimated in a non-worst-case manner. The novel observation is that such non-worst-case estimation assumes that the message dis-

tribution or the computation gate should follow some average-case or predefined behaviour; however, in the IND-CPA<sup>D</sup> attacker model, the adversary has the power to control the message distribution, causing it to deviate significantly from the average case. The adversary also has the power to select the functions and gates among the possible valid options.

We implement a key-recovery attack on the recent OpenFHE implementation, requiring only a single shared decryption output. The OpenFHE library, a successor to PALISADE, is chosen as the primary target implementation since it incorporates the noise-flooding countermeasure with the new secure noise bounds [Li+22] derived from differential privacy<sup>1</sup>. In contrast, Microsoft SEAL disallows the sharing of decryption results, while open-source HELib and HEAAN have not implemented countermeasures based on the latest CRYPTO 2022 research [Li+22]. We are capable of attacking OpenFHE as its noise estimation method aligns with the average-case noise estimation proposed in [Cos+22].

The experimental noise estimation employed in HELib necessitates that the server performing homomorphic evaluation adheres to the message/ciphertext distribution and the evaluation gates assumed by the user. However, these requirements contradict the adversary model of IND-CPA<sup>D</sup> security. Our analysis indicates that the flooding countermeasure with such noise estimation is also insecure.

Finally, as an additional finding, we show that the protection of utilizing a pseudorandom generator to produce deterministic noise with the ciphertext serving as an input seed does not improve security. Such a countermeasure has been suggested in [LM21; CHK20] and implemented in HELib. We have carefully designed a mechanism to bypass this security measure. The code for all of our experiments is public<sup>2</sup>.

## 1.2 Organizations

The remainder of this paper adheres to the following organization. Section 2 introduces the background on Approximate Homomorphic Encryption schemes. Section 3 presents the attack targeting implementations using non-worst-case noise estimation. Section 4 describes the attack method for deterministic noise estimation for “fresh ciphertexts”. Lastly, Section 5 offers concluding insights and additional discussions.

## 2 Preliminaries

In this section, we provide essential background information on Approximate Homomorphic Encryption schemes and describe their security notions.

<sup>1</sup>[https://github.com/openfheorg/openfhe-development/blob/main/src/pke/examples/CKKS\\_NOISE\\_FLOODING.md](https://github.com/openfheorg/openfhe-development/blob/main/src/pke/examples/CKKS_NOISE_FLOODING.md).

<sup>2</sup><https://github.com/d-nabokov/KRAonCKKS>

**Notation.** Let  $\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$  represent the sets of integers, rationals, reals, and complex numbers, respectively. Bold characters, such as  $\mathbf{a} = (a_1, \dots, a_n)$ , denote vectors. The dot product of two vectors is written as  $\langle \mathbf{a}, \mathbf{b} \rangle$ . For a probability distribution  $\mathcal{D}$ , we use  $x \leftarrow \mathcal{D}$  to indicate that  $x$  is sampled according to the distribution  $\mathcal{D}$ . We denote the additional noise of countermeasures with  $\epsilon$ , which, though not necessarily small, can be considered as random computational noise. Let  $\lfloor \cdot \rfloor$  denote the operation of rounding to the nearest integer. The security parameter is represented by  $\kappa$ . A function  $f$  is deemed negligible in  $\kappa$  if  $f(\kappa) = \kappa^{-\omega(1)}$ .

## 2.1 Lattices and the Ring-LWE problem

A lattice is a discrete subgroup of  $\mathbb{R}^n$ . Let  $n, d$  be two positive integers, and let  $\mathbf{b}_1, \dots, \mathbf{b}_d$  be linearly independent vectors in  $\mathbb{R}^n$ . The lattice they generate is  $\mathcal{L}(\mathbf{b}_1, \dots, \mathbf{b}_d) = \sum_{i=1}^d \mathbb{Z}\mathbf{b}_i$ . If  $d = n$ , the lattice is considered full-rank.

Let  $N = 2^k$  be a power of 2. Let  $\psi(N) = X^N + 1$  be a power-of-2 cyclotomic polynomial. Let  $R = \mathbb{Z}[X]/\psi(N)$ . For  $b \in R$ , we denote  $\|b\|_2$  the Euclidean norm of the vector of its coefficients. Let  $R_q = \mathbb{Z}_q[X]/\psi(N) = R/qR$ .

We define  $\mathcal{N}(\mu, \sigma)$  as the discrete Gaussian distribution with mean  $\mu$  and standard deviation  $\sigma$  – that is, a probability distribution supported on  $\mathbb{Z}$  with a probability mass function  $p(x) \propto \exp(-(x - \mu)^2/(2\sigma^2))$ . Such a distribution can be approximated by its continuous counterpart. Specifically, if  $x \leftarrow \mathcal{N}(\mu_1, \sigma_1)$  and  $y \leftarrow \mathcal{N}(\mu_2, \sigma_2)$ , respectively, the sum  $x + y$  is regarded as a draw from  $\mathcal{N}(\mu_1 + \mu_2, \sqrt{\sigma_1^2 + \sigma_2^2})$ . We can make the distribution over  $\mathbb{Z}_q$  by reducing the output mod  $q$ .

The best-known FHE schemes are built upon the Learning with Errors (LWE) problem [Reg05]. In the CKKS scenario, we focus more on its ring version, known as the Ring-LWE problem [LPR10]. The Ring-LWE distribution is obtained as follows.

**Definition 1.** Let  $k \geq 1, q \geq 2, \alpha \in \mathbb{R}, s \in R_q$ . The Ring-LWE distribution is obtained as follows:

$$\begin{aligned} a &\leftarrow \mathcal{U}(R_q); \\ e &\leftarrow R \quad \text{with coefficients} \leftarrow \mathcal{N}(0, \frac{\alpha q}{\sqrt{2\pi}}); \end{aligned}$$

The oracle subsequently returns  $(a, a \cdot s + e) \in R_q \times R_q$ .

## 2.2 Approximate Homomorphic Encryption

In their work [Che+17], Cheon et al. introduced the first approximate homomorphic encryption scheme, commonly referred to as CKKS, named after its creators. Here, we first present the definition of Homomorphic Encryption. Gentry's bootstrapping blueprint [Gen09] allows for the construction of a Fully Homomorphic Encryption (FHE) scheme from an HE scheme.

**Definition 2** (Homomorphic Encryption (HE)). *A (public-key) HE scheme comprises four polynomial time algorithms:*

- *A (randomized) key generation algorithm*  $\text{KeyGen} : 1^\kappa \rightarrow \mathcal{PK}^2 \times \mathcal{SK}$  *that generates the public key, evaluation key, and secret key.*
- *A (randomized) encryption algorithm*  
 $\text{Enc} : \mathcal{PK} \times \mathcal{M} \rightarrow \mathcal{C}.$
- *A (deterministic) decryption algorithm*  
 $\text{Dec} : \mathcal{SK} \times \mathcal{C} \rightarrow \mathcal{M}.$
- *A (deterministic) homomorphic evaluation function*  
 $\text{Eval} \forall k, \mathcal{PK} \times (\mathcal{M}^k \rightarrow \mathcal{M}) \times \mathcal{C}^k \rightarrow \mathcal{C}.$

In the given definition,  $\mathcal{PK}$  signifies the collection of public keys, inclusive of the evaluation keys, while  $\mathcal{SK}$  denotes the set of secret keys. Additionally,  $\mathcal{M}$  represents the message space, and  $\mathcal{C}$  stands for the ciphertext space. The function provided to the evaluation method is a circuit comprising multiple gates.

In standard HE schemes, the correctness of the scheme is defined as

**Definition 3** (HE Correctness). *For an arbitrary circuit  $\mathfrak{C}$  the following holds*

$$\text{Dec}_{\text{sk}}(\text{Eval}_{\text{ek}}(\mathfrak{C}, (c_1, \dots, c_k))) = \mathfrak{C}(m_1, \dots, m_k),$$

where  $(\text{pk}, \text{ek}, \text{sk}) = \text{KeyGen}(1^\kappa)$  and  $c_i = \text{Enc}_{\text{pk}}(m_i)$  for all  $i \leq k$ .

**Approximate HE schemes** Approximate HE schemes are a special type of HE schemes where the notion of correctness does not hold. They allow approximate arithmetic on encrypted data, considering errors of homomorphic operations as part of the computational errors in the approximate computation. As the first approximate HE proposal, CKKS supports two approximate homomorphic operations:

- $\text{Dec}_{\text{sk}}(\text{Add}(\text{Enc}_{\text{pk}}(m_1), \text{Enc}_{\text{pk}}(m_2))) \simeq m_1 + m_2,$
- $\text{Dec}_{\text{sk}}(\text{Mult}(\text{Enc}_{\text{pk}}(m_1), \text{Enc}_{\text{pk}}(m_2))) \simeq m_1 * m_2,$

with a tracker on the size of the noise. Next, we will delve into the specifics of the CKKS scheme, beginning with its method for message encoding.

**The CKKS message encoding.** In order to perform pointwise addition and multiplication of vectors in the SIMD manner, the CKKS scheme designs the following encoding method. Let  $\omega$  be the  $2N$ -th root of unity. Let  $\phi : R \rightarrow \mathbb{C}$  be the mapping from  $a(X) \in R$  to  $\phi(a) = \hat{\mathbf{a}} = (a(\omega^{4j+1}))_{j=0}^{N/2-1} \in \mathbb{C}^{N/2}$ . The CKKS scheme chooses a positive real  $\Delta$  called scaling factor and defines the encoding and decoding functions as

- $\text{Encode}(\mathbf{z} \in \mathbb{C}^{N/2}; \Delta) = \lfloor \Delta \cdot \phi^{-1}(\mathbf{z}) \rfloor$ ,
- $\text{Decode}(a \in R; \Delta) = \phi(\Delta^{-1} \cdot a) \in \mathbb{C}^{N/2}$ ,

where  $\phi^{-1}$  is the inverse function of  $\phi$ .

**The CKKS scheme [Che+17].** The CKKS scheme supports the encryption and decryption of floating-point inputs using the message encoding interface. We now consider the plaintext space as  $R_q$  and the ciphertext space as  $R_q^2$ . We have a ciphertext module  $q$  and a discrete Gaussian error distribution  $\chi$ , where  $\chi = \mathcal{N}(0, \sigma)$ . We bypass many intricate details, focusing on the primary procedures and attributes pertinent to this study. The following description is from [Li+22]. For a more comprehensive understanding, we refer the reader to [Che+17].

- **CKKS.KeyGen( $1^\kappa$ ):** Let  $w = w(\kappa)$  and  $p = p(\kappa, q)$ . We sample  $s$  from the set  $s \in \{-1, 0, 1\}^N$  with  $w$  non-zero entries in  $s$  and designate  $\text{sk}$  as  $(1, -s)$ . Alternatively,  $s$  can be sampled from a uniform distribution over  $\{-1, 0, 1\}^N$ . We sample  $a \leftarrow R_q$ ,  $e \leftarrow \chi$  and take  $\text{pk} = (a, b)$  with  $b = a \cdot s + e$ . We sample  $a' \leftarrow R_{pq}$ ,  $e' \leftarrow \chi$  and take  $\text{ek} = (a', b')$  with  $b' = a' \cdot s + e + p \cdot s^2 \pmod{pq}$ . Return  $(\text{sk}, \text{pk}, \text{ek})$ .
- **CKKS.Enc<sub>pk</sub>( $m$ ):** Let  $\xi$  represent the distribution over  $\{-1, 0, 1\}^N$ , where each coefficient is independently sampled as 1 or  $-1$  with a probability of  $1/4$ , and 0 with a probability of  $1/2$ . Then, we sample  $r \leftarrow \xi$ ,  $e_0, e_1 \leftarrow \chi$ , and return  $r \cdot \text{pk} + (m + e_0, e_1) \pmod{q}$ .
- **CKKS.Add( $c_0, c_1$ ):** Return  $c_0 + c_1 \pmod{q}$ .
- **CKKS.Mult<sub>ek</sub>( $c_0, c_1$ ):** Assume  $c_0 = (a_0, b_0)$  and  $c_1 = (a_1, b_1)$ . Return  $(a_2, b_2)$ , where  $(a_2, b_2) = (a_0 b_1 + a_1 b_0, b_0 b_1) + \lfloor p^{-1} \cdot a_0 a_1 \cdot \text{ek} \rfloor \pmod{q}$ .
- **CKKS.Dec<sub>sk</sub>( $c$ ):** For  $c = (a, b) \in R_q^2$ , return  $b - a \cdot s \pmod{q}$ .

Analyzing the addition operation in the worst-case scenario can be done straightforwardly. Suppose  $c_0$  and  $c_1$  are two ciphertexts with positive error bounds  $\text{err}b_0$  and  $\text{err}b_1$ , respectively. The ciphertext resulting from the Add operation on  $c_0$  and  $c_1$  would have an error bounded to  $\text{err}b_0 + \text{err}b_1$ .

### 2.3 Security notions

The standard notion of security for FHE is IND-CPA (Indistinguishability - Chosen Plaintext Attack) security defined as follows.

**Definition 4** (IND-CPA security). *We say that a FHE scheme  $(\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Eval})$  is IND-CPA secure if each efficient adversary  $\mathcal{A}$  has negligible advantage  $\text{Adv}_{\mathcal{A}}(\kappa)$ , where the advantage is defined as*

$$\text{Adv}_{\mathcal{A}}(\kappa) = \left| \Pr \left[ \mathcal{A} \xrightarrow{\text{Expr}_0} 1 \right] - \Pr \left[ \mathcal{A} \xrightarrow{\text{Expr}_1} 1 \right] \right|.$$

*The experiments are as follows :*

$$\begin{array}{ccc} \text{Expr}_b : & \mathcal{C} & \mathcal{A} \\ & (\text{sk}, \text{pk}, \text{ek}) \leftarrow \text{KeyGen} & \xrightarrow{\text{pk}} \\ & c \leftarrow \text{Enc}(\text{pk}, m_b) & \xleftarrow{m_0, m_1} \\ & & \xrightarrow{c} \text{Outputs a bit } b \end{array}$$

An additional security notion,  $\text{IND-CPA}^D$ , proves vital when the decryption outputs require sharing, as pointed out in [LM21]. For non-approximate HE, the  $\text{IND-CPA}^D$  notion is proven to be equivalent to the IND-CPA notion in [LM21]. However, for approximate HE, it represents a much stronger security notion. We present a formal definition as follows.

**Definition 5** ( $\text{IND-CPA}^D$  security [LM21]). *Given a public-key homomorphic (possibly approximate) encryption scheme  $\mathcal{E}$ , where  $\mathcal{E} = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Eval})$ , with plaintext space  $\mathcal{M}$  and ciphertext space  $\mathcal{C}$ , we define an experiment  $\text{Expr}_b^{\text{indcpa}^D}[\mathcal{A}]$ , parameterized by a bit  $b \in \{0, 1\}$  and involving an efficient adversary  $\mathcal{A}$  that is given access to the following oracles, sharing a common state  $S \in (\mathcal{M} \times \mathcal{M} \times \mathcal{C})^*$  consisting of a sequence of message-message-ciphertext triples:*

- *An encryption oracle  $E_{\text{pk}}(m_0, m_1)$  that, given a pair of plaintext messages  $m_0, m_1$ , computes  $c \leftarrow \text{Enc}_{\text{pk}}(m_b)$ , extends the state*

$$S := [S; (m_0, m_1, c)],$$

*with one more triplet, and returns the ciphertext  $c$  to the adversary.*

- *An evaluation oracle  $H_{\text{ek}}(g, J)$  that, given a function  $g : \mathcal{M}^k \rightarrow \mathcal{M}$  and a sequence of indices  $J = (j_1, \dots, j_k) \in \{1, \dots, |S|\}^k$ , computes the ciphertext  $c \leftarrow \text{Eval}_{\text{ek}}(g, S[j_1].c, \dots, S[j_k].c)$ , extends the state*

$$S := [S; (g(S[j_1].m_0, \dots, S[j_k].m_0), \\ g(S[j_1].m_1, \dots, S[j_k].m_1), c)]$$

*with one more triplet, and returns the ciphertext  $c$  to the adversary. Here and below  $|S|$  denotes the number of triplets in the sequence  $S$ , and  $S[j].m_0, S[j].m_1, S[j].c$  denote the three components of the  $j$ -th element of  $S$ .*

- A decryption oracle  $D_{sk}(j)$  that, given an index  $j \leq |S|$ , checks whether  $S[j].m_0 = S[j].m_1$ , and, if so, returns  $Dec_{sk}(S[j].c)$  to the adversary. (If the check fails, a special error symbol  $\perp$  is returned.)

The experiment is defined as

$$\begin{aligned} \text{Expr}_b^{\text{indcpa}^D}[\mathcal{A}](1^\kappa) : & (\text{sk}, \text{pk}, \text{ek}) \leftarrow \text{KeyGen}(1^\kappa) \\ & S := [] \\ & b' \leftarrow \mathcal{A}^{\text{E}_{pk}, \text{H}_{ek}, D_{sk}}(1^\kappa, \text{pk}, \text{ek}) \\ & \text{return}(b') \end{aligned}$$

The advantage of adversary  $\mathcal{A}$  against the  $\text{IND-CPA}^D$  security of the scheme is  $\text{Adv}_{\text{indcpa}^D}[\mathcal{A}](\kappa)$  defined as

$$\left| \Pr \left[ \text{Expr}_0^{\text{indcpa}^D}[\mathcal{A}](1^\kappa) = 1 \right] - \Pr \left[ \text{Expr}_1^{\text{indcpa}^D}[\mathcal{A}](1^\kappa) = 1 \right] \right|,$$

where the probability is over the randomness of  $\mathcal{A}$  and the experiment. The scheme  $\mathcal{E}$  is  $\text{IND-CPA}^D$ -secure if for any efficient (probabilistic polynomial time)  $\mathcal{A}$ , the advantage  $\text{Adv}_{\text{indcpa}^D}[\mathcal{A}](\kappa)$  is negligible in  $\kappa$ .

Another relevant security notion is the  $\text{KR}^D$  security, implied by the  $\text{IND-CPA}^D$  security. Although  $\text{KR}^D$  security is considered much weaker than the  $\text{IND-CPA}^D$  security, a key-recovery attack is predominantly employed to demonstrate the vulnerability of the analyzed schemes. This methodology aligns with the research presented in this paper. We present the formal definition of  $\text{KR}^D$  security in Definition 6.

**Definition 6** ( $\text{KR}^D$  security [Li+22]). Let  $\mathcal{E} = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Eval})$  be a public-key homomorphic (possibly approximate) encryption scheme with plaintext space  $\mathcal{M}$  and ciphertext space  $\mathcal{C}$ . We define a game  $\text{Expr}^{\text{KR}^D}[\mathcal{A}]$ , parameterized by an adversary  $\mathcal{A}$  that is given access to the (stateful) oracles  $E'_{pk}, H_{ek}, D_{sk}$ , where  $E'_{pk}(m) := E_{pk}(m, m)$  with oracles  $E_{pk}, H_{ek}, D_{sk}$  defined as in Definition 5.

The game is defined as

$$\begin{aligned} \text{Expr}^{\text{KR}^D}[\mathcal{A}](1^\kappa) : & \\ & (\text{sk}, \text{pk}, \text{ek}) \leftarrow \text{KeyGen}(1^\kappa) \\ & S := [] \\ & \text{sk}' \leftarrow \mathcal{A}^{E'_{pk}, H_{ek}, D_{sk}}(1^\kappa, \text{pk}, \text{ek}) \\ & \text{return}(\text{sk} == \text{sk}') \end{aligned}$$

The scheme  $\mathcal{E}$  is said to have  $k$  bits of  $\text{KR}^{\text{D}}$ -security if, for any adversary  $\mathcal{A}$ ,

$$k \leq \log_2 \frac{T(\mathcal{A})}{\text{Adv}_{\mathcal{A}}},$$

where  $T(\mathcal{A})$  is the running time of  $\mathcal{A}$ .

In [Li+22], the application of differential privacy technique results in a finer-grained definition of bit-security, parameterized by both a computational parameter  $c$  and a statistical parameter  $\nu$ . We define this security notion as follows.

**Definition 7** ( $(c, \nu)$ -bits of security [Li+22]). Let  $\Pi$  be a cryptographic primitive, and  $\mathcal{G}$  be an indistinguishability game. Let  $\text{Adv}_{\mathcal{A}}$  be the advantage of an adversary  $\mathcal{A}$  in breaking the security of  $\Pi$  in the  $\mathcal{G}$  game. The scheme  $\Pi$  has  $(c, \nu)$ -bits of  $\mathcal{G}$ -security if, for any adversary  $\mathcal{A}$ , either

$$\log_2 \frac{T(\mathcal{A})}{\text{Adv}_{\mathcal{A}}} \geq c, \quad \text{or}, \quad \log_2 \frac{1}{\text{Adv}_{\mathcal{A}}} \geq \nu. \quad (1)$$

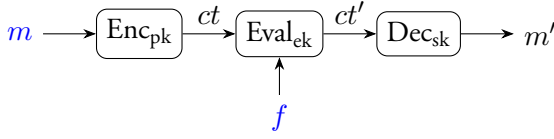
### 3 New key-recovery attacks

In this section, we improve upon key-recovery attacks initially presented by Li and Micciancio [LM21], when non-worst-case noise estimation is employed. We begin with the theoretical framework, then transition to the security implications for the OpenFHE and HELib homomorphic encryption libraries.

#### 3.1 Theoretical framework

**Attack model.** We adopt the adversary model defined in [LM21], as also illustrated in Definition 6 and Figure 1. In this model, the adversary gains access to three stateful oracles  $E'_{\text{pk}}$ ,  $H_{\text{ek}}$ , and  $D_{\text{sk}}$ , with  $E'_{\text{pk}}(m) := E_{\text{pk}}(m, m)$ . These oracles are defined in Definition 5. The adversary aims to recover the secret key  $\text{sk}$  and is capable of querying the three oracles, selecting messages and evaluation functions as inputs. Notably, this adversary remains passive; unlike an active adversary who can manipulate arbitrary ciphertexts, this model restricts the adversary to decryption queries on valid ciphertexts produced through legitimate operations, such as encryption and homomorphic computation via public interfaces or APIs provided by HE libraries.

**Real-world scenarios.** In practical applications, it is highly likely that decrypted data will be shared among parties who lack access to the secret key, making  $\text{IND-CPA}^{\text{D}}$  security indispensable. This sharing could facilitate significant applications like two-party computation for Privacy-Preserving Machine Learning as a Service (MLaaS), wherein the server owns a model it wishes to provision as a service, while the client



**Figure 1:** Figure illustrates a passive attacker scenario based on Li and Micciancio’s model [LM21]. In this context, the attacker may choose/know the plaintext  $m$  and the homomorphic function  $f$ . The attacker can eavesdrop on the ciphertexts  $ct$ ,  $ct'$  and decryption result  $m'$ , but is not allowed to tamper with (or inject) ciphertexts or alter the computation’s final outcome.

seeks to obtain a homomorphically computed inference on its confidential data. The ecosystem might comprise a user holding the secret key and other participants lacking it, who function as passive adversaries. Service providers contribute by supplying APIs and computational resources.

An interesting case study involves a commercial application that collects encrypted data from the user and leverages a cloud service’s API for homomorphic computations. We posit that the application can only invoke the API using ciphertexts authenticated by the user, thereby precluding the tampering or injection of fraudulent ciphertexts. The API invocation resembles the usage of the evaluation oracle  $H_{ek}$  as outlined in Definition 5. Should the user disclose the decrypted outcome, our novel attack strategy remains viable, even if non-worst-case noise-flooding countermeasures have been deployed.

**The Li-Micciancio attack.** Similar to [LM21], we use a simplified CKKS scheme without the encoding and decoding procedures as an example to explain the attack idea. The secret key is  $sk=(1, -s)$ , where  $s$  is a power-of-two cyclotomic ring element in  $R_q$ . The only unknown part of the secret key is  $s$ , so the recovery of  $sk$  is equivalent to recovering  $s$ . The adversary encrypts  $m$  as  $Enc(m) = (a, b) \in R_q^2$ , where  $a \in R_q$  is selected at random and  $b = a \cdot s + e + m$  for a polynomial  $e \in R_q$  where the coefficients of  $e$  are a small perturbation from the distribution  $\mathcal{N}(0, \sigma_1)$ . The decryption function computes  $Dec_s((a, b)) = b - a \cdot s = m + e$ . In the Li-Micciancio attack, the adversary  $\mathcal{A}$  selects the message  $m = 0$  and requests the computation of the identity function, effectively equivalent to performing no operation. The adversary then proceeds to request an approximate decryption and carries out subsequent computations of

$$b' = b - Dec_s((a, b)) = a \cdot s.$$

The secret key  $s$  can then be recovered by computing

$$s = a^{-1} \cdot b',$$

if  $a$  is an invertible element in  $R_q$ . Otherwise, the adversary requires for more decryption queries.

**Noise-flooding countermeasures.** Li and Micciancio have proposed a countermeasure from the noise-flooding technique, for protecting against their attack in [LM21]. The main idea is to introduce an additional noise in the decryption function. Thus, the decryption function of CKKS, named decryption for share, can be defined as

$$\text{Dec}_s^D((a, b)) = \text{Dec}_s((a, b)) + e_{\text{new}},$$

where  $e_{\text{new},i} \leftarrow \mathcal{N}(0, \sigma_2)$  and  $e_{\text{new},i}$  is the coefficient of  $e_{\text{new}}$ . When the adversary  $\mathcal{A}$  computes  $b' = b - \text{Dec}_s^D((a, b))$ , the output is  $a \cdot s + e_{\text{new}}$ ; if the noise is sufficiently large, then the new ring-LWE instance  $(a, a \cdot s + e_{\text{new}})$  is hard to solve.

**Achieving IND-CPA<sup>D</sup> through differential privacy [Li+22].** The general idea is to add a noise with sufficiently large variance. The newly proposed schemes, referred to as S-CKKS <sub>$\sigma$</sub> , are detailed in [Li+22]. Initially, they establish a noise bound, ct.t, designed to accommodate the worst-case noise growth, following which they add a component-wise noise,  $\mathcal{N}(0, \sigma \cdot \text{ct.t})$ , in the noise-flooding countermeasure. In [Li+22], the following lemma is proved.

**Lemma 1** ([Li+22]). *Assume the number of decryption oracle calls is bounded to  $q$  and  $\sigma = \sqrt{24qn}2^{\nu/2}$ . If CKKS is  $(c + \log_2 24)$ -bit IND-CPA-secure, then S-CKKS <sub>$\sigma$</sub>  is  $(c, \nu)$ -bit IND-CPA<sup>D</sup>-secure.*

On the other hand, for recent open-source implementations (e.g., in [Bad+22]), non-worst-case noise estimation approaches are preferred to enhance efficiency.

**New attack scenario.** We describe an attack scenario when the standard deviation  $\sigma_2$  of the added Gaussian noise in the noise-flooding countermeasure is a function that grows much slower than the estimated noise in the worst-case noise analysis.

In alignment with the IND-CPA<sup>D</sup> model outlined in Definition 5, the adversary has the capability to execute computations on valid ciphertexts using a chosen function. Specifically, the adversary employs the evaluation oracle  $H_{\text{ek}}$  for homomorphic calculations. Our primary focus is on the evaluation function  $g_t(\mathbf{c})$ , which performs homomorphic addition of  $t$  ciphertexts  $\mathbf{c} = (c_1, c_2, \dots, c_t)$ . This function, illustrated in Figure 2, can be executed through either an iterative addition over  $t - 1$  layers or a tree-based addition spanning  $\lceil \log_2(t) \rceil$  layers. Depending on the approach, the gate latency stands at either  $t - 1$  or  $\lceil \log_2(t) \rceil$ .

In this new attack scenario, we stick with the case where  $m = 0$ , first generate a ciphertext  $c_0 = (a, b)$ , and then homomorphically evaluate the addition  $g_t(\mathbf{c})$ ,



Figure 2: Iterative and tree-based addition for  $g_t(\mathbf{c})$  with  $t = 4$

where  $\mathbf{c}$  is a vector repeating the ciphertext  $c_0$  for  $t$  times. The noise, therefore, is grown from  $e$  to  $t \cdot e$ . The new decryption function  $\text{Dec}_s^D(\cdot)$  gives us the value of

$$e_{\text{total}} = t \cdot e + e_{\text{new}}.$$

We need the following lemma to characterize the distribution of conditional probability  $(e_i | e_{\text{total},i} = l)$ , where  $e_i$  and  $e_{\text{total},i}$  are the  $i$ -th coefficient of  $e$  and  $e_{\text{total}}$ , respectively.

**Lemma 2** ([Lee+18]). *For two positive numbers  $\sigma_a$  and  $\sigma_b$ , let  $X$  and  $Y$  be variables of distributions  $\mathcal{N}(0, \sigma_a)$  and  $\mathcal{N}(0, \sigma_b)$ , respectively. Let  $\sigma = \sqrt{\sigma_a^2 + \sigma_b^2}$ , then  $(X | X + Y = l)$  follows the distribution  $\mathcal{N}(l \cdot \frac{\sigma_a^2}{\sigma^2}, \frac{\sigma_a \sigma_b}{\sigma})$ .*

In the attack, we can obtain the value of  $e_{\text{total}}$  from the new decryption function. Conditioned on this observation, the distribution of the noise  $e$  needs to be updated. Since  $t \cdot e_i$  is sampled from  $\mathcal{N}(0, t \cdot \sigma_1)$  and  $e_{\text{new},i}$  is sampled from  $\mathcal{N}(0, \sigma_2)$ , from Lemma 2, we know  $(t \cdot e_i | e_{\text{total},i} = l)$  follows the distribution  $\mathcal{N}(l \cdot \frac{t^2 \sigma_1^2}{\sigma_2^2 + t^2 \sigma_1^2}, \frac{t \cdot \sigma_1 \cdot \sigma_2}{\sqrt{t^2 \cdot \sigma_1^2 + \sigma_2^2}})^3$  and, hence,  $(e_i | e_{\text{total},i} = l)$  follows the distribution  $\mathcal{N}(l \cdot \frac{t \sigma_1^2}{\sigma_2^2 + t^2 \sigma_1^2}, \frac{\sigma_1 \cdot \sigma_2}{\sqrt{t^2 \cdot \sigma_1^2 + \sigma_2^2}})$ . Thus, for each value of  $e_{\text{total},i}$ , we compute

$$b'_i = b_i - e_{\text{total},i} \cdot \frac{t \cdot \sigma_1^2}{\sigma_2^2 + t^2 \sigma_1^2}. \quad (2)$$

The new polynomial  $b'$  with the  $i$ -th coefficient  $b'_i$  satisfies

$$b' = a \cdot s + e', \quad (3)$$

<sup>3</sup>In [Lee+18], the authors assume a continuous Gaussian distribution. In contrast, our approach in Lemma 2 utilizes discrete Gaussians as approximations for their continuous counterparts. Empirical tests conducted with OpenFHE's discrete Gaussian distributions, as depicted in Figure 3, affirm the precision of our approximation methodology. For an in-depth discussion, refer to Section 3.2.

where the coefficients of  $e'$  follow the distribution of  $\mathcal{N}(0, \frac{\sigma_1 \cdot \sigma_2}{\sqrt{t^2 \cdot \sigma_1^2 + \sigma_2^2}})$ , we use  $\sigma_{\text{attack}}$  to denote the standard deviation of it.

**Intuition.** We next show applications where the standard deviation of the new noise variable can be very small, making the new Ring-LWE problem easy to solve. To simplify the analysis, we let  $t$  be arbitrarily large. This assumption permits us to focus on the leading terms and consider the asymptotic behavior. In the first scenario, where  $\sigma_2 = \sqrt{t}\sigma_1$ , the standard deviation is of the order  $\Theta(t^{-1/2})$ . Conversely, in the second scenario with  $\sigma_2$  as a constant, the standard deviation is  $\Theta(t^{-1})$ . In either case, it is feasible to choose a sufficiently large  $t$  to minimize the noise to negligible levels.

It is important to note that this assumption serves merely as a simplifying element for illustrative purposes. We are constrained by a noise budget and a target level of precision. In the attack scenarios explored throughout the paper, we begin operations using a fresh ciphertext with a small noise level. Consequently, the  $t$  values required to initiate the attack do not exceed the permissible noise thresholds in the associated libraries.

**Efficient implementation of  $g_t(\mathbf{c})$  for the adversary using  $\mathbf{c}$  as a  $t$ -fold repetition of  $c_0$ .** Let  $t_{\lambda_0-1} \dots t_1 t_0$  be the binary representation of  $t$ , where  $t_0$  is the least significant bit. In the tree-based addition variant of  $g_t(\mathbf{c})$ , where  $\mathbf{c}$  is a vector composed of  $t$  repetitions of the ciphertext  $c_0$ , each addition operation within the same layer receives and produces identical input and output. This condition leads to the transformation of  $g_t(\mathbf{c})$  into an alternate function, denoted as  $g'_t(\mathbf{c})$ , which effectively performs the following operations (which is similar to exponentiation by squaring<sup>4</sup>):

- Initialize: we assume that the most significant bit is set, i.e.  $t_{\lambda_0-1} = 1$ , let  $c_1 \leftarrow c_0$ .
- For  $i$  in the range  $\{1, \dots, \lambda_0 - 1\}$  iteratively compute

$$c_{i+1} \leftarrow \begin{cases} c_i + c_i & \text{if } t_{\lambda_0-1-i} = 0 \\ c_i + c_i + c_0 & \text{otherwise} \end{cases}$$

- Eventually output  $c_{\lambda_0}$ .

This implementation requires at most  $2\lambda_0$  homomorphic additions and exhibits a latency of  $\lambda_0$ . When querying the evaluation oracle using either the evaluation function  $g_t(\mathbf{c})$  or  $g'_t(\mathbf{c})$  with  $\mathbf{c}$  as a  $t$ -fold repetition of  $c_0$ , the resulting ciphertexts

<sup>4</sup>Here, we present the case for arbitrary  $t$  to provide a more general perspective on the attack. However, it's worth noting that in practical applications, one would likely choose  $t$  to be a power of 2. As a result, the function  $g'_t$  becomes considerably simpler, as it merely calculates  $c_{i+1} \leftarrow c_i + c_i$ .

are identical, while the amount of computation needed to evaluate  $g'_t$  is greatly reduced.

### 3.2 Average-case and empirical noise estimations are not $\text{KR}^D$ secure

In the threat model, the adversary may select messages and operations for evaluation. If this did not match with the computation or message distribution used to heuristically determine the error bound, then the security guarantee wrt. the Li-Micciancio attack can be destroyed.

The error bound should carefully record the real worst-case bound that can be achieved in the computation; otherwise, the adversary could fool the user. For instance, the in-trusted user may pick a different message and different evaluation circuit to lead to a much larger real error than the estimated error in the decryption client.

In [Cos+22], Costache et al. presented an average-case noise analysis for the CKKS scheme using the central limit theorem (CLT). The authors discussed that their approach is suitable for the noise-flooding countermeasure. For example, they wrote in [Cos+22] that

“Not only does our work provide tighter bounds for  $B_{\text{ctxt}}$  (resolving an open problem in [13]), but also our analysis enables us to directly characterise the distribution of  $e$  and its variance.”

In this subsection, we present a simple key-recovery attack showing that the average-case approach can be problematic when being applied in the mode releasing the decryption results.

**The CLT noise estimation.** In such a scenario, the precision tracker needs to make some heuristic assumptions such that the added or multiplied random variables are independent. Then, the tracker can get tighter noise bound using the central limit theorem. For instance, the addition of two ciphertexts will lead to a new ciphertext with noise variance the sum of the noise variances of the two input ciphertexts. In the worst-case analysis, we add the standard deviation instead.

The average-case precision tracker can be more accurate in the average case; the adversary, however, may be able to pick inputs to falsify the assumed independence assumption and make the average-case estimation inaccurate.

We consider the circuit of adding  $t$  ciphertexts. In the average case, adding  $t$  independent ciphertexts with noise  $\epsilon_i \sim \mathcal{N}(0, \sigma I_n)$  will lead to new ciphertexts with noise  $\epsilon \sim \mathcal{N}(0, \sqrt{t}\sigma I_n)$ . If the adversary picks  $t$  identical ciphertexts, then the noise distribution follows  $\epsilon \sim \mathcal{N}(0, t\sigma I_n)$ .

Importantly, the adversary can efficiently execute  $g_t(\mathbf{c})$  by employing  $g'_t(\mathbf{c})$ , where  $\mathbf{c}$  represents a  $t$ -fold repetition of a ciphertext  $c_0$ . When using a CLT noise tracker that assumes the independence of the two inputs in a homomorphic addition gate, one can still approximate the output ciphertext's estimated

noise as  $\epsilon \sim \mathcal{N}(0, \sqrt{t}\sigma I_n)$ . The actual noise distribution similarly adheres to  $\epsilon \sim \mathcal{N}(0, t\sigma I_n)$ .

**The empirical noise estimation.** Both the worst-case noise estimation and the CLT noise estimation can be designed so that a ciphertext inherently carries a noise bound, which homomorphic evaluations then automatically update. An alternative approach is what we term ‘empirical noise estimation’, where the user holding the secret key, or other involved parties, conduct preliminary experiments to set a noise bound for decryption with sharing. However, this empirical method generally presupposes that an adversary must perform certain computations, thereby contradicting the IND-CPA<sup>D</sup> adversary model, which allows the choice of the evaluation function. Additionally, the user in possession of the secret key may lack prior knowledge of the function to be evaluated, as could occur in cases involving private circuits. This lack of information renders the assumptions underlying empirical noise estimation invalid.

Estimating empirical noise can compromise noise-flooding protection due to the formal adversary model in IND-CPA<sup>D</sup> or KR<sup>D</sup> security, which allows the attacker to select valid ciphertexts and perform computations on them. Furthermore, the distribution of ciphertexts and computation gates significantly influences empirical noise estimation. The attacker can inflate the actual noise beyond the estimated noise by choosing identical ciphertexts as inputs to the simple  $g_t()$  function and selecting a much larger value of  $t$  than what was used in the empirical noise estimation.

To be specific, the noise bound predetermined by the user possessing the secret key for decryption essentially acts as a constant noise level for the adversary who selects the  $t$  value in the  $g_t()$  function. Utilizing the attack strategy outlined in Section 3.1, a passive adversary may choose a sufficiently large  $t$  value, thereby rendering the noise negligible.

### Experiments on average-case noise estimation

In this section, we present the experimental results on the OpenFHE library, focusing on average-case noise estimation.

**The noise estimation in OpenFHE.** OpenFHE suggests performing the computation in two steps. First, the noise is estimated on a freshly generated key pair with messages chosen from a suitable set of messages. The noise estimation is done by measuring the noise/precision-loss in the imaginary slots of the decrypted plaintext [Cos+22]. In this step, the value  $\sigma_{\text{est}}$  is computed. During the actual computation, the decrypted message is subjected to noise flooding with discrete Gaussian noise with a standard deviation of  $\sigma_2 = \sqrt{12q}2^{\nu/2}\sigma_{\text{est}}$ . Here,  $q$  represents the number of decryption queries ( $q = 1$  in our attack), and  $\nu \geq 30$  is the statistical security parameter.

The split of computation in two steps signifies OpenFHE’s use of an empirical noise estimation technique. Concurrently, OpenFHE intends to introduce a feature akin to HELib’s noise estimator, which associates a noise bound with each ciphertext and adjusts it dynamically during homomorphic operations (refer to [Bad+22]). This embedded noise bound in individual ciphertexts serves as an effective mechanism to reconcile disparities between predicted and actual noise levels that could arise due to a different adversary-chosen evaluation function.

OpenFHE recommends for the use of its own empirical noise evaluation strategy, initially suggesting a method akin to average-case noise estimation. The results from this method are, in specific cases under scrutiny, the same as those acquired through the CLT noise estimation method. Specifically, this involves gauging the noise or precision-loss in the imaginary slots of the decrypted plaintext [Cos+22]. In the discussion that follows, we deliberately apply identical computations in both phases, to negate the impact of function variations chosen by an adversary. Our attention is specifically geared towards OpenFHE’s average-case noise estimation, highlighting the disparities arising from the input message distributions for simple gates that an adversary could select.

In Section 3.2, we demonstrate that OpenFHE’s primary recommendation of empirical noise estimation is more vulnerable to attacks. Specifically, this vulnerability arises because the adversary has the freedom to select a different evaluation function.

**Key recovery attack on OpenFHE.** The attack scenario follows Section 3.1. The user sets up the parameters for the scheme that are provided in Table 1 and generates public and private keys. Then he estimates the noise of  $g_t$  for another freshly generated key pair for the adversary’s choice of  $t$  and initializes the oracles. The adversary then uses encryption and evaluation oracles to compute a ciphertext from  $g_t(\mathbf{c})$  and calls decryption oracle on it. The function  $g_t(\cdot)$  is implemented as  $g'_t(\cdot)$  by both the user and the adversary.

For the  $g_t(\cdot)$  function, the estimated noise  $\sigma_{\text{est}}$  is very close to  $\sqrt{t}\sigma_{1,\text{est}}$ , where  $\sigma_{1,\text{est}}$  is the noise estimation for a freshly generated ciphertext. We notice a possible bug in OpenFHE,  $\sigma_{1,\text{est}}$  is close to  $3.19\sqrt{2/3 \cdot n}$  for the secret with uniform ternary coefficients, whereas the actual noise  $\sigma_1$  of a fresh ciphertext is close to  $3.19\sqrt{4/3 \cdot n}$ . This implies that the injected noise is somewhat less than expected. Though this has a minor impact on the scheme’s security, making the attack slightly more effective, the attack can still proceed without this difference.

Treating  $\nu$  as a constant, from the formula for the standard deviation of the noise flooding  $\sigma_2 = \sqrt{12}q^{2\nu/2}\sigma_{\text{est}}$ , we obtain  $\sigma_2 = \Theta(\sqrt{t}\sigma_1)$ . As mentioned earlier, by choosing a sufficiently large  $t$ , we can make  $\sigma_{\text{attack}}$  negligible. In order to obtain the same value of  $\sigma_{\text{attack}}$  for scenarios when  $\nu = 0$  and  $\nu > 0$ , one need to increase  $\log t$  by  $\nu$ . To evaluate  $g'_t(\cdot)$ , one needs no more than  $2\lceil \log t \rceil$  ciphertext additions which translates to  $2\lceil \log t \rceil n$  additions over modulo  $q$ . In

|         | $n$      | $\log q$  | $\nu$ | precision | plaintext slots |
|---------|----------|-----------|-------|-----------|-----------------|
| OpenFHE | $2^{14}$ | 155 – 181 | 30    | 25        | 16              |

**Table 1:** Parameters used in the attack on OpenFHE. The modulus  $q$  is chosen by OpenFHE dynamically, it increases for higher values of  $t$ ; range for  $q$  is presented for the values  $46 \leq \log t \leq 57$  from the Figure 3. The value  $q$  is allowed to grow up to around  $2^{410}$  for the scheme to provide 128 bits of (quantum) security [Alb+18].

other words, the attack complexity linearly depends on  $\nu$ , which means that by simply increasing  $\nu$  the attack could not be prevented.

Let us emphasize once again that the attack relies on a single decryption call. The adversary decrypts  $t \cdot c_0$ , obtains  $e_{\text{total}}$ , and computes  $b'$ . In the real attack scenario, the adversary can utilize  $b'$ ,  $a$ , and an estimate of  $e'$  to extract information about  $s$ . However, in our experiments, we employ  $s$  to derive  $e'$  from  $b'$  and validate the theoretical estimates.

Figures 3 and 4 demonstrate  $\sigma_{\text{attack}}$  and the weight of  $e'$ , respectively. These figures clearly illustrate that the parameter  $\nu$  delays the attack, while the value of  $\sigma_{\text{attack}}$  remains the same. The parameters for  $\nu = 0$  are the same as in Table 1 except that  $\log q$  is lower. When  $\log t - \nu \geq 27$ , the trivial attack of computing  $s = a^{-1}b'$  becomes possible, as  $e'$  becomes 0 with a high probability. However, the retrieval of  $s$  is not limited to this trivial attack. Even with smaller values of  $t$ , we can still obtain an RLWE sample with noise smaller than that used in the public key and retrieve the secret key with additional post-processing.

Note that since  $e'$  is sampled from a discrete Gaussian distribution, it is natural for the empirical standard deviation to be smaller than the theoretical one for small  $\sigma_{\text{attack}}$ . This effect can be observed in Figure 3 when  $\log t - \nu \geq 24$ . A slight deviation from the theoretical prediction in the preceding part of the graph could be attributed to the fact that Lemma 2 was originally formulated for the continuous Gaussian distribution, whereas we employ the discrete Gaussian distribution, which is approximated by the continuous one.

OpenFHE supports both 64-bit and 128-bit precision computations and uses 64-bit computation by default. Since for the attack to work we need, say,  $t = 2^{57}$ , the noise grows significantly and to support that we need to compile the library with `NATIVE_SIZE = 128` option to support 128-bit precision computations.

Experiments were run on a computer with an Intel i7-11700K processor running at 3.60GHz using 64 GB of RAM. For the trivial attack with  $\log t = 57$  the running time of all oracles called by the adversary is 2.7 seconds.

After oracle calls the adversary need to compute  $b'$  as in Equation (2), then he can obtain  $s = a^{-1}b'$ . For simplicity this step was implemented in SageMath, but the computation is not optimized. The total time is over one minute, where most of the time is spent on computing  $a^{-1}$ .

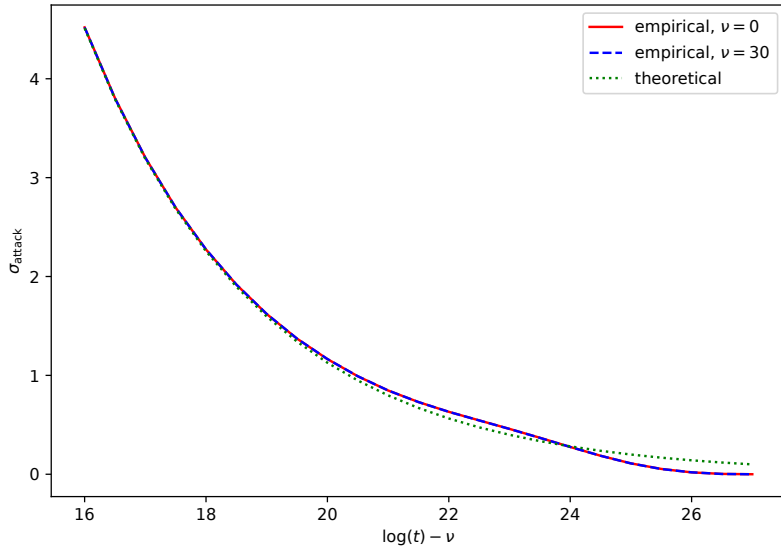
With values of  $\sigma_{\text{attack}}$  approaching 0, the coefficients of  $e'$  are usually zeroes with a few coordinates being  $-1$  or  $1$ . Outside of lattice reduction techniques, there is a simple post-processing idea, when one assumes that  $e'$  has  $i$  non-zero coefficients and brute-forces all possible  $\binom{n}{i} 2^i$  errors  $e''$  with  $i$  non-zero coefficients from  $\{-1, 1\}$  for  $i = 0, 1, \dots$ . Then the adversary can compute  $s' = a^{-1}(b' - e'')$  and check if the result is small (i.e. ternary for OpenFHE) and, if necessary, try to decrypt a ciphertext with a known message to avoid the false-positive result. Figure 5 demonstrates the success probability of the attack with and without simple post-processing, the data is the same as for Figures 3 and 4, i.e. we take empirical probability from 100 runs. The number of guesses for  $e''$  is  $\sum_{i=0}^2 \binom{n}{i} 2^i \approx 2^{29}$ , making the computation time reasonable. Guessing more positions can significantly increase the post-processing time.

**Summary.** When we focus on the parameter set outlined in Table 1 and choose  $t = 2^{57}$ , we observe an empirical success rate of 100% (100 successes in 100 trials) without the need for any post-processing. On our desktop system, the average time needed to produce the desired ciphertext to query to the decryption oracle is approximately 2.7 seconds, while the key recovery process takes up to one minute after receiving the decryption results. While it's possible to lower  $t$  to  $2^{56.5}$  if reasonable post-processing is employed, this modification has a minimal impact on the complexity and latency involved in creating the designed ciphertext. However, it considerably increases the complexity of key recovery. As a result, we recommend adhering to the straightforward version of the attack that excludes post-processing.

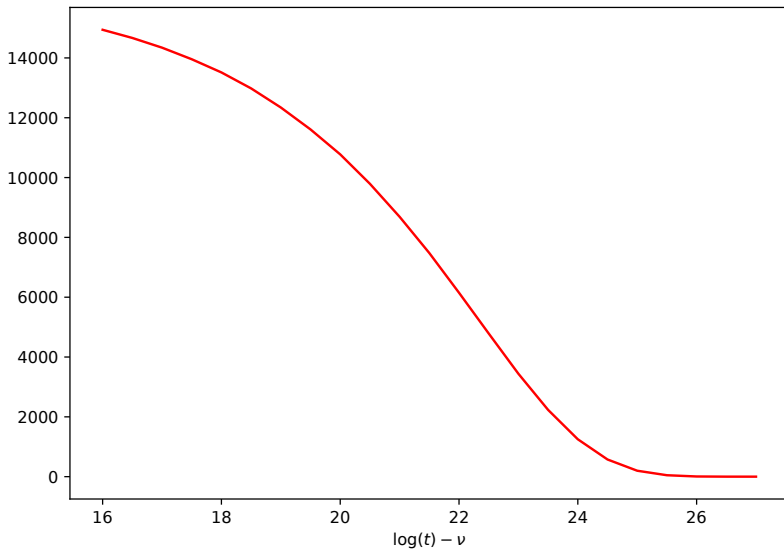
### Experiments on the empirical noise estimation

We now shift our attention to empirical noise estimation as recommended by OpenFHE. The attack scenario is the same as in Section 3.2, but now we assume that the user with secret key estimates the noise for  $g_t(\cdot)$  for some small  $t$ , say  $t = 15$ , expecting the server to do a few additions. The adversary, however, computes  $g_t(\cdot)$  for  $t = 2^{45}$  implementing  $g_t$  via  $g'_t$ . Upon receiving the decryption result, the adversary can effortlessly recover the secret  $s$  by computing  $s = a^{-1}b'$ . Note that the switch from average-case to empirical noise estimation in the described scenarios allows the adversary to reduce  $t$  from  $2^{57}$  to  $2^{45}$  for the straightforward attack without post-processing when targeting parameters from Table 1.

The required value of  $\log t$  increases with the value of the statistical parameter  $\nu$ . The scheme is using  $\nu = 30$ , therefore, even if we assume that the estimated noise is low, the starting value for  $t$  remains relatively high. Despite this, the attack can be more efficient than those discussed in Section 3.2. With the same computer platform, the adversary still requires less than three seconds to prepare the ciphertext for decryption and about one minute for key recovery.

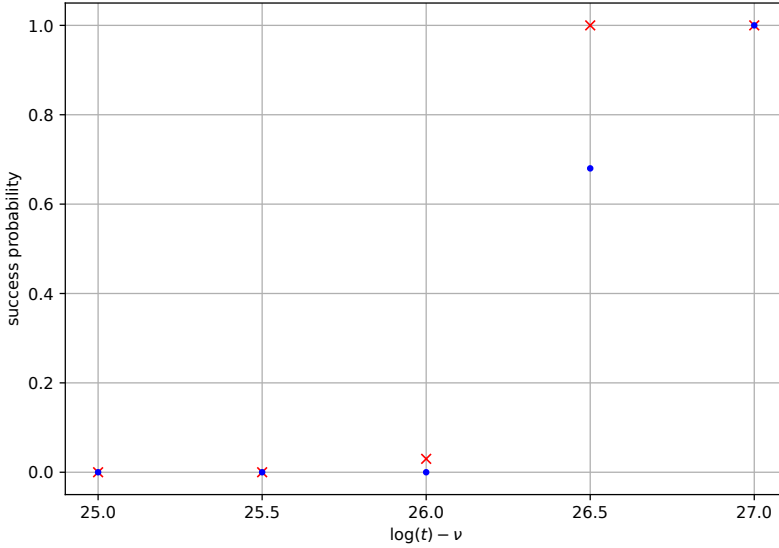


**Figure 3:** Standard deviation of  $e'$ . For empirical measurements we take average among 100 runs. The theoretical value is determined from Lemma 2 with continuous Gaussian distribution.



**Figure 4:** Average weight of  $e'$  among 100 runs.

If we consider HElib, the attack described in Section 3.2 is not successful. That



**Figure 5:** Empirical probability of successful key recovery attack: each data point based on 100 runs. Blue dots correspond to the case without post-processing, i.e. when the noise  $e'$  is zero, while red crosses show the probability of obtaining  $e'$  with weight less or equal 2, which allows for simple post-processing.

is because HELib keeps track of the error bound for the ciphertext. Most importantly, the error bound estimation employs worst-case analysis. This means that the noise after  $g_t$  with the same ciphertexts and the added noise during decryption have the similar standard deviation. However, the documentation of HELib suggests the usage of the function `ctxt.bumpNoiseBound` in case when the estimated noise is much larger than the real one, determined during the experiments. In other words, the usage of this function can transform the worst-case noise estimation into empirical noise estimation, thus potentially making the mentioned attack possible. We warn about the dangers of such function and relying on the average-case estimations since an attacker may not necessarily adhere to the underlying assumptions of the estimation.

## 4 On deterministic Gaussian noise for fresh ciphertext

In the conclusion section of [LM21], it was suggested to limit the number of decryption calls to 1 per ciphertext by simply replacing the seed of the pseudorandom function for the additional noise with the ciphertext and key derived from the decryption key. They claim that this proposed countermeasure can prevent an attack where the adversary makes the noise arbitrarily small by using a sufficiently large (still polynomial) number of calls for decrypting the same ciphertext. This coun-

termeasure has been implemented in the HELib library. The rounding protection employed in Lattigo can be regarded as a variant of adding deterministic noise. In [CHK20], Cheon, Hong, and Kim made similar claims, stating that “..an adversary can query the decryption only for fresh ciphertexts, one may choose much smaller additional noise..”. However, we will now demonstrate that the countermeasure involving deterministic noise generation does not provide additional security.

**The new idea.** Our novel observation reveals that the secret noise  $e$  of a ciphertext  $c$  may be exposed through querying decryption outputs of ciphertexts as a function incorporating  $c$ , rather than solely through the decryption outputs of the exact ciphertext  $c$ . Consequently, we generate multiple ciphertexts derived from  $c$  and obtain corresponding decryption outputs to gather information about the secret noise  $e$ . For example, for a ciphertext  $c$  which is an encryption of all zeros, we submit ciphertexts of  $t \cdot c$  for decryption, with varying values of  $t$ . This approach yields the noise  $t \cdot e + \epsilon(t \cdot c, s)$  for a specific integer  $t$ . As  $t$  changes, the pseudorandom generator’s seed alters. The error  $\frac{\epsilon(t \cdot c, s)}{t}$  remains random, ultimately allowing for the acquisition of multiple approximate samples of  $e$  to minimize the additional noise to an arbitrarily small amount.

The ciphertexts  $t \cdot c$  can be derived by homomorphically evaluating the scalar multiplication of  $c$  by  $t$  or the addition gate of  $g_t(c, \dots, c)$ . As detailed in Section 3, estimating the noise size of  $g_t(\cdot)$  using an average-case approach enables a more efficient attack on the scheme. We now illustrate that even with worst-case noise size estimation, incorporating deterministic noise for fresh ciphertexts fails to provide extra protection. A prime example is the current HELib implementation; when worst-case noise estimation with no empirical noise setting is employed, the standard deviation of  $\epsilon(t \cdot c, s)$  is equal to the one of  $t \cdot e$ , and, subsequently, the error  $\frac{\epsilon(t \cdot c, s)}{t}$  exhibits the same standard deviation as  $e$ . From this point forward, we assume that noise is estimated in the worst-case manner and modify our notation accordingly.

As described in Algorithm 1, we denote  $c_0 := (a, -as + e)$  a ciphertext encrypting zero, and for  $i > i_0$ ,  $c_i = (i \cdot a, -i \cdot as + i \cdot e)$  a fresh ciphertext of zero, which is the homomorphical multiplication of  $c_0$  by  $i$ . With a raw decryption query of  $c_i$ , an adversary can passively get  $i \cdot e$ . If we add a noise  $\epsilon(c_i, s)$  issued from a pseudorandom generator applied to  $(c_i, s)$  to hide this error, the attacker will get an approximation of  $i \cdot e + \epsilon(c_i, s)$ .

We denote  $z_i \leftarrow \text{DecForShare}_{\text{sk}}(c_i)$  the result from the decryption query followed by noise flooding and decoding. From these observations, an attacker can compute  $z'_i = \frac{1}{i} z_i$ , compute the empirical mean by  $\hat{z} = \sum_{i=1}^{\gamma} \frac{1}{\gamma} z'_i$  and encode it to have

**Algorithm 1** Key recovery attack on noise-flooding countermeasure with deterministic noise generation.

---

```

1: Input: Lattice parameters  $(n, \log q)$ , initial scaling factor  $\Delta_0$ 
2: Sample  $(\text{sk}, \text{pk}, \text{ek}) \leftarrow \text{KeyGen}(n, \log q, \Delta_0)$ 
3: Encrypt  $c_0 \leftarrow \text{Enc}_{\text{pk}}(\text{Encode}(\mathbf{0}; \Delta_0))$ 
4:  $\hat{z} \leftarrow \mathbf{0}$ 
5: for  $i_0 \leq i < \gamma + i_0$  do
6:   Homomorphically evaluate  $c_i \leftarrow g_i(c_0, c_0, \dots, c_0)$ 
7:   Decrypt  $z_i \leftarrow \text{DecForShare}_{\text{sk}}(c_i)$ 
8:    $z_i \leftarrow \frac{1}{i} z_i$ 
9:    $\hat{z} \leftarrow \hat{z} + \frac{1}{\gamma} z_i$ 
10: end for
11:  $m' \leftarrow \text{Encode}(\hat{z}; \Delta_0)$ 
12: Compute  $b' \leftarrow m' - b \in R_q$ , where  $(a, b) = c_0 \quad \triangleright b' \simeq a \cdot s + X_\gamma$ 
13: if POST-PROCESSING = 1 then
14:   Return  $s' := \text{PostPro}(a, b')$ , where  $\text{PostPro}(\cdot)$  represents a post-
     processing procedure, which could be enumeration or lattice reduction meth-
     ods
15: else
16:   Return  $s' := a^{-1} \cdot b'$  for the trivial attack when  $X_\gamma = 0$  with high
     probability
17: end if

```

---

$$\text{Encode}(\hat{z}) \simeq \Delta \varphi^{-1} \left( \frac{1}{\gamma} \sum_{i=1}^{\gamma} z'_i \right) \simeq \frac{\Delta}{\gamma} \sum_{i=1}^{\gamma} \varphi^{-1}(z_i) = e + X_\gamma$$

with

$$X_\gamma \simeq \frac{1}{\gamma} \left( \sum_{i=1}^{\gamma} \frac{1}{i} \epsilon(c_i, s) \right)$$

The value  $X_\gamma$  is the sum of independent centered Gaussian noises of the same standard deviation so it behaves as a centered Gaussian noise with a smaller variance. Doubling  $\gamma$  divides the variance by two. To have access to  $\eta$  bits of  $e$  with overwhelming probability, the attacker will only need to have  $\gamma = \Theta(\eta^2)$  decrypted results.

Contrastingly, without applying the countermeasure of deterministic noise generation for fresh ciphertexts, each query introduces a fresh noise with a standard deviation equal to that of  $e$ , equivalent to adding noise of  $\frac{\epsilon(i \cdot c, s)}{i}$ . To recover

|       | $n$      | $\log q$ | precision | plaintext slots |
|-------|----------|----------|-----------|-----------------|
| HElib | $2^{13}$ | 108      | 20        | $2^{12}$        |

**Table 2:** The targeted parameters used in the experiments on HElib.

$\eta$  bits of the secret  $e$ , we still need  $\gamma = \Theta(\eta^2)$  decrypted outputs.

In summary, the new method illustrates that employing deterministic noise for fresh ciphertexts offers no additional protection.

#### 4.1 Application to HElib

We implemented the attack described in Algorithm 1 for HElib. Secret key coefficients in this library are sampled from the ternary set, where the probability to sample 0 is  $1/2$ , and the rest probability mass is equally split between 1 and  $-1$ . Consequently, the expected noise of a fresh ciphertext is given by  $\sigma_1 = 3.2\sqrt{n+1}$ . Since HElib uses worst-case estimation for the noise and due to canonical embedding we have  $\sigma_{1,\text{est}} \simeq 3.2(\sqrt{2n \ln n} + 1)$ . At the end of the algorithm the standard deviation of  $X_\gamma$  is approximately

$$\sigma_{\text{attack}} = \frac{\sigma_{1,\text{est}}}{\sqrt{\gamma}}.$$

For our experiments, we selected parameters according to Table 2 to ensure a desired level of security, specifically, the overall security level is more than 128 bits, as outlined in [Alb+18]. We compared the theoretical value of  $\sigma_{\text{attack}}$  with the actual values obtained during our experiments. Figure 6 presents a graphical representation of these results. It is evident from the figure that our experimental data aligns closely with the theoretical predictions, validating the correctness of the results in Section 4.

We provide an overview of the implications of our attack on the HElib library. Given that HElib has not implemented any updates incorporating the countermeasure through differential privacy [Li+22], it lacks IND-CPA<sup>D</sup> secure. On the other hand, attack strategy detailed in this section requires millions of decryption queries to successfully retrieve the secret key when the worst-case noise bound for each ciphertext is dynamically updated. Consequently, when noise estimation is executed in a worst-case manner, the encryption scheme maintains its KR<sup>D</sup> security. Last, the attack discussed in Section 3.2 could work when targeting implementations with the empirical noise estimation method of invoking the function `ctxt.bumpNoiseBound` to manually adjust the noise level utilized in the decryption function for sharing.

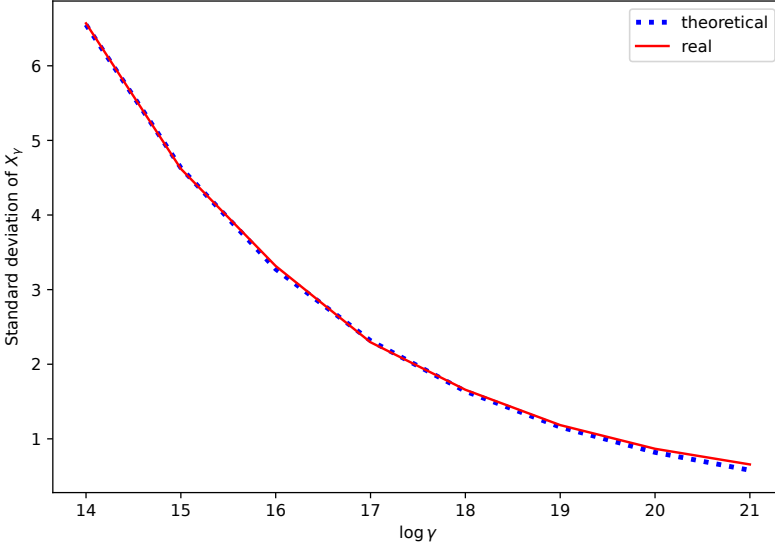


Figure 6: Standard deviation of  $X_\gamma$  according to the prediction and determined experimentally.

## 5 Concluding remarks

In this paper, we have proposed new key-recovery attacks on Approximate Homomorphic Encryption schemes that use noise-flooding countermeasures based on non-worst-case noise estimation, which are implemented to attain IND-CPA<sup>D</sup> security. We illustrated the attack process using the recent implementation of the OpenFHE library and showed how to recover the secret key with a single decryption output. Furthermore, we discussed the security implications of our attack on the HELib implementation, which allows the use of empirical noise estimation. As an additional finding, we revealed that using a pseudorandom generator to create deterministic noise in noise-flooding countermeasures does not enhance security.

**Mitigation and future works.** Our research underscores the importance of employing worst-case noise estimation in noise-flooding countermeasures. By adopting the security bounds proposed in [Li+22], which stem from differential privacy, and implementing a worst-case noise estimation strategy, our attack can be thwarted. Nevertheless, the application of such countermeasures can lead to a considerable reduction in performance, emphasizing the need to investigate more efficient countermeasures in future work.

Although one could formulate defenses against our attacks that employ the function  $g_t(\cdot)$  – for instance, by limiting the acceptable average-case noise bound for ciphertexts, thus disallowing large  $t$  values conducive to key recovery – such

countermeasures often incur significant performance costs. Moreover, they may prove ineffective against adversaries who craft new evaluation functions for emerging passive attacks. It remains an intriguing question as to whether we can develop a more efficient implementation using average-case noise estimation, while still preserving the scheme's IND-CPA<sup>D</sup> security (or perhaps only the KR<sup>D</sup> security).

## Acknowledgement

This work was supported by the Swedish Research Council (grant numbers 2019-04166 and 2021-04602); the Swedish Civil Contingencies Agency (grant number 2020-11632); the Swedish Foundation for Strategic Research (Grants No. RIT17-0005 and RIT17-0032); and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. The computations and simulations were partly enabled by resources provided by LUNARC.

## References

- [Alb+18] M. Albrecht et al. *Homomorphic Encryption Security Standard*. Tech. rep. Toronto, Canada: HomomorphicEncryption.org, Nov. 2018.
- [Bad+22] A. A. Badawi et al. *OpenFHE: Open-Source Fully Homomorphic Encryption Library*. Cryptology ePrint Archive, Paper 2022/915. <https://eprint.iacr.org/2022/915>. 2022.
- [Boe+19] F. Boemer et al. “nGraph-HE: a graph compiler for deep learning on homomorphically encrypted data”. In: *Proceedings of the 16th ACM International Conference on Computing Frontiers*. 2019, pp. 3–13.
- [BV11] Z. Brakerski and V. Vaikuntanathan. “Efficient Fully Homomorphic Encryption from (Standard) LWE”. In: *52nd FOCS*. Ed. by R. Ostrovsky. IEEE Computer Society Press, Oct. 2011, pp. 97–106.
- [Che+17] J. H. Cheon et al. “Homomorphic Encryption for Arithmetic of Approximate Numbers”. In: *ASIACRYPT 2017, Part I*. Ed. by T. Takagi and T. Peyrin. Vol. 10624. LNCS. Springer, Cham, Dec. 2017, pp. 409–437.
- [Che+18] J. H. Cheon et al. “Bootstrapping for Approximate Homomorphic Encryption”. In: *EUROCRYPT 2018, Part I*. Ed. by J. B. Nielsen and V. Rijmen. Vol. 10820. LNCS. Springer, Cham, Apr. 2018, pp. 360–384.
- [Che+19] J. H. Cheon et al. “A Full RNS Variant of Approximate Homomorphic Encryption”. In: *SAC 2018*. Ed. by C. Cid and M. J. Jacobson Jr. Vol. 11349. LNCS. Springer, Cham, Aug. 2019, pp. 347–368.

- [Chi+16] I. Chillotti et al. “Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds”. In: *ASIACRYPT 2016, Part I*. Ed. by J. H. Cheon and T. Takagi. Vol. 10031. LNCS. Springer, Berlin, Heidelberg, Dec. 2016, pp. 3–33.
- [CHK20] J. H. Cheon, S. Hong, and D. Kim. *Remark on the Security of CKKS Scheme in Practice*. Cryptology ePrint Archive, Report 2020/1581. 2020.
- [CKP19] J. H. Cheon, D. Kim, and J. H. Park. “Towards a Practical Cluster Analysis over Encrypted Data”. In: *SAC 2019*. Ed. by K. G. Paterson and D. Stebila. Vol. 11959. LNCS. Springer, Cham, Aug. 2019, pp. 227–249.
- [Cos+22] A. Costache et al. *On the precision loss in approximate homomorphic encryption*. Cryptology ePrint Archive, Report 2022/162. 2022.
- [Dat+19] R. Dathathri et al. “CHET: an optimizing compiler for fully-homomorphic neural-network inferencing”. In: *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*. 2019, pp. 142–156.
- [DM15] L. Ducas and D. Micciancio. “FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second”. In: *EUROCRYPT 2015, Part I*. Ed. by E. Oswald and M. Fischlin. Vol. 9056. LNCS. Springer, Berlin, Heidelberg, Apr. 2015, pp. 617–640.
- [FV12] J. Fan and F. Vercauteren. *Somewhat Practical Fully Homomorphic Encryption*. Cryptology ePrint Archive, Report 2012/144. 2012.
- [Gen09] C. Gentry. “Fully homomorphic encryption using ideal lattices”. In: *41st ACM STOC*. Ed. by M. Mitzenmacher. ACM Press, May 2009, pp. 169–178.
- [GSW13] C. Gentry, A. Sahai, and B. Waters. “Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based”. In: *CRYPTO 2013, Part I*. Ed. by R. Canetti and J. A. Garay. Vol. 8042. LNCS. Springer, Berlin, Heidelberg, Aug. 2013, pp. 75–92.
- [Han+19] K. Han et al. “Logistic regression on homomorphic encrypted data at scale”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 9466–9471.
- [Heaan] Heaan. <https://github.com/snucrypto/HEAAN>.
- [Helib] IBM. *Helib*. <https://github.com/homenc/HElib>.
- [Lattigo] EPFL-LDS. *Lattigo 2.2.0*. <https://github.com/tuneinsight/lattigo>.

- [Lee+18] J. Lee et al. *Instant Privacy-Preserving Biometric Authentication for Hamming Distance*. Cryptology ePrint Archive, Report 2018/1214. 2018.
- [Li+22] B. Li et al. “Securing Approximate Homomorphic Encryption Using Differential Privacy”. In: *CRYPTO 2022, Part I*. Ed. by Y. Dodis and T. Shrimpton. Vol. 13507. LNCS. Springer, Cham, Aug. 2022, pp. 560–589.
- [LM21] B. Li and D. Micciancio. “On the Security of Homomorphic Encryption on Approximate Numbers”. In: *EUROCRYPT 2021, Part I*. Ed. by A. Canteaut and F.-X. Standaert. Vol. 12696. LNCS. Springer, Cham, Oct. 2021, pp. 648–677.
- [LPR10] V. Lyubashevsky, C. Peikert, and O. Regev. “On Ideal Lattices and Learning with Errors over Rings”. In: *EUROCRYPT 2010*. Ed. by H. Gilbert. Vol. 6110. LNCS. Springer, Berlin, Heidelberg, May 2010, pp. 1–23.
- [Palisade] *PALISADE Lattice Cryptography Library (release 1.11.5)*. <https://palisade-crypto.org/>. Sept. 2021.
- [Par+19] S. Park et al. “Security-preserving Support Vector Machine with Fully Homomorphic Encryption.” In: *SafeAI@AAAI* 2301 (2019).
- [RAD+78] R. L. Rivest, L. Adleman, M. L. Dertouzos, et al. “On data banks and privacy homomorphisms”. In: *Foundations of secure computation* 4.11 (1978), pp. 169–180.
- [Reg05] O. Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In: *37th ACM STOC*. Ed. by H. N. Gabow and R. Fagin. ACM Press, May 2005, pp. 84–93.
- [Seal] W. Microsoft Research Redmond. *Microsoft SEAL*. <https://github.com/microsoft/SEAL>.



# LDPC Strikes Again: New Key-Recovery Chosen-Ciphertext Side-Channel Attacks on NTRU-based KEMs

## Abstract

In this work, we introduce novel techniques for adapting the SCA-LDPC framework to conduct efficient plaintext-checking (PC) oracle-based side-channel attacks on NTRU-style Key Encapsulation Mechanisms (KEMs). We specifically address the unique algebraic structure of the NTRU and NTRU Prime schemes, demonstrating how to extract information related to secret variables from side channels and modeled as Low-Density Parity-Check (LDPC) codes. Then, we can recover the complete secret key using an iterative decoding method. For both NTRU and NTRU Prime schemes, our approach achieves significant reductions in the required number of measurements compared to previous state-of-the-art under similar oracle noise assumptions, validated through extensive simulations, and exhibits robustness against decision errors in the constructed oracle.

Furthermore, we present the first documented power analysis attack targeting a first-order masked NTRU implementation. Using standard microcontroller hardware (ARM Cortex-M4), we successfully constructed a PC oracle achieving

a low 0.5% decision error rate; importantly, within a practical *cross-device* setting where training and attack devices differ. Our experimental results demonstrate that approximately 1250 side-channel measurements are sufficient to recover the secret key in this challenging scenario, closely aligning with our simulation findings. These results underscore the vulnerability of first-order masking for NTRU against this attack vector and motivate the thorough evaluation of higher-order countermeasures.

## 1 Introduction

Advancements in quantum computing pose a significant threat to traditional cryptographic systems, as quantum algorithms could potentially break widely used encryption methods. This challenge has propelled post-quantum cryptography (PQC) to the forefront of research, aiming to develop schemes resistant to quantum attacks. Among PQC candidates, lattice-based cryptography stands out due to its strong security foundations and practical performance. Notably, schemes based on the Learning With Errors (LWE) problem [Reg09] and NTRU-like schemes are prominent contenders, with the latter being renowned for their efficiency and suitability for practical applications.

Introduced in 1996 [Hof+99], the NTRU cryptosystem was a pioneering ring-based cryptographic system valued for its concise design and minimal computational requirements. NTRU-style proposals have seen deployment in secure communications (e.g., wolfSSL [wolfSSL], OpenSSH [OpenSSH]) and have undergone significant security enhancements, including proofs in the standard model [SS11] and variants like NTRU Prime [Ber+16]. Although the specific NTRU proposal [Che+20] was not selected for NIST standardization, its underlying principles remain influential. The NTRU problem has found widespread use in various cryptographic constructions, including encryption schemes, digital signatures, and other related applications [Hof+03], [Duc+13], [DLP14], [Fou+18]. Among these, the Falcon scheme, based on NTRU lattice problems, leverages algebraic properties defined on the structured lattice  $\mathbb{Z}_q[x]/(x^n + 1)$  to achieve highly compact key and signature sizes. Falcon has been officially standardized by NIST and is deployed in practical applications, such as Google Cloud's ALTS protocol [Goo22], where it combines NTRU-HRSS for a hybrid key exchange mechanism [Hül+17]. Furthermore, related schemes like NTRU+ [NTRU+] continue to be considered in initiatives such as the Korean PQC competition (KpQC) [KPQC]. Studying the security of NTRU provides valuable insights into potential vulnerabilities within this important class of ring-based PQC schemes.

Despite theoretical security, practical implementations face threats from side-channel attacks (SCAs). SCAs were first introduced by Kocher [Koc96b], with the core idea of exploiting physical leakages such as power consumption and timing information to extract sensitive data. For PQC, side-channel attacks can be broadly categorized into two types: the first relies on highly precise leakages in

the implementation, often requiring only a single or a few traces to recover the key or message; the second category is based on more common leaks in the implementation, often requiring the construction of a suitable side-channel oracle and extracting information related to the secret key by collecting a large number of traces.

In key-recovery chosen-ciphertext side-channel attacks (KR-CCA-SCAs), the adversary submits carefully crafted ciphertexts to the decryption or decapsulation algorithm and combines the responses with side-channel measurements in order to recover the secret key. Existing KR-CCA-SCAs can be roughly divided into two categories: one targets the re-encryption and ciphertext comparison steps of the FO transform, exploiting the sensitive information potentially leaked in these processes [GJN20b], [Guo+22b], [Rav+20b]; the other directly leverages the use of the secret key in the CPA-secure decryption procedure to obtain side-channel information [GJJ22b], [Ham+21b], [Ngo+21b], [Sch+22c], [GLG22c], [Guo+23b].

Modern NTRU variants achieve IND-CCA security mathematically [Che+20], mitigating traditional chosen-ciphertext attacks (CCAs) [JJ00; How+03]. However, side-channel assisted CCAs present a potent alternative. These attacks leverage physical leakage to construct plaintext-checking (PC) oracles, which reveal partial information about decryption outcomes, effectively bypassing formal CCA security transformations. For example, early timing and power attacks on NTRU were explored [SW07; Lee+10]. Ravi et al. [Rav+22a] demonstrated the feasibility of such attacks on NTRU-based Key Encapsulation Mechanisms (KEMs) by constructing specific ciphertexts to induce collisions related to the secret key. Concurrently, Ueno et al. [Uen+22a] presented generic power/EM attacks applicable to various KEMs using CCA transformations, evaluating their method on a masked pseudorandom function rather than a full NTRU implementation. While foundational, these works leave open the question of how to optimally leverage PC oracle information, including noisy oracle outputs from masked implementations, for highly efficient key recovery.

The SCA-LDPC framework [Guo+23b] offers a promising direction, demonstrating remarkable efficiency against masked LWE-based schemes like Kyber [Hei+22b] by modeling SCA leakage recovery as an LDPC decoding problem. This framework naturally incorporates error correction, making it potentially robust to oracle inaccuracies inherent in real-world measurements. However, adapting SCA-LDPC from LWE to NTRU-style schemes is non-trivial due to fundamental differences in their algebraic structures and decryption processes.

This motivates our central research question: *Can the error-correcting power of the SCA-LDPC framework be effectively adapted to the specific structure of NTRU-style schemes, enabling efficient key recovery even against masked implementations and in challenging, practical scenarios like cross-device attacks?* Answering this requires developing novel techniques to map the leakage from the algorithm's decryption process into an LDPC coding problem.

Table 1: Comparison with Previous Work

| Target Scheme        | Oracle Accuracy( $\rho$ ) | Work                  | Required Traces     |
|----------------------|---------------------------|-----------------------|---------------------|
| ntruhs2048509        | 1.00                      | Ravi et al. [Rav+22a] | 1791                |
| <b>ntruhs2048509</b> | <b>1.00</b>               | <b>This Work</b>      | <b>1243 (-31%)</b>  |
| ntruhs2048509        | 0.96                      | Ueno et al. [Uen+22a] | 5090                |
| <b>ntruhs2048509</b> | <b>0.96</b>               | <b>This Work</b>      | <b>2270 (-55%)</b>  |
| sntrup653            | 0.96                      | Ueno et al. [Uen+22a] | 13560               |
| <b>sntrup653</b>     | <b>0.96</b>               | <b>This Work</b>      | <b>5616 (-59%)</b>  |
| sntrup761            | 1.00                      | Ravi et al. [Rav+22a] | 3659                |
| <b>sntrup761</b>     | <b>1.00</b>               | <b>This Work</b>      | <b>2380 (-35%)</b>  |
| sntrup1277           | 0.96                      | Ueno et al. [Uen+22a] | 25875               |
| <b>sntrup1277</b>    | <b>0.96</b>               | <b>This Work</b>      | <b>10744 (-58%)</b> |

## 1.1 Contributions

The main contributions of this work are as follows:

- We introduce novel techniques specifically designed to adapt the SCA-LDPC framework [Guo+23b] for key recovery attacks against NTRU-style KEMs (focusing on [Che+20]). We detail how to leverage the algebraic structures of NTRU and NTRU Prime during decryption to construct tailored low-density parity checks to capture PC oracle outputs for LDPC decoding. This represents a non-trivial instantiation of the general framework for this class of schemes.
- Our research demonstrates significant performance improvements over existing methods for side-channel attacks targeting NTRU. The attack exhibits robustness against decision errors within the constructed oracle. For instance, simulations targeting a masked software scenario show that our attack requires only approximately 45% of the measurements compared to previous work [Uen+22a].
- In the context of side-channel attacks targeting NTRU Prime, our approach demonstrates a substantial performance improvement. Specifically, in the best-case scenario, our method requires only approximately 41% of the measurements compared to previous work [Uen+22a] to recover the complete secret key.
- We present the **first** documented power analysis attack on a first-order masked NTRU implementation [Cor+23b]. Using standard microcontroller hard-

ware (ARM Cortex-M4), we conducted a practical validation in a realistic and challenging *cross-device* setting, where the device used for training the leakage model differs from the target device under attack. Our methodology successfully built a PC oracle with a low decision error rate (0.5%) in this setting. This demonstration constitutes, to the best of our knowledge, the first practical power or electromagnetic attack reported against *any* masked NTRU-style KEMs in the cross-device setting.

- Our cross-device experiments show that approximately 1250 power traces suffice to recover the secret key of the first-order masked NTRU implementation, closely matching simulation results. This practical validation highlights the vulnerability of current first-order masking for NTRU to this attack vector and underscores the importance of evaluating the effectiveness of higher-order countermeasures in realistic attack scenarios.

The detailed performance comparison with previous state-of-the-art methods is summarized in Table 1. The code is available at <https://github.com/d-nabokov/NTRUmeetsLDPC.git>

This paper is an extended and revised version of [NTG25]. The primary contribution of this version is the introduction of a new key-recovery attack on NTRU Prime, for which we derive the special techniques needed to adapt our framework. Furthermore, we provide a more complete exposition of the overall approach and present a more extensive experimental validation.

## 1.2 Organization

This paper is organized as follows: Section 2 provides the necessary background information; Section 3 details our novel attack on NTRU using LDPC codes, and presents the simulation results of the approach; Section 4 extends our LDPC-based side-channel attack method to NTRU Prime, and provides simulation results; Section 5 introduces the first key-recovery attack on a masked NTRU implementation, validated through practical experiments; finally, Section 7 concludes the paper.

## 2 Preliminary

In this chapter, we introduce some necessary background knowledge. We first present some symbols and LDPC, followed by a description of the NTRU and NTRU Prime algorithms.

Throughout this paper, we use  $\mathbb{Z}[x]$  to denote the ring of polynomials with coefficients in  $\mathbb{Z}$ , and  $\mathbb{Z}_q[x]$  denotes the ring with coefficients modulo  $q$ . We define  $\Phi_1 = (x - 1)$ ,  $\Phi_n = (x^{n-1} + x^{n-2} + \dots + 1)$  and  $\Psi_n = x^n - x - 1$ , where  $n$  is a prime number.  $B^*$  denotes a byte array of arbitrary length. A polynomial is referred to as ternary if its coefficients belong to  $\{-1, 0, 1\}$ . We denote by  $\mathcal{T}$  the

sets of non-zero ternary polynomials of degree at most  $n - 2$ , and  $\mathcal{T}_+$  is the subset of  $\mathcal{T}$  consisting of polynomials that satisfy the non-negative correlation property. Additionally,  $\mathcal{T}(d)$  is a subset of  $\mathcal{T}$ , consisting of polynomials that have exactly  $d/2$  coefficients equal to  $+1$  and  $d/2$  coefficients equal to  $-1$ . Polynomials are represented by lowercase bold letters (e.g.,  $\mathbf{f}$ ) with  $\mathbf{f}[i]$  denoting a coefficient of  $\mathbf{f}$  corresponding to  $x^i$ . Similarly, vectors are denoted by lowercase bold letters. Finally,  $\text{Ber}_\rho$  denotes a Bernoulli distribution that outputs 1 with probability  $\rho$ , and 0 otherwise.

## 2.1 Error-Correcting Codes

**Low-Density Parity-Check (LDPC) codes** LDPC codes are a class of linear block codes with sparse parity-check matrices, proposed by Gallager in 1962 [Gal62]. LDPC codes exhibit sparsity in their parity-check matrix, ensuring that both decoding complexity and minimum distance increase linearly with the code length. This enables efficient encoding and decoding processes, such as iterative decoding on Tanner graphs using belief propagation.

**Concatenated codes** Concatenated codes are an effective method for constructing long codes from short ones, first proposed by Forney in 1965 [For65]. They consist of two parts: an inner code  $\mathcal{C}_{in} : \mathcal{A}^k \rightarrow \mathcal{A}^n$  and an outer code  $\mathcal{C}_{out} : \mathcal{B}^K \rightarrow \mathcal{B}^N$ , where  $|\mathcal{B}| = |\mathcal{A}|^N$ . The concatenated code is represented as  $\mathcal{C}_{con} : \mathcal{A}^{kK} \rightarrow \mathcal{A}^{nN}$  and decoded sequentially through the inner and outer code decoders. Maximum-likelihood decoding can be employed for inner codes, while efficient polynomial-time decoding (e.g., using LDPC codes) is possible for outer codes.

## 2.2 NTRU

The core of NTRU is a deterministic public key encryption scheme, and we describe the process of the core operations of NTRU in Algorithms 1 to 3. From it, the IND-CCA-secure KEM is constructed, which is shown in Algorithms 4 to 6. We recall in Table 2 the definitions of the sets of integer polynomials  $\mathcal{L}_f$ ,  $\mathcal{L}_g$ ,  $\mathcal{L}_r$ ,  $\mathcal{L}_m$ , and the embedding  $\text{Lift}$  for the two versions of NTRU, namely NTRU-HPS and NTRU-HRSS. To simplify the presentation, the following algorithms are based on the NTRU-HPS version, where  $\text{Lift}(m) = m$ . Then, we review the relevant parameters  $n$  and modulus  $q$  for the four versions of NTRU in Table 3.

## 2.3 NTRU Prime

The NTRU Prime is a suite of two IND-CCA secure KEMs, namely Streamlined NTRU Prime and NTRU LPrime, and operates in the field  $R_q := \mathbb{Z}_q[x]/(\Psi_n)$ , which is not cyclotomic. We present the core operations of NTRU Prime in Algorithms 7 to 9, and based on these operations, the construction of the IND-CCA-

| Scheme | $\mathcal{L}_f   \mathcal{L}_g$                 | $\mathcal{L}_r   \mathcal{L}_m$                 | Lift                                                                  |
|--------|-------------------------------------------------|-------------------------------------------------|-----------------------------------------------------------------------|
| HPS    | $\mathcal{T} \mid \mathcal{T}(\frac{q}{8} - 2)$ | $\mathcal{T} \mid \mathcal{T}(\frac{q}{8} - 2)$ | $\mathbf{m} \mapsto \mathbf{m}$                                       |
| HRSS   | $\mathcal{T}_+ \mid \Phi_1 \cdot \mathcal{T}_+$ | $\mathcal{T} \mid \mathcal{T}$                  | $\mathbf{m} \mapsto \Phi_1(\mathbf{m}/\Phi_1 \bmod^{\pm}(3, \Phi_n))$ |

**Table 2:** Definitions of Integer Polynomial Sets and Embedding for NTRU-HPS and NTRU-HRSS

|     | ntruhs2048509 | ntruhs2048677 | ntruhs4096821 | ntruhrss701 |
|-----|---------------|---------------|---------------|-------------|
| $n$ | 509           | 677           | 821           | 701         |
| $q$ | 2048          | 2048          | 4096          | 8192        |

**Table 3:** Parameters  $n$  and  $q$  for versions of NTRU

secure KEMs is illustrated in Algorithms 10 to 12. In these algorithms, the function `GenSmall()` samples small polynomials in  $\mathbb{Z}_3[x]/(\Psi_n)$ , while `GenShort()` samples polynomials of weight  $w$  in  $\mathbb{Z}_3[x]/(\Psi_n)$ . The function `Round()` rounds each coefficient of a given polynomial to the nearest multiple of 3. NTRU Prime is particularly challenging since the scheme uses rounded ciphertexts as well as conditional checks on the decrypted message. We review in Table 4 the relevant parameters  $n$ , modulus  $q$ , and the secret key weight parameter  $w$  for the three versions of NTRU Prime.

|     | sntrup653 | sntrup761 | sntrup1277 |
|-----|-----------|-----------|------------|
| $n$ | 653       | 761       | 1277       |
| $q$ | 4621      | 4591      | 7879       |
| $w$ | 288       | 286       | 492        |

**Table 4:** Parameters  $n$ ,  $q$  and  $w$  for versions of NTRU Prime.

## 2.4 SCA on NTRU based schemes

In 2021, Ravi et al. [Rav+22a] proposed a side-channel-assisted chosen-ciphertext attack on NTRU-based KEMs. For all parameter sets of NTRU and NTRU Prime, their proposed attacks can recover the complete secret key through a few thousand chosen ciphertext queries. The attacks they proposed focus on the decapsulation process, where in `KEM.Decaps`, the shared secret is obtained by hashing the ciphertext  $\mathbf{c}$ , the private key, and the public key. Their basic idea is to construct special ciphertexts such that the intermediate variables during the decapsulation process are closely related to the secret variables, and then recover the complete secret key using the information obtained from side channels.

---

**Algorithm 1** NTRU KeyGen(*seed*)

---

```

1:  $\mathbf{f} \leftarrow \mathcal{L}_f, \mathbf{g} \leftarrow \mathcal{L}_g$ 
2:  $\mathbf{f}_q \leftarrow (1/\mathbf{f}) \bmod (q, \Phi_n)$ 
3:  $\mathbf{h} \leftarrow (3 \cdot \mathbf{g} \cdot \mathbf{f}_q) \bmod (q, \Phi_1 \Phi_n)$ 
4:  $\mathbf{h}_q \leftarrow (1/\mathbf{h}) \bmod (q, \Phi_n)$ 
5:  $\mathbf{f}_p \leftarrow (1/\mathbf{f}) \bmod (3, \Phi_n)$ 
6: return  $((\mathbf{f}, \mathbf{f}_p, \mathbf{h}_q), \mathbf{h})$ 

```

---

**Algorithm 2** NTRU Encrypt( $\mathbf{h}, (\mathbf{r}, \mathbf{m})$ )

---

```

1:  $\mathbf{c} \leftarrow \mathbf{r} \cdot \mathbf{h} + \mathbf{m} \bmod (q, \Phi_1 \Phi_n)$ 
2: return  $\mathbf{c}$ 

```

---

**Algorithm 3** NTRU Decrypt( $((\mathbf{f}, \mathbf{f}_p, \mathbf{h}_q), \mathbf{c})$ )

---

```

1: if  $\mathbf{c} \neq 0 \bmod (q, \Phi_1)$  then
2:   return  $(0, 0, 1)$ 
3: end if
4:  $\mathbf{a} \leftarrow (\mathbf{f} \cdot \mathbf{c}) \bmod (q, \Phi_1 \Phi_n)$ 
5:  $\mathbf{e} \leftarrow \mathbf{a} \bmod (3, \Phi_n)$ 
6:  $\mathbf{m} \leftarrow (\mathbf{e} \cdot \mathbf{f}_p) \bmod (3, \Phi_n)$ 
7:  $\mathbf{r} \leftarrow ((\mathbf{c} - \mathbf{m}) \cdot \mathbf{h}_q) \bmod (q, \Phi_n)$ 
8: if  $(\mathbf{r}, \mathbf{m}) \in (\mathcal{L}_r, \mathcal{L}_m)$  then
9:   return  $(\mathbf{r}, \mathbf{m}, 0)$ 
10: else
11:   return  $(0, 0, 1)$ 
12: end if

```

---

We start with the intuition behind the attack. Let ciphertext  $\mathbf{c} = t + t \cdot \mathbf{h}$ , where  $t \in \mathbb{Z}^+$ . By examining the decryption procedure of NTRU, we have

$$\mathbf{a} = \mathbf{f} \cdot \mathbf{c} = \mathbf{f} \cdot t + \mathbf{f} \cdot t \cdot (3 \cdot \mathbf{g} \cdot \mathbf{f}_q) = t \cdot \mathbf{f} + 3t \cdot \mathbf{g},$$

where the coefficients of  $\mathbf{f}$  and  $\mathbf{g}$  belong to the set  $\{-1, 0, 1\}$ . When the  $i$ -th coefficients of both  $\mathbf{f}$  and  $\mathbf{g}$  are simultaneously 1 (or both  $-1$ ),  $|\mathbf{a}[i]|$  attains its maximum value. The event where the corresponding coefficients of two or more polynomials reach their maximum absolute value is called a collision. By choosing a suitable  $t$  such that  $3 \mid t$  and ensuring it satisfies

$$\mathbf{a}[j] \begin{cases} > q/2, & \text{if } j = i, \\ < q/2, & \text{if } j \neq i, \end{cases}$$

**Algorithm 4** NTRU KeyGen'(seed)

---

```

1:  $((\mathbf{f}, \mathbf{f}_p, \mathbf{h}_q), \mathbf{h}) \leftarrow \text{NTRU KeyGen}(seed)$ 
2:  $s \leftarrow \{0, 1\}^{256}$ 
3: return  $((\mathbf{f}, \mathbf{f}_p, \mathbf{h}_q, s), \mathbf{h})$ 

```

---

**Algorithm 5** NTRU Encapsulate(h)

---

```

1:  $coins \leftarrow \{0, 1\}^{256}$ 
2:  $(\mathbf{r}, \mathbf{m}) \leftarrow \text{Sampler}_{\mathbf{m}}(coins)$ 
3:  $\mathbf{c} \leftarrow \text{Encrypt}(\mathbf{h}, (\mathbf{r}, \mathbf{m}))$ 
4:  $k \leftarrow H_1(\mathbf{r}, \mathbf{m})$ 
5: return  $(\mathbf{c}, k)$ 

```

---

**Algorithm 6** NTRU Decapsulate((f, f<sub>p</sub>, h<sub>q</sub>, s), c)

---

```

1:  $(\mathbf{r}, \mathbf{m}, fail) \leftarrow \text{Decrypt}((\mathbf{f}, \mathbf{f}_p, \mathbf{h}_q), \mathbf{c})$ 
2:  $k_1 \leftarrow H_1(\mathbf{r}, \mathbf{m})$ 
3:  $k_2 \leftarrow H_2(s, \mathbf{c})$ 
4: if  $fail = 0$  then
5:   return  $k_1$ 
6: else
7:   return  $k_2$ 
8: end if

```

---

when  $\mathbf{a}$  is reduced modulo  $q$ , all coefficients except for  $\mathbf{a}[i]$  retain their true values and remain multiples of 3. More explicitly,

$$\mathbf{a} \bmod q = \mathbf{a} - q \cdot x^i,$$

and the intermediate variable

$$\mathbf{e} = \mathbf{a} \bmod 3 = (-q \bmod 3) \cdot x^i.$$

This approach ensures that, in the absence of a collision,  $\mathbf{a}[i] < q/2$ , and only when a collision occurs does  $\mathbf{a}[i]$  exceed the  $q/2$  threshold. Consequently, if  $\mathbf{e}[i] \neq 0$ , it indicates a collision at  $i$ . When there are no collisions,  $\mathbf{m}$  in line 6 of NTRU Decrypt is set to 0. On the contrary, if there is at least one collision, coefficients of  $\mathbf{m}$  can be seen as taking uniform values from  $\{-1, 0, 1\}$ .

## 2.5 PC oracle

A plaintext-checking oracle is a mechanism designed to determine whether a decrypted message matches a specific fixed value, typically using information ob-

---

**Algorithm 7** NTRU Prime KeyGen(*seed*)

---

```

1: while  $\mathbf{g}$  is not invertible in  $(3, \Psi_n)$  do
2:    $\text{coins} \leftarrow \mathcal{U}(B^*)$ ,  $\mathbf{g} \leftarrow \text{GenSmall}(\text{coins})$ 
3: end while
4:  $\mathbf{g}_3 \leftarrow (1/\mathbf{g}) \bmod (3, \Psi_n)$ 
5:  $\text{coins} \leftarrow \mathcal{U}(B^*)$ 
6:  $\mathbf{f} \leftarrow \text{GenShort}(\text{coins})$ 
7:  $\mathbf{h} \leftarrow \mathbf{g}/(3 \cdot \mathbf{f}) \bmod (q, \Psi_n)$ 
8: return  $((\mathbf{g}_3, \mathbf{f}), \mathbf{h})$ 

```

---

**Algorithm 8** NTRU Prime Encrypt( $\mathbf{h}, \mathbf{r}$ )

---

```

1:  $\mathbf{d} \leftarrow \mathbf{h} \cdot \mathbf{r} \bmod (q, \Psi_n)$ 
2:  $\mathbf{c} \leftarrow \text{Round}(\mathbf{d}) \bmod (q, \Psi_n)$ 
3:  $ct \leftarrow \text{Encode } \mathbf{c}$ 
4: return  $ct$ 

```

---

**Algorithm 9** NTRU Prime Decrypt( $((\mathbf{g}_3, \mathbf{f}), ct)$ )

---

```

1:  $\mathbf{c} \leftarrow \text{Decode}(ct) \bmod (q, \Psi_n)$ 
2:  $\mathbf{a} \leftarrow (3 \cdot \mathbf{f} \cdot \mathbf{c}) \bmod (q, \Psi_n)$ 
3:  $\mathbf{e} \leftarrow \mathbf{a} \bmod (3, \Psi_n)$ 
4:  $\mathbf{b}' \leftarrow (\mathbf{e} \cdot \mathbf{g}_3) \bmod (3, \Psi_n)$ 
5: if  $\text{Weight}(\mathbf{b}') = w$  then
6:   return  $\mathbf{r}' = \mathbf{b}'$ 
7: else
8:   return  $\mathbf{r}' = (1, 1, \dots, 1, 0, 0, \dots, 0)$ 
9: end if

```

---

tained from side-channel analysis [DAn+19c], [Rav+20b]. Side-channel information such as power consumption, electromagnetic emissions, or execution time can be exploited to infer sensitive data about the internal state of an algorithm. In our context, the PC oracle is utilized to check if the decrypted message  $\mathbf{m}$  is equal to zero. This potentially gives us one bit of information that we can connect to the secret key.

## 2.6 The threat model

This subsection outlines the threat model considered in our work. We focus on side-channel-assisted CCAs against NTRU-based KEMs. Our adversary, Malory, aims to recover the long-term secret key of a communicating party, Alice, during the key establishment phase. Malory crafts ciphertexts and sends them to Alice,

**Algorithm 10** NTRU Prime KeyGen'(seed)

- 
- 1:  $((\mathbf{f}, \mathbf{f}_p, \mathbf{h}_q), \mathbf{h}) \leftarrow \text{KeyGen}(seed)$
  - 2:  $s \leftarrow \{0, 1\}^{256}$
  - 3: **return**  $((\mathbf{f}, \mathbf{f}_p, \mathbf{h}_q, s), \mathbf{h})$
- 

**Algorithm 11** NTRU Prime Encapsulate(h)

- 
- 1:  $coins \leftarrow \mathcal{U}(B^*)$
  - 2:  $\mathbf{r} \leftarrow \text{GenShort}(coins)$
  - 3:  $\mathbf{c} \leftarrow \text{NTRU Prime Encrypt}(\mathbf{h}, \mathbf{r})$
  - 4:  $\mathbf{d} \leftarrow H_1(\mathbf{r}, \mathbf{h})$
  - 5:  $ct \leftarrow (\mathbf{c}, \mathbf{d}), K \leftarrow \mathcal{G}(1, \mathbf{r}, ct)$
  - 6: **return**  $(ct, K)$
- 

**Algorithm 12** NTRU Prime Decapsulate(((g<sub>3</sub>, f), h, ct))

- 
- 1:  $ct \leftarrow (\mathbf{c}, \mathbf{d})$
  - 2:  $\mathbf{r}' \leftarrow \text{NTRU Prime Decrypt}(ct, (\mathbf{g}_3, \mathbf{f}))$
  - 3:  $\mathbf{d}' \leftarrow H_1(\mathbf{r}', \mathbf{h})$
  - 4:  $\mathbf{c}' \leftarrow \text{NTRU Prime Encrypt}(\mathbf{h}, \mathbf{r}')$
  - 5:  $ct' \leftarrow (\mathbf{c}', \mathbf{d}')$
  - 6: **if**  $ct' = ct$  **then**
  - 7:     **return**  $K \leftarrow \mathcal{G}(1, \mathbf{r}', ct')$
  - 8: **else**
  - 9:     **return**  $K \leftarrow \mathcal{G}(1, coins', ct')$       $\triangleright coins'$  from  $B^{32}$  is a random secret
  - 10: **end if**
- 

exploiting potential side-channel leakages that may occur during the decapsulation process to mount CCAs. While IND-CCA secure NTRU-based KEMs are theoretically resistant to such attacks, our proposed attack leverages these side-channel vulnerabilities to recover Alice's secret key after a limited number of attempts.

Specifically, we propose Key-Recovery Chosen-Ciphertext Side-Channel Attacks with LDPC (KR-CCA-SCA-LDPC) targeting the NTRU-based KEMs. The main idea is to construct suitable ciphertexts that ensure a close relationship between intermediate variables and the secret key. When these ciphertexts are decapsulated using the target device, attackers can obtain information about the secret variables through side channels and model them as LDPC codes. The full secret key is then recovered through iterative decoding methods.

### 3 KR-CCA-SCA-LDPC on NTRU

In this section, we propose a novel side-channel attack method targeting NTRU KEMs. The basic idea is to construct suitable ciphertexts, and when the target device decapsulates them, obtain side-channel information related to secret variables through the PC oracle. This information is modeled as LDPC code. Finally, the full secret key is recovered using iterative decoding methods. We focus on the parameter set `ntruhs2048509`, though similar attacks can be applied to other parameter sets.

#### 3.1 The Main Attack

**Construct the base ciphertext** Our aim is to select appropriate parameters such that there is only a single collision at index  $i$ , meaning  $\mathbf{e}[i] \neq 0$ , while all other coefficients of  $\mathbf{e}$  are zero. To limit the number of collisions between  $\mathbf{f}$  and  $\mathbf{g}$ , following the approach in [Rav+22a], we choose the ciphertext as

$$\begin{aligned}\mathbf{c} &= t_1(x^{i_1} - x^{i_2} + \dots - x^{i_o}) + t_2(x^{j_1} + x^{j_2} + \dots + x^{j_p}) \cdot \mathbf{h} \\ &= t_1 \cdot \mathbf{d}_1 + t_2 \cdot \mathbf{d}_2 \cdot \mathbf{h},\end{aligned}$$

where  $\mathbf{d}_1$  and  $\mathbf{d}_2$  are polynomials with  $o$  and  $p$  coefficients of  $\pm 1$ , respectively, and  $o$  is even. Therefore, the corresponding  $\mathbf{a} = \mathbf{f} \cdot \mathbf{c}$  can be expressed as

$$\begin{aligned}\mathbf{a} &= \mathbf{f} \cdot (t_1 \mathbf{d}_1 + t_2 \mathbf{d}_2 \cdot \mathbf{h}) = t_1 \mathbf{f} \cdot \mathbf{d}_1 + t_2 \mathbf{d}_2 \cdot \mathbf{f} \cdot (3\mathbf{g} \cdot \mathbf{f}_q) \\ &= t_1 \mathbf{f} \cdot \mathbf{d}_1 + 3t_2 \mathbf{d}_2 \cdot \mathbf{g}.\end{aligned}$$

We assume a collision occurs at index  $i$ . Since

$$\begin{aligned}\mathbf{a}[i] &= t_1(\mathbf{f} \cdot x^{i_1})[i] - t_1(\mathbf{f} \cdot x^{i_2})[i] + \dots - t_1(\mathbf{f} \cdot x^{i_o})[i] \\ &\quad + 3t_2(\mathbf{g} \cdot x^{j_1})[i] + \dots + 3t_2(\mathbf{g} \cdot x^{j_p})[i],\end{aligned}$$

it follows that  $\mathbf{a}[i]$  reaches its maximum value when the  $i$ -th coefficients of  $(-1)^{\alpha-1} \mathbf{f} \cdot x^{i_\alpha}$  and  $\mathbf{g} \cdot x^{j_\beta}$  are all equal to 1, resulting in  $\mathbf{a}[i] = t_1 \cdot o + 3t_2 \cdot p$ , where  $1 \leq \alpha \leq o$  and  $1 \leq \beta \leq p$ . Note that here  $|\mathbf{a}[i]|$  could reach the maximum value when the mentioned coefficients are equal to  $-1$ , but we focus on the positive case for simplicity. Parameters  $t_1$  and  $t_2$  should satisfy the following three conditions:

$$3 \mid t_1, 3 \mid t_2, t_1 \cdot r + 3t_2 \cdot s \begin{cases} > q/2, & \text{if } r = o, s = p, \\ < q/2, & \text{otherwise,} \end{cases}$$

where  $0 \leq r \leq o$  and  $0 \leq s \leq p$ . By true single collision, we mean an event where only one coefficient of  $\mathbf{e}$  is non-zero, meaning that there is a single  $i$  for which  $|\mathbf{a}[i]| > q/2$ , whereas multiple collisions occur when more than one coefficient of  $\mathbf{e}$  is non-zero.

| Scheme        | $(o, p)$ | $(t_1, t_2)$ | Single true | Multiple |
|---------------|----------|--------------|-------------|----------|
| ntruhs2048509 | (4, 3)   | (147, 51)    | 0.1558      | 0.0148   |

**Table 5:** Concrete parameters for constructing base ciphertext in NTRU Schemes with collision probabilities

We choose parameters  $(o, p)$  such that a single collision occurs with high probability. Table 5 lists the concrete values of  $(o, p)$  and  $(t_1, t_2)$  together with the probability of obtaining true single and multiple collisions for the chosen parameters. These results were obtained through 1000 attack simulation trials using 1000 different secret keys. Finding the base ciphertext may require several attempts; for each of them, we generate new  $\mathbf{d}_1$  and  $\mathbf{d}_2$ .

**Construct the attack ciphertext** After obtaining the base ciphertext, we use the polynomials  $\mathbf{d}_1$  and  $\mathbf{d}_2$  to construct the attack ciphertext  $\mathbf{c}_{\text{att}}$  as follows:

$$\mathbf{c}_{\text{att}} = \ell_1 \cdot \mathbf{d}_1 + \ell_2 \cdot \mathbf{d}_2 \cdot \mathbf{h} + \ell_3 \cdot (x^{u_1} - x^{u_2} + \dots - x^{u_w}),$$

where  $\ell_1, \ell_2, \ell_3 \in \mathbb{Z}^+$  and  $0 \leq u_i < n$  for even  $w$ . Thus, for  $\mathbf{c}_{\text{att}}$ , we have

$$\begin{aligned} \mathbf{a} &= \mathbf{f} \cdot \mathbf{c}_{\text{att}} = \ell_1 \cdot \mathbf{d}_1 \cdot \mathbf{f} + 3\ell_2 \cdot \mathbf{d}_2 \cdot \mathbf{g} \\ &\quad + \ell_3 \cdot \mathbf{f} \cdot (x^{u_1} - x^{u_2} + \dots - x^{u_w}). \end{aligned}$$

Suppose there is a positive collision at index  $i$ , i.e.,  $\mathbf{d}_1 \cdot \mathbf{f}$  and  $\mathbf{d}_2 \cdot \mathbf{g}$  are  $o$  and  $p$ , resp. Let us define

$$\begin{aligned} \beta_i(u_1, \dots, u_w) &= (\mathbf{f} \cdot (x^{u_1} - x^{u_2} + \dots - x^{u_w}))[i] \\ &= \sum_{j=1}^w (-1)^{j-1} f[i - u_j \bmod n]. \end{aligned}$$

Then the coefficients of  $\mathbf{a}$  can be expressed as

$$\mathbf{a}[j] = \begin{cases} \ell_1 o + 3\ell_2 p + \ell_3 \beta_i(u_1, \dots, u_w), & \text{if } j = i \\ \ell_1 r + 3\ell_2 s + \ell_3 \beta_j(u_1, \dots, u_w), & \text{if } \begin{cases} j \neq i \text{ and} \\ (r, s) \neq (o, p) \end{cases} \end{cases}$$

It can be observed that  $\beta_i(u_1, \dots, u_w)$  belongs to  $\{-w, \dots, w\}$ . It is possible then to choose the parameters  $(\ell_1, \ell_2, \ell_3)$  such that  $\mathbf{a}[j] < q/2$  for  $j \neq i$  regardless of the value of  $\beta_j(u_1, \dots, u_w) \in \{-w, \dots, w\}$ , but  $\mathbf{a}[i] > q/2$  when  $\beta_i(u_1, \dots, u_w) \geq Z$  for some  $Z \in \{-w, \dots, w\}$  that depends on  $(\ell_1, \ell_2, \ell_3)$ . Thus, employing the PC oracle, we can extract information about  $\beta_i(u_1, \dots, u_w)$  which is directly connected to the secret key  $\mathbf{f}$ . In other words, by choosing ap-

appropriate  $(\ell_1, \ell_2, \ell_3)$ , we can query whether  $\beta_i(u_1, \dots, u_w) \geq Z$  for the chosen value of  $Z$ . This binary information can be viewed as a *distinguisher* for the secret variable  $\beta_i(u_1, \dots, u_w)$ . Thus, we refer to a distinguisher  $ds_Z$  as an oracle output corresponding to  $\beta_i(u_1, \dots, u_w) \geq Z$ . Note that if parameters  $(\ell_1, \ell_2, \ell_3)$  lead to the distinguisher  $ds_Z$ , taking  $-\ell_3$  instead of  $\ell_3$  leads to the query  $\beta_i(u_1, \dots, u_w) \leq -Z$  that we define as an distinguisher  $\bar{ds}_Z$ .

In the context of LDPC code,  $\beta_i(u_1, \dots, u_w)$  describes a row of the parity check matrix. Specifically, values  $u_1, \dots, u_w$  define which coefficients of  $\mathbf{f}$  (possibly multiplied by  $-1$ ) are participating in the parity check equation, where the result of the equation is exactly  $\beta_i(u_1, \dots, u_w)$ . Thus, each  $(u_1, \dots, u_w)$  defines a *check variable* or just *check*.

By varying the values of  $u_1, \dots, u_w$ , we can create different rows of the parity check matrix, i.e., obtain information about different subsets of coefficients of  $\mathbf{f}$ . In our attack, we generate a regular parity-check matrix with  $n$  columns (each corresponding to a coefficient of  $\mathbf{f}$ ) and  $k$  rows, where  $k$  is the attack parameter and depends on the oracle accuracy and the parameter set of the scheme. The matrix is constructed following the progressive edge-growth algorithm [HEA05].

**Construct the inner code** Recall that  $\beta_i(u_1, \dots, u_w) \in \{-w, \dots, w\}$  while each oracle call returns a single bit corresponding to the distinguishers  $ds_Z$  or  $\bar{ds}_Z$ , where  $Z = Z(\ell_1, \ell_2, \ell_3)$ .

Since  $\beta_i(u_1, \dots, u_w)$  can take multiple values, this naturally leads to an encoding problem. For example, when querying  $ds_1$  (i.e., whether  $\beta_i(u_1, \dots, u_w) \geq 1$ ), all values of  $\beta_i(u_1, \dots, u_w)$  greater than or equal to 1 are encoded as 1, while the rest are encoded as 0. This essentially splits the original interval  $\{-w, \dots, w\}$  into two segments. We can then adaptively refine these segments—similar to Huffman coding—by further splitting them to minimize the average encoding length. The goal is to maximize the amount of information gained from each oracle call from an information-theoretical perspective.

Let us illustrate for  $w = 6$ , which we use for our experiments in the noisy case. Here,  $\beta_i(u_1, \dots, u_w)$  takes values from  $\{-6, \dots, 6\}$ . We can begin by querying  $ds_1$ . If the oracle returns 1, we know (assuming no errors from the oracle) that  $\beta_i(u_1, \dots, u_w) \in \{1, \dots, 6\}$ . On the other hand, if the initial oracle response is 0, we conclude that  $\beta_i(u_1, \dots, u_w) \in \{-6, \dots, 0\}$ . We then query  $ds_2$  in the former case and  $\bar{ds}_{-1}$  in the latter. If necessary, we continue by adding  $ds_3$  or  $\bar{ds}_{-2}$ , thus, we uniquely determine value from  $\{-2, -1, 0, 1, 2\}$ , but encode  $\{3, 4, 5, 6\}$  and  $\{-6, -5, -4, -3\}$  the same way. Essentially, this process can be visualized as navigating a binary tree. The final encoding is presented in Table 6. Even though not all values are uniquely encoded, it is still enough for the key recovery as the probability of  $|\beta_i(u_1, \dots, u_w)| \geq 4$  is quite small. We also run the attack with  $w = 2$ , in this case, the encoding follows Table 6 except that we do not use  $ds_3$  and  $\bar{ds}_{-2}$ , resulted encoding is the same as used for NTRU Prime in Table 11(a).

| Value    | -6/-5/-4/-3 | -2 | -1 | 0 | 1 | 2 | 3/4/5/6 |
|----------|-------------|----|----|---|---|---|---------|
| Encoding | 0           | 0  | 0  | 0 | 1 | 1 | 1       |
|          | 1           | 1  | 1  | 0 | 0 | 1 | 1       |
|          | 1           | 1  | 0  |   |   | 0 | 1       |
|          | 1           | 0  |    |   |   |   |         |

**Table 6:** Inner code for  $\beta_i(u_1, u_2, u_3, u_4, u_5, u_6)$

To make key recovery possible, for each row of the parity check matrix that defines indices  $(u_1, \dots, u_w)$ , we adaptively call the PC oracle corresponding to the inner code from Table 6. Then we compute conditional probability

$$\Pr[y \mid \beta_i(u_1, \dots, u_w) = X]$$

for all  $X \in \{-w, \dots, w\}$ . The final step of the attack includes the call to the LDPC decoder; as an input to it, we submit all conditional probabilities for each  $\beta_i(u_1, \dots, u_w)$  corresponding to each row of the parity check matrix. Table 7 shows the parameters  $(\ell_1, \ell_2, \ell_3)$  that are necessary to construct the ciphertexts that will be used during the attack.

| Weight  | $(\ell_{11}, \ell_{12}, \ell_{13})$ | $(\ell_{21}, \ell_{22}, \ell_{23})$ | $(\ell_{31}, \ell_{32}, \ell_{33})$ |
|---------|-------------------------------------|-------------------------------------|-------------------------------------|
| $w = 2$ | (138,48,72)                         | (117,39,132)                        |                                     |
| $w = 6$ | (132,54,21)                         | (138,48,27)                         | (129,48,30)                         |

**Table 7:** Parameters to construct the attack ciphertext for ntruhs2048509. The values of  $(\ell_{j1}, \ell_{j2}, \ell_{j3})$  correspond to the check  $\beta_i(\cdot) \geq j$

### 3.2 Simulation Results

In this section, we present the experimental results, including simulation results and also an experimental validation on a ChipWhisperer platform.

This section investigates the impact of oracle inaccuracy on attack performance through simulation. We model realistic scenarios where constructing a perfect oracle is challenging due to factors like measurement noise or robust countermeasures such as masking. To simulate oracle inaccuracies, we introduce errors into the oracle output by adding a binary value sampled from a Bernoulli distribution  $\text{Ber}_{1-\rho}$ . This approach allows us to control the probability  $\rho$  of a correct oracle response. We present simulation results for two oracle accuracy levels:  $\rho = 1.00$  and  $0.96$ . The former corresponds to the perfectly accurate oracle used in [Rav+22a], while the latter aligns with the accuracy reported in [Uen+22a] for an oracle constructed by targeting a masked software implementation of a pseudorandom function.

Table 8 presents the number of side-channel traces required for successful key recovery under these simulated conditions. We define success as achieving an average number of erroneous positions in the recovered secret polynomial less than 1. This threshold is adopted since correcting a single error position through post-processing requires minimal effort. Compared to the approach by Ueno et al. [Uen+22a], which also investigates oracles with inaccurate responses, our method demonstrates a significant reduction in the required number of traces (as shown in Table 8).

The oracle built in [Rav+22a] has perfect accuracy. The LDPC framework, in contrast, is designed for noisy oracles; nevertheless, Table 8 shows that our approach shows improvement. This is largely attributable to more efficient encoding that requires fewer oracle calls per check.

Recall that the construction of the PC oracle comes with a few overheads, namely, one needs to find a base ciphertext and handle multiple collision cases. In [Rav+22a], authors adopt the early abort technique by analyzing the distribution of values obtained from the oracle and checking if it matches the expected one. If not, they restart the attack to avoid a multiple collision situation. We can use this approach for  $\rho = 1$  (for less accurate oracles, it is less efficient since there are more false negatives and false positives outcomes), thus, the overhead we have for our method is the same as in [Rav+22a]. And from 1162.51 average traces, assuming a true single collision and ignoring traces required to construct the base ciphertext, the total number of traces grows to 1243. Note that the reported number in [Rav+22a] includes the overheads.

For  $\rho = 0.96$ , the key recovery needs only 1990.08 traces without overheads. We estimate that this number should increase to around 2270 traces; however, the estimation does not include early aborts; thus, the number could be improved.

Similar to [Guo+23b], we can observe that the relative improvement in the number of traces is larger for less accurate oracles due to the great error-correcting capabilities of the LDPC decoder.

| $\rho = 0.96$         | Avg. number of traces | Avg. number of errors |
|-----------------------|-----------------------|-----------------------|
| Ueno et al. [Uen+22a] | 5090                  | 0/509                 |
| This work             | 2270 (-55%)           | 0.22/509              |
| $\rho = 1.00$         | Avg. number of traces | Avg. number of errors |
| Ravi et al. [Rav+22a] | 1791                  | 0/509                 |
| This work             | 1243 (-31%)           | 0/509                 |

**Table 8:** Comparison of simulation results between our approach and state-of-the-art for ntruhs2048509. We use  $w = 6$  with 709 checks (rows of the parity check matrix) for  $\rho = 0.96$  and  $w = 2$  with at most 509 checks for  $\rho = 1$

## 4 KR-CCA-SCA-LDPC on NTRU Prime

In this section, we demonstrate how to migrate the KR-CCA-SCA-LDPC attack to NTRU Prime KEMs. We begin with the construction of the base ciphertext  $\mathbf{c}_{\text{base}}$ , followed by the extraction of key-related side-channel information through carefully crafted attack ciphertexts, which is then modeled using LDPC codes. Finally, we present the complete key-recovery algorithm.

### 4.1 The Main Attack

**Construct the base ciphertext** To ensure the occurrence of a single true collision, we construct the base ciphertext as

$$\begin{aligned}\mathbf{c}_{\text{base}} &= t_1(x^{i_1} + x^{i_2} + \dots + x^{i_o}) + t_2(x^{j_1} + x^{j_2} + \dots + x^{j_p}) \cdot \mathbf{h} \\ &= t_1 \cdot \mathbf{d}_1 + t_2 \cdot \mathbf{d}_2 \cdot \mathbf{h},\end{aligned}$$

where  $\mathbf{d}_1$  and  $\mathbf{d}_2$  are polynomials with  $o$  and  $p$  coefficients of  $\pm 1$ , respectively. Hence, the corresponding  $\mathbf{a} = 3\mathbf{f} \cdot \mathbf{c}_{\text{base}}$  is

$$\begin{aligned}\mathbf{a} &= 3\mathbf{f} \cdot (t_1 \cdot \mathbf{d}_1 + t_2 \cdot \mathbf{d}_2 \cdot \mathbf{h}) \\ &= 3t_1 \cdot \mathbf{d}_1 \cdot \mathbf{f} + t_2 \cdot \mathbf{d}_2 \cdot \mathbf{g}.\end{aligned}$$

Since NTRU Prime works in the non-cyclic ring  $\mathbb{Z}_q[x]/(x^n - x - 1)$ , then we have

$$\begin{aligned}\mathbf{f} \cdot x^i \bmod (x^n - x - 1) &= \mathbf{f}[n-i] + (\mathbf{f}[n-i] + \mathbf{f}[n-i+1])x + \mathbf{f}[n-i+1]x^2 + \dots \\ &\quad + (\mathbf{f}[n-1] + \mathbf{f}[0])x^i + \mathbf{f}[1]x^{i+1} + \dots + \mathbf{f}[n-i-1]x^{n-1},\end{aligned}$$

all the coefficients of  $\mathbf{f} \cdot x^i$  belong to  $\{-2, -1, 0, 1, 2\}$ . We assume a collision occurs at index  $i$ . Given that

$$\begin{aligned}\mathbf{a}[i] &= 3t_1(\mathbf{f} \cdot x^{i_1})[i] + 3t_1(\mathbf{f} \cdot x^{i_2})[i] + \dots + 3t_1(\mathbf{f} \cdot x^{i_o})[i] \\ &\quad + t_2(\mathbf{g} \cdot x^{j_1})[i] + \dots + t_2(\mathbf{g} \cdot x^{j_p})[i],\end{aligned}$$

it follows that  $|\mathbf{a}[i]|$  reaches its maximum value when the  $i$ -th coefficients of  $\mathbf{f} \cdot x^{i_\alpha}$  and  $\mathbf{g} \cdot x^{j_\beta}$  are all equal to 2 (or  $-2$ , but here we focus on the positive case for the simplicity), resulting in  $\mathbf{a}[i] = 3t_1 \cdot 2o + t_2 \cdot 2p$ , where  $1 \leq \alpha \leq o$ , and  $1 \leq \beta \leq p$ . We therefore choose  $(t_1, t_2)$  that satisfy the following three conditions:

$$3 \mid t_1, 3 \mid t_2, 3t_1 \cdot r + t_2 \cdot s \begin{cases} > q/2, & \text{if } r = 2o, s = 2p, \\ < q/2, & \text{otherwise,} \end{cases}$$

where  $0 \leq r \leq 2o$  and  $0 \leq s \leq 2p$ .

**Effects of the Rounding Operation** In NTRU Prime Encrypt, a rounding function Round is introduced to reduce the ciphertext size, meaning that only the quotient of each coefficient upon division by 3 is transmitted. However, the base ciphertext that we constructed is not strictly a multiple of 3, and consequently, the decryption of it introduces a rounding noise  $\mathbf{e}' \in \mathbb{Z}_3[x]/(\Psi_n)$ . Therefore, the actual value of the base ciphertext during decryption is

$$\begin{aligned} \mathbf{c}_{\text{base}} &= \text{Round}(t_1 \cdot \mathbf{d}_1 + t_2 \cdot \mathbf{d}_2 \cdot \mathbf{h}) \\ &= t_1 \cdot \mathbf{d}_1 + t_2 \cdot \mathbf{d}_2 \cdot \mathbf{h} + \mathbf{e}', \end{aligned}$$

Therefore, the corresponding  $\mathbf{a} = 3\mathbf{f} \cdot \mathbf{c}_{\text{base}}$  is

$$\begin{aligned} \mathbf{a} &= 3\mathbf{f} \cdot (t_1 \cdot \mathbf{d}_1 + t_2 \cdot \mathbf{d}_2 \cdot \mathbf{h} + \mathbf{e}') \\ &= 3t_1 \cdot \mathbf{d}_1 \cdot \mathbf{f} + t_2 \cdot \mathbf{d}_2 \cdot \mathbf{g} + 3\mathbf{f} \cdot \mathbf{e}' \\ &:= \mathbf{a}_1 + \mathbf{a}_2, \end{aligned}$$

where  $\mathbf{a}_1 = 3t_1 \cdot \mathbf{d}_1 \cdot \mathbf{f} + t_2 \cdot \mathbf{d}_2 \cdot \mathbf{g}$  and we refer to  $\mathbf{a}_2 = 3\mathbf{f} \cdot \mathbf{e}'$  as the internal noise term.

Despite the small size of the noise term  $\mathbf{a}_2$ , its impact is still non-negligible. We denote the largest coefficient of the polynomial  $\mathbf{a}_1$  as  $m_1 = 3t_1 \cdot 2o + t_2 \cdot 2p$ . The second largest possible coefficient is called  $m_2$ . When constructing the base ciphertext, we choose parameters  $(t_1, t_2)$  such that  $m_1 > q/2$  and  $m_2 < q/2$ . We define

$$\begin{aligned} dm_1 &= |3t_1 \cdot 2o + t_2 \cdot 2p - q/2| \\ dm_2 &= \left| \max_{(r,s) \neq (2o,2p)} (3t_1 \cdot r + t_2 \cdot s) - q/2 \right| \end{aligned}$$

denote the distance between  $m_1$  and  $m_2$  from  $q/2$ , respectively. The presence of the noise term may change the values of  $(m_1 \bmod q)$  or  $(m_2 \bmod q)$ . For instance, when  $\mathbf{a}_1[i] = m_2$  and the corresponding noise component satisfies  $\mathbf{a}_2[i] > dm_2$ , this results in  $\mathbf{a}[i] = \mathbf{a}_1[i] + \mathbf{a}_2[i] > q/2$ , which consequently leads to  $\mathbf{e}[i] \neq 0$ . We call this event a false positive collision. Conversely, if  $\mathbf{a}_1[i] = m_1$  (due to a valid collision), but the corresponding noise component satisfies  $\mathbf{a}_2[i] < -dm_1$ , then  $\mathbf{a}[i] = \mathbf{a}_1[i] + \mathbf{a}_2[i] < q/2$  leading to  $\mathbf{e}[i] = 0$ , thereby masking the presence of the collision. We call this a false negative collision. Both false positive and false negative collisions are undesirable scenarios, rendering them ineffective for key recovery.

We choose parameters  $(o, p)$  such that a single collision occurs with high probability. Table 9 lists the concrete values of  $(o, p)$  and  $(t_1, t_2)$  together with the probability of obtaining true single, multiple collisions, false positive collision,

| Scheme     | $(o, p)$ | $(t_1, t_2)$ | $(dm_1, dm_2)$ | Single true | Multiple | False Positive | False Negative |
|------------|----------|--------------|----------------|-------------|----------|----------------|----------------|
| sntrup653  | (1, 3)   | (102, 309)   | (156, 150)     | 0.0455      | 0.0016   | 0.0016         | 0.0001         |
| sntrup761  | (1, 3)   | (102, 306)   | (153, 153)     | 0.0391      | 0.0011   | 0.0012         | 0.0001         |
| sntrup1277 | (1, 3)   | (171, 528)   | (255, 258)     | 0.0678      | 0.0035   | 0.0001         | $\approx 0$    |

**Table 9:** Concrete parameters for constructing base ciphertext in NTRU Prime with collision probabilities

and false negative collision for the chosen parameters. These results were obtained through 10000 attack simulation trials using 10000 different secret keys. For each secret key, the base ciphertext is regenerated on every attempt, with the polynomials  $\mathbf{d}_1$  and  $\mathbf{d}_2$  being regenerated.

**Construct the attack ciphertext** After deriving the base ciphertext, we leverage the polynomials  $\mathbf{d}_1$  and  $\mathbf{d}_2$  to formulate the attack ciphertext  $\mathbf{c}_{\text{att}}$  as follows (before rounding):

$$\mathbf{c}_{\text{att}} = \ell_1 \cdot \mathbf{d}_1 + \ell_2 \cdot \mathbf{d}_2 \cdot \mathbf{h} + \ell_3 \cdot (x^{u_1} + x^{u_2} + \dots + x^{u_w}),$$

where  $\ell_1, \ell_2, \ell_3 \in \mathbb{Z}^+$  and  $0 \leq u_i < n$  for even  $w$ . Thus, for  $\mathbf{c}_{\text{att}}$ , we have

$$\begin{aligned} \mathbf{a} &= 3\mathbf{f} \cdot \mathbf{c}_{\text{att}} = 3\ell_1 \cdot \mathbf{d}_1 \cdot \mathbf{f} + \ell_2 \cdot \mathbf{d}_2 \cdot \mathbf{g} \\ &\quad + 3\ell_3 \cdot \mathbf{f} \cdot (x^{u_1} + x^{u_2} + \dots + x^{u_w}). \end{aligned}$$

Suppose there is a positive collision at index  $i$ , i.e.,  $\mathbf{d}_1 \cdot \mathbf{f}$  and  $\mathbf{d}_2 \cdot \mathbf{g}$  are  $2o$  and  $2p$ , resp. We then define

$$\beta_i(u_1, \dots, u_w) = (\mathbf{f} \cdot (x^{u_1} + x^{u_2} + \dots + x^{u_w}))[i],$$

where

$$\mathbf{f} \cdot x^{u_1} = \begin{cases} f[i - u_1], & 0 \leq u_1 < i, \\ f[0] + f[n - 1], & u_1 = i, \\ f[n - 1 + i - u_1] + f[n + i - u_1], & i < u_1 < n. \end{cases}$$

Then the coefficients of  $\mathbf{a}$  can be expressed as

$$\mathbf{a}[j] = \begin{cases} 3\ell_1 2o + \ell_2 2p + 3\ell_3 \beta_j(u_1, \dots, u_w), & \text{if } j = i \\ 3\ell_1 r + \ell_2 s + 3\ell_3 \beta_j(u_1, \dots, u_w), & \text{if } \begin{cases} j \neq i \text{ and} \\ (r, s) \neq (2o, 2p) \end{cases} \end{cases}$$

We see that  $\beta_i(u_1, \dots, u_w)$  takes values in  $\{-2w, \dots, 2w\}$ . Similar to Section 3, we select appropriate parameters  $(\ell_1, \ell_2, \ell_3)$  such that  $\mathbf{a}[j] < q/2$  for  $j \neq i$

regardless of the value of  $\beta_j(u_1, \dots, u_w)$ , but  $\mathbf{a}[i] > q/2$  when  $\beta_i(u_1, \dots, u_w) \geq Z$  for some  $Z \in \{-2w, \dots, 2w\}$  that depends on  $(\ell_1, \ell_2, \ell_3)$ .

It should be noted that due to the influence of the internal noise term  $\mathbf{a}_2$ , the oracle's return value is not always accurate. The impact of internal noise is more intuitively reflected in the values of  $(dm_1, dm_2)$ . Therefore, we prioritize selecting parameters  $(\ell_1, \ell_2, \ell_3)$  with larger  $(dm_1, dm_2)$  as it reduces the probability of potential flipping caused by  $\mathbf{a}_2$ . Table 10 presents the specific parameters used in the attack ciphertexts.

| sntrup653                           |                |                                     |                |
|-------------------------------------|----------------|-------------------------------------|----------------|
| $(\ell_{11}, \ell_{12}, \ell_{13})$ | $(dm_1, dm_2)$ | $(\ell_{21}, \ell_{22}, \ell_{23})$ | $(dm_1, dm_2)$ |
| (90,288,30)                         | (48,42)        | (84,270,42)                         | (66,60)        |
| (96,282,30)                         | (48,42)        | (87,264,45)                         | (66,60)        |
| (102,276,30)                        | (48,42)        | (99,255,42)                         | (66,60)        |
| sntrup761                           |                |                                     |                |
| $(\ell_{11}, \ell_{12}, \ell_{13})$ | $(dm_1, dm_2)$ | $(\ell_{21}, \ell_{22}, \ell_{23})$ | $(dm_1, dm_2)$ |
| (90,285,30)                         | (45,45)        | (84,267,42)                         | (63,63)        |
| (96,279,30)                         | (45,45)        | (87,261,45)                         | (63,63)        |
| (105,270,30)                        | (45,45)        | (99,252,42)                         | (63,63)        |
| sntrup1277                          |                |                                     |                |
| $(\ell_{11}, \ell_{12}, \ell_{13})$ | $(dm_1, dm_2)$ | $(\ell_{21}, \ell_{22}, \ell_{23})$ | $(dm_1, dm_2)$ |
| (153,492,51)                        | (84,69)        | (153,447,75)                        | (111,111)      |
| (162,483,51)                        | (84,69)        | (156,444,75)                        | (111,108)      |
| (171,471,54)                        | (75,72)        | (165,438,72)                        | (111,105)      |

**Table 10:** Parameters to construct the attack ciphertext for NTRU Prime. The values of  $(\ell_{j1}, \ell_{j2}, \ell_{j3})$  correspond to the check  $\beta_i(\cdot) \geq Z$

**Construct the inner code** We adopt the same adaptive encoding method as described in Section 3 using  $w = 2$ . Although the internal noise term  $\mathbf{a}_2$  has a relatively small magnitude, it may cause certain ciphertext evaluations to shift from one distinguisher to another (for example, an element that should correspond to  $ds_0$  may be perturbed into  $ds_1$ ), thereby reducing the reliability of distinguisher selection. To mitigate this noise effect and maximize information utilization efficiency, we employ two internal encoding strategies in NTRU Prime. The first

strategy starts with the query  $\beta_i(\cdot) \geq 1$  (if the oracle can distinguish this condition, the encoding corresponding to Table 11(a) is adopted); the second strategy starts with  $\beta_i(\cdot) \geq 0$  (if the oracle can distinguish this condition, the encoding corresponding to Table 11(b) is adopted). To improve the statistical efficiency of encoding, we merge  $-4$  and  $-3$  into  $-2$ , and similarly merge  $3$  and  $4$  into  $2$ . Based on these encoding and selection strategies, the belief propagation algorithm can robustly recover the secret key.

Next, we describe how to determine the target distinguisher. For the case  $w = 2$ ,  $\beta_i(u_1, \dots, u_w) \in \{-4, \dots, 4\}$ , and its distribution is determined by the secret key coefficients. Since the weight distribution of the secret key is known, the expected distribution of  $\beta_i$  can be precomputed for a given sample size — for example, Table 12 illustrates the case of `sntrup653` (sample size = 200). Based on this distribution, the expected output counts corresponding to each distinguisher  $ds_Z$  ( $Z \in \{-4, \dots, 4\}$ ) can be derived. For instance, for  $ds_1$ , the expected total number of zeros is  $0.076 + 1.119 + 12.468 + 48.395 + 75.882 \approx 137.94$ , and the expected total number of ones is  $48.395 + 12.468 + 1.119 + 0.076 \approx 62.06$ . During the actual attack, we collect oracle responses for the constructed ciphertexts and record the observed counts of “0” and “1”. Then, we compare the observed counts with the expected counts corresponding to each candidate distinguisher  $ds_Z$ , compute their statistical distance, and select the candidate with the smallest distance as the best match. This procedure allows us to determine the most likely distinguisher, such as  $ds_0$  or  $ds_1$ .

| Value    | -4/-3/-2 | -1 | 0 | 1 | 2/3/4 |
|----------|----------|----|---|---|-------|
| Encoding | 0        | 0  | 0 | 1 | 1     |
|          | 1        | 1  | 0 | 0 | 1     |
|          | 1        | 0  |   |   |       |

(a) Inner code built with  $ds_1$  and  $ds_2$ 

| Value    | -4/-3/-2 | -1 | 0 | 1 | 2/3/4 |
|----------|----------|----|---|---|-------|
| Encoding | 0        | 0  | 1 | 1 | 1     |
|          | 1        | 0  | 1 | 0 | 0     |
|          |          |    |   | 0 | 1     |

(b) Inner code built with  $ds_0$  and  $ds_2$ **Table 11:** Inner codes for  $\beta(u_1, u_2)$ 

**Recover the secret key** We present the full procedure for recovering the secret key  $\mathbf{f}$  in Algorithm 13. For oracles with lower accuracy, we introduce redundancy queries to overcome the noise. Similarly to Section 3, we create a regular

| $\beta_i$ value | −4    | −3    | −2     | −1     | 0      | 1      | 2      | 3     | 4     |
|-----------------|-------|-------|--------|--------|--------|--------|--------|-------|-------|
| Expected count  | 0.076 | 1.119 | 12.468 | 48.395 | 75.882 | 48.395 | 12.468 | 1.119 | 0.076 |

**Table 12:** Expected distribution of  $\beta_i$  values for sntrup653 (sample size = 200)

parity-check matrix  $H$  with constant row weight  $w$ , where each row defines indices  $(u_1, \dots, u_w)$  such that the oracle is adaptively called on  $\beta_i(u_1, \dots, u_w)$ .

Concretely, in the algorithm we first construct attack ciphertexts and query the oracle on a number of initial rows of  $H$  (for example, the first  $k_1 = 200$  rows) to collect preliminary responses. Based on these responses we determine and confirm the required distinguisher type ( $ds_0$  or  $ds_1$ ). Once the distinguisher has been identified, we proceed to query the oracle on the remaining rows and collect the outputs; from all observations we compute the conditional probabilities associated with each index group. Finally, these conditional probabilities are supplied as soft information to the LDPC decoder for iterative decoding. The decoded candidate key is then validated by checking whether its weight matches the expected value; upon successful validation the complete secret key  $\mathbf{f}$  is recovered.

## 4.2 Simulation Results

In this section, we present experimental results of the KR-CCA-SCA-LDPC attack on NTRU Prime and analyze the impact of oracle inaccuracy on attack performance. We introduce inaccuracy into the oracle output by adding a binary value sampled from a Bernoulli distribution  $\text{Ber}_{1-\rho}$ . We present simulation results for two oracle accuracy levels:  $\rho = 0.96$  and  $\rho = 1.00$ . The former aligns with the results reported in [Uen+22a], while the latter corresponds to the perfectly accurate oracle used in [Rav+22a].

We run the simulations for different parameter sets and oracle accuracy to compare our results with existing work. For sntrup761 with  $\rho = 1.00$ , we use  $w = 1$ , the parity-check matrix is just the identity matrix. For  $\rho = 0.96$ , we consider parameters sntrup653 and sntrup1277, in both cases  $w = 2$  is used. The regular parity-check matrix consists of 1965 and 3831 rows for sntrup653 and sntrup1277, respectively.

Table 13 shows the number of side-channel traces required for successful key recovery under these simulation conditions. Compared to the approach by Ueno et al. [Uen+22a] (which also investigates oracles with inaccurate responses), our method achieves a significant reduction in the required number of traces. For sntrup653 and sntrup1277, our method reduces the number of traces by approximately 59% and 58%, respectively. In comparison to the scenario with perfect accuracy from [Rav+22a], our method reduces the required number of traces by about 35%.

**Algorithm 13** KR-CCA-SCA-LDPC on NTRU Prime

---

```

1: Input: Reference parameters, secret key weight  $w$ , number of parity-check rows  $k$ ,
   parameters  $k_1, k_2$ 
2: Output: Secret key  $\mathbf{f}$ 
3:  $(\mathbf{d}_1, \mathbf{d}_2) \leftarrow \text{GenBaseCiphertext}(t_1, t_2)$ 
4: Generate a regular parity-check matrix  $H$  with constant row weight  $w$  with  $k$  rows.
5: found_dist1  $\leftarrow$  false
6: for each parameter triple  $(\ell_{11}, \ell_{12}, \ell_{13})$  do
7:   Collect oracle responses for the first  $k_1$  rows of  $H$  using ciphertexts generated
   from  $(\ell_{11}, \ell_{12}, \ell_{13})$  and row.
8:   Analyze the distribution of '0's and '1's to determine a candidate distinguisher
    $ds_Z$ .
9:   if  $ds_Z$  is  $ds_0$  or  $ds_1$  then
10:     found_dist1  $\leftarrow$  true
11:     break
12:   end if
13: end for
14: if not found_dist1 then
15:   goto Step 1
16: end if
17: For rows  $0, \dots, k_1 - 1$ , call the PC oracle with  $\bar{ds}_Z$  to complete the  $(ds_Z, \bar{ds}_Z)$ 
   encoding (i.e., encoding from Table 11 without  $ds_2$ ).
18: For rows  $k_1, \dots, k_2 - 1$ , call the PC oracle with  $(ds_Z, \bar{ds}_Z)$  encoding.
19: Verify that the distribution of responses over all  $k_2$  rows matches the expected distri-
   bution for the chosen distinguisher.
20: if the distribution is not as expected then
21:   goto Step 1
22: end if
23: found_dist2  $\leftarrow$  false
24: for each parameter triple  $(\ell_{21}, \ell_{22}, \ell_{23})$  do
25:   For the first  $k_1$  rows, call the PC oracle with ciphertexts from  $(\ell_{21}, \ell_{22}, \ell_{23})$  to
   get responses for full  $(ds_Z, \bar{ds}_Z, ds_2, \bar{ds}_2)$  encoding.
26:   Analyze the resulting distribution to confirm if the distinguisher is indeed  $ds_2$ .
27:   if the distinguisher is confirmed as  $ds_2$  then
28:     found_dist2  $\leftarrow$  true
29:     break
30:   end if
31: end for
32: if not found_dist2 then
33:   goto Step 1
34: end if
35: For rows  $k_1, \dots, k - 1$ , finish the 4-distinguisher encoding by calling the PC oracle
   for  $ds_2$  and  $\bar{ds}_2$ .
36: Based on all oracle responses, compute the conditional probabilities  $\Pr[y \mid$ 
    $\beta_i(u_1, \dots, u_w) = X]$  for each possible value  $X \in \{-4, \dots, 4\}$ .
37:  $\mathbf{f}_{\text{candidate}} \leftarrow \text{LDPC\_Decode}(H, \text{Conditional Probabilities})$ 
38: return  $\mathbf{f}_{\text{candidate}}$ 

```

---

| <b>sntrup761</b>      |                       |                       |
|-----------------------|-----------------------|-----------------------|
| $\rho = 1.00$         | Avg. number of traces | Avg. number of errors |
| Ravi et al.           | 3659                  | 0/761                 |
| Our Method            | 2379.88 (−35%)        | 0.25/761              |
| <b>sntrup653</b>      |                       |                       |
| $\rho = 0.96$         | Avg. number of traces | Avg. number of errors |
| Ueno et al. [Uen+22a] | 13560                 | 0/653                 |
| Our Method            | 5615.79 (−59%)        | 0.36/653              |
| <b>sntrup1277</b>     |                       |                       |
| $\rho = 0.96$         | Avg. number of traces | Avg. number of errors |
| Ueno et al. [Uen+22a] | 25875                 | 0/1277                |
| Our Method            | 10743.62 (−58%)       | 0.48/1277             |

**Table 13:** Comparison of our approach with state-of-the-art simulation results on NTRU Prime.

Note that in our simulations, since the PC Oracle is not always perfectly accurate and due to the iterative decoder specifics, the average number of errors is non-zero, indicating that there may be recovery errors at certain positions of the key. We define “success” as having an average number of erroneous positions in the recovered secret polynomial equal to 1 or less; in this case, the secret key is recovered with very light post-processing (if necessary). When there are up to 20 incorrect positions in the recovered key, we define it as “partial success”; the real secret key could be recovered with involved post-processing that usually involves lattice reduction. Otherwise, mostly due to false positives and multiple collisions, the secret key can not be recovered. The specific probability values are shown in Table 14.

## 5 Key-recovery Attack on a Masked NTRU Implementation

In this section, we introduce and demonstrate the first key-recovery attack on a real-world masked NTRU implementation. Our experiments are based on the ChipWhisperer Husky toolkit, including the CW308 UFO board and the CW308T-STM32F4 target board with a 32-bit ARM Cortex-M4 CPU, operating at a frequency of 24 MHz, with traces sampled at the same frequency. The library is compiled using the -O3 optimization level. We focus on a first-order masked

| sntrup761     |             |                     |             |
|---------------|-------------|---------------------|-------------|
| $\rho = 1.00$ | Success Pr. | Partial Success Pr. | Failure Pr. |
| Our Method    | 0.946       | 0.039               | 0.015       |
| sntrup653     |             |                     |             |
| $\rho = 0.96$ | Success Pr. | Partial Success Pr. | Failure Pr. |
| Our Method    | 0.890       | 0.089               | 0.021       |
| sntrup1277    |             |                     |             |
| $\rho = 0.96$ | Success Pr. | Partial Success Pr. | Failure Pr. |
| Our Method    | 0.847       | 0.131               | 0.022       |

**Table 14:** Successful rate of our approach on NTRU Prime.

NTRU implementation [Cor+23b] and show how to mount an attack specifically targeting this implementation.

Unlike prior research, such as the work in [Uen+22a], our attack is the first to specifically target a real-world masked NTRU implementation. The attack described in [Uen+22a] employs a generic method applicable to various schemes—including lattice-based, code-based, and isogeny-based systems—that utilize CCA transformations. However, this prior work evaluates a masked software implementation of a pseudorandom function as an assumed building block, rather than investigating a real-world masked NTRU implementation. We target ntruhs2048509 parameter set for experiments, the attack directly extends to other parameters.

### 5.1 Leakage Detection

The leakage that we use for the attack comes from the function `masked_pack_S3`. It packs masked ternary polynomial  $m'$  into bytes, so that later in the decapsulation we can compute hash over  $r'$  and  $m'$ . Since  $m'$  is a ternary polynomial, 5 coefficients of it fit into a byte, thus, function `masked_pack_S3` in Algorithm 14 is called sequentially on chunks of five coefficients of  $m'$ . We notice a leakage from line 5, but there is a much stronger leakage at line 8, which we use for the attack. The function `convert_A2B_CGV14` takes 143 samples, our power traces consists of a concatenation of  $\lceil n/5 \rceil = \lceil 509/5 \rceil = 102$  calls to it, resulting in 14586 samples total.

To demonstrate the leakage, we use the popular TVLA framework [GJR+11] that uses Welch's  $t$ -test. Recall that  $m'$  during the decryption is set to either 0 or  $\pm x^i \cdot f_p$  depending on whether the secret-dependent coefficient of  $\mathbf{a}$  is pushed past  $q/2$ . Collected power traces over random keys are split into two sets: in the

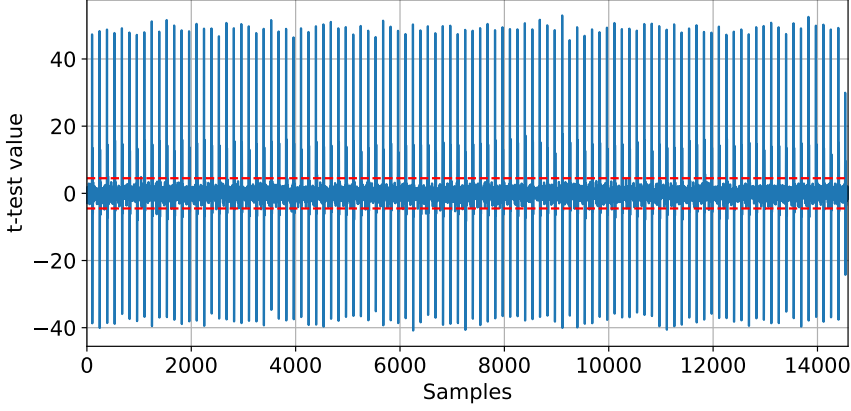


Figure 1: The  $t$ -test value for concatenated calls of `convert_A2B_CGV14` inside `masked_pack_S3`.

first set, we include all traces corresponding to  $m' = 0$ , and the rest are included in the second. The result of the  $t$ -statistic is shown in Figure 1. It is clearly visible that there are 102 peaks, one for each call of `convert_A2B_CGV14`.

## 5.2 Real-World Validation

The key recovery process comprises two distinct stages: profiling and attack, each conducted on a separate device. This setup presents a greater challenge compared to scenarios where the adversary can use the same device to both construct a model that extracts information from power traces and subsequently recovers the secret key.

The profiling stage uses a device  $D_1$ , over which the adversary has complete control. We collect 20,000 power traces for decapsulation on  $D_1$  using chosen ciphertexts. Each decapsulation call uses a ciphertext that results in the decrypted message  $m'$  being either 0 or  $\pm x^i \cdot f_p$ , both of which are masked; each call uses random keys. We train a neural network (NN) model over up to 20 epochs to distinguish between these two cases. The architecture of it is shown in Table 15.

The attack stage involves a different device,  $D_2$ , which the adversary has limited access to. The device  $D_2$  contains an unknown secret key that the adversary aims to recover. Using our trained model, we successfully recovered the secret key in 97 out of 100 cases, with a maximum of one coefficient difference, where the unsuccessful key recovery is due to a multiple collision case. Recall that the key recovery process involves (i) finding a base ciphertext and (ii) constructing attack ciphertexts to obtain information about the secret key. The total number of oracle calls per key is 1210.06, increasing to 1247.48 when accounting for retries in the 3 failed cases. One can try to detect a multiple collision case early and restart if it is detected; this allows us to reduce the total average to successfully recover the

---

**Algorithm 14** masked\_pack\_S3( $v_0, \dots, v_4, n$ )

---

**Require:**  $v_j = (v_{j,1}, \dots, v_{j,n})$  — arithmetic masking of ternary value modulo 3 with  $n$  shares

**Ensure:** Boolean masking of a byte  $\sum_{j=0}^4 3^j \cdot (\sum_{i=1}^n v_{j,i})$

1:  $p \leftarrow (0, \dots, 0)$

2: **for**  $j = 0$  **to** 4 **do**

3:    $w_j \leftarrow \text{masked\_convert\_mod3\_to\_modq}(v_j, 256) \triangleright w_j$  is arithmetic masking modulo 256

4:   **for**  $i = 1$  **to**  $n$  **do**

5:      $p_i \leftarrow p_i + 3^j \cdot w_{j,i} \bmod 256$

6:   **end for**

7: **end for**

8: **return** convert\_A2B\_CGV14( $p$ )    $\triangleright$  Transform arithmetic masking to boolean

---

key, but here the gain is not that large, so we decided not to over-complicate the presentation.

When tested on traces from  $D_2$ , the NN model achieves an accuracy of  $\rho = 0.995$ . As shown in Table 16, the real attack closely matches the simulation results. Let us emphasize that, in theory, key recovery could be achieved using only 1200.7 traces. However, due to the inherent overhead of the PC oracle—such as handling multiple collision cases and finding the base ciphertext—successful key recovery ultimately requires 1247.48 traces.

## 6 Discussion

### 6.1 Mathematical Aspects of the Attack

Our work introduces a significant mathematical enhancement to the SCA-LDPC framework, which directly improves the efficiency of constructing plaintext-checking (PC) oracles for attacks on NTRU-style KEMs. By refining the underlying mathematical method for information extraction, we substantially reduce the number of side-channel traces required for successful key recovery. The generic nature of our attack makes this reduction particularly significant, as it is not limited to specific masked platforms and can be broadly applied once a PC oracle can be constructed. The importance of this advancement is emphasized by the widespread adoption of NTRU-style KEMs in numerous cryptographic libraries and ongoing standardization efforts.

A key advantage of our approach lies in the dual functionality of Low-Density Parity-Check (LDPC) codes, which provide gains in both randomness extraction and error correction. This dual benefit is particularly impactful across different or-

| Layer type            | (Input, output) shape | # Parameters |
|-----------------------|-----------------------|--------------|
| Batch Normalization 1 | (14586, 14586)        | 58344        |
| Dense 1               | (14586, 64)           | 933568       |
| Batch Normalization 2 | (64, 64)              | 256          |
| ReLU                  | (64, 64)              | 0            |
| Dropout (0.2)         | (64, 64)              | 0            |
| Dense 2               | (64, 64)              | 4160         |
| Batch Normalization 3 | (64, 64)              | 256          |
| ReLU                  | (64, 64)              | 0            |
| Dense 3               | (64, 32)              | 2080         |
| Batch Normalization 3 | (32, 32)              | 128          |
| ReLU                  | (32, 32)              | 0            |
| Dense 4               | (32, 1)               | 33           |
| Sigmoid               | (1, 1)                | 0            |

**Table 15:** The architecture of the neural network used for the real-world attack on NTRU.

| $\rho = 0.995$ | Avg. number of traces | Avg. number of errors |
|----------------|-----------------------|-----------------------|
| Simulation     | 1195.51               | 0.03/509              |
| Real Attack    | 1200.7                | 0.06/509              |

**Table 16:** Average number of errors and traces for ntruhs2048509, where  $w = 6$  with 419 checks. Real attack assumes a single true collision and does not include overhead of finding a base ciphertext

acle accuracy scenarios. In high-accuracy scenarios, where the PC oracle is nearly perfect, randomness extraction is important as the LDPC framework makes the oracle’s output more uniform, thus extracting more information from a single call. The error-correction capability further ensures robustness against rare errors, leading to a more reliable attack. Conversely, in low-accuracy scenarios, where the oracle’s output is noisy (a common challenge when attacking platforms with strong countermeasures like masking), the inherent error-correction properties of LDPC codes become critical. This allows us to successfully recover the secret key even when the leakage is minimal and the oracle’s accuracy is low.

## 6.2 Physical Side Countermeasures

Given that NTRU and its variants are still integrated into multiple widely used libraries and are part of ongoing standardization initiatives, the development of effective physical side countermeasures is of paramount importance. Our results demonstrate that even first-order masked implementations can be vulnerable, highlighting the need for more robust defenses.

The central goal of any countermeasure should be to make the creation of accurate PC oracles prohibitively difficult for an attacker. While our attack was successful against a masked implementation, masking remains a main countermeasure. The challenge is to develop more efficient masking schemes that can withstand advanced attack vectors like ours. Higher-order masking is a logical next step, but its effectiveness must be rigorously evaluated in practical, real-world scenarios.

Another promising research direction is the combination of masking with other defense mechanisms, e.g., combining masking with shuffling techniques. Furthermore, it is essential to consider other approaches for building PC oracles. The emergence of microarchitectural attacks necessitates a thorough investigation of constant-time implementations for NTRU-based schemes. For instance, attacks like GoFetch [Che+24a] demonstrate how hardware-level optimizations designed to improve performance, such as Data Memory-Dependent Prefetchers (DMP), can inadvertently introduce new leakage channels for an adversary to exploit.

## 7 Conclusion

In this work, we have introduced novel methods for adapting the SCA-LDPC framework to PC oracle-based SCAs on NTRU-like KEMs. Our approach addresses the unique challenges posed by the algebraic structure of NTRU and NTRU Prime, enabling efficient key recovery with significantly fewer measurements compared to existing methods. Simulations demonstrated substantial performance enhancements and robustness against decision errors in the constructed oracle. Furthermore, we have presented the first documented real-world key-recovery side-channel attack on a masked NTRU implementation, achieving a plaintext-checking oracle with a low decision error rate of 0.5% in a cross-device setting.

Our findings demonstrate that first-order masking, as implemented in [Cor+23b], is insufficient to protect NTRU against this adapted SCA-LDPC attack, particularly highlighting the risks in cross-device scenarios. This motivates the urgent need for the development and, critically, the rigorous *evaluation* of more robust countermeasures for NTRU implementations. Future work should investigate the effectiveness of higher-order masking schemes, defenses like shuffling or random delays, and combinations thereof against this class of attacks. Additionally, as noted by Tanaka et al. [Tan+23], the NTRULPR variants of NTRU Prime can be enhanced using multi-value plaintext-checking (PC) oracles. It is plausible that our proposed LDPC method can also be adapted to the multi-value PC oracle scenario. We leave this investigation for future research.

## References

- [Ber+16] D. J. Bernstein et al. “NTRU Prime”. In: *IACR Cryptol. ePrint Arch.* (2016), p. 461.
- [Che+20] C. Chen et al. *NTRU*. Tech. rep. available at <https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions>. National Institute of Standards and Technology, 2020.
- [Che+24a] B. Chen et al. “{GoFetch}: Breaking {Constant-Time} Cryptographic Implementations Using Data {Memory-Dependent} Prefetchers”. In: *33rd USENIX Security Symposium (USENIX Security 24)*. 2024, pp. 1117–1134.
- [Cor+23b] J.-S. Coron et al. “Improved Gadgets for the High-Order Masking of Dilithium”. In: *IACR TCHES 2023.4* (2023), pp. 110–145.
- [DAn+19c] J.-P. D’Anvers et al. “Timing attacks on Error Correcting Codes in Post-Quantum Secure Schemes.” In: *IACR Cryptology ePrint Archive 2019* (2019), p. 292.
- [DLP14] L. Ducas, V. Lyubashevsky, and T. Prest. “Efficient identity-based encryption over NTRU lattices”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2014, pp. 22–41.
- [Duc+13] L. Ducas et al. “Lattice Signatures and Bimodal Gaussians”. In: *Cryptology Conference*. 2013.
- [For65] G. D. Forney. “Concatenated codes.” In: *Technical Report 440, MIT* (1965).
- [Fou+18] P.-A. Fouque et al. “Falcon: Fast-Fourier lattice-based compact signatures over NTRU”. In: *Submission to the NIST’s post-quantum cryptography standardization process* 36.5 (2018), pp. 1–75.
- [Gal62] R. Gallager. “Low-density parity-check codes”. In: *IRE Transactions on information theory* 8.1 (1962), pp. 21–28.
- [GJJ22b] Q. Guo, A. Johansson, and T. Johansson. “A key-recovery side-channel attack on Classic McEliece implementations”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2022), pp. 800–827.
- [GJN20b] Q. Guo, T. Johansson, and A. Nilsson. “A key-recovery timing attack on post-quantum primitives using the Fujisaki-Okamoto transformation and its application on FrodoKEM”. In: *Annual International Cryptology Conference*. Springer. 2020, pp. 359–386.

- [GJR+11] B. J. Gilbert Goodwill, J. Jaffe, P. Rohatgi, et al. “A testing methodology for side-channel resistance validation”. In: *NIST non-invasive attack testing workshop*. Vol. 7. 2011, pp. 115–136.
- [GLG22c] G. Goy, A. Loiseau, and P. Gaborit. “A new key recovery side-channel attack on HQC with chosen ciphertext”. In: *International Conference on Post-Quantum Cryptography*. Springer. 2022, pp. 353–371.
- [Goo22] Google Cloud. *Why Google Now Uses Post-Quantum Cryptography for Internal Comms*. <https://cloud.google.com/blog/products/identity-security/why-google-now-uses-post-quantum-cryptography-for-internal-comms>. 2022.
- [Guo+22b] Q. Guo et al. “Don’t reject this: Key-recovery timing attacks due to rejection-sampling in HQC and BIKE”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2022), pp. 223–263.
- [Guo+23b] Q. Guo et al. “SCA-LDPC: A Code-Based Framework for Key-Recovery Side-Channel Attacks on Post-quantum Encryption Schemes”. In: *ASIACRYPT 2023, Part IV*. Ed. by J. Guo and R. Steinfeld. Vol. 14441. LNCS. Springer, Singapore, Dec. 2023, pp. 203–236.
- [Ham+21b] M. Hamburg et al. “Chosen ciphertext k-trace attacks on masked cca2 secure kyber”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2021), pp. 88–113.
- [HEA05] X.-Y. Hu, E. Eleftheriou, and D. Arnold. “Regular and irregular progressive edge-growth tanner graphs”. In: *IEEE Transactions on Information Theory* 51.1 (2005), pp. 386–398.
- [Hei+22b] D. Heinz et al. *First-Order Masked Kyber on ARM Cortex-M4*. Cryptology ePrint Archive, Report 2022/058. 2022.
- [Hof+03] J. Hoffstein et al. “NTRUSIGN: Digital signatures using the NTRU lattice”. In: *Cryptographers’ track at the RSA conference*. Springer. 2003, pp. 122–140.
- [Hof+99] J. Hoffstein et al. “NTRU: A public key cryptosystem”. In: *NTRU Cryptosystems, Inc.(www.ntru.com)* (1999).
- [How+03] N. Howgrave-Graham et al. “The Impact of Decryption Failures on the Security of NTRU Encryption”. In: *CRYPTO 2003*. Ed. by D. Boneh. Vol. 2729. LNCS. Springer. 2003, pp. 226–246.
- [Hül+17] A. Hülsing et al. “High-speed key encapsulation from NTRU”. In: *International Conference on Cryptographic Hardware and Embedded Systems*. Springer. 2017, pp. 232–252.
- [JJ00] É. Jaulmes and A. Joux. “A Chosen-Ciphertext Attack against NTRU”. In: *CRYPTO 2000*. Ed. by M. Bellare. Vol. 1880. LNCS. Springer, Berlin, Heidelberg, Aug. 2000, pp. 20–35.

- [Koc96b] P. C. Kocher. “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems”. In: *CRYPTO’96*. Ed. by N. Koblitz. Vol. 1109. LNCS. Springer, Berlin, Heidelberg, Aug. 1996, pp. 104–113.
- [KPQC] *Korean Post-Quantum Cryptography Competition (KPQC)*. <https://www.kpqc.or.kr/>. Accessed: 2025-04-20. Korean Professional Qualification Center.
- [Lee+10] M. Lee et al. “Countermeasures against Power Analysis Attacks for the NTRU Public Key Cryptosystem”. In: *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* 93-A.1 (2010), pp. 153–163.
- [Ngo+21b] K. Ngo et al. “A side-channel attack on a masked IND-CCA secure saber KEM implementation”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2021), pp. 676–707.
- [NTG25] D. Nabokov, X. Tong, and Q. Guo. “Enhancing Key-Recovery Chosen-Ciphertext Side-Channel Attacks on NTRU Using LDPC”. In: (*to appear*) *2025 IEEE 43rd International Conference on Computer Design (ICCD)*. 2025.
- [NTRU+] *NTRU+*. <https://sites.google.com/view/ntruplus>. Accessed: 2025-04-20. 2025.
- [OpenSSH] OpenSSH. *OpenSSH Release 9.0*. <https://www.openssh.com/txt/release-9.0>. Accessed: 2024-10-28. 2024.
- [Rav+20b] P. Ravi et al. “Generic side-channel attacks on CCA-secure lattice-based PKE and KEMs”. In: *IACR transactions on cryptographic hardware and embedded systems* (2020), pp. 307–335.
- [Rav+22a] P. Ravi et al. “Will You Cross the Threshold for Me? Generic Side-Channel Assisted Chosen-Ciphertext Attacks on NTRU-based KEMs”. In: *IACR TCHES* 2022.1 (2022), pp. 722–761.
- [Reg09] O. Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In: *Journal of the ACM (JACM)* 56.6 (2009), pp. 1–40.
- [Sch+22c] T. Schamberger et al. “A power side-channel attack on the Reed-Muller Reed-Solomon version of the HQC cryptosystem”. In: *International Conference on Post-Quantum Cryptography*. Springer. 2022, pp. 327–352.
- [SS11] D. Stehlé and R. Steinfeld. “Making NTRU as Secure as Worst-Case Problems over Ideal Lattices”. In: *EUROCRYPT 2011*. Ed. by K. G. Paterson. Vol. 6632. LNCS. Springer, Berlin, Heidelberg, May 2011, pp. 27–47.

- [SW07] J. H. Silverman and W. Whyte. “Timing Attacks on NTRUEncrypt Via Variation in the Number of Hash Calls”. In: *CT-RSA 2007*. Ed. by M. Abe. Vol. 4377. LNCS. Springer, Berlin, Heidelberg, Feb. 2007, pp. 208–224.
- [Tan+23] Y. Tanaka et al. “Multiple-Valued Plaintext-Checking Side-Channel Attacks on Post-Quantum KEMs”. In: *IACR TCHES 2023.3* (2023), pp. 473–503.
- [Uen+22a] R. Ueno et al. “Curse of Re-encryption: A Generic Power/EM Analysis on Post-Quantum KEMs”. In: *IACR TCHES 2022.1* (2022), pp. 296–322.
- [wolfSSL] wolfSSL. *wolfSSL Products*. <https://www.wolfssl.com/products/wolfssl/>. Accessed: 2024-10-28.



# Single-Trace Key Recovery Attacks on HQC Using Valid and Invalid Ciphertexts

---

## Abstract

As the Hamming Quasi-Cyclic (HQC) cryptosystem was recently selected by NIST for standardization, a thorough evaluation of its implementation security is critical before its widespread deployment. This paper presents single-trace side-channel attacks that recover the full long-term secret key of HQC, experimentally evaluated on a protected Cortex-M4 implementation. We introduce two distinct attacks that significantly advance the state of the art: a passive attack that uniquely models key recovery as a moderate-density parity-check (MDPC) decoding problem from a single valid ciphertext, and an active chosen-ciphertext attack employing a new probing strategy on a linear combination of secret key components for significantly improved efficiency. Both attacks are enabled by a new information set decoding (ISD) variant that exploits soft side-channel information, a contribution of broader importance to code-based cryptography. Our experiments show that a single trace suffices for full key recovery under realistic conditions, effectively defeating countermeasures such as codeword masking for the first time. We also show that several existing defenses are ineffective against the new attacks.

## 1 Introduction

The impending arrival of large-scale quantum computers poses a significant threat to our current public-key cryptographic infrastructure, which largely relies on the

---

presumed hardness of integer factorization and the discrete logarithm problem. In response, the U.S. National Institute of Standards and Technology (NIST) initiated a comprehensive process to solicit, evaluate, and standardize quantum-resistant cryptographic algorithms. This multi-year effort has recently culminated in the selection of several schemes, including the Hamming Quasi-Cyclic (HQC) cryptosystem [Gab+25], a code-based Key Encapsulation Mechanism (KEM), which is designated for standardization.

As HQC transitions from a candidate to a global standard, the focus of the cryptographic community intensifies on its practical security. Beyond its theoretical foundations, a scheme’s resilience in real-world deployments hinges on its resistance to implementation attacks. Side-channel analysis, which exploits physical leakages like power consumption or electromagnetic emissions, represents one of the most potent threats to cryptographic hardware and software. Therefore, a thorough and continuous evaluation of HQC’s implementation security is not only critical for identifying potential vulnerabilities but also for developing robust and effective countermeasures before its widespread adoption.

**Motivation.** The implementation security of HQC has become a vibrant and fast-paced area of research with numerous side-channel and fault-injection attacks published, for instance [Waf+19; PT19; B  +19; GJN20a; Xag+21; Sch+22b; Guo+22a; Uen+22a; Hua+23a; GLG22b; Guo+23b; Tan+23; Pai+25; Goy+24; SGG24; B  +25; Gas+25a; Wei+25; DG25c; DG25b; Mai+25]. These attacks exploit a wide array of physical leakages—including timing, cache-timing, power, and electromagnetic (EM) emissions—as well as complex microarchitectural behaviors. Methodologically, these attacks range from the direct exploitation of physical leakages to the construction of powerful mathematical oracles, such as plaintext-checking (PC) or decryption-failure (DF) oracles that leverage the decapsulation behavior of the cryptosystem.

The study of HQC’s implementation security can be divided into two phases, mirroring the evolution of the scheme itself. The initial versions [Agu+17] of HQC used a concatenation of repetition and BCH codes. Following the demonstration of a decryption failure attack [GJ20] in 2020, the proposal transitioned to a more efficient concatenated coding scheme of Reed-Muller and Reed-Solomon codes. For this newer version, the attack landscape has evolved rapidly; the number of required decapsulation queries, or traces, has plummeted from hundreds of thousands [Guo+22a] in 2022 to as few as two in recent attacks [Pai+25; DG25b].

Among the spectrum of side-channel threats, single-trace attacks are particularly devastating. Their ability to succeed with a single observation makes them an ideal tool for a practical adversary: they are exceptionally difficult to detect and thwart, bypassing countermeasures based on failure monitoring or rate limiting. However, the true impact of such an attack depends on its target. While recent research has demonstrated the feasibility of single-trace attacks for recovering the ephemeral shared key [Goy+24], the recovery of the long-term secret key repre-

sents a far more catastrophic failure. A compromised long-term key allows an adversary to impersonate users, decrypt future communications, and potentially decrypt all previously recorded traffic, leading to a total and retroactive security collapse.

Despite the recognized severity of single-trace attacks, their application to long-term key recovery in HQC remains an open problem. State-of-the-art practical attacks targeting the long-term secret key require a minimum of two side-channel traces [Pai+25; DG25b]. Other single-trace investigations have been limited to simulations without experimental validation [Bai+25] or required a relatively high number of traces (83) on a real-world platform [Mai+25]. To close this critical gap, we propose two distinct single-trace attacks that achieve full secret key recovery on a Cortex-M4 platform. The first is a stealthy passive attack using a single valid ciphertext, while the second is an active chosen-ciphertext attack designed to succeed with significantly less leaked information than previously required.

## 1.1 Contributions

We present the first single-trace side-channel attacks that can achieve full secret key recovery on a protected implementation of the HQC KEM. Our attacks are built on three technical contributions:

- We introduce a novel passive side-channel attack (VC-KRA-MDPC) that models the key recovery process as a moderate-density parity-check (MDPC) decoding problem. This reframing significantly improves the information-theoretic efficiency of the attack, making it possible to recover the full secret key from a single side-channel trace of a valid HQC decapsulation.
- We design a new chosen-ciphertext attack (OT-FDA-CK) that enhances the capabilities of Full-Decryption (FD) attacks [DG25b]. The attack employs a novel probing strategy that extracts information about a linear combination of secret key components. This method maximizes the information gained from each oracle query, enabling a single-trace attack on hqc-1.
- We develop a new information set decoding (ISD) variant specifically tailored to process soft information derived from side-channel leakages. This algorithm serves as the efficient recovery backend for both of our attacks, converting probabilistic data into the concrete secret key. The technique is general and may hold independent interest for other applications in side-channel analysis of code-based cryptosystems.

We demonstrate the practical viability of these contributions using power leakages measured from an HQC implementation with codeword masking on a Cortex-M4 microcontroller. Our experiments show that a single trace suffices for full key

recovery on this target. Both attacks are inherently immune to simple countermeasures such as failure-detection mechanisms or ciphertext sanity checks, highlighting the limitations of current defenses.

## 1.2 New Techniques

Our single-trace attacks rely on three novel techniques that significantly improve the efficiency and practicality of side-channel analysis against HQC.

**Key recovery as MDPC decoding.** For our passive attack (VC-KRA-MDPC), we introduce a new conceptual framework that models the key recovery as a decoding problem for a MDPC code. We construct a parity check matrix from the public HQC parameters and define a codeword that concatenates the secret key components,  $x$  and  $y$ , with a corresponding syndrome vector. The noisy side-channel leakage obtained from the `expand_and_sum` operation provides a soft-information estimate of this syndrome [Pai+25]. We then apply an iterative belief propagation (BP) decoder, a well-studied area for MDPC codes [Mis+13; GJS16; Ara+22], initializing it with priors based on the known sparsity of the secret key and the channel probabilities from the side-channel measurements. Crucially, the presence of many short cycles in the code’s Tanner graph means the BP decoder’s soft outputs are not accurate posterior probabilities. While the ranking of bit reliabilities remains highly informative, this imperfection is a primary reason why the attack requires a larger amount of initial side-channel information to succeed. This reframing is nonetheless substantially more information-theoretically efficient than prior methods, reducing the required side-channel information to a level where a single trace is sufficient for a full key recovery.

**Combined probing for active attacks.** For our active chosen-ciphertext attack (OT-FDA-CK), we develop a more efficient probing strategy that improves upon the state-of-the-art FD oracle attack [DG25b]. Instead of probing the secret key components  $x$  and  $y$  in separate queries, our method crafts a single invalid ciphertext that manipulates the decapsulation process to depend on a linear combination of both. By carefully selecting the vectors  $r_1$ ,  $r_2$ , and  $e$ , we create a perturbation of the form  $X^l x + X^k y$  (where by default,  $l = k = 0$ ), allowing us to extract information about both secret components simultaneously from the FD oracle’s output. This combined probing approach extracts more information per query than the previous method in [DG25b], eliminating the need for a second trace.

**ISD with soft information.** The final step in both of our attacks relies on a new ISD algorithm. It replaces random permutation with a targeted, probabilistic approach, using reliability scores to guide information set selection and employing weighted sampling to prioritize bit positions most likely to be zero. This specialized

**Table 1:** Key-recovery side-channel attacks for hqc-1 (Cortex-M4 results).

| Attack                             | Ciphertext Type | Oracle Type                      | Info (bits)             | Traces |
|------------------------------------|-----------------|----------------------------------|-------------------------|--------|
| Maillet et al., CRYPTO’25 [Mai+25] | Valid           | $v - u \cdot y$ (expand_and_sum) | $\sim 17\,000$          | 83     |
| Paiva et al., PQCrypto’25 [Pai+25] | Valid           | $v - u \cdot y$ (expand_and_sum) | $\sim 7\,000$ per trace | 2      |
| VC-KRA-MDPC (Ours)                 | Valid           | $v - u \cdot y$ (expand_and_sum) | $\sim 1\,000$           | 1      |
| Dong & Guo, CHES’25 [DG25b]        | Invalid         | Full-Decryption (FD)             | $\sim 128$ per query    | 2      |
| OT-FDA-CK (Ours)                   | Invalid         | Full-Decryption (FD)             | $\sim 191$              | 1      |

ISD is crucial for efficiently converting the noisy beliefs—whether they are the approximate scores from BP or the more precise ones from the FD oracle—into the final secret key. This method sees a more natural fit in code-based cryptography; in contrast, exploiting such soft information in lattice-based schemes, for instance by combining BP with lattice reduction as explored in [Her+23; Her+25], remains a more complex problem that warrants further investigation.

### 1.3 Comparison with Related Works

The field of implementation security for HQC has seen a rapid acceleration of results, summarized in Table 1. Our work significantly advances this state-of-the-art. In the domain of passive attacks using valid ciphertexts, previous works like Maillet et al. [Mai+25] and Paiva et al. [Pai+25] also target the `expand_and_sum` leakage. However, the algebraic approach of Maillet et al. requires the recovery of a full intermediate vector  $v - u \cdot y$  (over 17 000 bits), leading to a practical requirement of **83 traces** on a Cortex-M4 platform. The method of Paiva et al. is more efficient but still requires **two traces**. In contrast, our method using MDPC decoding is information-theoretically superior, achieving a full key recovery from a **single trace** with only about 1 000 bits of information. This represents a more than 25-fold gain in efficiency over prior work in some high-noise scenarios<sup>1</sup>.

For active attacks using invalid ciphertexts, we improve upon the **two-trace** FD attack [DG25b], where secret components are probed separately. Our novel combined probing strategy targets a linear combination of both components, extracting more information per query and enabling a full key recovery with just a **single trace**. Crucially, we are the first to demonstrate that these single-trace key-recovery attacks are effective against implementations protected by codeword masking. These contributions collectively establish our methods as the most efficient and robust side-channel attacks against HQC to date.

<sup>1</sup>In an earlier version of their paper [Pai+25] (<https://eprint.iacr.org/archive/2023/1626/1697742959.pdf>), Paiva et al. reported they needed about 25 traces on average when the measured bit error rate was 0.35. Our method still succeeds with one trace.

## 1.4 Organization

The remainder of this paper is organized as follows. Section 2 provides background on the HQC KEM. We then present our two new attacks: VC-KRA-MDPC, a passive method using valid ciphertexts (Section 3), and OT-FDA-CK, an active chosen-ciphertext approach (Section 4). We describe the soft-information ISD post-processing in Section 5 and validate our attacks experimentally in Section 6. Finally, we discuss implications in Section 7 and conclude in Section 8.

## 2 Preliminaries

**Mathematical notations.** Let  $n$  be a positive integer. We consider the polynomial ring  $\mathcal{R} = \mathbb{F}_2[X]/(X^n - 1)$  defined over the binary field  $\mathbb{F}_2$ , where addition and subtraction are equivalent operations. An element  $h \in \mathcal{R}$  can be interchangeably viewed as a polynomial or as its corresponding length- $n$  binary vector, denoted  $\mathbf{h}$  over  $\mathbb{F}_2$ . The number of non-zero coefficients in  $h$  is its Hamming weight ( $w_H(h)$ ).

The notation  $\leftarrow \$ \mathcal{S}$  indicates a uniform random sampling from a given set  $\mathcal{S}$ . When sampling from the ring  $\mathcal{R}$  with a fixed Hamming weight  $w$ , we denote the operation as  $\leftarrow \$ (\mathcal{R}, w)$ . For integers  $a < b$ , the set of integers from  $a$  to  $b - 1$  is represented by  $[a..b]$ . Finally, a binary random variable  $X$  following the Bernoulli distribution with parameter  $p$  is denoted by  $X \sim \text{Ber}_p$ . Such a variable assumes the value 1 with probability  $p$  and 0 with probability  $1 - p$ .

The Shannon entropy of a discrete random variable  $X$  with probability mass function  $p(x)$  is  $H(X) = -\sum_x p(x) \log_2 p(x)$ . This simplifies to the binary entropy function,  $H_b(p) = -p \log_2(p) - (1 - p) \log_2(1 - p)$ , for a variable  $X \sim \text{Ber}_p$ . The mutual information  $I(X; Y)$  between two random variables  $X$  and  $Y$  measures their statistical dependence and is defined as  $I(X; Y) = H(X) - H(X|Y)$ .

### 2.1 Hamming Quasi-Cyclic (HQC)

Hamming Quasi-Cyclic (HQC) [Gab+25] is a code-based Key Encapsulation Mechanism (KEM) undergoing standardization by NIST, with its security rooted in the hardness of the quasi-cyclic syndrome decoding problem. The HQC construction follows a conventional methodology, starting with an IND-CPA secure Public Key Encryption (PKE) scheme, denoted HQC.CPAPKE, which is then elevated to an IND-CCA secure KEM, HQC.CCAKEM. This transformation leverages a modern instantiation of the Fujisaki-Okamoto (FO) transform [FO99], specifically the transform with implicit rejection as analyzed by Hofheinz, Hövelmanns, and Kiltz [HHK17]. This section first outlines the core PKE scheme before detailing its extension to a CCA-secure KEM.

### The core PKE scheme.

The HQC.CPAPKE scheme comprises three fundamental algorithms: KeyGen, Enc, and Dec. All polynomial operations are performed in the ring  $\mathcal{R} = \mathbb{F}_2[X]/(X^n - 1)$  for a prime  $n$ . Table 2 presents the three HQC parameter sets, hqc-1, hqc-3, and hqc-5, defined in [Gab+25].

**KeyGen.** The key generation algorithm samples a random polynomial  $h \leftarrow \$ \mathcal{R}$  and two sparse polynomials  $x, y \leftarrow \$ (\mathcal{R}, \omega)$ , where  $w_H(x) = w_H(y) = \omega$ . The secret key is the pair  $sk = (x, y)$ , and the public key is  $pk = (h, s)$ , where the syndrome is computed as  $s = x + h \cdot y$ .

**Enc.** To encrypt a message  $m$ , the algorithm first deterministically generates three error vectors from a seed  $\theta$ :  $r_1, r_2 \in \mathcal{R}$  of weight  $\omega_r$ , and  $e \in \mathcal{R}$  of weight  $\omega_e$ . The message  $m$  is encoded into a codeword  $m\mathbf{G}$  using a linear code  $\mathcal{C}$ , which is realized as a concatenation of a shortened Reed-Solomon code and a duplicated Reed-Muller code. The ciphertext  $(u, v)$  is then computed as:

$$u = r_1 + h \cdot r_2 \quad \text{and} \quad v = m\mathbf{G} + \text{Truncate}(s \cdot r_2 + e, n_1 n_2) \quad (1)$$

where  $\text{Truncate}(\cdot, n_1 n_2)$  reduces the polynomial to the length of the codeword space of  $\mathcal{C}$ .

**Dec.** Upon receiving a ciphertext  $(u, v)$ , decryption involves decoding the value  $V = v - \text{Truncate}(u \cdot y, n_1 n_2)$ . By substituting the expressions for  $u, v$ , and  $s$ , we derive the input to the decoder:

$$\begin{aligned} V &= m\mathbf{G} + \text{Truncate}(s \cdot r_2 + e - u \cdot y, n_1 n_2) \\ &= m\mathbf{G} + \text{Truncate}((x + h \cdot y) \cdot r_2 + e - (r_1 + h \cdot r_2) \cdot y, n_1 n_2) \\ &= m\mathbf{G} + \text{Truncate}(x \cdot r_2 - r_1 \cdot y + e, n_1 n_2) \end{aligned}$$

The result is the original codeword perturbed by an effective error vector  $e' = \text{Truncate}(x \cdot r_2 - r_1 \cdot y + e, n_1 n_2)$ . If  $w_H(e')$  is within the error-correction capacity of  $\mathcal{C}$ , the decoder recovers  $m$  successfully. Otherwise, a decryption failure occurs. Hereafter, the  $\text{Truncate}(\cdot, n_1 n_2)$  notation will be omitted for simplicity where the context is unambiguous.

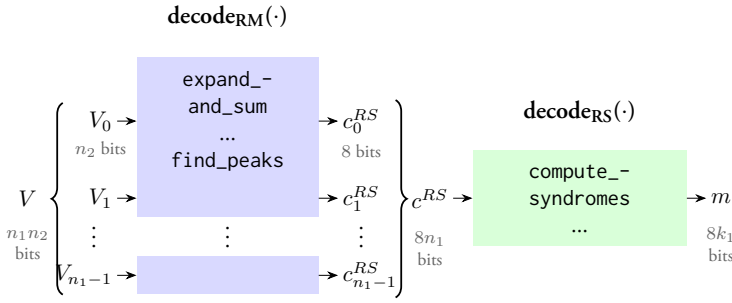
### Concatenated coding scheme in HQC.

The error-correcting code  $\mathcal{C}$  used in HQC is a two-layer concatenated code. The structure consists of an outer Reed-Solomon code and an inner duplicated Reed-Muller code.

The message  $m \in \mathbb{F}_{2^s}^{k_1}$  is first encoded by the outer code, which is a shortened Reed-Solomon (RS) code with parameters  $[n_1, k_1, d_{\text{RS}}]$  over the field  $\mathbb{F}_{2^s}$ . This

**Table 2:** HQC parameter sets. The inner code is a duplicated Reed-Muller code derived from a base  $[128, 8, 64]$  Reed-Muller code. Specifically, 3 duplications yield the  $[384, 8, 192]$  code, while 5 duplications produce the  $[640, 8, 320]$  code.

| Instance | RS-S  |       |          | Duplicated RM |          | $n$   | $\omega$ | $\omega_r = \omega_e$ |
|----------|-------|-------|----------|---------------|----------|-------|----------|-----------------------|
|          | $n_1$ | $k_1$ | $d_{RS}$ | $n_2$         | $d_{RM}$ |       |          |                       |
| hqc-1    | 46    | 16    | 31       | 384           | 192      | 17669 | 66       | 75                    |
| hqc-3    | 56    | 24    | 33       | 640           | 320      | 35851 | 100      | 114                   |
| hqc-5    | 90    | 32    | 59       | 640           | 320      | 57637 | 131      | 149                   |



**Figure 1:** Decoding of the concatenated Reed-Muller and Reed-Solomon codes.

produces an intermediate codeword  $m_1 \in \mathbb{F}_{2^8}^{n_1}$ . Each of the  $n_1$  bytes of this codeword is then individually encoded by the inner code. The inner code is a duplicated Reed-Muller (RM) code, derived from a base  $[128, 8, 64]$  RM code. Depending on the parameter set, this base code is duplicated 3 or 5 times to yield a  $[384, 8, 192]$  or  $[640, 8, 320]$  code, respectively.

The final codeword  $m\mathbf{G}$  is formed by concatenating the  $n_1$  binary output blocks from the inner encoder, resulting in a vector of length  $n_1 n_2$ . The decoding process, illustrated in Figure 1, reverses this procedure: the received vector is split into  $n_1$  blocks, each is decoded by the inner RM decoder to recover a byte, and the resulting sequence of bytes is then decoded by the outer RS decoder to recover the original message  $m$ .

### HQC Key Encapsulation Mechanism (KEM).

The construction of the IND-CCA secure KEM, referred to as HQC.CCAKEM, is detailed in Figure 2. This enhanced KEM variant builds upon HQC’s public-key encryption (PKE) core by integrating a suite of hash functions— $\mathcal{H}$ ,  $\mathcal{G}$ , and  $\mathcal{J}$ —to ensure security against active attackers.

A key feature of this construction is the de-randomization of the encryption process, which is achieved by using a seed  $\theta$ . This seed is derived by applying the hash function  $\mathcal{G}$  to a combination of the hashed public key  $\mathcal{H}(ek_{\text{KEM}})$ , a

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>Input:</b> <math>ek_{\text{KEM}}</math></p> <p><b>Output:</b> <math>K, c_{\text{KEM}} = (c_{\text{PKE}}, \text{salt})</math></p> <ol style="list-style-type: none"> <li>1: <math>m \leftarrow \mathbb{F}_2^{k_m}</math></li> <li>2: <math>\text{salt} \leftarrow \mathbb{F}_2^{128}</math></li> <li>3: <math>(K, \theta) \leftarrow \mathcal{G}(\mathcal{H}(ek_{\text{KEM}}) \parallel m \parallel \text{salt})</math></li> <li>4: <math>c_{\text{PKE}} \leftarrow \text{HQC.CPAPKE.Encrypt}(ek_{\text{KEM}}, m, \theta)</math></li> <li>5: <math>c_{\text{KEM}} \leftarrow (c_{\text{PKE}}, \text{salt})</math></li> </ol> <p>(a) Encaps</p> | <p><b>Input:</b> <math>dk_{\text{KEM}}, c_{\text{KEM}} = (c_{\text{PKE}}, \text{salt})</math></p> <p><b>Output:</b> <math>K'</math></p> <ol style="list-style-type: none"> <li>1: <math>(ek_{\text{KEM}}, dk_{\text{PKE}}, \sigma, \_) \leftarrow dk_{\text{KEM}} \triangleright \text{Parse key}</math></li> <li>2: <math>m' \leftarrow \text{HQC.CPAPKE.Decrypt}(dk_{\text{PKE}}, c_{\text{PKE}})</math></li> <li>3: <math>(K', \theta') \leftarrow \mathcal{G}(\mathcal{H}(ek_{\text{KEM}}) \parallel m' \parallel \text{salt})</math></li> <li>4: <math>c'_{\text{PKE}} \leftarrow \text{HQC.CPAPKE.Encrypt}(ek_{\text{KEM}}, m', \theta')</math></li> <li>5: <math>c'_{\text{KEM}} \leftarrow (c'_{\text{PKE}}, \text{salt})</math></li> <li>6: <b>if</b> <math>c_{\text{KEM}} \neq c'_{\text{KEM}}</math> <b>or</b> <math>m' = \perp</math> <b>then</b></li> <li>7:     <math>K' \leftarrow \mathcal{J}(\mathcal{H}(ek_{\text{KEM}}) \parallel \sigma \parallel c_{\text{KEM}})</math></li> <li>8: <b>end if</b></li> </ol> <p>(b) Decaps</p> |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Figure 2: HQC.CCAKEM.

randomly generated message  $m$ , and a randomly chosen 128-bit salt. During the decapsulation stage, a crucial re-encryption step is performed to verify the integrity and authenticity of the received ciphertext. If this check fails, or if the initial decryption is unsuccessful, a separate hash function  $\mathcal{J}$  is used to generate a pseudorandom key, preventing any information leakage to a potential attacker.

## 2.2 Codeword Masking Countermeasures

A prominent countermeasure against side-channel attacks on the decoding process is codeword masking [MSS13], a technique also recommended for protecting HQC in [Mai+25]. The core principle is to mask the sensitive data processed by the decoder. This is achieved by first generating a random message mask,  $m'$ , and encoding it into a corresponding codeword mask,  $c'$ . The decoder then processes the sum of the original ciphertext and this codeword mask. Due to the linearity of the code, the output is the original message masked by  $m'$ , i.e.,  $m + m'$ . The final plaintext,  $m$ , is retrieved by simply subtracting the random mask  $m'$ .

## 3 VC-KRA-MDPC: Single-Trace Key Recovery Attack with Valid Ciphertexts via MDPC Decoding

We present VC-KRA-MDPC, a novel single-trace attack that improves the side-channel attack of [Pai+25] on valid HQC ciphertexts targeting the `expand_and_sum` leakage. The original attack requires more information than can be obtained from a single trace. Our method overcomes this limitation by modeling the key-recovery problem as a moderate-density parity-check (MDPC) decoding problem. This approach makes the key recovery significantly more efficient and feasible with a single trace.

### 3.1 Threat Model

Our threat model considers a passive adversary aiming to recover the long-term secret key  $(x, y)$  of an HQC implementation. The attack is designed to be stealthy,

```

1 void expand_and_sum(rm_expanded_cdw *dest, rm_codeword_t src[]) {
2     // start with the first copy
3     for (int32_t part = 0; part < 4; part++) {
4         for (int32_t bit = 0; bit < 32; bit++) {
5             (*dest)[part * 32 + bit] = src[0].u32[part]>>bit&1;}}
6     // sum the rest of the copies
7     for (int32_t copy = 1; copy < MULTIPLICITY; copy++) {
8         for (int32_t part = 0; part < 4; part++) {
9             for (int32_t bit = 0; bit < 32; bit++) {
10                (*dest)[part * 32 + bit] += src[copy].u32[part]>>bit
                &1;}}}}

```

Listing 1: expand\_and\_sum

requiring side-channel traces from a valid decapsulation.

The attacker operates under a standard gray-box, profiled model. This assumes physical access for power or EM monitoring and an offline profiling phase on a similar device to create leakage templates. The adversary has full knowledge of the HQC algorithm, public key, implementation details, and the intermediate random vectors  $r_1$ ,  $r_2$ , and  $e$ . This is a practical assumption, as the adversary generates the ciphertext themselves. For example, Alice could recover Bob’s key by measuring a single power trace of his device’s decapsulation process.

### 3.2 Baseline Attack

Paiva et al. [Pai+25] exploit the leakage of the `expand_and_sum` operation during the Reed-Muller decoding process, where codeword in each Reed-Muller block is processed bit by bit. Let  $V_i \in \mathbb{F}_2^{n_2}$  be the input vector to the  $i$ -th Reed-Muller decoder block for each  $i$  in  $[0..n_1]$  as shown in Figure 1, representing a possibly corrupted codeword from the repeated Reed-Muller code used in HQC. Define the multiplicity as  $M$  and the block length  $n_2 = 128M$ .

As noted by Paiva et al. [Pai+25], the `expand_and_sum` operation shown in Listing 1 processes the input vector  $V_i$  (denoted as `src[]` in the listing) in two distinct steps to produce the output integer vector  $a_i \in \mathbb{Z}^{128}$  (via the pointer `dest`). In the *initialization step*, the algorithm processes only the first 128-bit repetition within the input block  $V_i$ . Each bit from this initial repetition is extracted and its integer value (0 or 1) is assigned as the initial value for the corresponding element  $a_i[j]$  in the output array. Following the initialization, the *accumulation step* processes the remaining  $M - 1$  repetitions within  $V_i$ . For each of these subsequent repetitions, the algorithm again extracts the bit at each position  $j$ . The integer value of this extracted bit (0 or 1) is then added to the existing value stored in the corresponding element  $a_i[j]$ .

Paiva et al. [Pai+25] use profiled side-channel attacks focusing on the power and EM leakages generated by these bit-wise manipulations. They are able to classify the value of each bit being processed, leading to the recovery of a noisy

version of the intermediate codeword  $V_i$ , denoted  $z_{V_i}$ , with a bit error rate  $\rho_1$ . The channel can be modeled as a binary symmetric channel (BSC) with crossover probability  $\rho_1$ , representing the bit error rate:  $P(z_{V_i}[j] \neq V_i[j]) = \rho_1$ .

Their post-processing method requires aggregating information to construct likelihood vectors, which are then used to heuristically guide an algebraic solver. In their experiments, two traces are needed for full key recovery.

### 3.3 Key Recovery as an MDPC Decoding Problem with BP

Our core contribution is to model the HQC key recovery task as a decoding problem for a moderate-density parity-check (MDPC) code. This allows the application of iterative belief propagation (BP) algorithms to output soft information about the key bits from noisy side-channel observations.

MDPC codes [Mis+13] are a class of linear codes defined by a parity-check matrix that is denser than that of traditional low-density parity-check (LDPC) codes. While early proposals for MDPC-based cryptosystems often focused on low-complexity, hard-decision decoding algorithms like Gallager’s bit-flipping [Gal63], their structure is also amenable to soft-decision BP decoders. Unlike sparse LDPC matrices, the parity-check matrix of an MDPC code typically has row and column weights proportional to the square root of the code length.

#### Parity check matrix and codeword definition.

We first construct the parity-check matrix  $\mathbf{H}$  for our code. Given  $\text{vec} \in \mathbb{F}_2^n$ , let  $\text{cir}(\text{vec})$  denote the circulant matrix induced by  $\text{vec}$ . For the valid-ciphertext attack, the matrix  $\mathbf{H}$  is formed by concatenating the circulant matrices derived from  $r_1$  and  $r_2$ , two vectors known to the attacker. The final parity-check matrix  $\mathbf{P}$  for our decoding problem is defined by appending the  $n \times n$  identity matrix  $\mathbf{I}_n$  to  $\mathbf{H}$ :

$$\mathbf{P} = [\mathbf{H}|\mathbf{I}_n] = [\text{cir}(r_2)|\text{cir}(r_1)|\mathbf{I}_n]. \quad (2)$$

Any codeword  $\hat{c} \in \mathbb{F}_2^{3n}$  of this code must satisfy the parity-check equation  $\mathbf{P}\hat{c}^T = \mathbf{0}^T \pmod{2}$ . We structure this codeword  $\hat{c}$  to contain the secret key components and a corresponding syndrome vector,  $\text{syn}$ :

$$\hat{c} = [x|y|\text{syn}]. \quad (3)$$

Substituting this structure into the parity-check equation directly yields the definition of the syndrome vector:  $\text{syn} = [x|y]\mathbf{H}^T \pmod{2}$ .

This framework allows us to use a noisy observation of the syndrome,  $z_{\text{syn}}$ , to recover soft information about the secret key components  $x$  and  $y$ . We employ the sum-product BP algorithm for this decoding task.

For hqc-1, the weights of  $r_1$  and  $r_2$  (75 each) result in an MDPC code with a row weight of 150 in its  $\mathbf{H}$  matrix. Additionally, due to HQC’s truncation, only the first  $n_1n_2$  rows of  $\mathbf{P}$ , corresponding to the observed portion of the syndrome,

receive initial channel information during decoding. Henceforth, all remaining analysis considers this truncated  $n_1 n_2 \times 3n$  parity-check matrix.

### LLR initialization.

The BP algorithm operates on log-likelihood ratios (LLRs), defined for a binary variable  $b$  given an observation  $z$  as  $L(b|z) = \ln \frac{P(b=0|z)}{P(b=1|z)}$ . We initialize the LLR for each bit of the codeword  $\hat{c} = [x|y|\text{syn}]$  as follows.

**Prior LLR for secret bits.** For the first  $2n$  positions corresponding to the secret key bits of  $[x|y]$ , the prior probability of being 1 is  $p_1 = \frac{\omega}{n}$  and being 0 is  $p_0 = 1 - p_1$ . The prior LLR for each secret bit position is thus:

$$L_{\text{prior}}(i) = \ln \frac{p_0}{p_1} = \ln \left( \frac{n - \omega}{\omega} \right). \quad (4)$$

**Channel LLR for syndrome bits.** For the last  $n$  bits corresponding to the syndrome, the initial LLR is determined by the noisy observations of each syndrome bit,  $z_{\text{syn}_i}$  for  $i \in [0..n]$ , obtained from the side-channel oracle (modeled as a BSC with bit error rate  $\rho_1$ ). The channel LLR for each syndrome bit is calculated as:

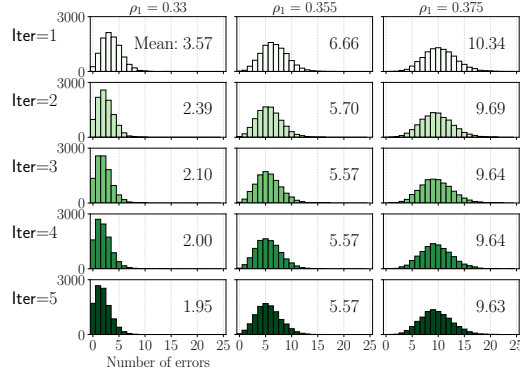
$$L_{\text{channel}}(i) = \ln \frac{P(z_{\text{syn}_i} | \text{syn}_i = 0)}{P(z_{\text{syn}_i} | \text{syn}_i = 1)} = (-1)^{z_{\text{syn}_i}} \ln \left( \frac{1 - \rho_1}{\rho_1} \right). \quad (5)$$

These prior and channel LLRs are concatenated to form the initial LLR vector  $L_{\text{init}} \in \mathbb{R}^{3n}$ .

### Iterative BP and bits ranking.

The sum-product BP algorithm is executed on the Tanner graph representation [Tan81] of the parity check matrix  $\mathbf{P}$ . The algorithm takes  $\mathbf{P}$  and  $L_{\text{init}}$  as input. It iteratively passes messages between variable and check nodes for a fixed number of iterations,  $\text{Iter}$ . This process refines the belief for each bit position in the codeword  $\hat{c}$ , ultimately yielding a final normalized posterior LLR vector,  $L_{\text{final}} \in \mathbb{R}^{3n}$  (details in Appendix Appendix B). The first  $2n$  elements of  $L_{\text{final}}$ , which contain the posterior beliefs for the secret key bits  $x$  and  $y$ , serve as the soft-information input to our new ISD variant for the final key recovery in Section 5.

We rank the secret key bit indices  $i \in [0..2n]$  based on their values in  $L_{\text{final}}$  from large to small. Indices with the largest values are considered most likely positions for zeros. We calculate the number of errors (i.e., actual ones) in the first  $n$  positions of the reordered key vector. To understand the performance of the BP algorithm, this statistic is calculated for various bit error rates  $\rho_1$  and the number of iterations  $\text{Iter}$ . Figure 3 presents the distribution of this statistic for three error rate levels and  $\text{Iter}$  up to 5. The results reveal three key insights: first,



**Figure 3:** Frequency plot of the number of errors in the first  $n$  positions of the reordered vector for hqc-1. Each plot is generated with 10 000 random key pairs.

even a single iteration is remarkably effective. Second, while further iterations yield improvements, the gains diminish rapidly, with performance largely stabilizing after three iterations. Third, this improvement with increased iteration is more pronounced for lower bit error rates. Consequently, we use the soft information after four iterations for the ISD algorithm.

## 4 OT-FDA-CK: Chosen-Ciphertext Attack with Invalid Ciphertexts Using Combined Key Components

We present OT-FDA-CK, a novel side-channel attack capable of recovering the long-term secret key of HQC. With a Full-Decryption (FD) oracle, the attack achieves full key recovery with a single chosen-ciphertext query, enabling a single-trace attack on the Cortex-M4 platform. Our work introduces a more efficient probing method targeting a combination of secret key components, advancing the OT-PCA framework [DG25c] and improving upon the two-call FD oracle attack [DG25b].

### 4.1 Threat Model

The adversary’s objective is to recover the long-term secret key,  $(x, y)$ , through an active chosen-ciphertext attack. In this scenario, the adversary submits a carefully crafted, invalid ciphertext to a device performing HQC decapsulation and measures the resulting physical side-channel leakages, such as power consumption or electromagnetic (EM) radiation.

The attack relies on the construction of a Full Decryption (FD) oracle, denoted  $\mathcal{O}_{\text{FD}}$ , from side-channel leakage. Unlike traditional binary PC and DF oracles—

versatile tools effective against many post-quantum schemes and instantiated from diverse leakages such as timing, power, or cache behavior—the FD oracle, firstly proposed for HQC in [DG25b], goes further. By exploiting the HQC decoder’s parallel architecture, it simultaneously reveals all  $n_1$  inner Reed-Muller decoding outputs in a single query. To model its behavior in a noisy environment, we define an inaccuracy parameter,  $\rho_2$ , as the probability that each Reed-Muller decoding from the FD oracle yields an incorrect output. The power of the FD oracle is the key element that allows the adversary to succeed with a single trace.

To build the FD oracle, our attack assumes a standard profiled side-channel model. For our experimental validation on the Cortex-M4, we presume the adversary uses a similar, or even identical, “clone” device to build the required leakage templates. We note, however, that this is a conservative assumption and not a fundamental requirement. As demonstrated in attacks on lattice-based cryptography [Ngo+21a; Guo+23b], the clone-device assumption can often be relaxed, with attacks remaining potent in cross-device settings. Given that the side-channel leakage for constructing an FD oracle for HQC [DG25b] is significantly stronger than for schemes like ML-KEM [Guo+23b], the feasibility of such cross-device attacks is further reinforced.

## 4.2 State of the Art: Probing Key Components Individually

The OT-PCA framework [DG25c; DG25b] uses the publicly available HQC decoder to construct offline templates (OT) capable of extracting soft information about the secret key components,  $x$  and  $y$ , separately. The central idea is that the secret key acts as a sparse perturbation to a carefully chosen error vector  $e$ . The resulting ciphertext, when processed by the decoder, produces different outputs (e.g., “success” or “failure” in the PC oracle case [DG25c]) that are correlated with the secret key’s structure.

The framework comprises of an offline phase and an online attack phase. During the offline phase, error patterns are optimized to maximize the information gain from each decoder query via a genetic-style algorithm. Then, OTs are built for each optimized pattern. The process involves generating a large number of low-weight vectors sampled according to the secret key’s distribution and categorizing them based on their decoding outcomes. Within each category, the empirical conditional probability of a secret bit position being one is estimated. The templates essentially map each possible decoder output to the probability of a secret bit being one at a given position [DG25c; DG25b].

During the actual attack, the adversary crafts ciphertexts by combining the optimized error vector  $e$  and strategically selected polynomials for  $r_1$  and  $r_2$  to probe the secret key parts  $x$  and  $y$  separately. After obtaining the decoder’s output, the attacker consults the pre-computed OTs to gather information regarding key bit positions’ likelihood of being one, which facilitates the final post-processing step.

Following the OT-PCA framework [DG25c], we assume the adversary, during the actual attack, has access to pre-computed OTs and optimized error vectors or has constructed them beforehand. Generating these resources is a one-time low-cost pre-computation that falls outside the scope of the actual attack phase, as they relate to the HQC construction itself rather than a specific device or key.

### 4.3 New Attack Methodology: Probing Combined Key Components

We propose an enhanced OT-PCA variant that improves attack efficiency by simultaneously targeting both secret key components,  $x$  and  $y$ , within a single decoding query. This is achieved by choosing special forms of  $r_1$  and  $r_2$  that result in a ciphertext with both  $x$  and  $y$  components. Specifically, the attacker sets  $r_1 = X^k$  and  $r_2 = X^l$ , with  $X^k$  and  $X^l$  polynomials with a Hamming weight of one. According to Equation (1), component  $u$  becomes  $u = X^k + h \cdot X^l$ , and  $v$  becomes  $v = m\mathbf{G} + s \cdot X^l + e$ . The input to the decoder,  $v - u \cdot y$ , is:

$$\begin{aligned} v - u \cdot y &= (m\mathbf{G} + s \cdot X^l + e) - (X^k + h \cdot X^l)y \\ &= m\mathbf{G} + e - X^k \cdot y + X^l(s - h \cdot y). \end{aligned} \quad (6)$$

As  $s = x + h \cdot y$ , the above can be written as:

$$v - u \cdot y = m\mathbf{G} + e - X^k \cdot y + X^l \cdot x. \quad (7)$$

Equation (7) shows the key innovation of our method: the ciphertext is influenced by both secret key parts,  $x$  and  $y$ . Specifically,  $x$  contributes through the shift  $l$  ( $X^l \cdot x$ ), while  $y$  contributes through the shift  $k$  ( $X^k \cdot y$ ). In contrast, previous approaches [SGG24; DG25c; DG25b] set either  $r_1$  or  $r_2$  to 0 to include only one key component in the ciphertext, forcing each component to be probed individually.

With an FD oracle defined in [DG25b] that outputs  $n_1$  RM decoding results simultaneously, our attack proceeds by setting the shift parameters  $k$  and  $l$  to 0. The ciphertext in Equation (7) then simplifies to  $v - u \cdot y = m\mathbf{G} + e - y + x$ , where the perturbation to  $e$  is the combined secret  $x + y$ . We configure  $e$  by placing the same length- $n_2$  optimized error pattern  $\epsilon$  in each of the  $n_1$  inner RM blocks.

Since the inner RM decoders act on disjoint length- $n_2$  blocks of the input, their operations are independent; and the probability distribution for any block of  $x + y$  is statistically the same. Consequently, our entire offline analysis can be simplified to focusing on a single, representative RM block.

#### Offline phase.

The offline phase aims to identify optimal error patterns  $\epsilon$  and to construct an OT for each. Both require querying the RM decoder with input  $\epsilon + \xi$ , where the

---

**Algorithm 1** FD attack with combined key components (online phase)

---

**Input:**  $h, s, m$  ( $:= 0$ ),  $salt$ , scheme parameters  $(n_1, n_2, \mathbf{G})$   
 $\mathcal{E} = \{\epsilon_1, \dots, \epsilon_t\}, \mathcal{T}_{\epsilon, m}$  for each  $\epsilon \in \mathcal{E}$

**Output:** Reliability score vector  $R$

```

1:  $R_\ell \leftarrow 1.0 \quad \forall \ell \in [0..n_1 n_2]$ 
2: for each  $\epsilon \in \mathcal{E}$  do
3:    $e \leftarrow (\epsilon, \epsilon, \dots, \epsilon)$   $\triangleright n_1$  blocks, each with length  $n_2$ 
4:    $r_1 \leftarrow X^0, r_2 \leftarrow X^0, u \leftarrow X^0 + h \cdot X^0$ 
5:    $c \leftarrow (u, m\mathbf{G} + s \cdot X^0 + e, salt)$ 
6:   Query oracle:  $(o_0, \dots, o_{n_1-1}) \leftarrow \mathcal{O}_{\text{FD}}^{\rho_2}(c)$ 
7:   for  $k \in [0..n_1]$  do  $\triangleright$  for each of the  $n_1$  parallel outputs
8:     for  $j \in [0..n_2]$  do
9:        $P \leftarrow \Pr(\xi_j = 1 \mid o_k)$  from  $\mathcal{T}_{\epsilon, m}$ 
10:       $\ell \leftarrow j + k \cdot n_2$   $\triangleright$  map position  $j$  in block  $k$  to global index  $\ell$ 
11:       $R_\ell \leftarrow R_\ell \cdot P$ 
12:    end for
13:  end for
14: end for
15: return  $R$ 
```

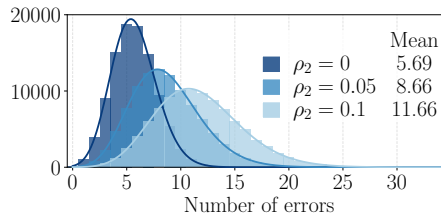
---

vector  $\xi$  models a length- $n_2$  block of the combined secret  $x + y$ . We generate  $\xi$  by summing two independent samples,  $\xi_x$  and  $\xi_y$ , each drawn according to the distribution of a secret key block.

To find an optimal  $\epsilon$ , we use a genetic algorithm with random starts similar to that in [DG25b], with the goal of maximizing the output entropy of the RM decoder. This iterative algorithm generates mutations of the current best pattern by flipping one or two bits. Each mutation is evaluated by repeatedly querying the decoder to compute the empirical entropy of its outputs, given by  $H = -\sum_{d \in \text{DO}} p_d \log_2(p_d)$ , where  $p_d$  is the empirical probability of decoding output  $d$  in the set of all obtained decoding outputs DO. The mutation yielding the highest entropy is selected to seed the next generation.

The resulting best error pattern for hqc-1 achieves an entropy of 4.15 bits. This increases the information extraction per FD oracle query to approximately 191 bits ( $46 \times 4.15$ ), a significant improvement over the 128 bits from the previous method in [DG25b], which targets key components individually.

Following the OT-PCA framework [DG25c], we then construct one template  $\mathcal{T}_{\epsilon, m}$  for each optimized error pattern  $\epsilon$  ( $m = 0$  by default). To do this, we generate 100 million random  $\xi$  vectors, group them by their RM decoding outputs, and discard any groups with fewer than 50 000 samples. We get 37 groups for the best error pattern for hqc-1. For each group, we calculate the empirical probability of being one for each position in the length- $n_2$  block. The  $\mathcal{T}_{\epsilon, m}$  maps each RM decoding output to a vector of conditional probabilities, where each entry represents the likelihood of a one at that bit position in one block of  $x + y$ .



**Figure 4:** Frequency plot of the number of errors in the first  $n$  positions of the reordered vector for different levels of oracle error probability  $\rho_2$  from hqc-1. Each level is drawn with 100 000 random key pairs.

### Online phase.

The online phase, shown in Algorithm 1, builds upon the FD attack in [DG25b]. Our attack’s core innovation is its ciphertext construction: setting specific values for  $r_1$  and  $r_2$  to probe the combination  $x + y$  in a single query.

For each selected error pattern  $\epsilon$ , a ciphertext is constructed and the FD oracle is queried; with a perfect oracle, a single  $\epsilon$  (and thus a single query) suffices. The algorithm then uses the oracle’s outputs to look up conditional probabilities in the pre-computed OTs. These probabilities are cumulatively multiplied into a reliability score vector,  $R$ , which contains the soft information on  $x + y$ . Crucially, the reliability score for each bit position of the combined secret  $x + y$  is then assigned to the corresponding bit positions in both  $x$  and  $y$ . The scores are used to rank all secret key bit positions by their likelihood of being zero. This ranking serves as the input for the post-processing step described in Section 5. Figure 4 shows the distribution of the number of errors in the first  $n$  positions of the reordered secret key vector given different levels of oracle error probability  $\rho_2$ .

## 5 Post-Processing with a New ISD Variant

We next describe a new post-processing method to recover the secret vector with information set decoding (ISD). ISD algorithms are a fundamental tool for solving the decoding problem for random linear codes, a problem central to code-based cryptography. The foundational concept was introduced by Prange [Pra62], with Stern’s subsequent work [Ste88] establishing the highly influential collision-based, meet-in-the-middle search strategy. This paradigm has been the basis for numerous modern improvements [MMT11; Bec+12; MO15] and has recently been complemented by distinct approaches such as sieving-based ISD [GJN24; Duc+24].

Moving to the context of side-channel analysis, the integration of soft information into ISD is critical. While there have been attempts to incorporate reliability information into Prange-style decoders, the substantially better performance of Stern’s algorithm makes it the natural foundation for such work. Pessl and Man-

gard [PM16] pioneered this by adapting Stern’s algorithm to use bit reliability, employing a column-swapping strategy analyzed with Markov chains. Building on this direction, we propose a new variant that also leverages soft information but through a different mechanism: weighted sampling within a constrained set of the most zero-like bit positions. We adopt this approach for its analytical simplicity, as the independence between trials facilitates a straightforward performance model, which we validate through simulation.

More advanced soft-decision decoders, such as SoftStern [Guo+17; Guo+19], also exist. These are one-pass algorithms that do not re-sample the information set; they allow for enumerating the most probable error patterns for the collision search and offer flexibility in memory usage. Combining their list generation with our iterative sampling is a promising direction for future work. Such a hybrid approach would require a dedicated evaluation, particularly to analyze the performance trade-offs under different memory constraints.

**Problem formulation.** The post-processing task is to solve for the low-weight secret key  $(x, y)$  given the public key and the soft information obtained from the previous phase. The core of this problem is the syndrome decoding equation derived from the public key, which can be expressed in matrix form as:

$$\begin{bmatrix} \mathbf{I}_n & \text{cir}(h) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = s, \quad (8)$$

where  $\text{cir}(h)$  is the circulant matrix generated from the public polynomial  $h$ .

To solve this system, we adapt the general ISD framework of Finiasz and Sendrier [FS09]. This framework operates by selecting a subset of  $n + \ell$  column positions (the information set) and transforming the corresponding submatrix of the full parity-check matrix into a quasi-standard form via Gaussian elimination:

$$\tilde{\mathbf{H}} := \begin{pmatrix} \mathbf{Q}' & \mathbf{0} \\ \mathbf{Q}'' & \mathbf{I}_{n-\ell} \end{pmatrix}, \quad (9)$$

where  $\mathbf{Q}'$  is an  $\ell \times (n + \ell)$  matrix. The key input to our algorithm is a vector of reliability scores for the secret key  $(x, y)$ , where each score indicates the likelihood of the corresponding bit being one.<sup>2</sup> Our primary contribution is to leverage this soft information to guide the ISD process. Instead of randomly selecting an information set, our method first identifies a small candidate pool containing only the most reliable (likely zero) positions. The  $n + \ell$  positions for the information set are then chosen from this constrained pool using a weighted sampling scheme.

---

<sup>2</sup>If only soft information for the  $2n_1n_2$  truncated entries is available, the remaining positions are assigned the a priori probability  $\omega/n$ .

## 5.1 A Modified ISD with Soft Information

Our algorithm proceeds in multiple independent trials. In each trial, an information set is first selected and processed to set up a collision search.

### Information set selection.

We begin by forming a candidate array of the  $\phi \cdot n$  secret bit positions most likely to be zero ( $1 < \phi < 2$ ), from which we will select the  $n + \ell$  positions for the information set. This array is partitioned into two disjoint sets of equal size. To introduce randomness during partitioning, the array is first segmented into contiguous, non-overlapping blocks of four elements with each block containing an outer pair (bit indices 0 and 3) and an inner pair (indices 1 and 2). Then, one pair is randomly assigned to a first set and the other to a second set. This procedure is applied to all blocks, creating two disjoint sets of size  $\frac{\phi \cdot n}{2}$  that form a complete partition of the candidate array. For each key, this partition is created only once and reused for all subsequent trials.

Each trial of the algorithm begins with a weighted sampling on the two sets, drawing  $\frac{n+\ell}{2}$  bit positions from the first to form subset  $\mathcal{S}_1$ , and  $\frac{n+\ell}{2}$  positions from the second to form subset  $\mathcal{S}_2$ . Together, these  $n + \ell$  positions form the information set. The weighted sampling is to make sure that bits more likely to be zero are more prone to be selected. For OT-FDA-CK, we use the inverse of the conditional probability of being one for each bit position as the weight, which is directly obtained from the online attack phase. For VC-KRA-MDPC, we observe that the probabilities of being one, inferred from the final normalized posterior LLRs in  $L_{\text{final}}$ , exhibit low variance, making direct use of these values akin to sampling from a uniform distribution. To amplify the underlying reliability ranking and make use of the good variation of  $L_{\text{final}}$ , we define the weight for bit position  $i$  using the transformation  $(1.1 \cdot \max_j (L_{\text{final}}(j)) - L_{\text{final}}(i))^{-1}$ .

The corresponding submatrix of the full parity-check matrix is then transformed via Gaussian elimination to yield a system with a partial syndrome of length  $\ell$ . For an  $n \times 2n$  parity-check matrix, the cost of this Gaussian elimination, denoted as  $C_{\text{Gauss}}$ , is  $O(n^3)$ . The concrete cost of a schoolbook implementation can be estimated as  $2^{43}$ ,  $2^{46}$ , and  $2^{48}$  for hqc-1, hqc-3, and hqc-5 respectively. The resulting partial syndrome becomes the target for the subsequent step.

### Collision search.

The search for a solution proceeds using a standard meet-in-the-middle approach to find a combination of two low-weight error patterns, one from each subset ( $\mathcal{S}_1$  or  $\mathcal{S}_2$ ), that solves the partial syndrome equation. Let the target weight in each subset be a small integer  $w_0$ .

First, a list  $L_1$  is constructed by enumerating all possible error patterns of weight bounded to  $w_0$  with support restricted to the positions in  $\mathcal{S}_1$ . For each

pattern, the corresponding  $\ell$ -bit partial syndrome is computed and stored, typically in a hash table for efficient lookup. Subsequently, we enumerate all error patterns of weight bounded to  $w_0$  with support in  $\mathcal{S}_2$ . For each of these patterns, its partial syndrome  $s_2$  is computed. We then check if the value  $s_{\text{tar}} - s_2$  exists in the hash table for  $L_1$ , where  $s_{\text{tar}}$  is the target partial syndrome from the Gaussian elimination step.

A successful collision identifies a candidate solution with a total weight of  $\leq 2w_0$  within the information set. This candidate must then be verified against the full syndrome decoding problem. For our chosen parameters (e.g.,  $\ell = 51$  for hqc-1), the partial syndrome is sufficiently long to ensure that the cost of handling random collisions and verifying them is negligible compared to the cost of list generation and search, assuming an  $O(1)$  memory access cost.

### Success condition and complexity.

A trial succeeds if it finds a low-weight key that passes all parity checks. Due to the design of HQC, this solution is guaranteed to be the unique secret key. In our simulations, since we know the true key, we can directly verify this success condition: a trial succeeds if the true secret key's partition across subsets  $\mathcal{S}_1$  and  $\mathcal{S}_2$  has a weight of at most  $w_0$  in each. The overall algorithm succeeds if any of its  $T$  independent trials succeeds.

The complexity of a single trial is dominated by the Gaussian elimination, list generation, and collision search. With a target error weight of  $w_0$  in each partition, the cost can be approximated as:

$$C_{\text{trial}} \approx C_{\text{Gauss}} + 2 \cdot \binom{\frac{n+\ell}{2}}{w_0} \cdot w_0 \cdot \ell.$$

Here,  $C_{\text{Gauss}}$  is the cost of the initial Gaussian elimination. The second term represents the cost of the meet-in-the-middle search, where we generate two lists of size  $\binom{(n+\ell)/2}{w_0}$  and the computation of each partial syndrome has a complexity proportional to  $w_0\ell$ .

The list size,  $\binom{(n+\ell)/2}{w_0}$ , determines the memory requirement. For hqc-1, where  $n = 17\,669$  and we set  $\ell = 51$ , choosing  $w_0 = 3$  leads to a list size of approximately  $\binom{8860}{3} \approx 2^{36.5}$ . Storing one of these lists may require hundreds of gigabytes of RAM, for an efficient implementation. The time complexity for this trial is dominated by the list operations, estimated at  $2 \cdot 2^{36.5} \cdot w_0\ell \approx 2^{45}$  operations, which is slightly higher than the initial  $C_{\text{Gauss}} \approx 2^{43}$ . For higher security levels, a smaller  $w_0$  (e.g.,  $w_0 = 2$ ) may be necessary, or algorithms such as SoftStern [Guo+17; Guo+19] that allow adjusting the list size in a fine-grained manner.

The total complexity of the attack is therefore  $T \cdot C_{\text{trial}}$ . The number of trials an adversary can perform is ultimately limited by their computational budget. For

example, with a total post-processing budget of  $2^{60}$  operations, an adversary could execute approximately  $T = 2^{60}/C_{\text{trial}} \approx 2^{15}$  trials.

## 5.2 Simulation Results

### Parameter choice and simulated success rate.

The algorithm's performance depends on two key parameters: the candidate pool size factor,  $\phi$ , and the number of trials,  $T$ . The parameter  $\phi$  governs a tradeoff between the number of errors in the candidate array and the sampling flexibility.

A small  $\phi$  yields a candidate array nearly the same size as the  $n + \ell$  positions to be sampled. While this pool contains the fewest total errors, it offers little sampling flexibility as the number of errors in the final subsets ( $\mathcal{S}_1$  and  $\mathcal{S}_2$ ) is nearly determined by the number of errors present in this constrained pool. Conversely, a larger  $\phi$  introduces a vast number of additional zero positions into the pool albeit at the cost of also including some lower-ranked errors. It provides the crucial flexibility for the sampling process to “replace” an error from the top-ranked portion with a zero from the extended section. Ultimately, this flexibility can increase the probability of drawing two subsets with fewer errors, which is especially important when oracle accuracy is lower.

Figure 5 plots the simulated success rates of OT-FDA-CK with a perfect oracle and 10 000 random keys for two choices of  $\phi$ , 1.1 versus 1.2, against the number of trials  $T$  up to  $2^{15}$ . A significant advantage for the smaller  $\phi$  value is evident when  $T$  is small. This performance gap narrows as the number of trials grows. In all remaining simulations, we set  $\phi = 1.2$  by default.

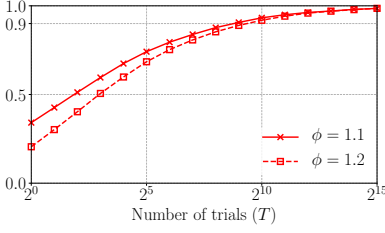
### Modeling the success rate.

Our goal is to find the minimum  $T$  required to achieve a desired success rate (e.g., 95%). For a given key, as each trial is independent, the probability of success after  $T$  trials,  $P_{\text{succ}}$ , can be modeled by the one-shot success probability,  $P_{\text{os}}$ , as  $P_{\text{succ}} = 1 - (1 - P_{\text{os}})^T$ .

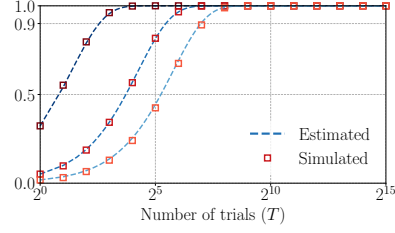
We first validate this theoretical model. To obtain an empirical estimate of  $P_{\text{os}}$  for a given key, we perform 10 000 independent tests, each with  $T$  bounded by  $2^{15}$ . The value for  $P_{\text{os}}$  is calculated as the proportion of tests that succeed on the very first trial. Figure 6 shows this validation for three distinct random keys. In all three cases, the simulation success rates align closely with their corresponding curves from theoretical estimation, confirming the model's accuracy.

### Analysis of the one-shot success probability.

While the model holds for a single key,  $P_{\text{os}}$  varies across different keys. To predict the average success rate at a given  $T$ , we analyze the distribution of  $P_{\text{os}}$  using 1 000 random keys. For each key,  $P_{\text{os}}$  is calculated as the success rate over 100 000 single-trial tests ( $T = 1$ ).



**Figure 5:** Simulated success rates with different  $\phi$ .  $x$ -axis in log scale.



**Figure 6:** Estimated vs. simulated success rates for three distinct keys.  $x$ -axis in log scale.

Figure 7(a) shows the histogram of  $P_{os}$  for VC-KRA-MDPC with  $\rho_1 = 0.355$ . The distribution is highly polarized. The BP decoder effectively solves “easy” key instances, pushing their success probability to nearly one within one trial. However, it has little impact on “harder” instances, which remain clustered at a low success probability. This creates a large gap in the distribution, with very few keys having a  $P_{os}$  between 0.6 and 1.0.

Figure 7(c) shows the histogram of  $P_{os}$  for OT-FDA-CK with a perfect oracle. In contrast, without the BP step, the data points are predominantly clustered at low success probabilities, and the remaining values spread continuously across the range. For both attacks, a guaranteed failure ( $P_{os} = 0$ ) is very rare, accounting for only 0.4% and 0.7% of keys, respectively.

### Estimating expected success rate.

With the empirical distribution of  $P_{os}$ , we can estimate the expected success rate  $P_{succ}$ , which is given by the expectation of the success function:  $P_{succ} = \mathbb{E}[f(P_{os})]$ , where  $f(p) = 1 - (1 - p)^T$ .

We perform a bootstrap estimation to project the evolution of the expected success rates and its 95% confidence interval for  $T$  up to  $2^{20}$ . Figure 7(b) and Figure 7(d) show the estimated curves for the two attacks respectively. The estimation predicts that about  $T = 2^{11}$  trials are required to achieve a 95% average success rate for VC-KRA-MDPC with  $\rho_1 = 0.355$  and OT-FDA-CK with a perfect oracle. As shown in the figures, the estimation curves for these two specific configurations align closely with the success rates in simulation, validating the model.

We can quantify our VC-KRA-MDPC attack from an information-theoretic perspective. Modeling the power leakage for each bit from `expand_and_sum` as a BSC channel with crossover probability  $\rho_1$ , a single trace provides  $n_1 n_2 (1 - H_b(\rho_1))$  bits of information. For hqc-1, our error rate of  $\rho_1 = 0.355$  yields approximately 1 090 bits. In contrast, the experiments in [Pai+25] report a lower error rate of 0.14—yielding roughly 7 300 bits per trace—yet their method requires two traces. This demonstrates our MDPC model’s superior efficiency.

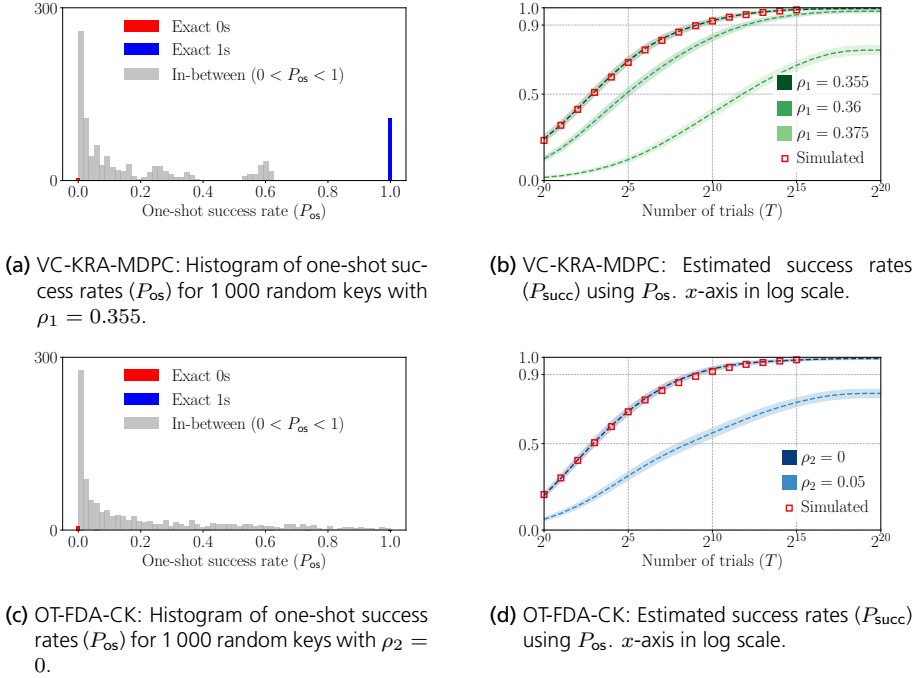


Figure 7: Post-processing success rates for the two single-trace attacks on hqc-1.

## 6 Validations

This section presents a practical, real-world validation of the single-trace attacks proposed in this paper. We evaluate the effectiveness of our attacks on a recent HQC implementation protected with codeword masking. Our evaluation follows a standard profiled side-channel attack methodology: first, we characterize the device’s leakages and build templates on a clone device; subsequently, we use these templates in an attack phase to recover the secret key from a single power-consumption trace captured from the target.

Our experiments are based on the ChipWhisperer Husky toolkit, including the CW308 UFO board and the CW308T-STM32F405 target board with a 32-bit ARM Cortex-M4 CPU, operating at a frequency of 25 MHz. Power traces are sampled at 200 MHz. The Husky’s hardware limits trace acquisition to approximately 131 000 samples per capture. To analyze longer functions, we performed multiple captures with identical inputs, adjusting the measurement offset for each. This sequence of captures is treated as a single trace, as no new information is learned beyond the extended observation window.

We target the most recent reference implementation of HQC [Gab+25] (commit from 26 Aug, 2025) for the hqc-1 parameter set. The source code was compiled using the arm-none-eabi-gcc compiler with the -O3 optimization level.

## 6.1 Targeting Codeword Masking with Valid Ciphertexts

### Bypassing codeword masking.

The codeword masking relies on the linearity of the underlying codes. A random message mask  $m'$  is generated, encoded into a codeword mask  $c'$ , and added to the decoder input. If an attacker can recover  $m'$ , the countermeasure is trivially defeated.

We recover  $m'$  by targeting the Reed-Solomon encoder, as shown in its simplified form in Listing 2. The authors of [Goy+24] demonstrated that the Galois field multiplication, `gf_mul`, leaks significant information about its first operand. In the reference HQC implementation, this first operand (`gate`) is directly derived from the secret message mask  $m'$ , while the second operand is a known constant.

Following a standard profiling procedure, we collected 10 000 traces of the Reed-Solomon encoder for random messages. Each trace is split into  $k_1$  segments according to the outer loop; each segment contains  $d_{\text{RS}}$  multiplications that share the same first operand. We use Fisher’s linear discriminant analysis (LDA) to train a classifier that predicts the byte value of this first operand.

As input to the classifier, we use a reduced trace obtained via a standard  $t$ -test: for each bit  $b_i$  of the `gate` variable, we compute a list of points of interest (PoI)  $\text{lst}_i$ . We then form the union  $\bigcup_i \text{lst}_i$  of these bit-wise PoI lists and select the leftmost and rightmost points  $a$  and  $b$  from that union; the reduced trace is built from PoI equal to  $[a..(b+1)]$ . This procedure yields a single, compact window that covers all bit-specific informative samples while remaining suitable as an input to LDA. We train  $k_1$  separate models in total, one per segment. For `hqc-1` this produces 310 000 traces per model, which we split 90%/10% into training and validation sets. The classifiers achieve an average accuracy of 0.9997 across all models for a *single* multiplication. Since each input is processed by  $d_{\text{RS}}$  independent multiplications, aggregating the per-multiplication predictions reduces the overall error probability to a negligible value. Finally, the mask  $m'$  can be recovered from the `gate` values using Gaussian elimination.

These results indicate that codeword masking does not provide meaningful protection. Even for a relatively modest single-multiplication success rate 0.765, the probability to fully recover  $m'$  with simple majority voting reaches 0.99. Therefore, the single-multiplication accuracy 0.9389 reported in [Goy+24] (the discrepancy with our result is attributable to differences in platform and measurement setup) is already sufficient to remove the mask with overwhelming probability.

### Obtaining the leakage for VC-KRA-MDPC.

With the mask  $m'$  recovered, we can compute the corresponding codeword mask  $c' = m' \mathbf{G}$  and subtract it, thereby exposing the original value  $c = m \mathbf{G} + x \cdot r_2 + y \cdot r_1 + e$ . The goal of VC-KRA-MDPC is to obtain a noisy observation of this value,  $\tilde{c} \approx c + c'$ , from which we can approximate the syndrome  $\text{syn} = x \cdot r_2 + y \cdot r_1$ .

```

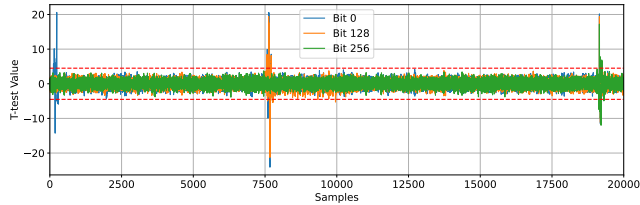
1 void reed_solomon_encode(uint8_t *cdw, const uint8_t *msg) {
2     memset(cdw, 0, PARAM_N1);
3     for (i = 0; i < PARAM_K; ++i) {
4         gate = msg[PARAM_K - 1 - i] ^ cwd[PARAM_N1 - PARAM_K - 1];
5         for (j = 0; j < PARAM_G; ++j) {
6             tmp[j] = gf_mul(gate, PARAM_RS_POLY[j]);
7         }...}...}

```

Listing 2: reed\_solomon\_encode (simplified)

Table 3: Average accuracy of recovering bits from expand\_and\_sum for hqc-1.

| Bit Range | 0–127  | 128–255 | 256–383 |
|-----------|--------|---------|---------|
| Accuracy  | 0.9698 | 0.8144  | 0.7159  |



**Figure 8:** The  $t$ -test value for output bit 0 of `expand_and_sum`. The latter part of the function is truncated as the  $t$ -test value is within red lines until the end of the function.

This leakage is sourced from the `expand_and_sum` function (shown in Listing 1) within the Reed-Muller decoder. The function’s structure reveals that each output bit in the dest array depends on  $M$  distinct bits from the input src array, spaced 128 bits apart. This dependency is visualized in the  $t$ -test results shown in Figure 8, where the leakage related to the processing of bits 0, 128, and 256 for the first output bit is clearly visible.

Inspired by this structure, we built templates to recover the  $M$ -bit tuples that contribute to each output bit simultaneously, rather than recovering the codeword bit-by-bit. For each of 128 tuples, we trained an LDA model on 10 000 traces of the first call to `expand_and_sum` during decapsulation. The resulting bit-wise recovery accuracies are detailed in Table 3. As expected, the accuracy slowly decreases for later bits as there are fewer points in the trace leaking information about them.

### Attack results.

To evaluate the success rate of the full single-trace attack, we performed 100 independent trials. In each trial, a new key pair and a valid ciphertext were generated.

```

1 void compute_syndromes(uint16_t *syndromes, uint8_t *cdw) {
2     for (size_t i = 0; i < 2 * PARAM_DELTA; ++i) {
3         for (size_t j = 1; j < PARAM_N1; ++j) {
4             syndromes[i] ^= gf_mul(cdw[j], alpha_ij_pow[i][j - 1]);
5             syndromes[i] ^= cdw[0];
6         }
7     }
8 }

```

Listing 3: compute\_syndromes

From a single power trace of the decapsulation, we first executed our attack on the Reed-Solomon encoder to recover the mask  $m'$ , which was successful in all 100 trials. We then used our models to recover the masked noisy codeword  $\tilde{c}$  from  $n_1$  expand\_and\_sum functions. From this, we computed the Log-Likelihood Ratios (LLRs) for the syndrome vector, incorporating the bit-wise probabilities from our models.

This soft information was fed into the Belief Propagation (BP) decoder. For all 100 keys, after a single iteration of BP, there were zero errors remaining in the first  $n$  positions of the reordered key vector. A final Gaussian elimination step was sufficient to recover the full secret key, achieving a 100% success rate. This result confirms the practical viability of the VC-KRA-MDPC attack, even against a codeword-masked implementation.

## 6.2 Invalid Ciphertext Case

We validated the OT-FDA-CK attack against the HQC implementation protected by codeword masking. The countermeasure was bypassed by first recovering the random mask message  $m'$ , as detailed in Section 6.1, and then constructing a FD oracle for the masked Reed-Muller outputs. The linearity of the Reed-Muller code allows us to subtract the known mask from the oracle's output to obtain the true decoded values.

The oracle targets the gf\_mul operation within the compute\_syndromes function (Listing 3). The first operand to gf\_mul corresponds to the output of an inner Reed-Muller decoder. This byte leaks significantly, enabling near-perfect recovery. For hqc-1, each Reed-Muller output is used in 30 separate multiplications, providing ample leakage to create an oracle with no observed errors.

Our experiments on the Cortex-M4 platform confirm that we can perfectly recover the last 45 Reed-Muller decoding outputs, thus constructing a perfect *partial* FD oracle. This strong leakage source was previously identified in [DG25b]. To assess its impact, we simulated the attack for hqc-1 over 10 000 random keys. With a 45-output oracle, we achieved a 90% key recovery rate after  $2^{10}$  ISD trials and 98% after  $2^{15}$  trials. This is only marginally less effective than an attack using a full 46-output oracle, with the performance gap diminishing as the number of trials increases (see Figure 10 in the appendix for a direct comparison).

## 7 Discussion

### 7.1 On Attack Methods

**Information-theoretical analysis.** From an information-theoretic perspective, our proposed attacks demonstrate a significant leap in efficiency compared to previous works, requiring substantially fewer bits of information from the side channels to achieve full key recovery. This reduction is a key enabler for the success of our single-trace attacks.

For attacks using *valid ciphertexts*, the contrast is stark. The method by Maillet et al. [Mai+25] is the most demanding, requiring the recovery of  $u \cdot y$ , on the order of  $n$  bits, which is  $\sim 17\,000$  bits for hqc-1. This high threshold explains the practical need for 83 traces in their experiment on the Cortex M4 platform. The attack by Paiva et al. [Pai+25] is more efficient but still requires approximately 7 000 bits per trace, totaling  $\sim 14\,000$  bits for their two-trace attack. Our passive attack, VC-KRA-MDPC, radically lowers this requirement. By modeling the problem as MDPC decoding, we need only about 1 000 bits of information, making a single-trace recovery feasible.

In the realm of CCA, our active attack (OT-FDA-CK) is also superior. The original FD-oracle attack [DG25b] required two oracle calls for hqc-1, with each revealing around 128 bits, meaning the side channel had to leak a minimum of 256 bits. In contrast, our improved OT-FDA-CK attack creates an FD oracle with an output entropy of approximately 191 bits. This indicates that our OT-FDA-CK method is likely the most efficient for scenarios involving protected implementations or high-noise environments where physical side-channel leakage is limited.

**Combining leakage sources.** The invalid ciphertext setting grants an attacker significantly more freedom, enabling a single-trace attack that is more powerful than either of our standalone attacks (VC-KRA-MDPC and OT-FDA-CK). While both attacks model the key recovery as a decoding problem on the vector  $e' = e + r_1 \cdot y + r_2 \cdot x$  where  $e$ ,  $r_1$ , and  $r_2$  are known, the CCA adversary setting provides a critical advantage: the attacker can choose  $r_1$  and  $r_2$  to be extremely sparse (e.g., weight 1, 2, or 3).

This choice allows for a combination of leakages. Specifically, combining the noisy information from the `expand_and_sum` function provides a better estimation of  $e'$  than what the FD oracle provides alone, which in turn frames a fundamentally easier decoding problem. Also, the ability to select sparse  $r_1$  and  $r_2$  allows us to construct a significantly sparser LDPC code for post-processing. This is a marked improvement over the MDPC code derived from the denser, randomly generated vectors in the valid ciphertext attack. The transition to a sparser code improves the performance of BP algorithms, making the attack using combined leakage more robust to noise. Consequently, the attack is ideal for analyzing well-protected implementations or high-noise environments.

**Alternative methods.** In developing our OT-FDA-CK attack, we explored alternative probing strategies, such as increasing the Hamming weight of the intermediate polynomials  $r_1$  and  $r_2$ . The hypothesis was that a larger, denser perturbation in the decoder’s input vector ( $v - u \cdot y = m\mathbf{G} + e + \sum_j X^{l_j}x - \sum_i X^{k_i}y$ ) would increase the chance of decoding to a faraway codeword. This, in turn, would increase the entropy of the Reed-Muller decoding output, thus yielding more information from the FD oracle.

However, our experiments showed that this method did not improve upon the version presented and, in fact, performed worse. A possible explanation is that an LDPC decoding problem with a denser parity-check matrix demands substantially more information to be effectively resolved.

Interestingly, we did observe that this method could be used to construct a more powerful Multi-Value Plaintext-Checking (MV-PC) oracle-based attack than the one presented in [DG25b]. A detailed analysis of this improved MV-PC attack is beyond the scope of this paper and we leave its exploration for future research.

## 7.2 Countermeasures

The powerful single-trace attacks presented in this paper render many existing countermeasure strategies ineffective, highlighting an urgent need for more robust defenses. We analyze the shortcomings of two primary approaches: low-cost detection schemes and masking.

**Low-cost, detection-based countermeasures.** Low-cost countermeasures typically rely on detecting anomalous behavior, such as failing a sanity check on the ciphertext’s structure (e.g., checking if it is too sparse or skewed) or observing an excessive number of decapsulation failures for a given key-pair. Upon detection, the device might refresh its secret key or shut down. However, these methods have known drawbacks. Besides being bypassable, as shown in the context of Kyber [Rav+24a], they can introduce new vulnerabilities. Forcing a fallback to key generation, for instance, shifts the operational focus to a process that is inherently more vulnerable and costly to secure than decapsulation itself. Furthermore, an adversary could deliberately trigger repeated failures to mount a Denial-of-Service (DoS) attack, creating an additional attack surface.

Our single-trace attacks render such strategies largely ineffective. The passive attack using a valid ciphertext (VC-KRA-MDPC) is particularly stealthy, as it does not trigger decapsulation failures, making it invisible to these detection mechanisms. While the active, oracle-based attack (OT-FDA-CK) does induce errors, the detection-based approach fails to protect past communications. For instance, if a static key is used over an extended period (e.g., 30 days), an adversary can archive the encrypted traffic and mount the attack on the final day. A successful key recovery would retroactively compromise all data exchanged during that period. Our

current chosen ciphertext selection in OT-FDA-CK also adds the public key  $h$  to  $u$  ( $s$  and  $e$  to  $v$ )—all of which are dense vectors in our setting—making simple sanity checks on ciphertext sparsity ineffective.

**Masking countermeasures.** Codeword masking has been suggested as a defense for the `expand_and_sum` operation in HQC [Mai+25]. However, our experimental results on the Cortex-M4—a crucial benchmark for embedded security—demonstrate that this countermeasure is insufficient to prevent our single-trace key recovery.

True protection against these attacks will likely require more granular, gate-level masking schemes that introduce more fresh randomness into sensitive computation. The high performance overhead of such schemes, however, remains a significant barrier. The development of efficient yet secure masking is therefore a critical direction for future research. This is especially pertinent given that even well-regarded first-order masking schemes for lattice-based cryptography have been broken with a small number of traces, even in a cross-device setting [Guo+23b]. Since the side-channel leakage from HQC appears stronger than that from Kyber, the effectiveness of any proposed masking implementation must be rigorously evaluated against FD-based attacks like ours.

While recent work on masked HQC implementations [Goy+25] is a commendable step, public and open-source libraries are essential. The community requires rigorously-vetted implementations to properly evaluate the practical security of HQC before its widespread deployment as a standard.

## 8 Concluding Remarks

In this work, we have presented significant side-channel threats to HQC, a new NIST PQC KEM standard, driven by three methodological contributions and a validation using power consumption measurements taken with a ChipWhisperer device. Our methodological advances include: 1) a novel passive attack that uniquely reframes key recovery as a MDPC decoding problem; 2) a new active attack employing an efficient chosen-ciphertext strategy to probe a linear combination of secret key components; and 3) a new ISD variant that exploits reliability information, crucial for both attacks. We have validated these methods with a Cortex-M4 platform, demonstrating for the first time that even a codeword-masked implementation can be broken with a single trace. Critically, low-cost detection-based countermeasures are ineffective to both attacks.

This work opens several avenues for future research. First, given that our attacks defeat codeword masking, a critical area for future work is the development of more efficient and secure masking schemes. A second direction is the exploration of new leakage sources on various hardware platforms, such as FPGAs, to construct powerful oracles. Since the Full Decryption (FD) oracle for HQC is a recent development, understanding its practical feasibility and variations across

different implementations is a critical next step. Finally, we highlight the need for a more rigorous, quantitative analysis of combined leakage attacks, especially when targeting state-of-the-art defenses such as the new masked implementation proposed in [Goy+25].

## Acknowledgement

This work was supported by the Swedish Research Council (grant numbers 2021-04602 and 2023-03654), the Swedish Civil Contingencies Agency (grant number 2020-11632), the Crafoord Foundation, and the Wallenberg AI, Autonomous Systems and Software Program (WASP), funded by the Knut and Alice Wallenberg Foundation. The computations and simulations were partly enabled by resources provided by LUNARC.

## References

- [Agu+17] C. Aguilar-Melchor et al. *HQC*. Tech. rep. available at <https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions>. National Institute of Standards and Technology, 2017.
- [Ara+22] N. Aragon et al. *BIKE*. Tech. rep. available at <https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions>. National Institute of Standards and Technology, 2022.
- [Bäe+19] C. Bäetu et al. “Misuse Attacks on Post-quantum Cryptosystems”. In: *EUROCRYPT 2019, Part II*. Ed. by Y. Ishai and V. Rijmen. Vol. 11477. LNCS. Springer, Cham, May 2019, pp. 747–776.
- [Baï+25] C. Baïsse et al. “Secret and shared keys recovery on hamming quasi-cyclic with SASCA”. In: *DCC 93.6 (2025)*, pp. 2137–2157.
- [Bec+12] A. Becker et al. “Decoding Random Binary Linear Codes in  $2^{n/20}$ : How  $1 + 1 = 0$  Improves Information Set Decoding”. In: *EUROCRYPT 2012*. Ed. by D. Pointcheval and T. Johansson. Vol. 7237. LNCS. Springer, Berlin, Heidelberg, Apr. 2012, pp. 520–536.
- [DG25b] H. Dong and Q. Guo. “Multi-Value Plaintext-Checking and Full-Decryption Oracle-Based Attacks on HQC from Offline Templates”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems 2025 (2025)*, pp. 254–289.

- [DG25c] H. Dong and Q. Guo. “OT-PCA: New Key-Recovery Plaintext-Checking Oracle Based Side-Channel Attacks on HQC with Offline Templates”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2025.1 (Jan. 2025). Date of Publication: 2024/12/09, pp. 251–274.
- [Duc+24] L. Ducas et al. “Asymptotics and Improvements of Sieving for Codes”. In: *EUROCRYPT 2024, Part VII*. Ed. by M. Joye and G. Leander. Vol. 14657. LNCS. Springer, Cham, May 2024, pp. 151–180.
- [FO99] E. Fujisaki and T. Okamoto. “Secure Integration of Asymmetric and Symmetric Encryption Schemes”. In: *CRYPTO’99*. Ed. by M. J. Wiener. Vol. 1666. LNCS. Springer, Berlin, Heidelberg, Aug. 1999, pp. 537–554.
- [FS09] M. Finiasz and N. Sendrier. “Security Bounds for the Design of Code-Based Cryptosystems”. In: *ASIACRYPT 2009*. Ed. by M. Matsui. Vol. 5912. LNCS. Springer, Berlin, Heidelberg, Dec. 2009, pp. 88–105.
- [Gab+25] P. Gaborit et al. *Hamming Quasi-Cyclic (HQC)*. <https://pqc-hqc.org>. Version 2025/08/22. HQC Submission Team, Aug. 2025. eprint: 2508.022.
- [Gal63] R. G. Gallager. *Low-Density Parity-Check Codes*. M.I.T. Press Research Monograph Series 21. Cambridge, MA: M.I.T. Press, 1963.
- [Gas+25a] S. Gast et al. “CounterSEVeillance: Performance-Counter Attacks on AMD SEV-SNP”. In: *NDSS 2025*. The Internet Society, Feb. 2025.
- [GJ20] Q. Guo and T. Johansson. “A New Decryption Failure Attack Against HQC”. In: *ASIACRYPT 2020, Part I*. Ed. by S. Moriai and H. Wang. Vol. 12491. LNCS. Springer, Cham, Dec. 2020, pp. 353–382.
- [GJN20a] Q. Guo, T. Johansson, and A. Nilsson. “A Key-Recovery Timing Attack on Post-quantum Primitives Using the Fujisaki-Okamoto Transformation and Its Application on FrodoKEM”. In: *CRYPTO 2020, Part II*. Ed. by D. Micciancio and T. Ristenpart. Vol. 12171. LNCS. Springer, Cham, Aug. 2020, pp. 359–386.
- [GJN24] Q. Guo, T. Johansson, and V. Nguyen. “A new sieving-style information-set decoding algorithm”. In: *IEEE Transactions on Information Theory* (2024).
- [GJS16] Q. Guo, T. Johansson, and P. Stankovski. “A Key Recovery Attack on MDPC with CCA Security Using Decoding Errors”. In: *ASIACRYPT 2016, Part I*. Ed. by J. H. Cheon and T. Takagi. Vol. 10031. LNCS. Springer, Berlin, Heidelberg, Dec. 2016, pp. 789–815.

- [GLG22b] G. Goy, A. Loiseau, and P. Gaborit. “A New Key Recovery Side-Channel Attack on HQC with Chosen Ciphertext”. In: *Post-Quantum Cryptography - 13th International Workshop, PQCrypto 2022*. Ed. by J. H. Cheon and T. Johansson. Springer, Cham, Sept. 2022, pp. 353–371.
- [Goy+24] G. Goy et al. “Single trace HQC shared key recovery with SASCA”. In: *IACR TCHES 2024.2* (2024), pp. 64–87.
- [Goy+25] G. Goy et al. *Side-Channel Sensitivity Analysis on HQC: Towards a Fully Masked Implementation*. Cryptology ePrint Archive, Paper 2025/1344. 2025.
- [Guo+17] Q. Guo et al. “Information set decoding with soft information and some cryptographic applications”. In: *2017 IEEE International Symposium on Information Theory (ISIT)*. IEEE. 2017, pp. 1793–1797.
- [Guo+19] Q. Guo et al. “Some cryptanalytic and coding-theoretic applications of a soft stern algorithm.” In: *Advances in Mathematics of Communications* 13.4 (2019).
- [Guo+22a] Q. Guo et al. “Don’t Reject This: Key-Recovery Timing Attacks Due to Rejection-Sampling in HQC and BIKE”. In: *IACR TCHES 2022.3* (2022), pp. 223–263.
- [Guo+23b] Q. Guo et al. “SCA-LDPC: A Code-Based Framework for Key-Recovery Side-Channel Attacks on Post-quantum Encryption Schemes”. In: *ASLACRYPT 2023, Part IV*. Ed. by J. Guo and R. Steinfeld. Vol. 14441. LNCS. Springer, Singapore, Dec. 2023, pp. 203–236.
- [Her+23] J. Hermelink et al. “Belief Propagation Meets Lattice Reduction: Security Estimates for Error-Tolerant Key Recovery from Decryption Errors”. In: *IACR TCHES 2023.4* (2023), pp. 287–317.
- [Her+25] J. Hermelink et al. “A Generic Framework for Side-Channel Attacks Against LWE-Based Cryptosystems”. In: *EUROCRYPT 2025, Part VIII*. Ed. by S. Fehr and P.-A. Fouque. Vol. 15608. LNCS. Springer, Cham, May 2025, pp. 3–32.
- [HHK17] D. Hofheinz, K. Hövelmanns, and E. Kiltz. “A Modular Analysis of the Fujisaki-Okamoto Transformation”. In: *TCC 2017, Part I*. Ed. by Y. Kalai and L. Reyzin. Vol. 10677. LNCS. Springer, Cham, Nov. 2017, pp. 341–371.
- [Hua+23a] S. Huang et al. “Cache-Timing Attack Against HQC”. In: *IACR TCHES 2023.3* (2023), pp. 136–163.
- [Mai+25] N. Maillet et al. “Key Recovery from Side-Channel Power Analysis Attacks on Non-SIMD HQC Decryption”. In: *CRYPTO 2025, Part V*. Ed. by Y. T. Kalai and S. F. Kamara. Vol. 16004. LNCS. Springer, Cham, Aug. 2025, pp. 70–102.

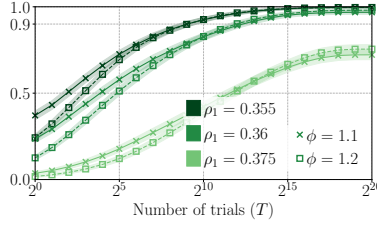
- [Mis+13] R. Misoczki et al. “MDPC-McEliece: New McEliece variants from Moderate Density Parity-Check codes”. In: *Proceedings of the 2013 IEEE International Symposium on Information Theory, Istanbul, Turkey, July 7-12, 2013*. IEEE, 2013, pp. 2069–2073.
- [MMT11] A. May, A. Meurer, and E. Thomae. “Decoding Random Linear Codes in  $\tilde{O}(2^{0.054n})$ ”. In: *ASIACRYPT 2011*. Ed. by D. H. Lee and X. Wang. Vol. 7073. LNCS. Springer, Berlin, Heidelberg, Dec. 2011, pp. 107–124.
- [MO15] A. May and I. Ozerov. “On Computing Nearest Neighbors with Applications to Decoding of Binary Linear Codes”. In: *EUROCRYPT 2015, Part I*. Ed. by E. Oswald and M. Fischlin. Vol. 9056. LNCS. Springer, Berlin, Heidelberg, Apr. 2015, pp. 203–228.
- [MSS13] D. Merli, F. Stumpf, and G. Sigl. *Protecting PUF Error Correction by Codeword Masking*. Cryptology ePrint Archive, Report 2013/334. 2013.
- [Ngo+21a] K. Ngo et al. “A Side-Channel Attack on a Masked IND-CCA Secure Saber KEM Implementation”. In: *IACR TCHES 2021.4* (2021), pp. 676–707.
- [Pai+25] T. B. Paiva et al. “Et tu, Brute? Side-Channel Assisted Chosen Ciphertext Attacks Using Valid Ciphertexts on HQC KEM”. In: *Post-Quantum Cryptography - 16th International Workshop, PQCrypto 2025, Part II*. Ed. by R. Niederhagen and M.-J. O. Saarinen. Springer, Cham, Apr. 2025, pp. 294–321.
- [PM16] P. Pessl and S. Mangard. “Enhancing Side-Channel Analysis of Binary-Field Multiplication with Bit Reliability”. In: *CT-RSA 2016*. Ed. by K. Sako. Vol. 9610. LNCS. Springer, Cham, Feb. 2016, pp. 255–270.
- [Pra62] E. Prange. “The use of information sets in decoding cyclic codes”. In: *IRE Transactions on Information Theory* IT-8 (1962), pp. 5–9.
- [PT19] T. B. Paiva and R. Terada. “A Timing Attack on the HQC Encryption Scheme”. In: *SAC 2019*. Ed. by K. G. Paterson and D. Stebila. Vol. 11959. LNCS. Springer, Cham, Aug. 2019, pp. 551–573.
- [Rav+24a] P. Ravi et al. “Defeating Low-Cost Countermeasures against Side-Channel Attacks in Lattice-based Encryption A Case Study on Crystals-Kyber”. In: *IACR TCHES 2024.2* (2024), pp. 795–818.
- [Sch+22b] T. Schamberger et al. “A Power Side-Channel Attack on the Reed-Muller Reed-Solomon Version of the HQC Cryptosystem”. In: *Post-Quantum Cryptography - 13th International Workshop, PQCrypto 2022*. Ed. by J. H. Cheon and T. Johansson. Springer, Cham, Sept. 2022, pp. 327–352.

- [SGG24] R. L. Schröder, S. Gast, and Q. Guo. “Divide and Surrender: Exploiting Variable Division Instruction Timing in HQC Key Recovery Attacks”. In: *USENIX Security 2024*. Ed. by D. Balzarotti and W. Xu. USENIX Association, Aug. 2024.
- [Ste88] J. Stern. “A method for finding codewords of small weight”. In: *International colloquium on coding theory and applications*. Springer. 1988, pp. 106–113.
- [Tan+23] Y. Tanaka et al. “Multiple-Valued Plaintext-Checking Side-Channel Attacks on Post-Quantum KEMs”. In: *IACR TCHES 2023.3* (2023), pp. 473–503.
- [Tan81] R. M. Tanner. “A Recursive Approach to Low Complexity Codes”. In: *IEEE Transactions on Information Theory* 27.5 (1981).
- [Uen+22a] R. Ueno et al. “Curse of Re-encryption: A Generic Power/EM Analysis on Post-Quantum KEMs”. In: *IACR TCHES 2022.1* (2022), pp. 296–322.
- [Waf+19] G. Wafo-Tapa et al. *A Practicable Timing Attack Against HQC and its Countermeasure*. Cryptology ePrint Archive, Report 2019/909. 2019.
- [Wei+25] H. Weissteiner et al. “TEEcorrelate: An Information-Preserving Defense against Performance-Counter Attacks on TEEs”. In: *USENIX Security 2025*. 2025.
- [Xag+21] K. Xagawa et al. “Fault-Injection Attacks Against NIST’s Post-Quantum Cryptography Round 3 KEM Candidates”. In: *ASIACRYPT 2021, Part II*. Ed. by M. Tibouchi and H. Wang. Vol. 13091. LNCS. Springer, Cham, Dec. 2021, pp. 33–61.

## Appendix A: Additional Figures

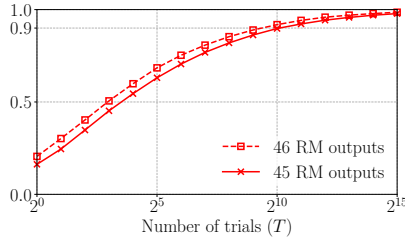
**Figure 9.** Figure 9 shows the estimated success rates obtained from simulated  $P_{os}$  values for VC-KRA-MDPC. The results align with the expectation that  $\phi = 1.1$  yields a higher success rate for a small number of trials ( $T$ ). As  $T$  increases, however, the performance with  $\phi = 1.2$  becomes comparable and can eventually be superior, especially when the bit error probability ( $\rho_1$ ) becomes higher.

**Figure 10.** Figure 10 presents a comparison of simulated success rates for the OT-FDA-CK attack, assuming a perfect Full-Decryption (FD) oracle. The simulation, conducted over 10 000 random keys, contrasts the scenario where the oracle provides all 46 Reed-Muller decoding outputs with a scenario where it provides only 45 outputs. The latter case corresponds to the experimental validation detailed in Section 6.2, where we demonstrate that a perfect oracle for the



**Figure 9:** Estimated success rates ( $P_{\text{succ}}$ ) using  $P_{\text{os}}$  via bootstrapping.  $x$ -axis in log scale.

last 45 Reed-Muller outputs can be constructed by exploiting leakage from the `compute_syndrome` function in the Reed-Solomon decoder. The results show that the missing first Reed-Muller decoding output has only a minor impact on the attack's success rate. This effect is most pronounced for a small number of trials ( $T$ ), while the performance gap between the two scenarios becomes negligible as  $T$  increases.



**Figure 10:** Impact of a partial oracle on OT-FDA-CK attack success rate. The plot compares simulated success rates using a perfect FD oracle with 46 RM outputs versus one with 45 RM outputs, averaged over 10 000 keys.  $x$ -axis in log scale.

## Appendix B: Sum-Product BP Algorithm

This section provides a detailed description of the sum-product belief propagation (BP) algorithm used for decoding in our VC-KRA-MDPC attack, as introduced in Section 3.3. We use a normalized version to stabilize convergence. The algorithm operates on the Tanner graph representation [Tan81] of the parity-check matrix  $\mathbf{P} = [\mathbf{H}|\mathbf{I}_n]$ , where  $\mathbf{H} = [\text{cir}(r_2)|\text{cir}(r_1)]$ . The goal is to compute the posterior log-likelihood ratios (LLRs) for the bits of the codeword  $\hat{c} = [x|y|\text{syn}]$  given the initial LLR vector,  $L_{\text{init}}$ .

Let  $v_i$  denote the  $i$ -th variable node (corresponding to the  $i$ -th bit of  $\hat{c}$ ) and  $c_j$  denote the  $j$ -th check node (corresponding to the  $j$ -th row of  $\mathbf{P}$ ). The algorithm iteratively passes messages between these nodes.

Let  $L_{v_i \rightarrow c_j}$  denote the message from variable node  $v_i$  to check node  $c_j$ . This message represents the LLR of the variable node  $v_i$  based on information from all connected check nodes *except*  $c_j$ . Let  $L_{c_j \rightarrow v_i}$  denote the message from check node  $c_j$  to variable node  $v_i$ . This message represents the extrinsic information about  $v_i$  provided by the parity-check equation at  $c_j$ , based on information from all other connected variable nodes. The algorithm proceeds as follows:

**1. Initialization.** For each variable node  $v_i$  and each adjacent check node  $c_j$ , the initial message is set to the LLR of the corresponding bit from  $L_{\text{init}}$ .

$$L_{v_i \rightarrow c_j}^{(0)} = L_{\text{init}}(i),$$

where  $L_{\text{init}}(i)$  is defined by Equations (4) and (5) in the main text. The messages from check nodes to variable nodes are initialized to zero, i.e.,  $L_{c_j \rightarrow v_i}^{(0)} = 0$ .

**2. Iterative updates.** For a fixed number of iterations,  $t = 1, 2, \dots, \text{Iter}$ , the following messages are computed.

**Check node to variable node update.** The message from a check node  $c_j$  to a variable node  $v_i$  is calculated based on the incoming messages from all other variable nodes  $v_{i'}$  connected to  $c_j$  (where  $i' \neq i$ ). The update rule is:

$$L_{c_j \rightarrow v_i}^{(t)} = 2 \cdot \text{atanh} \left( \prod_{i' \in \mathcal{N}(c_j) \setminus \{i\}} \tanh \left( \frac{L_{v_{i'} \rightarrow c_j}^{(t-1)}}{2} \right) \right),$$

where  $\mathcal{N}(c_j)$  is the set of indices of variable nodes connected to check node  $c_j$ . This operation effectively computes the LLR for bit  $i$  assuming the parity-check equation at  $c_j$  is satisfied.

**Variable node to check node update.** The message from a variable node  $v_i$  to a check node  $c_j$  is the sum of its initial LLR and the incoming messages from all other check nodes  $c_{j'}$  connected to  $v_i$  (where  $j' \neq j$ ).

$$L_{v_i \rightarrow c_j}^{(t)} = L_{\text{init}}(i) + \sum_{j' \in \mathcal{N}(v_i) \setminus \{j\}} L_{c_{j'} \rightarrow v_i}^{(t)},$$

where  $\mathcal{N}(v_i)$  is the set of indices of check nodes connected to variable node  $v_i$ .

**3. Normalized belief updates.** In each iteration  $t$ , after receiving messages from all connected check nodes, the posterior LLR for each variable node is updated

and normalized. This normalized belief,  $L_{\text{post}}^{(t)}(i)$ , is calculated as:

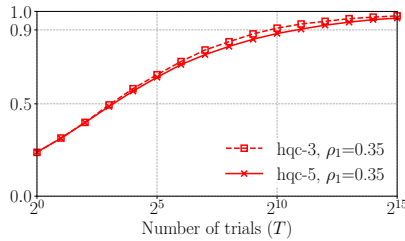
$$L_{\text{post}}^{(t)}(i) = \frac{L_{\text{init}}(i) + \sum_{j \in \mathcal{N}(v_i)} L_{c_j \rightarrow v_i}^{(t)}}{|\mathcal{N}(v_i)| + 1}.$$

The final output of the algorithm,  $L_{\text{final}}$ , is the vector of posterior LLRs from the last iteration, i.e.,  $L_{\text{final}}(i) = L_{\text{post}}^{(\text{iter})}(i)$ . This vector is then used as the soft-information input for the ISD-based post-processing step described in Section 5.

## Appendix C: Broad Applicability of the VC-KRA-MDPC Attack

Our new VC-KRA-MDPC attack has broad applicability. This attack exploits the same `expand_and_sum` leakage source identified by Maillet et al. in [Mai+25]. The authors of [Mai+25] claim their attack “targets standard Instruction Set Architectures (ARM T32, RISC-V, x86-64) and compiler optimization level” and works on both optimized (Non-SIMD) and reference implementations. Since our VC-KRA-MDPC attack targets the identical leakage source but employs a far more efficient key recovery approach, it inherits this wide-ranging applicability and significance.

As shown in Figure 11, the VC-KRA-MDPC attack remains highly effective against hqc-3 and hqc-5, achieving high success probabilities even with a significant bit error rate of  $\rho_1 = 0.35$ .



**Figure 11:** Simulated success rates for VC-KRA-MDPC on hqc-3 and hqc-5 with bit error rate  $\rho_1 = 0.35$ . Each series uses 10 000 keys and the number of trials  $T$  is bounded by  $2^{15}$ .  $w_0 = 3$ .  $x$ -axis in log scale.



# Unlocking the True Potential of Decryption Failure Oracles: A Hybrid Adaptive-LDPC Attack on ML-KEM Using Imperfect Oracles

---

## Abstract

Side-channel attacks exploiting Plaintext-Checking (PC) and Decryption Failure (DF) oracles are a pressing threat to deployed post-quantum cryptography. These oracles can be instantiated from tangible leakage sources like timing, power, and microarchitectural behaviors, making them a practical concern for leading schemes based on lattices, codes, and isogenies. In this paper, we revisit chosen-ciphertext side-channel attacks that leverage the DF oracle on ML-KEM. While DF oracles are often considered inefficient compared to their binary PC counterparts in lattice-based schemes, we demonstrate that their full potential has been largely unrealized.

We introduce a novel attack framework that combines adaptive query generation with belief propagation for Low-Density Parity-Check (LDPC) codes. Our methodology crafts carefully balanced parity checks over multiple secret coefficients, maximizing the Shannon information extracted from each oracle query, even in the presence of significant noise. This approach dramatically reduces the number of queries required for a full key recovery, achieving near-optimal

efficiency by approaching the theoretical Shannon information bound. For ML-KEM-768 with an oracle accuracy of 95%, our attack requires only 2 950 queries (a 1.35 ratio to the Shannon lower bound), establishing that a well-designed DF attack can surpass the efficiency of state-of-the-art binary PC attacks.

To validate the practical impact of our findings, we apply our framework to the recent GoFetch attack, showing significant gains in this real-world, microarchitectural side-channel scenario. Our method reduces the required measurement traces by over an order of magnitude and eliminates the need for computationally expensive post-processing, enabling a full key recovery on higher-security schemes previously considered intractable.

## 1 Introduction

The recent conclusion of the National Institute of Standards and Technology (NIST) post-quantum cryptography (PQC) standardization process has established CRYSTALS-Kyber as a primary global standard for public-key encryption and key establishment. Now formally standardized as the Module-Lattice-based Key-Encapsulation Mechanism (ML-KEM) in FIPS 203 [Nat24], its widespread deployment is imminent. This landmark decision has intensified research into its practical security, especially its resilience against side-channel attacks. Kyber’s design follows a common paradigm in modern cryptography: a public-key encryption (PKE) scheme with chosen-plaintext attack (CPA) security is first constructed, which is then transformed into a key encapsulation mechanism (KEM) with chosen-ciphertext attack (CCA) security using a variant of the Fujisaki–Okamoto (FO) transform [FO99]. This transformation, while providing theoretical security, introduces new attack surfaces at the implementation level.

Among the various side-channel vectors, oracle-based attacks represent a particularly potent threat. This powerful methodology abstracts the physical leakage of a device into a conceptual black box, or “oracle”, which reveals specific information about the internal state or outputs during cryptographic operations. By carefully crafting ciphertexts and feeding them to the decapsulation procedure, an attacker can use the oracle output to recover the secret key.

The concept of exploiting decryption outcomes in lattice-based cryptography can be traced back to early reaction attacks [HGS99] and has evolved into a sophisticated taxonomy of oracles. The most frequently discussed are the decryption failure (DF) oracle and the plaintext-checking (PC) oracle. A general PC oracle [Bäe+19; Uen+22a], given a chosen ciphertext  $\tilde{c}$  and a message  $m$ , reveals whether the PKE decryption of  $\tilde{c}$  equals  $m$ . A DF oracle [GJN20a] is similar to the general PC oracle, by viewing that the chosen ciphertext  $\tilde{c}$  is typically (but not necessarily) modified from a valid ciphertext  $c$  corresponding to a plaintext  $m$  (the “reference plaintext”). The DF oracle, similar to the general PC oracle, tells whether the PKE decryption result of  $\tilde{c}$  is equivalent to the reference plaintext  $m$ .

If the oracle returns 1, then it is a decryption success; otherwise, it is a decryption failure.

In the area of lattice-based cryptography, the term PC oracle by default refers to the binary PC oracle, a specialized variant introduced by D’Anvers et al. [DAn+19b] and later utilized by Ravi et al. [Rav+20a]. With this oracle, an attacker crafts ciphertexts such that the decrypted message  $m'$  depends on a single coefficient of the secret key, taking one of two distinct values—specifically, an all-zero vector versus a sparse vector with a single nonzero element. It is important to note that general PC oracles, DF oracles, and binary PC oracles extract at most 1 bit of information per oracle call. While more powerful variations exist, such as the multi-value PC oracle [Raj+23; Tan+23; SLZ24; GMÅ24; Li+25] and the full decryption oracle [Xu+20; Ngo+21a; Rav+22a; Guo+23b], these advanced oracles require significantly more information from a single query, which limits their use-cases.

The abstraction of oracles offers a clean and effective separation between the physical discovery of a leakage and its mathematical exploitation. This allows researchers to concentrate on developing efficient key-recovery algorithms based on specific oracle types, while others focus on identifying physical leakages to instantiate those oracles, ultimately leading to generic attack methodologies applicable across lattice-based, code-based, and isogeny-based KEMs [Uen+22a]. Furthermore, this approach facilitates the exploitation of a wide range of side-channel leakages. These leakages range from strong to extremely subtle and include timing discrepancies [Eat+18; PT19; DAn+19b; GJN20a; Guo+22a; Ber+25; CVE24], power and electromagnetic emanations [Rav+20a; Xu+20; Ngo+21a; Uen+22a; Sch+22b; Rav+22a; Raj+23; Tan+23; Guo+23b; She+23b; DG25c], and complex microarchitectural behaviors such as attacks exploiting dynamic frequency scaling [Wan+22], cache timing [Hua+23a], contention on execution units due to variable-time instructions [SGG24], data memory-dependent prefetchers [Che+24b], and contention on shared resources in trusted execution environments [Gas+25b; Wei+25].

Furthermore, in contrast to attacks targeting operations that directly manipulate secret values, such as the Number Theoretic Transform [PPM17], oracle-based attacks are considerably more challenging to detect and mitigate. This difficulty arises because such attacks exploit information leakages that have an indirect and often obscured relationship with the secret key. Last, the implications of oracle-based attacks extend beyond the realm of traditional side-channel analysis, e.g., in key-mismatch and misuse attacks [Flu16; DFR18; B  e+19; Qin+21; GM23] against IND-CPA secure KEMs or in fault-injection attacks [Xag+21; Mon+24]. This is because such attack vectors can be used to construct required PC-style oracles, effectively transforming the problem into one solvable with a PC oracle-based attack.

**Motivation.** Recent research into PC oracle-based side-channel attacks against lattice-based cryptographic schemes has largely followed two trends. The first focuses on optimizing query efficiency for *perfect* oracles. This line of work, initiated by Qin et al. [Qin+21], employs adaptive techniques, such as using Huffman coding on oracle outputs, to dramatically reduce the number of queries required for a key recovery. Subsequent work has refined this approach, achieving query counts that approach the theoretical Shannon limit [GM23]. The second trend confronts the more practical scenario of *inaccurate* oracles, which arise from environmental noise, measurement limitations, or active countermeasures. Here, methods have been proposed that combine adaptivity with error detection codes [She+23b]. However, these approaches tend to lose efficiency as the oracle’s error rate increases, as their reliance on techniques like majority voting requires a high number of traces to maintain accuracy. Non-adaptive approaches using Low-Density Parity-Check (LDPC) codes, such as the SCA-LDPC framework proposed in [Guo+23b], offer strong error-correction capability. However, they lack the information gain from adaptive queries, and their evaluation on binary PC or DF oracles in the lattice-based setting has been limited.

Within this context, a prevailing belief has formed that binary PC oracles are fundamentally more powerful and efficient than DF oracles for lattice-based schemes (see e.g. [Rav+24b]). This view, however, overlooks a crucial advantage of DF oracles: flexibility. This flexibility is twofold. First, certain physical leakages can naturally instantiate a DF oracle such as the timing leakage in [GJN20a]. Second, DF oracles are more general; they trigger when decryption fails to an *unspecified* message, whereas a binary PC oracle requires the decrypted message to match one of two precisely crafted, distinct values. This gives an attacker greater freedom in constructing chosen ciphertexts. Despite this, existing attacks based on DF oracles are often considered inefficient. For instance, a recent attack by Hermelink et al. [Her+24] on ML-KEM-768 required 7 000 perfect DF oracle queries for a full key recovery—a query count 4.49 times the information-theoretic lower bound. In stark contrast, our method achieves a full key recovery with only 2 950 queries against a noisy oracle, demonstrating a dramatic leap in efficiency. The recent GoFetch attack [Che+24b] serves as a prime example of these challenges. While it successfully constructs a DF oracle against Kyber on Apple M-series CPUs, its practical efficiency is hampered by significant limitations. The attack can only partially recover secret key coefficients, necessitating a computationally expensive, multi-hour lattice reduction step for full key recovery. Furthermore, it requires a large number of traces to overcome measurement noise, highlighting the inefficiencies of current DF oracle exploitation techniques.

This paper challenges the perception of DF oracle inferiority. We investigate whether the DF oracle is truly less efficient than the binary PC oracle, or if its full potential has simply not been realized. This leads to the core problem we address: *How can we design a more query-efficient oracle-based attack, especially for inaccurate oracles, and how closely can we approach the information-theoretic lower bound?*

**Table 1:** Comparison of binary PC and DF Oracle-Based Key-Recovery Attacks on ML-KEM.

| Work                                      | Oracle Accuracy Level ( $\rho$ ) | # Oracle Calls (Approx.) | Theoretical Lower Bound <sup>b</sup> | Ratio <sup>c</sup> |
|-------------------------------------------|----------------------------------|--------------------------|--------------------------------------|--------------------|
| Scheme: ML-KEM-512                        |                                  |                          |                                      |                    |
| Hermelink et al. [HPP21] (2021)           | Perfect (100%)                   | 5 750                    | 1 195                                | 4.81               |
| Hermelink et al. [Her+23] (2023)          | 100%; 90%                        | 5 500; 12 000            | 1 195; 2 250                         | 4.6; 5.33          |
| Ueno et al. [Uen+22a] (2022)              | 96%                              | 7 680                    | 1 577                                | 4.87               |
| Shen et al. [She+23b] (2023) <sup>d</sup> | 95%                              | 3 875                    | 1 674                                | 2.31               |
| <b>This Work</b>                          | <b>95%</b>                       | <b>2 400</b>             | <b>1 674</b>                         | <b>1.43</b>        |
| Scheme: ML-KEM-768                        |                                  |                          |                                      |                    |
| Hermelink et al. [HPP21] (2021)           | Perfect (100%)                   | 6 750                    | 1 560                                | 4.33               |
| <b>This Work</b>                          | <b>95%</b>                       | <b>2 950</b>             | <b>2 186</b>                         | <b>1.35</b>        |

<sup>b</sup>Calculated as  $H_2(\text{sk})/(1 - H_2(\rho))$ , where  $H_2(\text{sk})$  is the Shannon entropy of the full secret key ( $\approx 1\,195$  bits for ML-KEM-512,  $\approx 1\,560$  bits for ML-KEM-768) and  $1 - H_2(\rho)$  is the maximum information per query with noise level  $\rho$ .

<sup>c</sup>Ratio of Oracle Calls to the Theoretical Lower Bound. A lower ratio indicates higher efficiency.

<sup>d</sup>This is a follow-up for [Qin+21; GM23] that extends Huffman coding approach to the noisy oracles. While Huffman coding is the most efficient method for perfect oracles, the focus of this paper is on the noisy case.

Our work explores this question by developing a novel DF oracle-based attack, demonstrating that a carefully constructed attack methodology allows DF oracles to achieve, and even surpass, the efficiency of their binary PC-based counterparts, particularly in noisy, realistic environments.

**New Techniques.** To answer the research questions posed, we introduce a novel attack methodology that combines the principles of adaptive querying with the powerful error-correction capabilities of LDPC decoding. Our technique is centered on two core concepts.

First, we design queries that create high-information parity checks involving multiple secret key coefficients. Instead of targeting individual coefficients, which often leads to unbalanced and less informative oracle outputs, our method constructs linear combinations of several secret coefficients. This approach is powerful for two reasons: 1) it directly generates the parity checks required by an LDPC decoder, and 2) it allows us to maximize the information gained from each query. By carefully selecting the coefficients and a corresponding threshold, we can balance the oracle’s output distribution (i.e., make the probability of success or failure close to 0.5). This technique effectively functions as a randomness extractor, ensuring each oracle call yields close to the channel capacity, a maximum of Shannon entropy one can obtain with the noisy oracle.

Second, our attack is structured as a multi-phase, adaptive framework. The attack begins by issuing a series of these information-maximized queries. The noisy oracle responses are then processed by a belief propagation decoder, which computes a posterior probability distribution for each targeted secret coefficient, si-

multaneously aggregating information and correcting errors. The crucial step is the adaptive nature of the subsequent phase: based on these posterior probabilities, the attacker generates new, targeted queries designed specifically to resolve the highest remaining uncertainties. This adaptive loop—querying, decoding, and re-adapting—maximizes efficiency at each step. This hybrid approach overcomes the fragility of purely adaptive methods in noisy settings while improving the query count compared to purely non-adaptive schemes. The combination allows us to significantly reduce the number of traces for a full key recovery, demonstrating that DF oracles can be highly efficient.

**Research Contributions.** Our core contributions are:

- We introduce a novel hybrid attack framework that combines adaptive querying with belief propagation for Low-Density Parity-Check (LDPC) codes. Our method crafts ciphertexts that generate balanced, high-information parity checks over multiple secret key coefficients. This approach maximizes the Shannon information extracted from each query, proving highly effective even when interacting with imperfect or noisy oracles.
- We demonstrate that a well-designed Decryption Failure (DF) oracle attack can achieve, and in noisy environments surpass, the efficiency of state-of-the-art binary Plaintext-Checking (PC) oracle attacks. Our extensive simulations show a dramatic reduction in the required number of oracle queries. For instance, on ML-KEM-768 with an oracle accuracy of 95%, our attack requires only 2 950 queries for a full key recovery (a ratio of just 1.35 to the lower Shannon’s bound). A detailed comparison with previous works is presented in Table 1.
- We validate the practical impact of our findings by applying our framework to the GoFetch attack scenario [Che+24b]. Our methodology achieves significant gains in this real-world setting, overcoming previous limitations. We reduce the required measurement traces by more than an order of magnitude, eliminate the need for computationally expensive post-processing, and enable the practical full key recovery on higher-security schemes, which was intractable with the original GoFetch method.

The source code is available at <https://doi.org/10.5281/zenodo.17899501>.

**Outline.** The remainder of this paper is organized as follows. Section 2 provides the necessary background on ML-KEM and defines the threat model. Section 3 details our novel key-recovery methodology. In Section 4, we apply our framework to the GoFetch attack scenario to demonstrate its practical effectiveness. We then discuss the broader implications of our findings in Section 5. Finally, Section 6 concludes the paper and outlines directions for future research.

## 2 Preliminaries

In this section, we provide the necessary background, covering the fundamental aspects of the ML-KEM scheme and outlining the threat model assumed in this work.

**Notation.** We use lowercase bold letters (e.g.,  $\mathbf{v}$ ) for polynomials or vectors, and uppercase bold letters (e.g.,  $\mathbf{A}$ ) for matrices. For a polynomial  $\mathbf{v}$ , its  $i$ -th coefficient is denoted by  $\mathbf{v}[i]$ . All arithmetic on polynomial coefficients is performed modulo  $q$ . Polynomials reside in the ring  $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$ , where  $n = 256$ . We represent values from  $\mathbb{Z}_q$  as  $\{-\lfloor q/2 \rfloor, \dots, \lfloor q/2 \rfloor\}$  for odd  $q$ . For a vector  $\mathbf{s} = (s_0, \dots, s_{d-1})$  composed of  $d$  polynomials from  $\mathcal{R}_q$ , we abuse notation and write  $\mathbf{s}[i]$  for  $0 \leq i < d \cdot n$  to refer to  $\mathbf{s}_{\lfloor i/n \rfloor}[\lfloor i \bmod n \rfloor]$ .

We use  $H_2$  for the Shannon binary entropy and  $\text{Ber}_\rho$  for the Bernoulli distribution with parameter  $\rho$ .

### 2.1 ML-KEM (Kyber)

ML-KEM, previously named CRYSTALS-Kyber [Sch+20], is a key encapsulation mechanism (KEM) selected by the U.S. National Institute of Standards and Technology (NIST) as a primary standard for post-quantum cryptography [Nat24]. The scheme's security is based on the hardness of the Module Learning with Errors (MLWE) problem.

To achieve security against chosen-ciphertext attacks (IND-CCA), ML-KEM employs a variant of the Fujisaki-Okamoto (FO) transform [FO99]. This transform converts an IND-CPA secure public-key encryption scheme (denoted ML-KEM.CPAPKE) into an IND-CCA secure KEM (denoted ML-KEM.CCAKEM).

The core procedures for key generation, encapsulation, and decapsulation are detailed in Algorithm 1. These descriptions are simplified for clarity, omitting implementation-level optimizations like the Number Theoretic Transform (NTT). The complete specifications are available in the official documentation [Nat24].

In ML-KEM, a smart design includes compression and decompression functions that reduce ciphertext sizes by operating coefficient-wise on polynomials. They are defined as follows:

**Definition 1 (Compression).** *The function  $\text{Comp}_q(x, d) : \mathbb{Z}_q \rightarrow \mathbb{Z}_{2^d}$  is defined as:*

$$\text{Comp}_q(x, d) = \left\lfloor \frac{2^d}{q} \cdot x \right\rfloor \pmod{2^d}. \quad (1)$$

**Definition 2 (Decompression).** *The function  $\text{Decomp}_q(x, d) : \mathbb{Z}_{2^d} \rightarrow \mathbb{Z}_q$  is defined as:*

$$\text{Decomp}_q(x, d) = \left\lfloor \frac{q}{2^d} \cdot x \right\rfloor. \quad (2)$$

---

**Algorithm 1** A Simplified View of the ML-KEM

---

ML-KEM.CCAKEM.KeyGen()

**Output:**  $sk', pk$

- 1:  $z \xleftarrow{\$} \{0, 1\}^{256}$
- 2:  $(\mathbf{A}, \mathbf{s}, \mathbf{e}) \xleftarrow{\$} \mathcal{R}_q^{d \times d} \times \mathcal{B}_{\eta_1}^d \times \mathcal{B}_{\eta_1}^d$
- 3:  $\mathbf{b} \leftarrow \mathbf{A}\mathbf{s} + \mathbf{e}$
- 4:  $pk \leftarrow (\mathbf{b}, \rho)$ , where  $\rho$  is the seed for  $\mathbf{A}$
- 5:  $sk \leftarrow \mathbf{s}$
- 6:  $sk' \leftarrow (sk, pk, \mathcal{H}(pk), z)$
- 7: **return**  $(pk, sk')$

ML-KEM.CCAKEM.Encaps( $pk$ )

**Input:**  $pk = (\mathbf{b}, \rho)$

**Output:**  $K, ct$

- 8:  $\mathbf{m} \xleftarrow{\$} \{0, 1\}^{256}$
- 9:  $(K, r) \leftarrow \mathcal{G}(\mathbf{m}, \mathcal{H}(pk))$
- 10:  $\mathbf{A} \leftarrow \text{Gen}(\rho)$
- 11:  $\mathbf{r} \xleftarrow{\$} \mathcal{B}_{\eta_1}^d; \mathbf{e}_1 \xleftarrow{\$} \mathcal{B}_{\eta_2}^d; \mathbf{e}_2 \xleftarrow{\$} \mathcal{B}_{\eta_2}^d$
- 12:  $\mathbf{u} \leftarrow \mathbf{A}^T \mathbf{r} + \mathbf{e}_1$
- 13:  $v \leftarrow \mathbf{b}^T \mathbf{r} + \mathbf{e}_2 + \text{Decomp}_q(\mathbf{m}, 1)$
- 14:  $c_1 \leftarrow \text{Comp}_q(\mathbf{u}, d_u)$
- 15:  $c_2 \leftarrow \text{Comp}_q(v, d_v)$
- 16:  $ct \leftarrow (c_1, c_2)$
- 17: **return**  $(K, ct)$

ML-KEM.CCAKEM.Decaps( $sk', ct$ )

**Input:**  $sk' = (sk, pk, h, z), ct = (c_1, c_2)$

**Output:**  $K'$

- 18:  $\mathbf{u}' \leftarrow \text{Decomp}_q(c_1, d_u)$
  - 19:  $v' \leftarrow \text{Decomp}_q(c_2, d_v)$
  - 20:  $\mathbf{m}' \leftarrow \text{Comp}_q(v' - sk^T \mathbf{u}', 1)$
  - 21:  $(K', r') \leftarrow \mathcal{G}(\mathbf{m}', h)$
  - 22:  $\bar{K} \leftarrow \mathcal{J}(z, ct)$
  - 23:  $ct' \leftarrow ML - KEM.CPAPKE.\text{Enc}(pk, \mathbf{m}', r')$
  - 24: **if**  $ct \neq ct'$  **then**
  - 25:      $K' \leftarrow \bar{K}$
  - 26: **end if**
  - 27: **return**  $K'$
-

Table 2: Parameter sets for ML-KEM [Sch+20]

| Scheme      | $n$ | $d$ | $q$  | $\eta_1$ | $\eta_2$ | $(d_u, d_v)$ |
|-------------|-----|-----|------|----------|----------|--------------|
| ML-KEM-512  | 256 | 2   | 3329 | 3        | 2        | (10, 4)      |
| ML-KEM-768  | 256 | 3   | 3329 | 2        | 2        | (10, 4)      |
| ML-KEM-1024 | 256 | 4   | 3329 | 2        | 2        | (11, 5)      |

The scheme also employs hash functions  $\mathcal{J}$ ,  $\mathcal{G}$ , and  $\mathcal{H}$ . ML-KEM is specified for three security levels—ML-KEM-512, ML-KEM-768, and ML-KEM-1024—each with a unique parameter set, as detailed in Table 2. All variants use the prime modulus  $q = 3329$ . The module rank,  $d$ , is set to 2, 3, or 4 for the respective security levels. Secret and error polynomials are sampled from a centered binomial distribution,  $\mathcal{B}_\eta$ , parameterized by  $\eta_1$  and  $\eta_2$ .

## 2.2 The Threat Model

In our threat model, we consider a side-channel adversary whose goal is to recover the secret key from an ML-KEM implementation. The adversary has the capability to perform a chosen-ciphertext attack: they can submit crafted ciphertexts to the decapsulation device and observe physical side-channel information (e.g., timing, power) emitted during the computation. We abstract this physical leakage into a conceptual tool known as an “oracle”, which the adversary queries to gain information about the secret.

This oracle-based approach is a powerful and widely used methodology in side-channel analysis. It creates a clean separation between the physical discovery of a leakage and its mathematical exploitation. This allows for the development of generic key-recovery algorithms that are applicable across a wide range of cryptographic schemes and physical platforms.

The oracles central to attacks on PQC KEMs such as ML-KEM, including the ones discussed in this paper, are generally categorized as follows:

- **Binary Plaintext-Checking (PC) Oracle:** This is the most common oracle model for lattice-based schemes that allows for a given ciphertext  $\tilde{c}$  to obtain information about the corresponding decrypted plaintext  $m'$ . An attacker crafts ciphertexts such that  $m'$  depends on a single coefficient of the secret key. This causes  $m'$  to take one of two distinct, predictable values, for instance, “the decrypted plaintext will be 0” (an all-zero vector) or “a unit vector  $0^{i-1} \| 1 \| 0^{l-i-1}$ ” (a single 1 at a specific position, with all other bits being 0). The side-channel leakage, therefore, acts as a simple binary distinguisher for a secret coefficient. More powerful, though less common, variants like the multi-value PC oracle also exist.

- **General PC and Decryption-Failure (DF) Oracle:** A more general primitive is an oracle that, given a ciphertext  $\tilde{c}$  and a reference message  $m$ , reveals whether the decrypted plaintext  $m'$  is equal to  $m$ . In the context of FO-transformed KEMs, the General PC oracle and the DF oracle effectively collapse into the same side-channel observation [Uen+22a] and can be treated as mathematically equivalent. In this work, we adopt the term *DF Oracle* for this primitive. We do so for two reasons: (1) to clearly distinguish our approach from the restrictive binary PC oracle described above, and (2) to align with the physical reality of our target scenario (GoFetch [Che+24b]), where the leakage manifests as a system-level failure (e.g., a pointer dereference fault) when decryption yields an unexpected value.

As both the binary PC and DF oracles described here provide at most one bit of information per query, we collectively refer to them as *one-bit-bound* oracles.

These abstract one-bit-bound oracle models are remarkably generic; key-recovery algorithms based on them are applicable not only to lattice-based schemes but also to code-based and isogeny-based KEMs. Furthermore, they are versatile and can be instantiated by exploiting a wide spectrum of side-channel leakages—from strong to extremely subtle—including timing discrepancies [Eat+18; PT19; DAn+19b; GJN20a; Guo+22a; Ber+25; CVE24], power and electromagnetic emanations [Rav+20a; Xu+20; Ngo+21a; Uen+22a; Sch+22b; Rav+22a; Raj+23; Tan+23; Guo+23b; She+23b; DG25c], and complex microarchitectural behaviors [Wan+22; Hua+23a; SGG24; Che+24b; Gas+25b; Wei+25]. While the oracle abstraction is general, its concrete instantiation is highly platform-dependent, relying on the specific leakage characteristics of the target’s hardware and software. In this work, we assume the existence of such an oracle to simulate the performance of the key recovery; its instantiation within the GoFetch attack scenario will be detailed in Section 4.

### 3 New Key-Recovery Approach for ML-KEM

This section outlines our new key-recovery strategy for ML-KEM while drastically reducing the number of oracle queries.

#### 3.1 Information-Theoretic Foundations

As established in Section 2.2, a chosen oracle returns a single binary value. Because of noise, imperfect encodings, and conditional mutual information, the amount of Shannon information we obtain per query is generally *strictly less than one bit*. Fewer bits per query translate directly into more queries before the secret is recovered, so we are motivated to maximize the information carried by each oracle response.

**Setting.** Let  $S$  denote the random variable we wish to learn; for concreteness, you may think of  $S = s[i] \in \{-2, -1, 0, 1, 2\}$ <sup>1</sup> when we deal with ML-KEM-768. Because  $S$  is finite, we can list all its possible values in a table together with their probabilities. A side-channel adversary crafts a ciphertext that leads to the computation of a predicate

$$X = f(S) \in \{0, 1\},$$

that is detectable via side-channels. This function we call an *encoding* of  $S$ . For instance, one may take  $f(S) = [s[i] > 0]$ , yielding the truth vector  $(0, 0, 0, 1, 1)$  for  $s[i] \in \{-2, -1, 0, 1, 2\}$ .

**Noisy Oracle.** In practice, the oracle is noisy: instead of  $X$  we observe  $Y = X \oplus E$ , where  $E \leftarrow \text{Ber}_{1-\rho}$  is independent of  $S$  and the oracle accuracy  $\rho > \frac{1}{2}$ . Upon seeing  $Y$  we gain the mutual information

$$I(S; Y) = H(S) - H(S | Y).$$

Maximizing  $I(S; Y)$  therefore minimizes the expected number of queries. The next lemma shows that information is maximized when the observable output  $Y$  is *balanced*.

**Lemma 1** (Balanced leakage maximizes information). *Fix the oracle accuracy  $\rho \in (\frac{1}{2}, 1]$  and let  $f: \text{supp}(S) \rightarrow \{0, 1\}$  be an arbitrary predicate. Write  $q := \Pr[f(S) = 1]$  and  $r := \Pr[Y = 1] = (1 - \rho) + (2\rho - 1)q$ . Then*

$$I(S; Y) = H_2(r) - H_2(1 - \rho) = H_2(r) - H_2(\rho),$$

where  $H_2$  is the binary entropy function. The quantity  $I(S; Y)$  achieves its maximum  $1 - H_2(\rho)$  when and only when  $r = \frac{1}{2}$ .

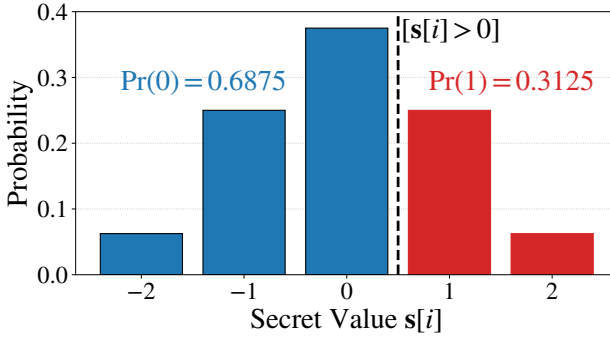
*Proof.* Because  $X = f(S)$  is deterministic,  $H_2(Y | S) = H_2(Y | X)$  and hence

$$I(S; Y) = H_2(Y) - H_2(Y | S) = H_2(Y) - H_2(Y | X) = I(X; Y).$$

Maximizing  $I(X; Y)$  therefore *also* maximizes  $I(S; Y)$ , and it is enough to work with the simpler binary random variable  $X$ .

For a binary-symmetric channel with crossover probability  $1 - \rho$ , we have  $I(X; Y) = H_2(Y) - H_2(Y | X) = H_2(r) - H_2(1 - \rho)$ , where  $r = (1 - \rho) + (2\rho - 1)q$ . The second term is constant, and  $H_2(r)$  is strictly concave with a unique maximum at  $r = \frac{1}{2}$ . Hence  $I(X; Y)$ , and therefore  $I(S; Y)$ , is maximized when  $r = \frac{1}{2}$ .  $\square$

<sup>1</sup>Later in the attack, we group together several coefficients, but setting with one coefficient should be sufficient for the present discussion.



**Figure 1:** An example of a simple highly unbalanced encoding  $[s[i] > 0]$  on a single secret coefficient for ML-KEM-768

**Implications for Attack Design.** The preceding lemma says that every oracle call can obtain at most  $1 - H_2(\rho)$  bits of information; this upper bound is achieved *only* when the observable bit  $Y$  is perfectly balanced. Note here that if  $\Pr[X = 1]$  is close to 0.5, then  $\Pr[Y = 1]$  is even closer to 0.5, thus, finding a good balanced encoding guarantees that  $Y$  is well-spread. So, for now we focus on  $X$  for clarity, but our attack searches for encodings maximizing how balanced  $Y$  is, i.e., it depends on the accuracy of the oracle  $\rho$ .

One of our goals is to select such a balanced encoding. Unfortunately, one can not choose any binary string for an encoding: the set of possible encodings is dictated by the oracle itself, for example, in a typical binary PC oracle attack, the message is forced to be all zero except for a single bit. This leads to encodings of the form  $s[i] > t$  or  $s[i] < t$  for  $t \in \{-2, 1, 0, 1, 2\}$  that are generally poorly balanced. See example in Figure 1.

In the following we show a new construction that allows much more balanced encodings which leads to a better information per oracle call extraction and reduction of the number of traces.

### 3.2 Creating Basic Chosen Ciphertexts

Our main idea is to make the oracle output depend not on a single coefficient of the secret key, but on several of them simultaneously. Specifically, we define the target random variable  $S$  as an  $\omega$ -dimensional vector composed of  $\omega > 1$  chosen coefficients from the secret key  $\mathbf{s}$ . The random variable  $S$  can thus take on up to  $|\mathcal{B}_{\eta_1}|^\omega$  distinct vector values. This vastly larger support space alleviates the problem of probability being concentrated on a single value, which is crucial for creating well-balanced encodings.

This approach extends the work of [Guo+23b], which proposed targeting *the sum* of several secret coefficients. While targeting the sum also improves the balancing problem, it leads to an unavoidable loss of information through aliasing.

For instance, the secret vectors  $(0, -1)$  and  $(-1, 0)$  are indistinguishable as their sum is  $-1$  in both cases. Our method is more general as it targets the joint distribution of the vector directly, allowing us to potentially distinguish every possible outcome. This increased granularity, however, comes at the price of a more computationally intensive key-recovery phase.

Let us briefly revisit the decapsulation procedure. The secret key  $\mathbf{s}$  consists of  $d$  polynomials,  $\mathbf{s} = (\mathbf{s}_0, \dots, \mathbf{s}_{d-1})$ , where  $\mathbf{s}_i \in \mathcal{R}_q$ . Similarly, for a ciphertext  $(\mathbf{u}, \mathbf{v})$ , we have  $\mathbf{u} = (\mathbf{u}_0, \dots, \mathbf{u}_{d-1})$  for  $\mathbf{u}_i, \mathbf{v} \in \mathcal{R}_q$ . The message is computed as  $\mathbf{m}' \leftarrow \text{Comp}_q(\mathbf{v}' - \mathbf{s}^T \mathbf{u}', 1)$ , where  $\mathbf{u}'$  and  $\mathbf{v}'$  are obtained via application of the  $\text{Decomp}_q$  function. The message bit  $\mathbf{m}[j]$  is set to 0 if  $(\mathbf{v}' - \mathbf{s}^T \mathbf{u}')[j]$  is closer to 0 (modulo  $q$ ) and to 1 otherwise. The switch happens at  $\pm(\lfloor \frac{q}{4} \rfloor + 1)$ . Thus, we always choose  $\mathbf{u}$  in a way so that for each coefficient  $c$  of  $\mathbf{s}^T \mathbf{u}'$  we have  $|c| \leq \lfloor \frac{q}{4} \rfloor$ . This guarantees that the decrypted message would be exactly  $\mathbf{m}$  except for positions with additional noise in  $\mathbf{v}$ .

More concretely, let  $\mathbf{v}' = \text{Decomp}_q(\mathbf{m}, 1) + \delta$ , then the input to  $\text{Comp}_q$  at position  $0 \leq j < n$  is

$$(\mathbf{v}' - \mathbf{s}^T \mathbf{u}')[j] = \mathbf{m}[j] + \underbrace{\delta[j]}_{\text{additional noise}} - \underbrace{(\mathbf{s}^T \mathbf{u}') [j]}_{\text{secret dependent}}.$$

A decryption failure for positive  $\delta[j]$  happens if

$$\delta[j] - (\mathbf{s}^T \mathbf{u}') [j] > \left\lfloor \frac{q}{4} \right\rfloor,$$

or, rearranging,

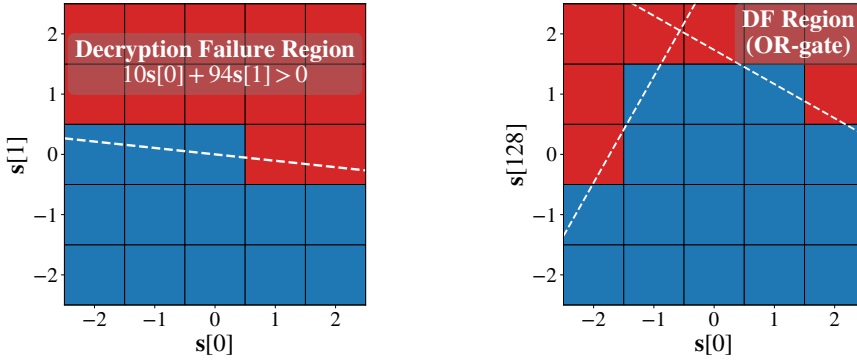
$$(\mathbf{s}^T \mathbf{u}') [j] < \delta[j] - \left\lfloor \frac{q}{4} \right\rfloor.$$

Similarly, for negative  $\delta[j]$  we have  $(\mathbf{s}^T \mathbf{u}') [j] > \delta[j] + \lfloor \frac{q}{4} \rfloor$ . Due to the choice of  $q$  and construction of  $\text{Decomp}_q$  function, the value  $\delta[j] \pm \lfloor \frac{q}{4} \rfloor$  is a multiple of  $\left\lfloor \frac{q}{2^{d_v}} \right\rfloor$ . Thus, by carefully choosing the added noise at position  $j$ , we can trigger a decryption failure if

$$\begin{aligned} (\mathbf{s}^T \mathbf{u}') [j] &> k \cdot \left\lfloor \frac{q}{2^{d_v}} \right\rfloor, \text{ or} \\ (\mathbf{s}^T \mathbf{u}') [j] &< -k \cdot \left\lfloor \frac{q}{2^{d_v}} \right\rfloor, \end{aligned} \tag{3}$$

where  $k \in \{0, 1, 2, 3\}$ . In other words, the decryption failure at position  $j$  happens when one of the inequalities from 3 is satisfied. Which inequality we look at depends on the sign of the added noise; both can not be satisfied at the same time.

The first type of encoding we work on is created when exactly one value of  $\delta$



(a) Encoding over linear combination of  $\omega$  secret coefficients

(b) Combination of several inequalities

**Figure 2:** A visualization of encoding types that are used in the key-recovery process for  $\omega = 2$ . (a) Basic encoding that is represented by a single inequality, geometrically is determined by a hyperplane, separating  $\omega$ -dimensional space into two regions, (b) using DF oracle, we can combine multiple inequalities/hyperplanes, implementing logical OR over these inequalities to achieve more balanced encodings.

is nonzero at position  $j$  and  $\mathbf{u}$  has  $\omega$  nonzero values at arbitrary positions. Note that setting  $\mathbf{u}_0[i_1] = z$ , results in  $(\mathbf{s}^T \mathbf{u}')[j]$  having a value

$$\begin{cases} -z \cdot s_0[j - i_1], & \text{if } i_1 \leq j, \\ z \cdot s_0[n - i_1 + j], & \text{otherwise} \end{cases}$$

due to multiplication in  $\mathcal{R}_q$ . This notation can easily become cumbersome; thus, instead of describing how to choose  $\mathbf{u}$ , we focus on the inequality determining if decryption failure is triggered. Then this encoding type could be described by a single inequality (or a predicate)

$$z_0 \mathbf{s}[i_0] + \dots + z_{\omega-1} \mathbf{s}[i_{\omega-1}] \leq T, \quad (4)$$

where indices  $(i_0, \dots, i_{\omega-1})$  could be chosen freely,  $T$  is the multiple of  $\left\lfloor \frac{q}{2^{d_v}} \right\rfloor$ , the comparison sign (greater or less) depends on the sign of  $T$  and values  $(z_0, \dots, z_{\omega-1})$  are from image of  $\text{Decomp}_q(\cdot, d_u)$  with additional restriction  $|\eta_1 \cdot \sum_{i=0}^{\omega-1} z_i| < \left\lfloor \frac{q}{4} \right\rfloor$  to avoid triggering decryption failure for positions different from  $j$ . So, in words, we can create a ciphertext that, with the help of the oracle, should reveal information about the linear combination of  $\omega$  secret coefficients, in particular, where this linear combination is greater or less than some threshold value of our choice. See a low-dimensional example in Figure 2a.

The random variable  $S$  is then  $S = (\mathbf{s}[i_0], \dots, \mathbf{s}[i_{\omega-1}])$ . The complexity

of our key-recovery process depends on the support size of  $S$ , which is  $|\mathcal{B}_{\eta_1}|^\omega$ . Generally, a larger  $\omega$  allows for more balanced encodings at the price of exponentially slower key recovery. We choose  $\omega = 4$  for our attack as it provides a good trade-off, and an additional reason is revealed in Section 3.3.

### 3.3 Utilizing DF Benefits: Complex Encoding

Even though the truth table for  $S$  is quite large (i.e.,  $7^4 = 2401$  for ML-KEM-512 and  $5^4 = 625$  for ML-KEM-768 and ML-KEM-1024), the encoding we can obtain from Section 3.2 is still highly structural as it is a hyperplane separating vectors in two classes and, for example, if the inequality  $(\mathbf{s}^T \mathbf{u}')[j] > 0$  is satisfied for some  $\mathbf{s}[i] = 1$  where  $\mathbf{s}[i]$  is part of  $S$ , then it would also be satisfied for  $\mathbf{s}[i] = 2$ . This is rather unfortunate since the more structure we have, the less likely to have a well-balanced encoding.

We propose a new idea that allows us to increase the number of possible encodings that are less structured, which allows a more efficient (in number of oracle calls) key-recovery process. The idea is based on the benefits that the DF oracle provides, specifically, that it can detect if the decrypted message differs from the intended message at *any* position. In Section 3.2, similarly to a typical binary PC oracle attack, we choose only one nonzero position in  $\delta$ . But now we allow  $\mathbf{m}'$  to be different from  $\mathbf{m}$  at *a few* positions.

As was shown previously, after fixing  $\mathbf{u}$  to have exactly  $\omega$  nonzero positions and setting  $\delta[j]$  to be nonzero, we get an inequality over  $\omega$  secret coefficients. Therefore, we can choose  $\delta[j_1]$  and  $\delta[j_2]$  to be nonzero, then the decryption failure would occur if the inequality at position  $j_1$  *or* the inequality at position  $j_2$  is satisfied. However, a naive choice of  $j_1$  and  $j_2$  would result in an encoding over two disjoint sets of secret coefficients, increasing the dimension of  $S$  to  $2\omega$  and making key recovery much more computationally heavy.

Luckily, the negacyclic ring structure of Kyber provides a powerful solution. By carefully selecting the nonzero positions in both  $\mathbf{u}'$  and the noise  $\delta$  to be related by cyclic shifts, we can force all the resulting inequalities to depend on the exact same set of  $\omega$  secret coefficients. We now give an example that is easily generalizable. Set

$$\begin{aligned}\mathbf{u}'_0 &= z_0 + z_1x^{64} + z_2x^{128} + z_3x^{192} \\ \delta &= \delta_0 + \delta_1x^{64} + \delta_2x^{128} + \delta_3x^{192},\end{aligned}$$

while other polynomials of  $\mathbf{u}$  are  $\mathbf{0}$ . A careful inspection of the computation of

$\mathbf{v}' - \mathbf{s}^T \mathbf{u}'$  reveals the 4 inequalities:

$$\begin{cases} z_0 \mathbf{s}_0[0] + z_1 \mathbf{s}_0[64] + z_2 \mathbf{s}_0[128] + z_3 \mathbf{s}_0[192] \leq T_0 \\ -z_3 \mathbf{s}_0[0] + z_0 \mathbf{s}_0[64] + z_1 \mathbf{s}_0[128] + z_2 \mathbf{s}_0[192] \leq T_1 \\ -z_2 \mathbf{s}_0[0] - z_3 \mathbf{s}_0[64] + z_0 \mathbf{s}_0[128] + z_1 \mathbf{s}_0[192] \leq T_2 \\ -z_1 \mathbf{s}_0[0] - z_2 \mathbf{s}_0[64] - z_3 \mathbf{s}_0[128] + z_0 \mathbf{s}_0[192] \leq T_3 \end{cases}$$

where threshold  $T_i$  depends on  $\delta[64i]$ . Instead of a single inequality over  $\omega$  variables, the oracle now depends on  $\omega$  inequalities over the same variables. The coefficients of the linear combination could be chosen freely for the first inequality, later we apply a negacyclic shift. Notably, despite some limitations on the linear combination coefficients, we still can choose  $T_i$  freely and independently which greatly increases the flexibility of constructing an encoding and, consequently, how balanced it is. The visualization of this approach for  $\omega = 2$  is shown in Figure 2b.

This approach implicitly uses the fact that  $\omega$  has to be a divisor of  $n$ . And since  $n = 256$ ,  $\omega$  has to be a power of 2,  $\omega = 4$  is a reasonable choice as  $\omega \geq 8$  gets too expensive. This is because the runtime of our key-recovery scales with  $|\mathcal{B}_{\eta_1}|^\omega$ .

The idea is generalizable in a way that we can choose which block of secret key to target, then we set  $\mathbf{u}_i$  to have nonzero values at positions  $(j_1, j_1 + 64, j_1 + 128, j_1 + 192)$ , while for  $\mathbf{v}$  at positions  $(j_2, j_2 + 64, j_2 + 128, j_2 + 192)$  for  $0 \leq j_1, j_2 < n$ . This leads to 4 inequalities over 4 secret coefficients from  $\mathbf{s}_i$ . We call this type of encoding *full rotation encoding*. See more details and examples in Section Appendix A.

### 3.4 Key-Recovery Strategy

In Sections 3.2 and 3.3 it was shown how to obtain a single bit from an oracle that maximizes the amount of information we learn. In order to recover the whole secret key, many more bits are required. We can define a lower bound on the number of required oracle calls using channel capacity and entropy of the key as  $H_2(\mathbf{s})/(1 - H_2(\rho))$ . In this Section, we discuss the strategy of combining several calls to maximize information gain.

#### Multibit Encoding

The first straightforward approach is to keep the subset of secret coefficients fixed, then several different encodings are used. Thus, each vector of the random variable  $S$  is encoded with several bits. The estimation of the received information in this case essentially follows Section 3.1. Calling oracle  $k$  times on  $S$  gives us  $\mathbf{y}$ , which is a binary string of length  $k$ , so that we obtain multibit encoding of  $S$ . If  $\mathbf{y}$  is uniformly distributed, then we obtain the maximum possible amount of information  $k(1 - H_2(\rho))$ , thus, the goal of making  $\mathbf{y}$  balanced stays the same, but now instead of a single bit we try to balance  $k$  bits.

The common approach is to use something akin to Hamming adaptive encoding, where the first bit is chosen to be as balanced as possible, while the choice of the subsequent bit encoding depends on the previous bits, again trying to make a balanced bit. This adaptive approach, however, struggles when the accuracy of the oracle  $\rho$  drops since incorrect early bit negatively affects subsequent choices. Therefore, as our approach is designed to work with inaccurate oracles, we try to limit adaptivity for now.

We create multibit encodings for full rotation checks that were described in Section 3.3 as they provide the most flexibility and create the most balanced encodings. Ideally, we just put together  $k$  balanced encodings and obtain close to  $k(1 - H_2(\rho))$  amount of information. Unfortunately, the encodings are not independent in general. Even if each bit is perfectly balanced on its own, the bitstring  $\mathbf{y}$  is not uniform. This leads to an interesting scenario: the more bits we add to a multibit encoding of  $S$ , the less amount of information each subsequent bit provides, as we are slowly exhausting nearly independent encodings. We stress again that each bit encoding is fixed in advance before the attack starts.

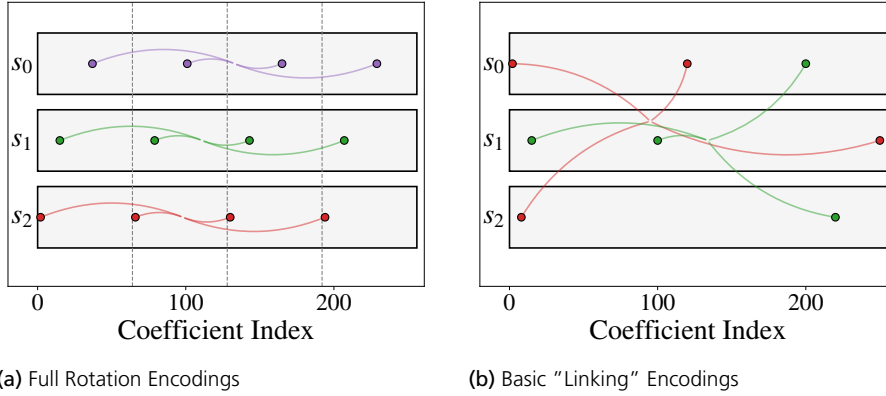
We search for this multibit encoding first by randomly sampling 10 000 balanced full rotation 1-bit encodings, then we greedily try to combine them with a goal to get as uniform  $\mathbf{y}$  as possible. In our experiments, we found 7 bits to be a good cutoff. For  $\rho = 0.95$ , the theory suggests the upper bound of information to be 4.99522 bits, while we can get (on average) 4.66449 bits for ML-KEM-512 and 4.67581 bits for ML-KEM-768, i.e., we lose about 0.3 bits of information per 7 calls.

We split all secret coefficients into  $\frac{n}{\omega}d$  sets of  $\omega$  coefficients. Then, for each set, we call the oracle 7 times. But it is not enough to recover the whole secret. Adding more bits to multibit encoding gives diminishing returns, a different approach is needed to maximize further extracted information.

### Batches: New Adaptive Approach

Since the multibit full rotation encodings were applied to independent subsets of  $\mathbf{s}$ , the total information gathered in this initial phase is simply the sum of the mutual information from each subset. However, to resolve the remaining uncertainty, we must introduce new encodings that link these previously independent groups, as illustrated in Figure 3. A significant challenge arises here: estimating the information gain from a new linking query is nontrivial, as it is statistically dependent on 16 previous secret coefficients. A direct computation of the overall mutual information quickly becomes intractable.

To overcome this, we propose an adaptive, iterative strategy. After the initial phase of full rotation encodings, we compute a posterior probability mass function (PMF) for each secret coefficient based on the oracle outputs. This belief state then guides the selection of new queries. For any given set of four coefficients, we can efficiently search for a nearly optimal balanced encoding. We achieve this



**Figure 3:** Illustration of the two-phase query strategy. (a) In the initial phase, full rotation encodings create strong, independent checks over structured, disjoint groups of coefficients (e.g., within  $s_0$ ,  $s_1$ , etc.). (b) In the batch phase, basic encodings create flexible "linking" checks that connect arbitrarily chosen coefficients across the different groups.

by checking a pre-selected set of approximately 12 000 candidate encodings and, based on the current posterior PMFs, quickly selecting the one that yields the most balanced oracle response, thereby maximizing the expected information gain. To manage this process, we again partition all secret coefficients into  $\frac{n}{\omega}d$  disjoint subsets and apply one such adaptively chosen encoding to each. We define this full set of  $\frac{n}{\omega}d$  new queries as a *batch*.

To integrate the information from this new batch and update our belief state, we employ a belief propagation (LDPC) decoder, following the SCA-LDPC framework [Guo+23b]. The decoder processes the responses from *all* oracle queries made so far—both the initial full rotation encodings and all subsequent batches. This not only combines the information but also provides powerful error-correction. The output is a new, more refined set of marginal PMFs for each secret coefficient. The attack proceeds by iteratively adding new batches of adaptively generated queries and running the LDPC decoder, progressively reducing the uncertainty until the secret key is recovered.

### Hybrid Adaptive Algorithm Using LDPC

The complete hybrid attack strategy, combining the initial information gathering with the adaptive batch-based approach, is outlined in Algorithm 2. The process begins by querying the oracle with a precomputed set of 7-bit full rotation encodings to establish a baseline of information. Based on these initial responses, we compute a preliminary posterior probability mass function (PMF) for each secret coefficient using a maximum-likelihood approach.

The attack then enters an iterative loop, where in each round, or *batch*, we partition the secret coefficients and apply new, adaptively chosen single-bit basic encodings to link them. The responses from each batch are collected and then processed, along with all previous data, by an LDPC decoder. This step updates the PMFs for all coefficients, refining our belief state and leveraging the error-correcting properties of the decoder. This loop continues for a fixed number of batches.

---

**Algorithm 2** Hybrid Adaptive Key-Recovery Strategy
 

---

**Input:** Oracle noise level  $\rho$ , Maximum number of batches  $B_{max}$ , Entropy threshold  $B_{entr}$

**Output:** Estimated secret key  $\hat{s}$

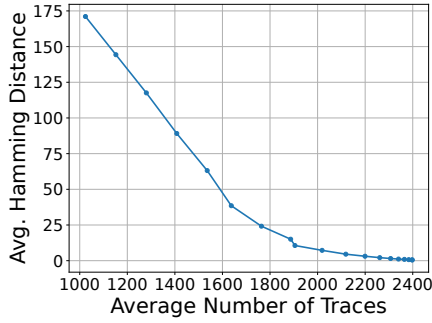
```

1:  $C_{FR} \leftarrow$  Precomputed 7-bit full rotation encoding
2:  $D_{FR} \leftarrow \emptyset$   $\triangleright$  Storage for full rotation oracle responses
3:  $D_{batch} \leftarrow \emptyset$   $\triangleright$  Storage for batch oracle responses
4: for all polynomials  $s_k$  in  $s$  do
5:   for all  $j \in \{0, \dots, 63\}$  do
6:      $I \leftarrow \{j, j + 64, j + 128, j + 192\}$   $\triangleright$  Select coefficient indices
7:      $y \leftarrow$  empty array of size 7
8:     for  $i \in \{0, \dots, 6\}$  do
9:        $ct_i \leftarrow \text{BuildFRCt}(C_{FR}, I, i, k)$   $\triangleright$  Build ciphertext for  $i$ -th bit for  $s_k$ 
10:       $y[k] \leftarrow \text{QueryOracle}(ct_i)$ 
11:    end for
12:    Add  $(I, C_{FR}, y)$  to  $D_{FR}$ 
13:  end for
14: end for
15:  $pmfs \leftarrow \text{ComputeInitialPMFs}(D_{FR})$   $\triangleright$  Initial belief via Max-Likelihood
16: for  $b \in \{1, \dots, B_{max}\}$  do
17:    $P \leftarrow \text{RandomPartition}(\{0, \dots, n \cdot d - 1\}, \text{size}=\omega)$ 
18:   for all subset of indices  $I \in P$  do
19:     if  $H_2(pmfs[I]) < B_{entr}$  then continue;  $\triangleright$  Skip if the chosen coefficients have low total entropy
20:      $enc \leftarrow \text{FindBalancedEncoding}(I, pmfs)$   $\triangleright$  Adaptively find best encoding
21:      $ct \leftarrow \text{BuildBatchCt}(enc, I)$ 
22:      $y \leftarrow \text{QueryOracle}(ct)$   $\triangleright$  Get 1 noisy bit
23:     Add  $(I, c, y)$  to  $D_{batch}$ 
24:   end for
25:    $pmfs \leftarrow \text{LDPC-Decode}(D_{FR} \cup D_{batch})$   $\triangleright$  Update beliefs
26: end for
27:  $\hat{s} \leftarrow \text{EstimateKeyFromPMFs}(pmfs)$   $\triangleright$  Make final decision
28: return  $\hat{s}$ 

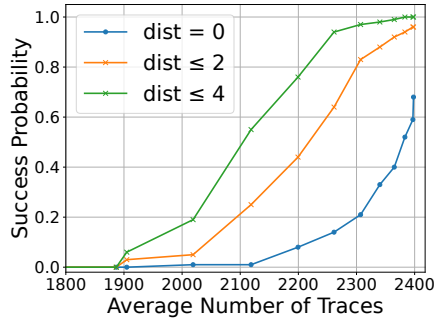
```

---

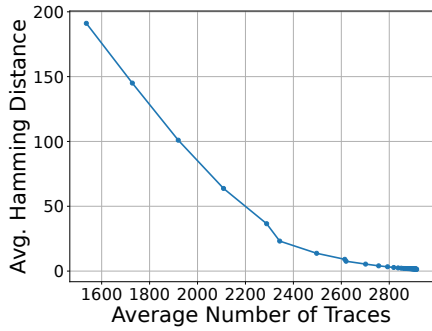
To evaluate the effectiveness of our strategy, we ran simulations for both ML-KEM-512 and ML-KEM-768 using an oracle with  $\rho = 0.95$ . The results, averaged over 100 keys, are presented in Figure 4. We measure the attack's progress



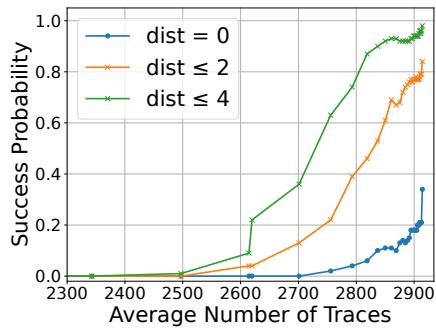
(a) ML-KEM-512



(b) ML-KEM-512



(c) ML-KEM-768



(d) ML-KEM-768

**Figure 4:** Attack results for ML-KEM-512 (top row) and ML-KEM-768 (bottom row), averaged over 100 keys. The *left column* shows the decrease in the number of incorrect secret coefficients as more oracle calls are made. The *right column* shows the final key-recovery success rate with no post-processing, light, and medium post-processing. Each data point is obtained after running the full LDPC decoder on the accumulated information from each batch of oracle queries.

using the *Hamming distance*, defined as the number of incorrect coefficients in the recovered by LDPC secret key compared to the correct one. As shown in the Figure, the average Hamming distance steadily decreases as more information is gathered through successive batches of oracle queries. A key finding is that a perfect recovery (Hamming distance of 0) does not always happen. Often, the attack terminates with a small number of errors in the recovered key. This remaining uncertainty can be efficiently resolved with light post-processing. Specifically, we have implemented a meet-in-the-middle approach over two lists of size about  $2^{20}$ . For the implementation written in C on a single thread, we can recover the key for a Hamming distance of 4 or less in less than a second.

### 3.5 Investigating Extremely High-Noise Oracle

For completeness, we estimate the performance of our approach when the oracle is extremely noisy. To reuse the analysis and experiments already presented in this paper, we construct a higher-accuracy oracle by repeating independent calls to the original (noisier) oracle and combining the outcomes. In other words, we reduce the noisy-oracle setting to the higher-accuracy setting treated elsewhere in the main body.

Concretely, let the basic (Bernoulli) oracle have accuracy  $\rho$ . After  $n$  repeated calls we observe  $\mathbf{y}$  containing  $k$  zeros. The likelihood of these observations assuming the true value is 0 is

$$\Pr(\mathbf{y} \mid 0) = \rho^k (1 - \rho)^{n-k},$$

while assuming the true value is 1 it is

$$\Pr(\mathbf{y} \mid 1) = (1 - \rho)^k \rho^{n-k}.$$

The likelihood ratio (and hence the posterior odds, up to the prior) is therefore

$$\frac{\Pr(\mathbf{y} \mid 0)}{\Pr(\mathbf{y} \mid 1)} = \left( \frac{\rho}{1 - \rho} \right)^{2k-n}.$$

We work in the log domain to avoid numerical underflow. We repeat oracle calls until the posterior probability for one hypothesis exceeds a prescribed threshold; equivalently, we stop when the log-likelihood ratio crosses the threshold

$$\log \frac{\rho'}{1 - \rho'},$$

which corresponds to a target effective accuracy  $\rho'$ . This procedure is essentially majority voting with soft information; the resulting posterior is passed into the decoder as soft information.

For ML-KEM-768, we take the noisy oracle accuracy  $\rho = 0.515$ , reported for the masked-hardware scenario in [Uen+22a]. Lifting the oracle to  $\rho' = 0.95$ , we can achieve on average 4 291 390 traces which has a ratio of 1.79 to the theoretical lower bound.

After we submitted this paper in August 2025 we became aware of a related paper in the ePrint by Hermelink, Mårtensson, and Tran [HMT25], which also touches upon very noisy oracles and reports results for the noisy binary PC oracle. They report a mean of 19 319 143 traces for ML-KEM-768 for  $\rho = 0.51$ , which has a ratio of 3.57 to the lower bound. Our approach with lifting to  $\rho' = 0.95$  achieves on average 9 690 509 traces with a ratio of 1.79, demonstrating the practical benefit of our attack.

We deliberately fixed  $\rho' = 0.95$  because the parameters and encodings were already optimized for that target. We believe, however, that further gains are possible: one could target a smaller accuracy  $\rho' < 0.95$  and optimize all the parameters for it. We leave the analysis of the optimal choice of  $\rho'$  as an open problem.

## 4 Application to GoFetch Attack on Kyber

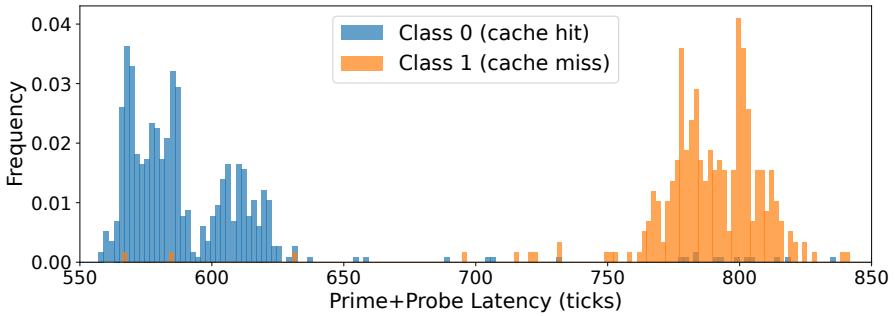
### 4.1 Adversary Model and Attack Phases

The GoFetch attack [Che+24b] is a microarchitectural side-channel attack capable of breaking constant-time cryptographic implementations. It exploits a hardware feature known as the Data Memory-Dependent Prefetcher (DMP), which is present in Apple M-series and Intel’s 13th generation CPUs.

Unlike traditional prefetchers that predict memory access patterns based on address history, the DMP inspects the content of data being loaded into the cache. If any data value resembles a memory pointer, the DMP attempts to “dereference” it by prefetching data from the supposed memory address. This behavior creates a powerful side channel: an attacker can craft a cryptographic input such that an intermediate value, dependent on a secret key, appears as a valid pointer if and only if the secret holds a specific value. By monitoring cache activity to detect whether the DMP prefetches from the target of this “pointer”, the attacker can learn information about the secret key.

In our attack, this mechanism is used to construct a decryption failure (DF) oracle. The attacker crafts a ciphertext that, upon decryption, produces a message containing either a target pointer or a slightly modified version with one flipped bit. The secret-dependent bit flip is designed such that the DMP is triggered on the target pointer in one case but not the other. By observing the DMP’s activity, the attacker can distinguish between these two outcomes, effectively creating a DF oracle to extract the secret key. Our method employs a ciphertext construction that allows targeting all secret key coefficients. This enables a full key recovery without the need for the computationally intensive lattice reduction step required by the original GoFetch approach and leverages a more robust statistical model for interpreting noisy measurements.

**Adversary Model.** The adversary executes unprivileged code on the same machine as the victim, but the attacker and victim processes operate in separate address spaces and do not share memory. This assumption, consistent with the GoFetch paper, precludes the use of common Flush+Reload cache attacks. Instead, the attack leverages a separate thread to perform high-resolution timing measurements for a Prime+Probe [OST06b] attack. Following the GoFetch methodology, our attack is divided into two primary stages: a **calibration phase** and a **key-recovery phase**. While we assume the availability of high-resolution timers typical of native execution environments, we consider the further restriction of browser-



**Figure 5:** Timing difference between cache hit and cache miss during the attack (1 000 measurements)

based sandboxes—where such timers may be jittered or unavailable—to be outside our current scope and a subject for future work.

**Phase 1: Calibration.** The calibration phase prepares the environment for the attack and profiles the target system’s timing characteristics. This phase consists of three steps. First, an **Eviction Set Search** is performed to find groups of memory addresses that map to the same cache set, which is necessary for the Prime+Probe technique. This initial step does not require interaction with the victim process. Second, the attacker builds **Compound Eviction Sets**, as introduced by GoFetch. These are pairs of L2 eviction sets chosen so that one evicts the victim buffer where the crafted pointer may reside during computation, while the other targets the cache set of the pointer’s eventual dereference. Finding these sets is essential for isolating the DMP’s activity and requires sending a large volume of ciphertexts to the victim. These two steps are essentially unchanged from GoFetch, as it is the necessary precondition to exploit the DMP. The final step is **Profiling**, where we model the distributions of cache hits and misses as Gaussian. We collect 1 000 timing measurements to establish parameters  $(\mu_0, \sigma_0)$  for cache hits (Class 0) and  $(\mu_1, \sigma_1)$  for cache misses (Class 1, DMP activation), as illustrated in Figure 5. This profiling step may be optional on similar hardware, but we include it explicitly as our results differ from those in [Che+24b, Figure 9], likely due to hardware differences between our MacBook Air (Apple M1) and their Mac Mini (Apple M1), both running macOS 13.5.

The rationale for this profiling is to frame the oracle’s response as a hypothesis testing problem. During the attack, for a new set of  $k$  timing measurements  $\mathbf{t} = \{t_1, \dots, t_k\}$ , we evaluate two competing hypotheses: Hypothesis 0 ( $H_0$ ), that  $\mathbf{t}$  was sampled from the cache-hit distribution  $\mathcal{N}(\mu_0, \sigma_0^2)$ , and Hypothesis 1 ( $H_1$ ), that it was sampled from the cache-miss distribution  $\mathcal{N}(\mu_1, \sigma_1^2)$ . Our goal is to determine the posterior probability of each hypothesis, for which we calculate

$P(H_0|\mathbf{t})$  using Bayes' theorem with balanced priors

$$\Pr(H_0|\mathbf{t}) = \frac{\Pr(\mathbf{t}|H_0) \Pr(H_0)}{\Pr(\mathbf{t})}, \quad \Pr(H_0) = \Pr(H_1) = 0.5.$$

During the attack, we filter outliers (e.g., timings greater than 1 500 ticks) and accumulate measurements until either  $\Pr(H_0|\mathbf{t})$  or  $\Pr(H_1|\mathbf{t})$  exceeds a confidence threshold of 0.85. This method provides a more stable oracle despite measurement noise. While the posterior probability provides valuable soft information, we surprisingly found that key recovery was more effective when treating the oracle as a binary output (i.e., outputting 0 if  $\Pr(H_0|\mathbf{t}) \geq 0.5$ ). This observation may warrant further investigation.

**Phase 2: Key Recovery.** The key-recovery phase, which follows the strategy detailed in Section 3.4, is orchestrated by three cooperating processes. The first process implements the logic of the attack, deciding which ciphertext to test next based on the oracle's previous responses. The second process is responsible for interacting with the victim; it takes a ciphertext from the first process, sends it to the victim, executing the Prime+Probe measurement loop. This process uses a dedicated thread to create a high-resolution timer for its measurements and reports back the final conditional probability after several measurements. The victim process in a loop sends its public key, receives a ciphertext, then runs a decapsulation. While the first two processes could be merged into a single one, we separated the main logic into a distinct Python process for convenience during implementation.

## 4.2 Executing the Attack

To instantiate the decryption failure oracle, we craft ciphertexts that cause the decrypted message to contain a specific pointer,  $\text{ptr}$ , which is learned during the calibration phase. The core of the technique is to create a predicate, dependent on the secret key, that determines whether the decrypted message contains the exact  $\text{ptr}$  or a slightly modified version,  $\text{ptr}'$ . By observing whether the DMP dereferences  $\text{ptr}$ , we can learn the outcome of the predicate. As established in [Che+24b], the DMP is insensitive to bit flips in the upper 8 and lower 7 bits of a 64-bit pointer, a property that was a limitation but turns out as a helpful tool in our case.

**Ciphertext for Basic Encodings.** To create the encodings described in Section 3.2, we first define  $\text{ptr}' = \text{ptr} \oplus (1 \ll i)$ , where the bit at position  $i$  is within the DMP's sensitive region ( $7 < i < 56$ ) and  $\text{ptr}[i] = 1^2$ . We then construct the ciphertext component  $\mathbf{v}'$  such that the decrypted message is an array of pointers, starting with  $\text{ptr}'$  followed by three random pointers that reside within the same 4 GB

---

<sup>2</sup>There is an off-by-one error comparing an inequality over  $\text{ptr}'[i] = 1$  to  $\text{ptr}[i] = 0$ , thus, we fix  $\text{ptr}'[i] = 0$  to follow Section 3.

Table 3: Aggregated results for 73 successful ML-KEM-512 key recoveries

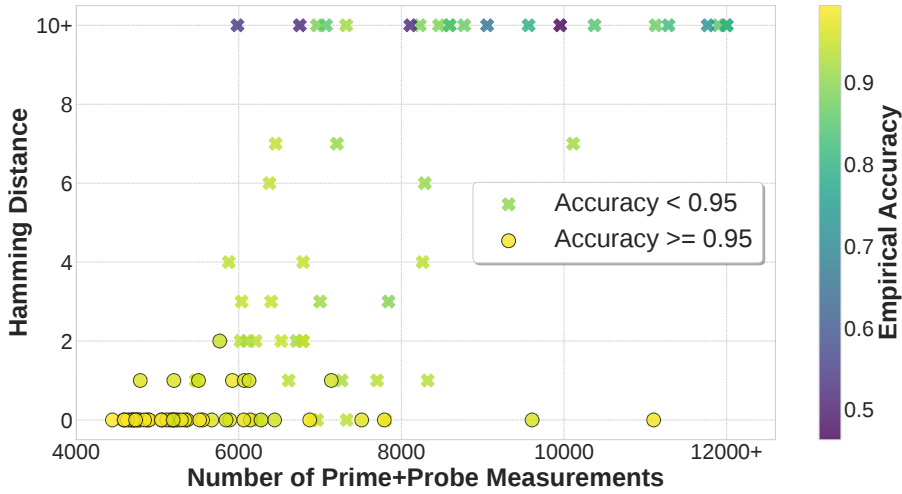
| Measurement repetitions | Empirical accuracy | Traces  | Hamming distance |
|-------------------------|--------------------|---------|------------------|
| 2.69                    | 0.9614             | 6047.84 | 0.73             |

memory region. The DMP will prefetch  $\text{ptr}$  if and only if the decryption result is modified back from  $\text{ptr}'$  to  $\text{ptr}$ , which happens when the chosen secret-dependent predicate is satisfied.

Crucially, for our attack, the value of  $\mathbf{v}$  remains fixed across all oracle queries. To target different coefficients of the secret key  $\mathbf{s}$ , we modify the nonzero positions within the ciphertext component  $\mathbf{u}$ . This approach allows us to probe any coefficient of the secret key, thereby overcoming the DMP sensitivity limitations that constrained the original GoFetch attack and enabling a full key recovery without subsequent lattice reduction.

**Ciphertext for Full Rotation Encodings.** Constructing the full rotation encodings from Section 3.3 is less straightforward, as the goal is to detect if the decrypted message contains  $\text{ptr}$  at *any* of four positions. A naive idea is to construct  $\mathbf{v}'$  to correspond to a message of four identical pointers,  $[\text{ptr}', \text{ptr}', \text{ptr}', \text{ptr}']$ . The rationale is that if the secret causes any of these to become  $\text{ptr}$ , the DMP will prefetch it, and probing will result in a cache miss. Unexpectedly, this approach does not work. Our experiments show that the oracle is only sensitive to the first two pointers; for example, if the decrypted message is  $[\text{ptr}', \text{ptr}', \text{ptr}, \text{ptr}']$ , no prefetch occurs. This implies that the DMP may stop scanning an array of pointers after encountering the same repeating value at the beginning. This behavior, which reveals additional logic within the DMP, might be explained by a mechanism like the history filter mentioned in [Che+24b], designed to prevent redundant operations on repetitive data.

We bypass this limitation by constructing  $\mathbf{v}'$  such that the intended decrypted message is an array of slightly different pointers:  $[\text{ptr}' \oplus 1, \text{ptr}' \oplus 2, \text{ptr}' \oplus 4, \text{ptr}' \oplus 8]$ . The DMP ignores the lower 7 bits of a pointer, so these values all target the same L2 cache line. However, despite the prefetch targeting the same cache line, the pointers are technically different, which avoids the problem of the DMP halting on repetitive values. This allows us to create a reliable oracle that triggers if any of the four secret-dependent predicates are satisfied. We also note an unexplained empirical observation: this technique only works reliably when the flipped bit  $i$  in the construction of  $\text{ptr}'$  satisfies  $i \geq 14$ . We leave a full investigation of this behavior to future work.



**Figure 6:** Distribution of final Hamming distance versus the total number of Prime+Probe measurements for 100 ML-KEM-512 key-recovery attempts. Each point’s color indicates the empirical oracle accuracy for that run. Attack parameters were based on  $\rho = 0.95$ ; thus, circles nicely follow the theory

**Results of the Attack.** We ran the attack on 100 random secret keys to assess its performance. The initial profiling was performed only once, and the resulting Gaussian parameters were used for all subsequent tests. The results for ML-KEM-512 are shown in Figure 6, where the most accurate oracle cases are centered in the lower-left corner. Choosing a different confidence threshold can increase the chance of having a more accurate oracle. A full key recovery takes approximately 25 minutes, with nearly half of this time dedicated to the calibration phase. As expected with a timing-based side channel, the results show considerable variance, with some runs exhibiting significant noise. In 73 of the 100 trials, the recovered secret key had a Hamming distance of 4 or less from the true key, which we classify as a success. The aggregated statistics for these successful runs are presented in Table 3. The number of traces reported includes only those made during the key-recovery phase.

As seen in Figure 6, the empirical accuracy of the oracle exhibits significant variability. Our experiments reveal that the attack occasionally suffers from bursts of consecutive incorrect oracle responses. This phenomenon is likely attributed to system-level noise, such as operating system interrupts, context switches, or background processes. These events can inadvertently evict cache lines during the Prime+Probe measurement window, causing false cache misses that the attacker misinterprets as DMP activity. Despite these occasional bursts of noise, we find that increasing measurement redundancy to guarantee a success rate close to 1 is suboptimal. In our current setup, it is cheaper (in terms of number of traces) to accept a non-zero failure rate and simply rerun the key recovery phase when

necessary.

### 4.3 Comparison with the GoFetch Attack

Our proposed attack demonstrates a significant improvement in efficiency and capability when compared to the GoFetch attack presented in [Che+24b]. The advantages are evident in several key areas.

Most critically, our method drastically reduces the number of traces required for key recovery. This is due to a fundamental difference in how the two attacks handle measurement noise. The GoFetch attack on Kyber-512 relies on repetition to create a *highly accurate* oracle; it uses 32 traces for each ciphertext query, effectively performing a majority vote to overcome noise. This approach leads to a total trace count of 100 352:

$$\begin{array}{rcl}
 & 392 \text{ coefficients} & \\
 \times & 8 \text{ ciphertexts/coefficient} & \\
 \times & \mathbf{32} \text{ traces/ciphertext} & \\
 \hline
 = & 100\,352 \text{ traces} &
 \end{array}$$

This extensive data collection is coupled with a computationally intensive post-processing step, which the authors report requires an average of 7.5 hours on a high-performance server (an Intel Xeon Platinum 8352Y with 128 logical cores and 1.48 TB of RAM). In stark contrast, our methodology is designed to work directly with noisy oracle responses. By integrating powerful error-correction techniques, we eliminate the need for extensive repetition, requiring only a small number of traces per query. This results in a tenfold reduction in the total online trace acquisition. Although our experimental setup exhibits a slightly lower noise floor (as seen by comparing Figure 5 with [Che+24b, Figure 9]), this dramatic improvement is overwhelmingly attributable to our novel attack methodology, not the hardware platform.

Finally, our attack overcomes a major limitation of the GoFetch methodology that prevents it from scaling to higher-security schemes. The GoFetch attack is fundamentally constrained by the physical behavior of the Data Memory-dependent Prefetcher (DMP). As detailed in their analysis, the DMP ignores the leading 8 and trailing 7 bits of any 64-bit value it interprets as a pointer. This means that for every block of 64 secret coefficients, a portion of the secret remains unrecoverable through their leakage mechanism. For ML-KEM-512, this results in 392 of 512 recoverable coefficients, leaving 120 unknown. While this smaller number of unknowns can be solved with a computationally intensive lattice reduction, the problem becomes intractable for ML-KEM-768 and ML-KEM-1024. For these larger parameter sets (768 and 1024 coefficients, respectively), the number of unrecoverable coefficients increases proportionally, making the search space for the final lattice reduction step too large to be practical. Our method does not

suffer from this constraint and can recover all secret coefficients, enabling a full key recovery even for these schemes with a higher security level.

## 5 Discussions

We now discuss the key properties of our attack methodology that highlight its practical significance, as well as the implications for potential countermeasures.

### 5.1 Attack Properties

**Near-Optimal Efficiency.** Our attack is "near-optimal" in its query efficiency, approaching the theoretical Shannon lower bound for key recovery. By achieving a query-to-bound ratio as low as 1.35, our method demonstrates high efficiency with little room for further improvement. This result demonstrates that the information-theoretic bound is, in this case, a tight bound, which allows for an accurate evaluation of the security margins of cryptographic systems.

**Extension to Multi-Value PC Oracles.** Our attack's core methodology could be extended beyond one-bit-bound oracles to more powerful multi-value PC variants. Finding optimally balanced queries in this larger information space is a nontrivial optimization problem that we leave for future work.

**Applicability for ML-KEM-1024.** While our presentation focuses on ML-KEM-512 and ML-KEM-768, the attack framework extends directly to ML-KEM-1024. We prioritized the lower-security variants as they are more commonly targeted in side-channel analysis, particularly in the context of resource-constrained and embedded devices. The attack on ML-KEM-1024 would follow the same methodology as for ML-KEM-768. Although the secret key space for ML-KEM-1024 is larger by a factor of  $4/3$ , we anticipate that the required number of traces would increase by a smaller margin. This expected gain in efficiency stems from ML-KEM-1024's larger compression parameters ( $d_u, d_v$ ). The reduced compression allows for more fine-grained control when constructing the inequalities that form our oracle queries, enabling the creation of more balanced encodings. As each query would yield more information on average, the total number of traces needed for a full key recovery would be proportionally lower.

### 5.2 Countermeasures

**Implications for One-Bit-Bound Oracle Attacks.** Our work improves the mathematical exploitation of one-bit-bound oracles. Consequently, traditional software hardening approaches like masking and blinding, which aim to increase the noise and reduce the signal of side-channel leakage, remain conceptually valid. However, their practical security must be re-evaluated. By demonstrating a key recovery

that requires significantly fewer traces, we show that an amount of noise previously considered sufficient to deter an attacker may no longer be adequate. Implementations may require more robust hardening (e.g., higher-order masking) to provide the same level of security, which in turn impacts performance. The trade-off between security and efficiency for these countermeasures must be reconsidered in light of our more potent attack methodology. For instance, first-order masked software has been shown to yield high oracle accuracy (e.g.,  $\rho \approx 0.96$  [Uen+22a]), and our analysis in Section 3.5 indicates that even highly noisy hardware-masked scenarios ( $\rho \approx 0.515$ ) are potentially vulnerable.

**Implications for the GoFetch Scenario.** In the specific context of the GoFetch attack, our method improves the efficiency of exploiting the DF oracle, but it does not alter the fundamental leakage source: the Data Memory-Dependent Prefetcher (DMP). Therefore, the countermeasures proposed in the original GoFetch paper remain fully effective. Disabling the DMP via an ISA extension or restricting cryptographic computations to cores where the DMP is not active would completely mitigate this specific attack vector by eliminating the oracle at its source. Our work simply makes the leakage that is available far more dangerous.

## 6 Conclusion and Future Works

In this paper, we have proposed a new methodology for decryption failure (DF) oracle attacks on ML-KEM that challenges their perceived inefficiency. Our approach integrates adaptive query generation with the error-correction power of LDPC codes, introducing a hybrid framework. By constructing queries that yield balanced, high-entropy parity checks over multiple secret coefficients, we maximize the information gained from each oracle call. This technique significantly reduces the number of traces required for key recovery, demonstrating that a carefully constructed DF oracle attack can be more potent than its binary PC counterparts, especially in noisy environments. We validate the real-world applicability of our attack by applying it to the GoFetch side-channel vulnerability, achieving a substantial reduction in the required number of traces for successful key recovery.

We selected ML-KEM as our primary target due to its standardization as FIPS 203 and its anticipated long-term importance among lattice-based KEMs. This methodology can be broadened to encompass other lattice-based proposals, potentially guiding the selection of PQC standards beyond those established by NIST. Future work should include extending our current approach to address multi-value plaintext checking oracle attacks [Raj+23; Tan+23] on ML-KEM, aiming for performance optimization that approaches information-theoretic lower bounds—particularly in protected platforms with less accurate oracles. Further research should focus on identifying additional leakage sources from diverse physical platforms to instantiate the required DF oracle and apply our new attack framework; high-order masked platforms are particularly interesting cases. Additionally,

the broader implications of our new DMP reverse-engineering results warrant further exploration. The halting scan behavior we identified, along with its bypass technique, could be leveraged to enhance attacks on other cryptographic implementations. Finally, given that more powerful attacks can render even protected implementations vulnerable with fewer traces needed for key recovery, designing more efficient countermeasures offering higher-order protection at a reduced performance cost is an important area of investigation. While our current adversary model focuses on an unprivileged user on a multi-user system with access to high-resolution timers, exploring this attack within a browser sandbox remains a compelling direction for future work.

## Acknowledgement

This work was supported by the Swedish Research Council (grant numbers 2021-04602 and 2023-03654), the Swedish Civil Contingencies Agency (grant number 2020-11632), the Crafoord Foundation, and the Wallenberg AI, Autonomous Systems and Software Program (WASP), funded by the Knut and Alice Wallenberg Foundation. The computations and simulations were partly enabled by resources provided by LUNARC.

## Ethical Considerations

Our primary contribution is a general mathematical framework for dramatically improving oracle-based side-channel attacks, validating this framework through a specific instantiation against the GoFetch vulnerability on Apple hardware. In conducting this work, we navigated the ethical challenge of balancing the public benefit of identifying this heightened risk against the potential harm of publishing a more potent attack methodology. Below, we discuss the impact on stakeholders, our responsible disclosure process, and the broader implications for the security community.

**Stakeholder Analysis and Impact.** We considered the potential harm to several parties. First, for hardware vendors (in this case, Apple), our attack significantly reduces the resources required to exploit the Data Memory-Dependent Prefetcher (DMP) vulnerability, making the threat more immediate and accessible than previously understood. Specifically, larger security parameters are no longer secure. Second, for cryptographic designers (such as the NIST and CRYSTALS/ML-KEM teams), our findings demonstrate that any side-channel leakage instantiating a 1-bit oracle is far more dangerous than previously believed, capable of being exploited with near-optimal efficiency. Finally, users of affected systems face a higher risk of key theft.

However, we believe the long-term benefits to these stakeholders outweigh the risks. By demonstrating the severity of these improved attacks, we motivate the development and adoption of robust mitigations that will ultimately protect end-users and strengthen cryptographic libraries against future oracle-based threats. As a temporary measure, disabling DMP prevents this particular attack.

**Responsible Disclosure.** Given the increased severity of our new technique compared to the original GoFetch disclosure, we adhered to a strict responsible disclosure process.

We notified Apple and the CRYSTALS (Kyber/ML-KEM) team of our findings, providing them with the details of our improved attack efficiency. This ensures that software and hardware vendors have the necessary information to assess the urgency of deploying mitigations.

Our practical validation was exclusively performed on Apple silicon. While similar DMP mechanisms exist in other architectures (e.g., Intel), we focused our disclosure on the validated target. As the specifics of our attack rely on the M-series implementation and Intel is already aware of general DMP attack techniques, we did not extend the disclosure to Intel, focusing instead on the verified vulnerability.

**Broader Implications and Dual Use.** We acknowledge that our primary contribution has dual-use potential. While it empowers security researchers to identify and test vulnerabilities with greater efficiency, it could theoretically be used by adversaries to optimize attacks on other leakage sources (power, EM, or timing). However, we contend that suppressing this methodology would be more harmful to the security community. By publishing the mathematical underpinnings of these attacks, we enable defenders to accurately model the risk of “minor” leakages and design countermeasures that render 1-bit oracles imperfect or impossible to exploit in practical settings.

## Open Science

To facilitate the verification of our results and to encourage further research, we make our simulation code, attack implementation, and the data used to generate the figures in this paper publicly available in an open repository <https://doi.org/10.5281/zenodo.17899501>.

## References

- [Bäe+19] C. Bäetu et al. “Misuse Attacks on Post-quantum Cryptosystems”. In: *EUROCRYPT 2019, Part II*. Ed. by Y. Ishai and V. Rijmen. Vol. 11477. LNCS. Springer, Cham, May 2019, pp. 747–776.

- [Ber+25] D. J. Bernstein et al. “KyberSlash: Exploiting secret-dependent division timings in Kyber implementations”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2025.2 (2025), pp. 209–234.
- [Che+24b] B. Chen et al. “GoFetch: Breaking Constant-Time Cryptographic Implementations Using Data Memory-Dependent Prefetchers”. In: *USENIX Security*. 2024.
- [CVE24] CVE-2024-37880. *The Kyber reference implementation before 9b8d306, when compiled by LLVM Clang through 18.x with some common optimization options, has a timing side channel that allows attackers to recover an ML-KEM 512 secret key in minutes*. <https://www.cvedetails.com/cve/CVE-2024-37880/>. Accessed: 2025-07-11. 2024.
- [DAn+19b] J.-P. D’Anvers et al. *Timing attacks on Error Correcting Codes in Post-Quantum Schemes*. Cryptology ePrint Archive, Report 2019/292. 2019.
- [DFR18] J. Ding, S. R. Fluhrer, and S. RV. “Complete Attack on RLWE Key Exchange with Reused Keys, Without Signal Leakage”. In: *ACISP 18*. Ed. by W. Susilo and G. Yang. Vol. 10946. LNCS. Springer, Cham, July 2018, pp. 467–486.
- [DG25c] H. Dong and Q. Guo. “OT-PCA: New Key-Recovery Plaintext-Checking Oracle Based Side-Channel Attacks on HQC with Offline Templates”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2025.1 (Jan. 2025). Date of Publication: 2024/12/09, pp. 251–274.
- [Eat+18] E. Eaton et al. “QC-MDPC: A Timing Attack and a CCA2 KEM”. In: *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018*. Ed. by T. Lange and R. Steinwandt. Springer, Cham, 2018, pp. 47–76.
- [Flu16] S. Fluhrer. *Cryptanalysis of ring-LWE based key exchange with key share reuse*. Cryptology ePrint Archive, Report 2016/085. 2016.
- [FO99] E. Fujisaki and T. Okamoto. “Secure Integration of Asymmetric and Symmetric Encryption Schemes”. In: *CRYPTO’99*. Ed. by M. J. Wiener. Vol. 1666. LNCS. Springer, Berlin, Heidelberg, Aug. 1999, pp. 537–554.
- [Gas+25b] S. Gast et al. “CounterSEVeillance: Performance-Counter Attacks on AMD SEV-SNP”. In: *Network and Distributed System Security Symposium 2025: NDSS 2025*. 2025.

- [GJN20a] Q. Guo, T. Johansson, and A. Nilsson. “A Key-Recovery Timing Attack on Post-quantum Primitives Using the Fujisaki-Okamoto Transformation and Its Application on FrodoKEM”. In: *CRYPTO 2020, Part II*. Ed. by D. Micciancio and T. Ristenpart. Vol. 12171. LNCS. Springer, Cham, Aug. 2020, pp. 359–386.
- [GM23] Q. Guo and E. Mårtensson. “Do Not Bound to a Single Position: Near-Optimal Multi-positional Mismatch Attacks Against Kyber and Saber”. In: *Post-Quantum Cryptography - 14th International Workshop, PQCrypto 2023*. Ed. by T. Johansson and D. Smith-Tone. Springer, Cham, Aug. 2023, pp. 291–320.
- [GMÅ24] Q. Guo, E. Mårtensson, and A. Åström. “The Perils of Limited Key Reuse: Adaptive and Parallel Mismatch Attacks with Post-processing Against Kyber”. In: *CiC 1.3* (2024), p. 21.
- [Guo+22a] Q. Guo et al. “Don’t Reject This: Key-Recovery Timing Attacks Due to Rejection-Sampling in HQC and BIKE”. In: *IACR TCHES 2022.3* (2022), pp. 223–263.
- [Guo+23b] Q. Guo et al. “SCA-LDPC: A Code-Based Framework for Key-Recovery Side-Channel Attacks on Post-quantum Encryption Schemes”. In: *ASIACRYPT 2023, Part IV*. Ed. by J. Guo and R. Steinfeld. Vol. 14441. LNCS. Springer, Singapore, Dec. 2023, pp. 203–236.
- [Her+23] J. Hermelink et al. “Belief Propagation Meets Lattice Reduction: Security Estimates for Error-Tolerant Key Recovery from Decryption Errors”. In: *IACR TCHES 2023.4* (2023), pp. 287–317.
- [Her+24] J. Hermelink et al. “The Insecurity of Masked Comparisons: SCAs on ML-KEM’s FO-Transform”. In: *ACM CCS 2024*. Ed. by B. Luo et al. ACM Press, Oct. 2024, pp. 2430–2444.
- [HGS99] C. Hall, I. Goldberg, and B. Schneier. “Reaction Attacks against several Public-Key Cryptosystems”. In: *ICICS 99*. Ed. by V. Varadharajan and Y. Mu. Vol. 1726. LNCS. Springer, Berlin, Heidelberg, Nov. 1999, pp. 2–12.
- [HMT25] J. Hermelink, E. Mårtensson, and M. Tran. “Noise-Tolerant Plaintext-Checking Oracle Attacks—A Soft-Analytic Approach Applied to ML-KEM”. In: *Cryptology ePrint Archive* (2025).
- [HPP21] J. Hermelink, P. Pessl, and T. Pöppelmann. “Fault-enabled chosen-ciphertext attacks on Kyber”. In: *International Conference on Cryptology in India*. Springer. 2021, pp. 311–334.
- [Hua+23a] S. Huang et al. “Cache-Timing Attack Against HQC”. In: *IACR TCHES 2023.3* (2023), pp. 136–163.

- [Li+25] J. Li et al. “Grafted Trees Bear Better Fruit: An Improved Multiple-Valued Plaintext-Checking Side-Channel Attack Against Kyber”. In: *Design, Automation & Test in Europe Conference, DATE 2025, Lyon, France, March 31 - April 2, 2025*. IEEE, 2025, pp. 1–7.
- [Mon+24] P. Mondal et al. “A Practical Key-Recovery Attack on LWE-Based Key-Encapsulation Mechanism Schemes Using Rowhammer”. In: *ACNS 2024, Part III*. Ed. by C. Pöpper and L. Batina. Vol. 14585. LNCS. Springer, Cham, Mar. 2024, pp. 271–300.
- [Nat24] National Institute of Standards and Technology. *Module-Lattice-Based Key-Encapsulation Mechanism Standard*. Tech. rep. FIPS PUB 203. U.S. Department of Commerce, Aug. 2024.
- [Ngo+21a] K. Ngo et al. “A Side-Channel Attack on a Masked IND-CCA Secure Saber KEM Implementation”. In: *IACR TCHES 2021.4* (2021), pp. 676–707.
- [OST06b] D. A. Osvik, A. Shamir, and E. Tromer. “Cache attacks and countermeasures: the case of AES”. In: *Cryptographers’ track at the RSA conference*. Springer. 2006, pp. 1–20.
- [PPM17] R. Primas, P. Pessl, and S. Mangard. “Single-Trace Side-Channel Attacks on Masked Lattice-Based Encryption”. In: *CHES 2017*. Ed. by W. Fischer and N. Homma. Vol. 10529. LNCS. Springer, Cham, Sept. 2017, pp. 513–533.
- [PT19] T. B. Paiva and R. Terada. “A Timing Attack on the HQC Encryption Scheme”. In: *SAC 2019*. Ed. by K. G. Paterson and D. Stebila. Vol. 11959. LNCS. Springer, Cham, Aug. 2019, pp. 551–573.
- [Qin+21] Y. Qin et al. “A Systematic Approach and Analysis of Key Mismatch Attacks on Lattice-Based NIST Candidate KEMs”. In: *ASIACRYPT 2021, Part IV*. Ed. by M. Tibouchi and H. Wang. Vol. 13093. LNCS. Springer, Cham, Dec. 2021, pp. 92–121.
- [Raj+23] G. Rajendran et al. “Pushing the Limits of Generic Side-Channel Attacks on LWE-based KEMs - Parallel PC Oracle Attacks on Kyber KEM and Beyond”. In: *IACR TCHES 2023.2* (2023), pp. 418–446.
- [Rav+20a] P. Ravi et al. “Generic Side-channel attacks on CCA-secure lattice-based PKE and KEMs”. In: *IACR TCHES 2020.3* (2020), pp. 307–335.
- [Rav+22a] P. Ravi et al. “Will You Cross the Threshold for Me? Generic Side-Channel Assisted Chosen-Ciphertext Attacks on NTRU-based KEMs”. In: *IACR TCHES 2022.1* (2022), pp. 722–761.

- [Rav+24b] P. Ravi et al. “Side-channel and Fault-injection attacks over Lattice-based Post-quantum Schemes (Kyber, Dilithium): Survey and New Results”. In: *ACM Trans. Embed. Comput. Syst.* 23.2 (2024), 35:1–35:54.
- [Sch+20] P. Schwabe et al. *CRYSTALS-KYBER*. Tech. rep. available at <https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions>. National Institute of Standards and Technology, 2020.
- [Sch+22b] T. Schamberger et al. “A Power Side-Channel Attack on the Reed-Muller Reed-Solomon Version of the HQC Cryptosystem”. In: *Post-Quantum Cryptography - 13th International Workshop, PQCrypto 2022*. Ed. by J. H. Cheon and T. Johansson. Springer, Cham, Sept. 2022, pp. 327–352.
- [SGG24] R. L. Schröder, S. Gast, and Q. Guo. “Divide and Surrender: Exploiting Variable Division Instruction Timing in HQC Key Recovery Attacks”. In: *USENIX Security 2024*. Ed. by D. Balzarotti and W. Xu. USENIX Association, Aug. 2024.
- [She+23b] M. Shen et al. “Find the Bad Apples: An efficient method for perfect key recovery under imperfect SCA oracles - A case study of Kyber”. In: *IACR TCHES 2023.1* (2023), pp. 89–112.
- [SLZ24] M. Shao, Y. Liu, and Y. Zhou. “Pairwise and Parallel: Enhancing the Key Mismatch Attacks on Kyber and Beyond”. In: *ASIACCS 24*. Ed. by J. Zhou et al. ACM Press, July 2024.
- [Tan+23] Y. Tanaka et al. “Multiple-Valued Plaintext-Checking Side-Channel Attacks on Post-Quantum KEMs”. In: *IACR TCHES 2023.3* (2023), pp. 473–503.
- [Uen+22a] R. Ueno et al. “Curse of Re-encryption: A Generic Power/EM Analysis on Post-Quantum KEMs”. In: *IACR TCHES 2022.1* (2022), pp. 296–322.
- [Wan+22] Y. Wang et al. “Hertzbleed: Turning Power Side-Channel Attacks Into Remote Timing Attacks on x86”. In: *USENIX Security 2022*. Ed. by K. R. B. Butler and K. Thomas. USENIX Association, Aug. 2022, pp. 679–697.
- [Wei+25] H. Weissteiner et al. “TEEcorrelate: An Information-Preserving Defense against Performance-Counter Attacks on TEEs”. In: *USENIX Security 2025*. 2025.

- [Xag+21] K. Xagawa et al. “Fault-Injection Attacks Against NIST’s Post-Quantum Cryptography Round 3 KEM Candidates”. In: *ASIACRYPT 2021, Part II*. Ed. by M. Tibouchi and H. Wang. Vol. 13091. LNCS. Springer, Cham, Dec. 2021, pp. 33–61.
- [Xu+20] Z. Xu et al. *Magnifying Side-Channel Leakage of Lattice-Based Cryptosystems with Chosen Ciphertexts: The Case Study of Kyber*. Cryptology ePrint Archive, Report 2020/912. 2020.

## Appendix A: Extended Analysis of Encoding Strategies

To provide better intuition for the “Full Rotation” encoding chosen in Section 3.3, we analyze three different main strategies for constructing oracle queries. We utilize a simplified “toy model” to make the index arithmetic transparent.

**Toy Model and Math Recap.** We consider a polynomial ring with degree  $n = 16$ . We impose a computational constraint  $\omega \leq 4$ , meaning the set of secret key coefficients influencing a *single* oracle query must have cardinality at most 4.

Recall from Section 3.2 that the decryption failure oracle checks inequalities based on the value  $(-\mathbf{s}^T \mathbf{u})[i]$ . By selecting specific nonzero positions in the noise vector  $\mathbf{v}$  (which correspond to check indices  $i$ ), we determine which linear combinations are validated.

**Strategy 1: Arbitrary disjoint encoding.** The simplest approach is to target an arbitrary set of  $\omega$  secret coefficients. We construct  $\mathbf{u}$  with  $\omega$  nonzero terms and set  $\mathbf{v}$  to have exactly one nonzero coefficient at index  $i$ . We can target any subset of the key; however, in general, it is difficult to have a balanced encoding.

**Strategy 2: The sliding window method.** We choose  $\mathbf{u}$  with nonzero coefficients separated by a fixed distance  $d$  (modulo  $n$ ). We then check multiple positions in  $\mathbf{v}$ , also separated by distance  $d$ . This causes the linear combination to “slide” over the secret key, creating a chain of overlapping dependencies.

*Example:* Let  $\mathbf{u} = z_0x^{15} + z_1x^{13}$  (distance 2). We check positions  $i \in \{0, 2, 4\}$ .

- $i = 0$ :  $(-\mathbf{s}^T \mathbf{u})[i] = s_1z_0 + s_3z_1$ .
- $i = 2$ :  $(-\mathbf{s}^T \mathbf{u})[i] = s_3z_0 + s_5z_1$ .
- $i = 4$ :  $(-\mathbf{s}^T \mathbf{u})[i] = s_5z_0 + s_7z_1$ .

The total set of involved variables is  $\{s_1, s_3, s_5, s_7\}$ , which has size 4 (satisfying  $\omega \leq 4$ ); see visual representation in Table 4.

**Table 4:** Sliding Window Dependencies ( $n = 16, \omega = 4$ ). Shaded cells indicate involved coefficients.

|         | $s_0$ | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ | $s_6$ | $s_7$ | ... |
|---------|-------|-------|-------|-------|-------|-------|-------|-------|-----|
| $i = 0$ | ·     | $z_0$ | ·     | $z_1$ | ·     | ·     | ·     | ·     | ... |
| $i = 2$ | ·     | ·     | ·     | $z_0$ | ·     | $z_1$ | ·     | ·     | ... |
| $i = 4$ | ·     | ·     | ·     | ·     | ·     | $z_0$ | ·     | $z_1$ | ... |

**Analysis:** With sliding window we can use  $1 < k < \omega$  inequalities, each over  $\omega - k + 1$  variables. The flexibility is quite high, as the distance could be chosen arbitrarily. Despite the overall encoding depending on  $\omega$  variables, each inequality has fewer than  $\omega$  variables, which reduces the granularity and the probability of having a balanced encoding. Additionally, some variables participate in more inequalities than others, as seen in Table 4,  $s_1$  is present only in the first inequality, while  $s_3$  is present in two of them. This strategy is not used in our paper, but we believe it might be of use in a different scenario.

**Strategy 3: Full rotation encoding.** This approach, described in Section 3.3, constructs a “closed loop”. Each out of  $\omega$  inequalities has the same set of  $\omega$  variables.

*Example:* Let  $\mathbf{u} = z_0x^{15} + z_1x^{11} + z_2x^7 + z_3x^3$ . We select non-zero positions  $i \in \{1, 5, 9, 13\}$  in  $\mathbf{v}$ . Carefully calculating  $(-\mathbf{s}^T \mathbf{u})[i]$  leads to Table 5.

**Table 5:** Full Rotation Dependencies. Note that every check column is fully populated.

|          | $s_2$  | $s_6$  | $s_{10}$ | $s_{14}$ |
|----------|--------|--------|----------|----------|
| $i = 1$  | $z_0$  | $z_1$  | $z_2$    | $z_3$    |
| $i = 5$  | $-z_3$ | $z_0$  | $z_1$    | $z_2$    |
| $i = 9$  | $-z_2$ | $-z_3$ | $z_0$    | $z_1$    |
| $i = 13$ | $-z_1$ | $-z_2$ | $-z_3$   | $z_0$    |

**Analysis:** The full rotation encoding allows us to consider  $\omega$  inequalities over  $\omega$  variables, which favorably compares to the sliding window approach; thus, each encoding is more balanced, and the key recovery is more efficient. The downside is the lack of flexibility in the choice of secret variables. In our approach, we resolve this in the attack’s second phase by adding random “linking” queries.

**Constraints and Extensions.** Two important implementation details distinguish these methods:

1. **Divisibility:** The full rotation approach relies on symmetric spacing, requiring  $\omega$  to be a divisor of  $n$ . Since  $n = 256$ ,  $\omega$  must effectively be a power

of 2. Strategies 1 and 2 do not have this restriction and can support any integer  $\omega$ .

2. **Cross-Polynomial Linking:** In ML-KEM, the secret key is a vector of polynomials. The full rotation method is rigid and operates within a single polynomial ring. However, strategies 1 and 2 can be easily adapted to construct  $\mathbf{u}$  with nonzero coefficients in different polynomial slots. We use the first strategy in the batch phase to create links between the disjoint sets of variables recovered by the full rotation phase.

---

# Popular Science Summary

---



# Popular Science Summary

---

## The Lock and the Locksmith

Every time you send a message, make a payment, or log into a website, cryptography is working invisibly in the background to keep your information safe. Modern encryption works like a padlock with a secret key: only the person who holds the right key can open it. For decades, the locks we have relied on have been strong enough — but a new threat is emerging on the horizon.

Quantum computers, machines that exploit the strange laws of quantum physics, have the potential to shatter many of the cryptographic locks that protect the internet today. While a powerful enough quantum computer does not yet exist, the concern is serious enough that governments and standards bodies around the world are already preparing. The United States National Institute of Standards and Technology (NIST) has spent nearly a decade running a global competition to find new kinds of cryptographic algorithms, the ones that even a quantum computer cannot break. Several winners have now been announced, and their deployment is beginning.

But here is the uncomfortable truth: a mathematically secure algorithm is only as safe as its implementation. A perfectly designed padlock is useless if the metal it is cast from is brittle, or if it leaks information about its key through a tiny crack. This thesis is about finding those cracks. Before attackers do.

## Listening to What a Computer Whispers

Computers are not silent. When a processor performs a computation, it consumes power, emits electromagnetic radiation, and takes a measurable amount of time. These physical signals are usually considered irrelevant noise, but for an attacker with the right equipment and knowledge, they can be a treasure trove of secret information. This class of techniques is known as *side-channel attacks*.

Imagine trying to open a heavy combination safe. Instead of spending years guessing every possible sequence of numbers, a safecracker presses a stethoscope to the metal door to listen for the faint internal clicks as the dial turns. Side-channel attacks work on the same principle: not breaking the algorithm directly, but eavesdropping on the physical world around it. This thesis constructs and evaluates such attacks against real hardware running real implementations of the

new post-quantum cryptographic algorithms that NIST has selected or shortlisted for standardization.

The central contribution is a new general-purpose attack framework that is substantially more efficient than prior methods. The key insight is that previous attacks wasted most of the available information by targeting one tiny piece of the secret key at a time. The new approach instead crafts cleverly designed decryption requests that cause the device to simultaneously reveal information about many pieces of the key at once. These partial observations are then assembled into a mathematical structure known as a low-density parity-check code. This structure is borrowed from the field of error-correcting codes used in wireless communications. A decoding algorithm then reconstructs the full secret key from the collected measurements.

The fewer measurements an attack requires, the more dangerous it is in practice. A device that must be monitored for thousands of operations gives its owner time to notice something is wrong, demands prolonged physical access, and produces enormous amounts of data that must be stored and processed. An attack that succeeds in a handful of measurements, by contrast, could be carried out in seconds by anyone briefly in the vicinity of the target — a colleague, a customs officer, or a malicious charging station. In the extreme, a *single* measurement suffices: the attacker need only observe the device once, with no repeated access and no possibility of detection. Pushing the measurement count down is therefore not merely a theoretical exercise in efficiency; it directly determines whether an attack is a laboratory curiosity or a genuine real-world threat. This thesis significantly advances the state of the art, reducing the number of measurements required to break protected implementations of several algorithms. And, for the first time, achieving full secret key recovery from a single measurement on a protected HQC implementation.

## Attacks Without a Wire

Not all side channels require physical access to a device. Modern processors are engineered for raw speed, and their aggressive performance optimizations can inadvertently leak secrets through purely digital means. One attack in this thesis targets Apple Silicon chips using a hardware feature designed to speed up memory access. By carefully crafting decryption requests and observing which memory addresses the processor automatically prefetches, an attacker sharing the same machine can extract a secret key. The new framework reduces the number of queries needed from over one hundred thousand to around six thousand, and extends the attack to stronger security levels that were previously out of reach.

## Privacy-Preserving Computation and Its Pitfalls

A separate line of work in this thesis examines *homomorphic encryption*, a remarkable cryptographic technology that allows computations to be performed directly

on encrypted data without ever decrypting it. This has obvious appeal for cloud computing: a hospital could send encrypted patient records to a cloud server, have the server compute a diagnosis, and receive an encrypted result. All of that without the server ever seeing the raw data. One prominent scheme for this purpose is CKKS, which is implemented in several widely used open-source libraries.

The research demonstrates that a known theoretical weakness in CKKS — the fact that decryption results reveal information about the secret key — is far more exploitable in practice than previously believed. In particular, a popular countermeasure that adds carefully sized random noise before sharing a decryption result was found to be broken in OpenFHE, the first major library to claim a secure implementation. The vulnerability arises because the library estimates the size of the noise in an optimistic, average-case manner. An attacker can deliberately violate this assumption by submitting specially constructed ciphertexts, causing the true noise to be far larger than the library expects and rendering its protective flooding noise ineffective. The full secret key of OpenFHE can be recovered in under three seconds on a laptop. The finding underscores that the gap between a provably secure design and a securely deployed system can be treacherously narrow.

### **Why This Work Matters**

The transition to post-quantum cryptography is one of the largest infrastructure upgrades the internet has ever undergone. Billions of devices will need to be updated, and the new algorithms will protect sensitive data for decades to come. Getting the implementations right from the start is therefore not an academic exercise, it is a matter of practical security for everyone who relies on the internet.

This thesis demonstrates that several published, protected implementations of leading post-quantum schemes have concrete and exploitable vulnerabilities. By identifying these weaknesses and developing the tools and techniques to exploit them, the work provides the research community and industry with the knowledge needed to build stronger countermeasures before widespread deployment. The full source code for all attacks is publicly released, so that implementers can test their own systems and engineers can build better defenses. The goal is not to undermine post-quantum cryptography, but to ensure that when we finally lock the door against the quantum threat, the lock itself is built to last.